A High-Dimensional Visualization System with Applications in Portfolio Management

Amy Tian

Advisor: Professor Han Liu

Submitted in partial fulfillment

of the requirements for the degree of

Bachelor of Science in Engineering

Department of Operations Research and Financial Engineering

Princeton University

June 2017

I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

_____

Amy Tian

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Amy Tian

Abstract

Often in modern multivariate analyses, data analysts rely solely on statistical estimators to explore the data. We are interested in visualizing high-dimensional correlation graphs as a way to verify numerical tests of dependence, which have far-reaching financial implications. High-dimensional visualization is problematic because (1) the number of plots to sort through increases quadratically as the number of variables increase, (2) it is tedious to verify numerical results with visual results and vice versa. We present a visualization tool that actively learns user preferences, applies the fitted classifier to unlabeled data, and outputs the difference among the numerical graph $G^{\text{num}} = (V, E)$ and the visual graph $G = (V, E)$. As a specific response to the aforementioned problems, we focus on the active learning and graph comparison components of the visualizer system. Both employ simulation studies in order to select the best procedure for use in the selection of healthcare stocks for a portfolio. The data is run through the visualization system, which utilizes the $----$ active learning algorithm and $----$ graph comparison method to select the correlation graph G^{num} closest to the system's output G. The portfolios are then put through a "buy and hold" strategy. Yearly returns are compiled, and the portfolio corresponding the selected correlation graph G^{num} is one of the top performers (??). The visualization tool provides a simple and intuitive way to improve predictive and portfolio management methods in the financial industry. Moreover (and arguably more importantly), it increases standardization in the data analysis process, thereby increasing accountability in an industry where ambiguity can mean a global financial crisis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem statement

More than 2.5 quintillion bytes of data are produced daily as the field of data analysis continues to grow. "Statistical thinking and methodology" has become the framework for disciplines such as education, agriculture, economics, biology, medicine, astronomy, geology, and physics [6], but there is still a lack of accountability and consistency in the field. What is striking in the current practice of data analysis is the lack of progress on this particular subject beyond the development of numerical methods. The rapid increase in computing resources has led to the proliferation of high-dimensional datasets, which are more tedious to efficiently understand patterns in the data and verify numerical estimators. In fact, the "physical limitations of display devices and our [human's] visual system prevent the direct display and instantaneous recognition of structures with higher dimensions than two or three" [12]. One solution is to manually plot each explanatory variable against the response variable, but this becomes computationally tedious and unfeasible to sort through when there are even a few hundred variables. This problem gets even more complicated when considering interaction terms (various transformations or combinations of explanatory variables).

Although methods for dimension reduction have been developed [12], it is still unclear how the analyst can easily check the resulting model to ensure that the variables which were culled in the dimension reduction process are actually undesirable. Thus, the problem with current high-dimensional visual analysis is two-fold: (1) there are too many potential plots to sort through manually, and (2) it is tedious to verify numerical results with visual results and vice versa.

In computer science, a framework for "clean code" has been extensively documented and is the accepted industry standard for writing, interacting with, and thinking about code. But in empirical data analysis with large datasets, analysts blindly depend on estimators and hypothesis tests to explore the data and have no justification of their analyses aside from asymptotic, mathematical guarantees. Furthermore, since each estimator inherently performs well or poorly under different settings, data analysts are unable to differentiate between the properties intrinsic to the dataset and the spurious properties the estimators added. Little, a Professor of Biostatics at the University of Michigan, notes that "developing good statistical solutions to real applied problems, based on good science rather than 'cookbookery,' is far from easy" [10]. This lack of agreement and "cookbook" mentality of data analyses has far-reaching consequences. It is simple to run the data through a list of many estimators and cherry-pick the most "interesting" result. Similarly, an analyst can remove undesirable data points without justification or unknowingly fit egregiously incorrect models. Regardless of whether all these situations are performed maliciously or with good intentions, the art of data analysis is unclear without standards. The lack of clear-cut guidelines makes it difficult for analysts to discern the "truth" from the data and avoid the aforementioned pitfalls while simultaneously making it difficult for consumers of the resulting analyses to evaluate how trustworthy it is. This mentality arises due to the difficulty in visualizing high-dimensional data; plotting is

one of the most consistent and universally interpretable "sanity checks" for numerical results.

Consider the following scenario with two different univariate datasets (Appendix A.1 and A.2). The problem is if x contains explanatory power of y. Common numerical analysis techniques yield the results summarized in Table 1.1.

Table 1.1: Numerical analysis in the univariate case. The results suggest that the data are uncorrelated. For Dataset 1, refer to Appendix A.1. For Dataset 2, refer to Appendix A.2

Dependency test	**Dataset 1**	**Dataset 2**
Linear regression	$y = 0.461 + 0.008x$	$y = -0.131 - 0.2699x$
p-values	(2e-16) (0.911)	(0.488) (0.190)
Conclusion	Insignificant	Insignificant
ANOVA p-value	0.9109	0.1896
Conclusion	Insignificant	Insignificant
Shapiro p-value	0.5795	0.1632
Conclusion	Normally-distributed residuals	Normally-distributed residuals
Pearson's correlation	-0.1886	0.0113
p-values	0.1896	0.9109
Conclusion	Uncorrelated	Uncorrelated

Supposing that an analyst must rely on numerical tests alone, the reasonable conclusion to reach would be that x and y are uncorrelated. Given the power to plot quickly and efficiently, however, an analyst would quickly discover that the data exhibits a strong dependency (Figure 1.1). There are certainly many more ways to numerically analyze the data, and in retrospect, it can be argued that an analyst might have tried an estimator that captured the dependency properly. Even then, without the ability to plot, the previous numerical results (which were strongly uncorrelated) cast doubt on the sole correlated estimator.

It is interesting to note that Dataset 2 (Figure 1.1, right) is clearly linear yet common tests of linear correlation (linear regression, Pearson's correlation. See Sec-

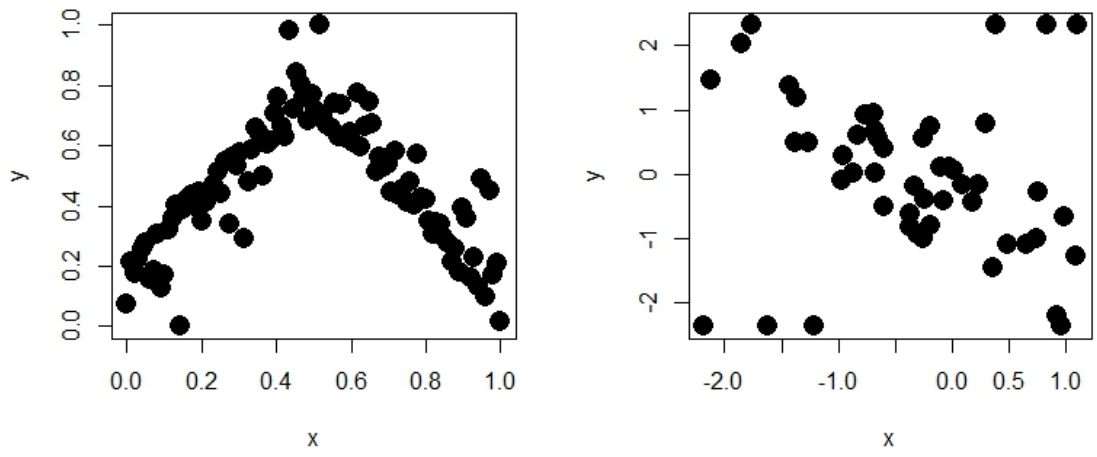

Figure 1.1: Visual analysis in the univariate case. The data exhibits a strong visual dependency but fails common numerical tests of dependence (Table 1.1). *Left*: Dataset 1 (Appendix A.1), *Right*: Dataset 2 (Appendix A.2)

tion 1.2) are not significantly different from 0 (Table 1.1). Indeed, the data used in these examples was purposefully constructed to be dependent but bypass common tests for dependency. However, if it is possible to construct datasets in one-dimension that evade commonly-used numerical methods, it is believable that it is even easier to construct analogous datasets in higher dimensions. Hence, no such standards of "clean analysis" currently exist in data science despite its importance in financial decisions, judicial evidence, government policy, and scientific discovery. Verification of numerical methods is especially important in finance as equity markets are large and involve billions of dollars; portfolio selection often involves determining the relationship among as many stocks as possible in order to be thorough and achieve the best possible portfolio.

1.2 Correlation graphs

Correlation graphs are one way to discover the dependency structure among different stocks whose returns may be represented as random variables following some distribution. Let $G^{\text{num}} = (V, E)$ be an undirected graph with vertices $V_1, ..., V_d$ (a d-dimensional distribution) and edges $E_{i,j} \in \{0, 1\}$. We set $E_{i,j} = 1$ when there is an edge between V_i and V_j, and 0 otherwise. An edge is drawn between V_i and V_j iff the two random variables are correlated. This graph can be drawn from a correlation matrix $\sum$ where $\sum_{i,j} = corr(V_1, V_2)$ with the following heuristic:

If $\sum_{i,j} > p$, draw edge $E_{i,j}$ where p is the p-value for the desired confidence level

---

We differentiate between "visual correlation" (which can be thought more of as "pairs of variables that marginally appear dependent") and the more common mathematical interpretation of "correlation". More specifically, we would like to visually understand what a correlation graph looks like and compare it to correlation graphs constructed with the traditional interpretation of correlation. Two variables that have a correlation coefficient near 0 may not necessarily be uncorrelated. A scatter plot can reveal this by showing outliers or patterns in the data that the analyst wasn't expecting; this was seen in Figure 1.1 (right). But in order to confirm that the coefficient used applies to each potential relationship, the analyst must plot all possible sets of data, which we have already established as computationally infeasible and tedious to sort through in high dimensions. We will present a solution in Section 1.4. What follows is an overview of common numerical methods to estimate the correlation between two random variables.

1.2.1 Pearson's correlation

Pearson's correlation measures the linear dependence among two random variables X and Y. In a population, the correlation is given by

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathrm{E}(XY) - \mathrm{E}(X)\mathrm{E}(Y)}{\sqrt{\mathrm{E}(X^2) - \mathrm{E}(X)^2}\sqrt{\mathrm{E}(Y^2) - \mathrm{E}(Y)^2}}$$

The formulation above is not as useful in practice as datasets are regarded as samples of a population. Given n observations, the sample expectation is given by the formula $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. By substituting into the above and multiplying by n^2/n^2, we can estimate the Pearson's correlation with the following:

$$\rho_{x,y} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}}$$

such that $-1 \leq \rho \leq 1$. With perfect positive and negative linear dependence respectively, $\rho = \pm 1$. It is important to note that $\rho = 0$ does not necessarily indicate independence, though it is an indication of **linear** independence.

1.2.2 Spearman's correlation

Spearman's correlation is more broad than Pearson's; it measures monotonic dependence among two random variables X and Y. Monotonic functions are either strictly increasing or decreasing; while linear functions are monotonic, monotonic functions are not necessarily linear. Subsequently, Spearman's correlations may also capture non-linear dependencies. Spearman's correlation is computed by computing the Pearson's correlation among "ranked variables". Each sample observation x_i of X is ranked from 1 to n based on its position relative to $x_j, j \in \{1, ..., n\} \backslash i$. The ranking is also computed for all observations of Y. We then define the difference of a

sample (x_i, y_i) as $d_i = x_i - y_i$ and compute Spearman's correlation as

$$\rho_{x,y} = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}$$

such that $-1 \leq \rho \leq 1$. With perfect increasing and decreasing monotonic dependence respectively, $\rho = \pm 1$. Again, it is important to note that $\rho = 0$ does not necessarily indicate independence, though it is an indication of **monotonic** independence.

1.2.3 Kendall's tau

Similar to Spearman's correlation, Kendall's tau is another method of identifying monotonic dependence among two random X and Y as it also computes correlation among ranked variables. However, it does not utilize the difference among a single sample. Instead, it compares pairs of samples among each other. For $i \neq j$, (x_i, y_i) and (x_j, y_j) are defined as "concordant" if the ranks of both elements agree i.e. $x_i > x_j$ and $y_i > y_j$ or $x_i < x_j$ and $y_i < y_j$. Pairs are defined as "discordant" if the ranks of both elements disagree i.e. $x_i > x_j$ and $y_i < y_j$ or $x_i < x_j$ and $y_i > y_j$. In the case where ranks of either element are equal, the pair is ignored. Let $c =$ the number of concordant pairs and $d =$ the number of discordant pairs. Then Kendall's tau is computed as

$$\tau_{x,y} = \frac{c-d}{n(n-1)/2}$$

such that $-1 \leq \tau \leq 1$. Kendall's tau is less sensitive to errors in the data as its correlation is based on sample pairs rather than deviations within an observation, though the resulting values tend to result in the same interpretations. As with Spearman's correlation, $\tau = \pm 1$ with perfect increasing and decreasing monotonic dependence respectively. Furthermore, $\tau = 0$ does not necessarily indicate independence, though it is an indication of **monotonic** independence.

1.2.4 Distance correlation

While the aforementioned correlation metrics are well-known and commonly used, they are constrained to monotonic functions. Distance correlation was first proposed in 2007 as a way to further test for non-monotone dependence between X and Y [18]. The distance correlation is a function of the distance covariance and distance variance of X and Y. We define the $n \times n$ matrices a and b as the distance matrices of X and Y respectively. The elements $a_{k,l}$ and $b_{k,l}$ are respectively defined as $||X_k - X_l||$ and $||Y_k - Y_l||$ for all $k, l = 1, 2, ..., n$ where $||z||$ is the Euclidean norm $\sqrt{z_1^2 + ... + z_n^2}$. Then we define the distance covariance and distance variance as

- $\mathrm{dCov}(X, Y) = \sqrt{\frac{1}{n^2} \sum_{k=1}^{n} \sum_{l=1}^{n} A_{k,l} B_{k,l}}$
- $\mathrm{dVar}(X) = \mathrm{dCov}(X, Y)$
- $\mathrm{dVar}(Y) = \mathrm{dCov}(Y, Y)$

where $A_{k,l} = a_{k,l} - \bar{a}_{k,} - \bar{a}_{,l} + \bar{a}$ and $\bar{a}_{k,}$ is the kth row mean of a, $\bar{a}_{,l}$ is the lth column mean of a, and $\bar{a}$ is grand mean of a. Similarly, $B_{k,l} = b_{k,l} - \bar{b}_{k,} - \bar{b}_{,l} + \bar{b}$ and $\bar{b}_{k,}$ is the kth row mean of b, $\bar{b}_{,l}$ is the lth column mean of b, and $\bar{b}$ is grand mean of b. The distance correlation is then defined as

$$\mathcal{R}_{X,Y} = \frac{\mathrm{dCov}(X, Y)}{\sqrt{\mathrm{dVar}(X)\mathrm{dVar}(Y)}}$$

such that $0 \leq \mathcal{R} \leq 1$. As before, $\mathcal{R} = 1$ indicates perfect dependence. However, the interpretation of $\mathcal{R} = 0$ is different, which is one of its two important properties [18]:

1. X and Y may be of different dimensions.
2. $\mathcal{R} = 0$ if and only if X and Y are independent.

1.3 Portfolio management

An important application of correlation graphs is in modeling the dependencies among financial equities. Determining the relationship among various stocks is especially useful when managing portfolios. One such methodology is the "buy and hold" tactic where an investor selects a portfolio of stocks and never rebalances. The idea is to select diversified stocks with low correlation and positive drift such that losses are offset by gains, and the portfolio gains on average. This strategy is especially useful when transaction costs are high as fees are prohibitive to rebalancing gains.

In a world with low transaction costs, however, there is more to be gained (less to be lost, alternatively) by frequently rebalancing the portfolio rather than holding. The concept for stock picking is similar to that of "buy and hold", though there are some key difference. Another component for successful rebalancing gains is for the stock returns to be relatively independent [11]. Thus, when one stock goes down, the others do not fall with it; with perfect independence, rebalancing gains become a function of the volatility of the stocks rather than of stock returns. This is further important because frequent asset trading may move the market and lead to unexpected price swings. With independent stocks, the price risk associated with rebalancing is minimized. In summary, rebalancing gains are best suited to markets with low correlation, independent returns, and high volatility [11].

Another financial application is the creation of a predictive stock model. Predictive models are a form of "model selection" where dependence rather than independence is important. Model selection addresses the problem of determining which explanatory variables (commonly seen as columns of the matrix X) are informative to explain the variation in the response variable (commonly seen as the vector Y). This is also related to the concept of "sparsity," a statistical term referring to the fact that many coefficients of a fitted model should be 0. Model selection with sparsity aids in the interpretability of the model since there are fewer variables for data analysts to

understand. This is another way to view rebalancing; the predictive model, if it is to be believed, can signal price swings to come which open up arbitrage opportunities. Furthermore, stocks which are uncorrelated are those with insignificant coefficients in the resulting model; this "negative space" point of view is partly utilized in the stock selection methodology for correlation graphs proposed in Section 5.2.

It must be noted that correlation alone cannot capture the full complexity of these financial applications; in fact, cases such as that of Figure 1.1 (right) reflect the limitations of correlation coefficients and reinforce the importance of visualization. An unassuming analyst might select the data as their correlation is not significantly different from zero (Table 1.1, Dataset 2), but it turns out that they have selected a stocks which we can see are highly correlated! However, it is still an important component of portfolio management in theory and in practice. Regardless, correlation graphs and their financial applications are not solutions to the two problems proposed in Section 1.1. Rather, correlation graphs and their financial applications are a concrete application of whatever the proposed solution is (A detailed application may be found in Chapter 5, and a stock selection methodology for correlation graphs is proposed in Section 5.2). What follows is a roadmap of the high-dimensional visualization solution that is developed further in this paper.

1.4 Summary

In this work, we tackle the problem by developing a sophisticated visualization system (abbreviated VS) to explore the data and numerical model in a different way. Specifically, we focus on two aspects of the VS which address the two problems raised in Section 1.1: (1) the procedure of sorting through plots is efficiently automated by learning the user's interests, and (2) the procedure of comparing the numerical and visual output is also automated. This allows the system to find visually interesting

relationships that the numerical model may have missed and/or toss out relationships which turn out to be uninteresting. Furthermore, this allows future analysts to combine visual feedback with the numerical feedback from estimators to make better decisions during data analysis and provide clear justification of their decisions.

A broad overview of the VS and its framework may be found in Chapter 2. Chapter 3 then focuses on the active learning stage of the VS, which is part of the first solution to the problems raised in Section 1.1. The chapter details and simulates various active learning methods to be used in the financial application in Chapter 5. Chapter 4 focuses on the second solution to the problems raised in Section 1.1. The chapter is concerned with the VS output, which quantifies the differences between numerical and visual correlation graphs for the user. Subsequently, the chapter details various graph comparison methods and ultimately selects one for usage in the financial application in Chapter 5. Finally, Chapter 6 recaps the work and presents future extensions of the VS.

Chapter 2

Visualization System

Regardless of whether high dimensional data visualization methods are computationally heavy or interaction heavy, user interactivity is a critical component of high dimensional visualization analysis; it is simply a question of what degree [12]. It is not enough to simply have user interaction, however. Given d variables, there are a total of $\binom{d}{2}$ possible scatter plots of the data. This blows up almost quadratically as d increases, which is infeasible for the analyst to sort through in any reasonable amount of time. Subsequently, automation is another necessary element in the task of visualizing high dimensional datasets.

We develop a system that first learns what visual patterns the data analyst finds promising, querying the user where the decision boundary is ambiguous. It then automatically iterates through thousands of possible plots and returns an adjacency matrix that captures the classifications of variable pairs. The VS may then return several plots for the user, perform line-up tests to refine the tree 6.1.5, or compare the visual graph with a numerical graph of the user's choice. This allows users to compare and contrast visual feedback with numerical algorithms for improved model selection. Figure 2.1 is a visual summarization of the system.

Figure 2.1: Broad overview of the visualization system.

Section 2.1 describes characteristics of an interesting plot; mindfulness and incorporation of these ideas in the VS facilitates improves user accuracy in stage 1 . Sections 2.2 and 2.3 provide a brief overview of stage 1 and stage 2 respectively (see Figure 2.1) while Section 2.4 describes the focus of the rest of the work which respond to the problems posed in Section 1.1.

2.1 Scatterplot characterization

2.1.1 Characteristics of a "good" plot

The simplest scatterplot is the response against the observed variables. This, however, may not be the best way to ascertain independence for the user. This notion is illustrated in Figure 2.2. The left plot appears to be independent as it's a cluster of points near the origin, but it's not entirely clear due to the multitude of stray points outside of $y \in (-2, 2)$ and $x \in (-1.5, 1.5)$. By looking at the outliers, it could also be argued that there is some dependency. However, applying the CDF in both directions creates a plot distributed on (0,1). This transformation is non-destructive and preserves dependency in the data if it exists. The data is clearly independent as the points appear to be uniformly distributed within the box.

Restricting the plot to a unit box allows analyst's visual systems to focus on locations where there is low spatial frequency, which is ideal for detecting dependence [8].

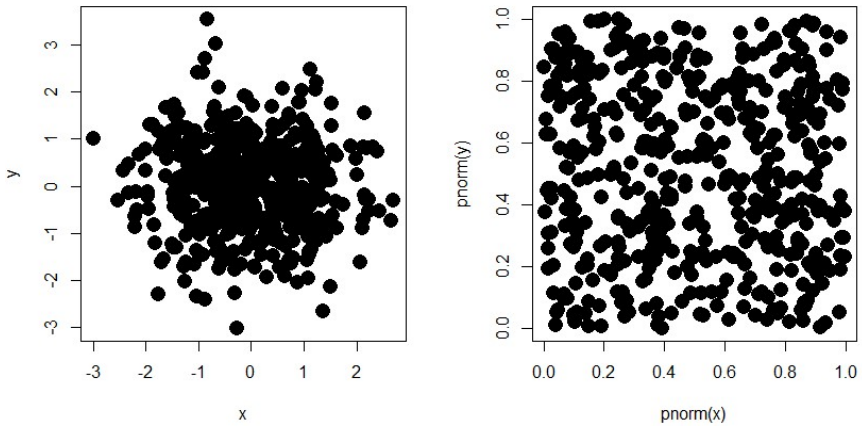

Figure 2.2: A plot of y against x with no transformation (left) and after the CDF is applied in both directions (right). The code for this example may be found in Appendix A.3

The effects of this can be progressively observed by looking from the left to the right in Figure 2.3 below.

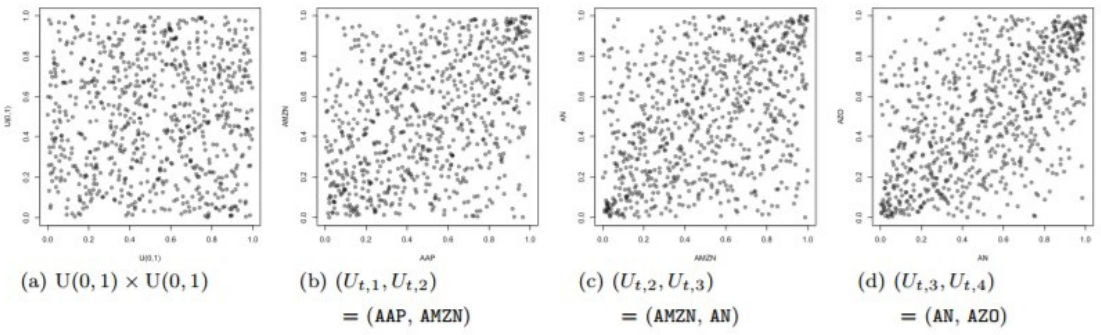

Figure 2.3: Scatterplots of (a) independent $U(0,1)$ random variables and (b,c,d) the pseudo-observation pairs $(U_{t,j}, U_{t,j+1}), j \in \{1,2,3\}$. Ticker abbreviations: AAP = Advanced Auto Parts, AMZN = Amazon.com Inc, AN = AutoNation Inc., AZO = AutoZone Inc. Figure from Hofert and Oldford 2016 [8]

2.1.2 Feature extraction from plot

In order for the active learning classifier to properly understand and classify all $\binom{d}{2}$ plots (Sections 2.2 2.3), the features must be extracted from each plot. The more useful criteria there are, the more sophisticated the classification will be.

Numerical features

Our goal is to quantify various features of a scatter plot for the computer, and that does include numerical features. The following features are currently implemented in the VS:

- Correlation coefficients and their p-values (See Section 1.2 for more details on the various types of quantifiers)
- Kullback-Leibler divergence criterion
- Chi-square test of independence and its p-value

Visual features

What is more challenging is to find a way to quantify the visual features of scatter plots. This may be done by looking for concentration of points in various spaces of the plot domain. The following features are currently implemented in the VS:

- Middle box criterion: Percentage of points near the center of the plot
- LR criterion: The percentage of points that lie above and below the linear regression line
- Clustering criterion: The percentage quantile of the ratio between the largest and next-largest distance
- Visual trend criterion: A higher value suggests a greater visual trend. This is computed by max(PosTrendCriterion, NegTrendCriterion) which is computed from the percentage of points in the upper left and bottom right, and bottome left and upper right, respectively

2.2 Active learning (Stage 1)

The main goal of stage 1 is to learn the user's interests. This requires the system to select ("query") data (which are n characterizations of $\binom{d}{2}$ plots in the case of the VS as described in Section 2.1.2) for the analyst (the "oracle") to label (classify). This is **stage 1**. The learner may then utilize a classification model (discriminant analysis, naive bayes, decision tree(s), logistic regression, etc.) that trains on labeled data to "learn" user interests. The user's interests are encoded in a classifier (some instance of the classification model) that is applied to automatically label the rest of the data (For more on the semantic differences between "classification model" and "classifier" in this body of work, see Figure 2.4). This is **stage 2** (Section 2.3). As such, it is important to make the process as efficient as possible to avoid redundancy for the end user. There are various methods that may be used for querying in stage 1 [5]:

- **Supervised learner**: This learner queries a single, random subset of all unlabeled data. It ignores the rest of the data when refining the classifier
- **Semisupervised learner**: Similar to a supervised learner, a semisupervised learner queries a single, random subset of all unlabeled data but proceeds to utilize the remaining unlabeled data to better inform the final classifier
- **Active learner**: An active learner selects its queries in a non-random, intelligent manner to reduce the hypothesis space $\mathcal{H}$ of all possible classifiers that may explain the data.

It has been shown that when a learning algorithm is allowed to choose its next query, it performs better with less training; as such, we choose to utilize active learning to select the plots to be queried by the oracle in stage 1 [17]. Chapter 3 goes into detail on different active learning methodologies as this section is primarily focused on its role in the system.

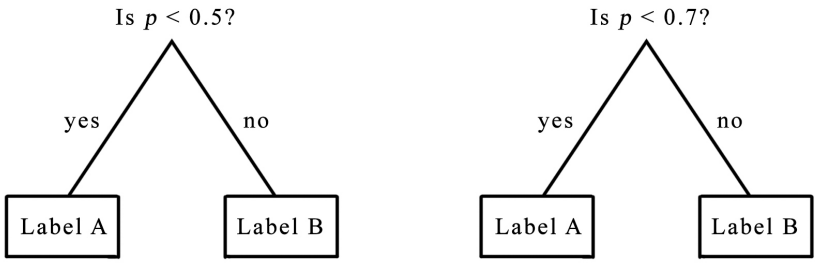

Figure 2.4: Although both figures on the left and right are slightly different classifiers, they are both instances of (extremely simple) decision trees, a type of classification model. Other models include discriminant analysis, naive bayes, random forest, logistic regression, etc. See Section 2.3.1 for more details on trees.

2.2.1 Initialization of active learner

It is problematic to start from scratch; how does the system determine the best first point of ambiguity when it knows nothing (the hypothesis space is everything)? A classic method is to simply select k random data instances for the user to label. As initialization is not the focus of this work, the VS currently utilizes this methodology.

Alternatively, we can exploit the fact that the user is already providing a numerical model that they believe to be a good representation of the data which they would like the visualization system to check visually. Given this data, the system may build a classifier that utilizes the various properties of the plots to determine whether one is interesting or uninteresting. Doing so greatly narrows the hypothesis space and makes it easier to determine points of ambiguity. However, to reconcile with the fact that the user wishes to check the numerical model and may not necessarily believe it is a good representation of fit, the learner must check whether the initial classifier is a proper fit (This may be achieved with line-up tests, which are described briefly in Section 6.1.5). As the user then proceeds to label various conditional plots (queried by the active learner) as "interesting" or "not interesting," the learner better understands the users interest, and the final classifier continues to evolve and improve.

2.2.2 Query selection

Post-initialization, the active learner cleverly queries vital plots so that the system can best learn the users interests. The system first determines which features it is uncertain about classifying and then returns a plot matching those characteristics to the user. This allows the system to utilize its classification model of choice to build a better classifier more efficiently. **It is important to distinguish between the active learner, which selects the next queries from the pool of unlabeled data and *may use its own classification model(s) to aid in query selection*, and the VS, which uses a *single classification model* to fit both the initialization and actively selected queries in order to build a model to classify the user's interests and label the remaining plots in Stage 2** (Section 2.3). Various active learning (selection) algorithms include uncertainty sampling, query by committee, query by bagging, and min-max clustering, all of which are described more thoroughly in Chapter 3.

2.3 Automated plot generation (Stage 2)

2.3.1 Decision tree classification of user interests

Given initialization and actively selected labeled data, what classification models can the visualization system use to create a final fitted model of user interests? A *decision tree* is composed of nodes (which correspond to classification labels) and branches (which correspond to decision boundaries). A tree is constructed at each node by sampling all M possible vertical and horizontal splits in the sample space and selecting the split which minimizes the *Gini criterion*. A mapping of the sample space to a tree is shown in Figure 2.5. The Gini criterion measures the homogeneity of the nodes in each side of the proposed split and is computed as follows (for a vertical

split)

$$G = N_{left} \sum_{k=1}^{K} p_{k,left}(1 - p_{k,left}) + N_{right} \sum_{k=1}^{K} p_{k,right}(1 - p_{k,right})$$

where N_s is the number of nodes in side s of a split ($s = \{left, right\}$ in a vertical split, and $s = \{top, bottom\}$ in a horizontal split) and $p_{k,s}$ is the fraction of class label k on side s of the split. Decision trees are a more sophisticated classification method than simple linear regression and retains interpretability. The root node is, naturally, most important as it corresponds to a split that optimizes the Gini criterion while the terminal nodes can be thought of as the data's homogeneous clusters. However, decision trees are also unstable; perturbing a single data point may change the entire tree. Furthermore, due to the nature of vertical and horizontal splitting,

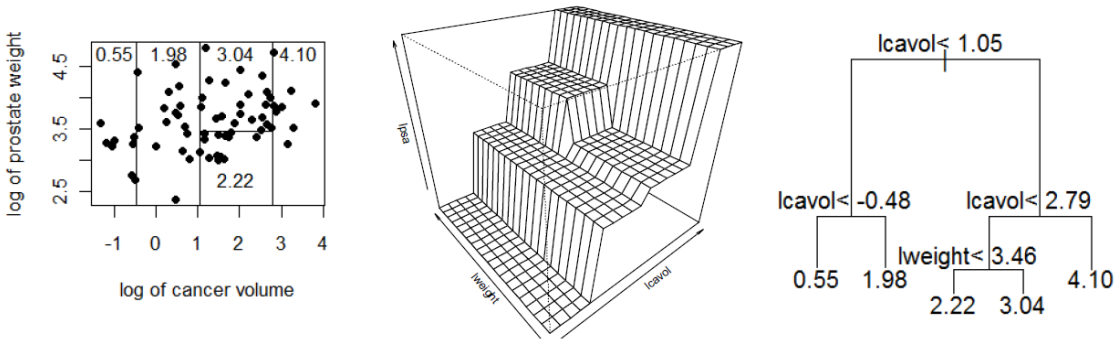

Figure 2.5: Mapping the sample space (left) to a decision tree (right). Images from Cutler [3].

A *random forest* provides solutions to the problems that a single decision tree faces. In a random forest, a forest is composed of many trees "grown" from random partitions of the labeled set (the training set). Furthermore, each decision tree is constructed by finding the best split among $m \in M$ splits. This allows each tree to specialize on a subset of the data, creating a more informative aggregate. A simplistic example of a forest with 2 trees may be found in Figure 2.4. Each tree in the forest has a vote of weight one for each unlabeled data, and the forest is

aggregated by majority vote, which makes the resulting decision boundaries more stable (less variant) on average. On the other hand, a forest with many trees is difficult to visualize. However, there are methods to simplify a forest into a single tree for the purposes of visualization (rather than for usage in classification, as the issue with stability would then remain). One such method for single tree approximate is presented by Zhou and Hooker [22]. Utilization of such a methodology aids in user interpretability of the system output while maintaining the robustness of a forest. As such, the VS sets the choice of classification model to random forest by default but allows the user to specify their preferred classification model if they wish to do so.

2.3.2 User interaction with active learning output

The system has now learned which of the unlabeled plots may be of interest to the user. The final learned classifier is used to fit the rest of the $\binom{d}{2}$ unlabeled scatterplots, and a visual graph $G = (V, E)$ may be built from these labels with the following heuristic where $i, j \in \{1, ..., d\}$:

If label$(i, j) > 0$ (i.e. "interesting" instead of "uninteresting"), draw edge $E_{i,j}$

Furthermore, the VS may provide a visualization of the resulting classifier itself such as the decision tree itself (as discussed earlier). The active learning output may also be visualized as a heat map. A classic heat map represents each pair of variables as colors from a bivariate spectrum. Heat maps are difficult to interpret as it is one-dimensional; each end of the color spectrum represents minimum and maximum values respectively. It is unclear what the max or the min is as it depends on the domain of the whatever the heat map is plotting; as such, the minimum may not necessarily be negative while the maximum may not necessarily be positive. Furthermore, the subtle variations in hue between colors make it difficult to compare relative ranking among different pairs for colors that are similar. Buja *et al.* propose an alternate,

clearer method of visualizing the heat map, termed the "association navigator" [2]. The association navigator is two-dimensional: the size of each color corresponds to its value while the color represents a positive or negative value [2]. As such, the association navigator only utilizes two colors rather than a spectrum of colors, making it simple to distinguish between the two. By simplifying the color scheme and adding the dimension of size, the association navigator makes it much easier to interpret and compare different pairs of data at a glance. The stark difference between a traditional heatmap and the association navigator when applied to the same dataset may be seen in Figure 2.6. The VS provides both options, allowing the analyst to select whichever is easier to for them to interpret. With these visualizations of the active learning output, the user should be able to understand his/her own interests.

Figure 2.6: Heatmap versus association navigator example.

2.3.3 System output

There are three options at the end of stage 2. Two of them are concrete outputs that may be easily used in an analyst's final report, and the third is a refinement of the active learning component in stage 1.

- **Automatic plot generation:** The VS compiles a selection of the most interesting and non-interesting plots along with their associated transformation variables.
- **Graph comparison:** The VS accepts a numeric graph (For example, a correlation graph G^{num} generated with numerical correlation coefficients as described in Section 1.2) and measures the difference between the numeric graph G^{num}

and visual graph G (the active learning output). Details on different graph comparison methods may be found in Chapter 4.

- **Line-up test:** In the event that the classifier is not a satisfactory representation of the analyst's interests, the VS may utilize line-up tests to help determine where to query from further. For more details on this methodology, see Section 6.1.5.

With the focus on correlation graphs in this work, graph comparison is the most useful output from the visualization system. As such, it is one of our primary VS focuses. The next section provides a roadmap for the rest of this work, which goes into detail on two aspects of the VS.

2.4 Specific focus: AL and GC

The active learning component is the bread and butter of the visualization system and forms the first stage of the system. Furthermore, it solves for the tedious nature of classifying $\binom{d}{2}$ plots in high-dimensional visualization , which was first brought up in Section 1.1. The second issue with high-dimensional visualization is the verification of the numerical results against the visual result. This is a question of the VS output, and graph comparison is a solution to this problem that is both informative to the analyst and further useful in the case of correlation graphs, which we are interested in for their financial applications. As such, the primary focus of this work that follows are active learning methods (Chapter 3) and graph comparison methods (Chapter 4). These methods are applied to the current iteration of the VS, which is then utilized in conjugation with numerical correlation graphs to perform stock selection.

It should be noted that the VS is a large project and may certainly be further refined despite the work that we do to refine two major components of the system. Ideas for future extensions of the VS may be found in Section 6.1. There are several recommendations for improvements that can be made to different aspects of the system which are not the focus on this work.

Chapter 3

Active learning

Active learning is a subset of machine learning wherein an algorithm is "trained" on labeled data instances in order to learn from and make predictions on the data. Consider a large set of unlabeled data X with a hidden label from a finite set Y that can be queried from some human "oracle"; we would like to learn a good classifier of the data, some mapping $h : X \to Y$ from the set $\mathcal{H}$ without making too many queries [5]. Active learning is the process of intelligently selecting the queries (a constrained resource) to learn as much as possible. When a learning algorithm is allowed to choose its next query, it performs better with less training [17]. To put it more concretely, the error (a measurement of the difference between the predicted labels and the "true" labels) converges to zero faster. This property is especially desirable when labeled data are difficult, time-consuming, or computationally expensive to obtain. Stage 1 of the VS (Section 2.2) is one such classification task where $Y \in \{\text{interesting, non-interesting}\}$. Classification and filtering tasks are both tedious and redundant, which necessitates intelligent selection of queries [17]. What follows is an active learning literature review in Section 3.1, an overview of active learning methods and their algorithms in Section 3.2, and a simulation study in Section 3.3 to

determine the method that is best-suited for usage with the VS in Chapter 5's equity application.

3.1 Literature review

It is important to consider the situation in which the learning algorithm receives the points in which it selects a query from. There are three different scenarios in which the learner may request queries as described by Settles [17]:

1. **Membership query synthesis**: The learner may select a query from an unlabeled samples in X.
2. **Stream-based selective sampling**: An unlabeled sample is randomly selected from X, and the learner decides whether to query or not.
3. **Pool-based sampling**: k unlabeled samples are randomly selected from X, and the learner picks one to query.

While the methods above may apply to many different active learning environments, the specific situation for the VS is pool-based selective sampling. Because it is computationally expensive to extract features from every single $\binom{d}{2}$ plot (Section 2.1.2), membership query synthesis is not an option. While stream-based selective sampling may work, it has a similar problem as membership query synthesis because there are no constrains on the number of samples that the algorithm may choose to discard.

The next problem, then, is to determine the informativeness of unlabeled instances that are presented by the methodologies above. This is the crux of active learning as it allows for intelligent selection of queries. Dasgupta identifies two different approaches to active learning that are fundamental drivers to the process of query selection [5]:

1. **Efficient search through hypothesis space $\mathcal{H}$**: The idea is to select a query that shrinks $\mathcal{H}_t$, the set of all possible classifiers at time t that explain the labeled data, as much as possible.
2. **Exploiting cluster structure in data**: The idea is to cluster the data and select queries based on cluster structure (i.e. query from each cluster). While

clusters may be split over time if they are discovered to be non-homogenous, a learner may leverage situations where clusters are fairly homogeneous in order to classify points by propagating the labels to its neighbors.

Uncertainty sampling and query by committee are active learning algorithms that are a form of the aforementioned efficient search through the hypothesis space. These algorithms may be found in Sections 3.2.1 to 3.2.3 respectively. We also present a clustering partitioning algorithm in Section 3.2.4 that seeks to exploit the cluster structure in the data.

3.2 Overview of active learning methods

In this section, we present algorithms for specific active learning methods and a reference to the implementation code, which can be found in the appendix. The `activelearning` package by `ramhiser` [15] has adapted much of the methods reviewed in Settles' work [17]. As such, most of the implementation code has been adapted and reworked (with the inclusion of substantial parts written from scratch) from the `activelearning` package, which is too general for our purposes.

3.2.1 Uncertainty sampling

In uncertainty sampling, the active learner selects the query q that it is most uncertain on how to label; in other words, the algorithm queries the label that has the highest posterior probability [9]. With binary classification labels such as the VS system (either "interesting" or "non-interesting"), this reduces to the case of querying the instance whose posterior probability of being "interesting" is closest to 0.5 [9]. While the algorithm presented later follows this methodology and may subsequently only be used with classification models that are able to encompass posterior probability computations, there has been much work done to expand uncertainty sampling to non-probabilistic classifiers such as decision trees and nearest-neighbor [17].

Uncertainty sampling is simple but not without problem. Given that the number of possible classification labels is $k > 2$, uncertainty sampling only considers the information about the most probable label i and ignores the other possible labels $j \in \{1, ..., k\} \backslash i$. "Margin sampling" and "entropy" are variants that try to solve for these problems, but both reduce to the scheme above (selecting q with a posterior probability closest to 0.5) when $k = 2$ [17].

We have developed an algorithm for uncertainty sampling based on the literature review (see Appendix A.4 for code):

Algorithm 1 Uncertainty sampling (as described by Settles [17])

procedure (X is a $n \times d$ matrix of d observations of all n variables, y is an n-length vector of labels for each variable in X (y_i=N/A when X_i, has no label))
    $tout \leftarrow \text{train}(X^{labeled}, y^{labeled}, \text{classifier})$
    $p \leftarrow \text{predict}(tout, X^{unlabeled}, \text{posterior prob.} = TRUE)$
    loop from $i = 1$ **to** $\text{len}(p)$:
        $p_i \leftarrow |p_i - 0.5|$
    return $\text{where}(p == \min(p))$

By searching for the most "uncertain" point, uncertainty sampling is able to further refine the classifier as the oracle must label the point either "interesting" or "non-interesting". This can be viewed as a search within the hypothesis space $\mathcal{H}$ that contains multiple classifiers, instances of a single classification model.

3.2.2 Query by committee

Query by committee is a clearer case of efficient search through the hypothesis space. In query by committee, a "committee" of classification models are trained on the current labeled instances. Each model represents a competing hypotheses, and its prediction for an unlabeled query candidate q is a "vote" of weight one on q; the most disagreeable candidate is then selected [17]. Settles reviews various methods for committee selection for both probabilistic and non-probabilistic models, though

the implementation of the algorithm (Appendix A.5) allows the user to specify the committee members [17].

We may write the basic algorithm as follows:

Algorithm 2 Query by committee (as described by Settles [17])

1: **procedure** (X is a $n \times d$ matrix of d observations of all n variables, y is an n-length vector of labels for each variable in X (y_i=N/A when X_i, has no label). Let C be the vector of all committee members)
2: **loop from** $i = 1$ **to** len(C):
3: $tout_i \leftarrow \text{train}(X^{labeled}, y^{labeled}, C_i)$
4: $p_{i,} \leftarrow \text{predict}(tout_i, X^{unlabeled})$
5: $d \leftarrow \text{disagreement}(p)$
6: **return** where($d == \max(d)$)

While it is simpler to constantly maintain the same committee throughout, it would be more informative to prune the committee as the algorithm proceeds; it may simply be the case that a model is ill-suited for the problem at hand and consistently returns predictions that skew the voting procedure. Subsequently, a model is removed from the committee if it is consistently out-of-line with whatever the "true" label of q_t (the next query point decided and then queried at time t) turns out to be. It is important to note that the "true" label is not known until *after* the algorithm has return the index of q_t. Subsequently, the committee from t is pruned at time $t + 1$ at the start of the algorithm using the retrieved label from time t. It should be noted that the pruning function should only be run after a good number of iterations have passed to allow the error ratios to converge (Otherwise, there is no room for learning). In our implementation, we implement the pruning algorithm after $iter/2$ queries where $iter$ is the maximum number of queries allowed. Furthermore, the query by committee methodology described by Settles is only focused on selection of q independent of the final classification model once the rest of the training data is selected [17]. However, there is merit in maintaining the final pruned committee as its own classification model after the querying is complete and the training data is selected. This may

be achieved by selecting labels on unlabeled instances via majority vote. Further details on implementation and performance may be seen in the simulation study (Section 3.3). The revised selection algorithm is as follows (see Appendix A.5 for code):

Algorithm 3 Query by committee (revised framework)

procedure (X is a $n \times d$ matrix of d observations of all n variables, y is an n-length vector of labels for each variable in X (y_i=N/A when X_i, has no label). Let C be the vector of all committee members, E be the error ratio of the respective committee members (initialized to 0), $0 < \epsilon < 1$ be some threshold for the error ratio, and t be the current iteration of QBC starting at $t = 1$)
    function QBC
        loop from $i = 1$ **to** len(C):
            $tout_i \leftarrow \text{train}(X^{labeled}, y^{labeled}, C_i)$
            $p_{i,} \leftarrow \text{predict}(tout_i, X^{unlabeled})$
        $d \leftarrow \text{disagreement}(p)$
        return $j = \text{where}(d == \max(d))$
    function ORACLE
        return l, the label of $X_{j,}$
    $y_j \leftarrow l$
    function PRUNE (Let *iter* be the total active learning budget. When iteration $i \in [1, iter] > iter/2$, run PRUNE)
        $prune = [\ ]$ (empty vector)
        loop from $i = 1$ **to** len(C):
            If $(p_{i,j} == y_j)$ **then** $iv = 0$ **Else** $iv = 1$
            $E_i = E_i + \frac{iv - E_i}{t}$
            If $E_i > \epsilon$ **then** $prune$.append(i)
        $t++$
        return $prune$
    loop from $i = 1$ **to** len($prune$):
        Delete E_{prune_i}, C_{prune_i}, $p_{prune_i,}$, and $tout_{prune_i}$

The algorithm contains a generic disagreement function, so it is important to note that there are different measures of disagreement among the committee members. There are two main methods of disagreement measurement described by Settles [17]:

1. **Vote entropy**: The disagreement for variable d is computed

$$x^*_{VE} = \arg\max_x - \sum_{j \in \text{all possible labels}} \frac{V(y_j)}{\text{len}(C)} \log \frac{V(y_j)}{\text{len}(C)}$$

   where $V(y_j)$ is the number of votes each possible label y_i for d received. The interested reader may refer to [4] for further details on this methodology.

2. **Kullback-Leibler divergence**: The disagreement is computed

$$x^*_{KL} = \arg\max_x \frac{1}{\text{len}(C)} \sum_{i \in 1}^{\text{len}(C)} \sum_{j \in \text{all possible labels}} \text{P}_{C_i}(y_j|x) \log \frac{\text{P}_{C_i}(y_j|x)}{\text{P}_C(y_j|x)}$$

   Since C was defined as the committee, we can interpret $\text{P}_C(y_j|x) = \frac{1}{\text{len}(C)} \sum_{k=1}^{\text{len}(C)} \text{P}_{C_k}(y_j|x)$ as the probability that the label with the most votes will be y_j where $\text{P}_{C_k}(y_j|x)$ is the probability that committee member k votes y_j for variable d.The interested reader may refer to [13] for further details on this methodology.

An implementation of vote entropy disagreement can be found in Appendix A.6. This function was imported from the `activelearning` package developed by `ramhiser` [15] and simply calls the `entropy` package in `R`.

3.2.3 Query by bagging

Query by Bagging and Boosting was proposed as a way to improve the performance of a single classifier by forming of committee of classifiers trained on random (weighted, in the case of Boosting) samples of the labeled data [1]. Bagging, quite simply, uniformly samples k training sets from the labeled data to form a committee of k classifiers trained on the same classification model [1]. The unlabeled data with the most disagreement among the committee members is then selected as the next oracle query (see Section 3.2.2 for details on disagreement measures). Thus, the algorithm is as follows (see Appendix A.7 for code):

Algorithm 4 Query by bagging (as described by Abe and Mamitsuka [1])

procedure (X is a $n \times d$ matrix of d observations of all n variables, y is an n-length vector of labels for each variable in X (y_i=N/A when $X_{i,}$ has no label). num_class is the desired number of committee members. Let $r \in (0, 1)$ such that $r * \text{len}(X^{labeled})$ rounded is the number of points to randomly sample from the labeled set.)
    loop from $i = 1$ **to** num_class:
        $idx \leftarrow \text{unif\_sample}(labeled, r * \text{len}(X^{labeled}))$
        $tout_i \leftarrow \text{train}(X_{idx,}, y_{idx}, \text{classifier})$
        $p_{i,} \leftarrow \text{predict}(tout_i, X^{unlabeled})$
    $d \leftarrow \text{disagreement}(p)$
    return $\text{where}(d == \max(d))$

Since each iteration of the active learner is a new process of random committee training and selection, there is no need to (1) maintain error ratios to prune the starting committee or to (2) use majority vote for the final fitted model over a random forest, which the VS utilizes for stage 2 (Section 2.3).

3.2.4 Min-max clustering

The main principle of the Min-Max Approach is to query points that are close to the the current labeled set but far from each other [20]. Given natural clustering in the data, the active learned would be able to select a healthy sample from each cluster with this methodology with the hope that each cluster is fairly homogeneous (Section 3.1). As such, the methodology is able to exploit the clustering structure of data but may perform more poorly (converge more slowly i.e. require more queries to get to a reasonable error level) in datasets that do not naturally form clusters. Naturally, there are two important considerations:

- **Initialization:** How does the active learner find the clusters in the first place? This is important because selecting points near the centers of the clusters, the densest regions, allows the algorithm to converge faster [20].
- **Query selection:** How does the active learner quantify the distance between data points? Euclidean distance is a common metric of the straight-line distance between points in the Euclidean space.

Vu *et al.* has proposed the creation of a k-nearest neighbors graph to measure the local density of each data point; the most dense points may be selected to initialize the active learner. Since the VS will be initialized randomly, we do not concern ourselves too much with min-max clustering initialization; the interested reader may refer to [20] for more of the mathematical details and a demonstration of viability. Instead, we present the simple algorithm for the actual active learning (querying) process (see Appendix A.8 for code):

Algorithm 5 Min-max clustering (as described by Vu *et al.* [20])

procedure (X is a $n \times d$ matrix of d observations of all n variables, y is an n-length vector of labels for each variable in X (y_i=N/A when $X_{i,}$ has no label))
    loop from $i = 1$ **to** len($y^{unlabeled}$):
        min $\leftarrow \infty$
        loop from $j = 1$ **to** len($y^{labeled}$):
            $d \leftarrow$ distance($X_{i,}, X_{j,}$)
            If (min $> d$) : min $\leftarrow d$
        $q_i \leftarrow$ min
    return where($q ==$ max (q))

3.3 Simulations

We utilize a simulation study in order to determine the method that is best-suited for usage with the VS in Chapter 5's equity application. To recap, these are key features of the visualization system that should be (and are) captured in the simulations:

- **Initialization (Section 2.2.1):** Initializing the active learner begins with a random selection of k data that is presented to the oracle for classification. Each simulation is initialized with 10 randomly selected data points.
- **Pool-based sampling (Section 3.1):** After initialization, k unlabeled samples are randomly selected from X, and the active learner picks one to query. Each simulation iteration (of the AL algorithm) is presented 15 unlabeled points to query from.
- **Random forest (Section 2.3.1):** The VS's overall classification model (for use when initialization and querying are complete) is a random forest. Each

active learning method in the simulations optimize for the final random forest classification model by utilizing random forests in their selection process (excluding QBC due to the nature of the algorithm). Instead, the QBC simulations have been run with a committee of classification models that includes random forest (See Section 3.3.3 for the full list). Furthermore, the QBC simulations have been run with both (1) majority vote and (2) random forest as the overarching classification, as well as (1) with pruning and (2) without pruning.

- **Bivariate classification:** The classification of user interests have two possible labels/levels: "interesting" and "not interesting". The simulations also use data with two levels of classification (Section 3.3.1).

Finally, it should be noted that each active learning algorithm is given a budget of 50 queries (50 progressive iterations of a single trial).

3.3.1 Data

The data is taken from the MNIST database of handwritten digits (`http://yann.lecun.com/exdb/mnist/`). The digits $(0, 1, ..., 9)$ have already been classified and may be visualized in the form of a 28×28 pixel array; the darkness of each pixel is represented by a value from 0 (light) to around 300 (dark). Each image has been transformed into a single 784-length vector by "unfurling" each row and adding it to the last column of the row above it. For ease of use and computation efficiency, the data has been further compressed to a 196-length vector (14×14 pixel image). In order to maintain the condition of bivariate classification, two out of ten digits were selected. The digits 7 and 9 were selected as they are rather similar, making it more difficult for an active learning algorithm to correctly parse the data with few queries (As opposed to 1 and 0, which are very different on the visual plane). The MNIST training set contains 60,000 total data points while the testing set contains 10,000 total data points. This was far too many, so the simulator selects 250 random samples (125 of each digit) from the training set to make up the final data. The final dataset may be visualized in Figure 3.1. The functions for working with the MNIST data were adapted from file `gist:39760` [14] and may be found in Appendix A.9.1.

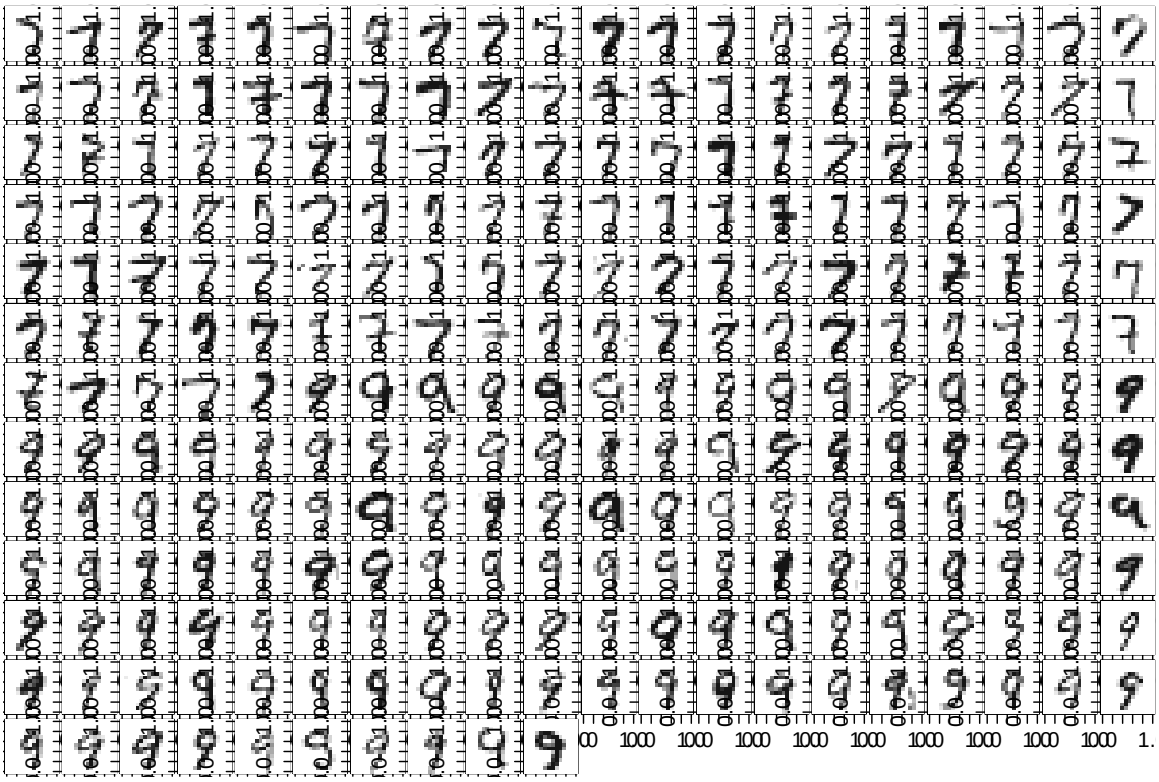

Figure 3.1: MNIST data used in the simulations. The digits 7 and 9 are relatively similar visually, making the simulations more realistic as it is harder for the classificaiton models to tell the difference.

3.3.2 Evaluation

The performance of a learner at any given point in time (at any iteration i of simulation number $k \in [1, 25]$) is encapsulated in its error ratio ϵ. That is, given the current labeled set, the **main classifier (Random Forest model)** is trained, and the entire dataset is predicted given the current classifier. The predictions are stored in an n-length vector p. We know the true labels ahead of time (thanks to the MNIST dataset), and they are stored in an n-length vector y'. The predictions are compared against the true labels and the error ratio of iteration i is given by

$$\epsilon_i = \frac{1}{250}\left(\sum_{z=1}^{250} \mathbf{1}_{p_z \neq y'_z}\right)$$

where $\mathbf{1}_{p_z \neq y'_z}$ is an indicator variable that is 1 when $p_z \neq y'_z$ and 0 otherwise. These 50 error ratios then form one vector ϵ_s, the result of trial s. To select the best active learning algorithm *on average*, each algorithm's error ratios are averaged over 25 total trials. This helps offset the randomness of the initialization and pooling scheme. Then the final error ratio of an algorithm is given by

$$\epsilon = \frac{1}{25}\left(\sum_{i=1}^{25} \epsilon_s\right)$$

The final framework for computing ϵ_s, a single trial's vector of error ratios, is summarized as follows (This procedure is encapsulated by the simulation engine code in Appendix A.9.2):

Algorithm 6 Computing ϵ_s, a single trial's vector of error ratios

procedure (X is a $n \times d$ matrix of d observations of all n variables, y is an n-length vector of labels for each variable in X (y_i=N/A when X_i, has no label). y' is an n-length vector of **true** labels for each variable in X. *iter* is the maximum number of queries allowed per trial)
    loop from $i = 1$ **to** *iter*:
        $idx \leftarrow$ active_learning_method$(X, y, ...)$
        QUERY $X_{idx,}$
        $tout \leftarrow \text{train}(X^{labeled}, y^{labeled}, \text{randomforest})$
        $p \leftarrow \text{predict}(tout, X)$
        $res_i \leftarrow \frac{\text{length(which}(p \neq y'))}{\text{length}(y')}$
    return res

3.3.3 Summary of methods

What follows is a summary of the active learning methods used in the simulation with details such tuning parameter values.

Table 3.1: A summary of the active learning methods tested in the simulation. **Note:* The classification model is the main classification model that is used to fit the error, not the classification model(s) used in the active learning methods (those are parameters). The classification model in the simulator is akin to the main classification model in the VS.

AL method	**Simulations**	**Classification model***	**Parameters**
Random sampling (CONTROL)	25 trials, 50 iterations (queries allowed) each	Random forest	*Classifier*: None
Uncertainty sampling 3.2.1	25 trials, 50 iterations (queries allowed) each	Random forest	*Classifier*: Random forest
Query by committee 3.2.2	25 trials, 50 iterations (queries allowed) each	Random forest	*Committee*: RF, NB, SVM, PLS, *Disagreement*: Vote entropy, *C_Pruning*: T, ϵ: 0.5
		Majority committee vote	*Committee*: RF, NB, SVM, PLS, *Disagreement*: Vote entropy, *C_Pruning*: T, ϵ: 0.5
		Random forest	*Committee*: RF, NB, SVM, PLS, *Disagreement*: Vote entropy, *C_Pruning*: F, ϵ: N/A
		Majority committee vote	*Committee*: RF, NB, SVM, PLS, *Disagreement*: Vote entropy, *C_Pruning*: F, ϵ: N/A
Query by bagging 3.2.3	25 trials, 50 iterations (queries allowed) each	Random forest	*Classifier*: Random forest, *Disagreement*: Vote entropy, *num_class*: 5, r: 0.75
Min-max clustering 3.2.4	25 trials, 50 iterations (queries allowed) each	Random forest	*Classifier*: None, *Distance*: Euclidean

A summary of each classification model used in the stating committee for QBC implementation is as follows:

- **Random Forest:** A random forest is a collection of decision trees which are grown from independent draws of the training set. A more detailed description can be found in Section 2.3.1.
- **Naive Bayes:**
- **SVM:**
- **Partial Least Squares:**

The call to each active learning function is controlled by the AL engine code in Appendix A.9.3. By implementing the active learning call in this way, the main AL functions are hidden from the end user so that they cannot call the functions directly, which may lead to bypassing checks and/or improperly calling functions.

3.3.4 Results

The full simulator which calls the aforementioned engines, runs the 25 simulations, and plots the results may be found in Appendix A.9.4. The results are summarized in the line plot below.

(Discuss how QBC actually spikes up when the pruning occurs, contrary to intuition - perhaps there is room for a future extension there. Note that the simulations are in no way completely comprehensive/exhaustive, and there is further work to be done ... For example, trying other classification models within the active learning algorithms - who knows, maybe that would've been better! Or trying different QBC committees, different tuning parameters i.e. pruning factor is 0.75 instead of 0.5, or you prune later/earlier)

Chapter 4

Graph comparison

Graph comparison may be reduced to a problem of measuring the difference between two graphs. A primary goal of the visualization system is to allow the user to better verify their numerical results with visual intuition. As such, the graph comparison measure is an important output of the VS that allows the user to quantify the difference between the methods, prompting the user to rethink their methods as needed. Furthermore, graph comparison may be directly used to compare visual and numeric correlation graphs, which we utilize to select stocks for a portfolio in Chapter 5.

4.1 Literature review

Graphic summarization

4.2 Overview of methods

4.2.1 Edge difference

4.2.2 Histogram of edge density

4.2.3 Shortest-path matrix

4.2.4 Graph distance kernel

4.3 Examples

Table summarization of method outputs when applied to various examples we've thought of previously

Chapter 5

Application and results

5.1 Application of the visualization system

Let X be a $n \times d$ data matrix where there are n observations of d variables. The application of the visualization system is as follows:

1. Create four different numerical correlation graphs $G_i^{\text{num}} = (V, E)$ for all $i \in \{1, ..., 4\}$, one for each correlation coefficient described in Section 1.2.

2. Run the VS on the same dataset. Stage 1 of the system utilizes the active learning algorithm selected in Chapter 3 to learn what is "correlated" and "not correlated." Stage 2 of the system then iterates through all possible unlabeled plot pairs and returns a graph $G = (V, E)$ where

---

For $i, j \in \{1, ..., d\}$, $E_{i,j}$ exists if the plot of j against i is "correlated"

   As a reminder, correlation is used loosely to refer to a visual interpretation of the mathematical term.

3. Utilize the graph comparison algorithm selected in Chapter 4 for each pair (G, G_i^{num}) for all $i \in \{1, ..., 4\}$. Select $G_i^{\text{num}} = (V, E)$ such that $GC(G, G_i^{\text{num}}) \leq GC(G, G_j^{\text{num}})$ for all $j \neq i$. The numerical correlation graph G_i^{num} is the graph which best matches the visual interpretation of the relationships among the dataset's variables.

5.2 Stock selection methodology

Given a correlation graph $G = (V, E)$, we wish to select k stocks (represented as vertices) such that each stock is as independent as possible of the other stocks. Two random variables X and Y are independent if and only if their joint cumulative distribution function may be written as $F_{X,Y}(x, y) = F_X(x)F_Y(y)$. With this formal definition, Santos *et al.* notes that "we say two random variables X and Y are dependent if they are not independent" so the problem then becomes one of "how to measure and detect dependence from the observation of the two random variables"[16]. With the correlation graph G, we now have a measure of dependence. However, it was made clear that, in general, it is incorrect to say that "uncorrelated" equates "independence". Regardless, non-correlation is still a useful tool to inform stock selection for portfolio management (Section 1.3). So, we select k stocks such that each stock is as *uncorrelated* as possible with the other stocks.

As V_i and V_j have no edge when $\text{Corr}(i, j)$ is below some threshold p, we might consider the following stock selection strategy:

Algorithm 7 Naive stock selection strategy

procedure (k is the number of stocks to select and A is the adjacency matrix of $G = (V, E)$)
    $d \leftarrow \text{len}(A)$
    $z \leftarrow \text{Perm}(d, k)$*
    $min = \infty$
    $index = 0$
    loop from $i = 1$ **to** $\binom{d}{k}$:
        $c = 0$
        loop from $j = 1$ **to** k:
            loop from $l = 1$ **to** k:
                $c \leftarrow c + A_{z_{i,j}, z_{i,l}}$
        If $c < min$ **then**
            $min = c$
            $index = i$
    return $z_{index,}$

*Perm(d, k) is a function that returns all possible permutations of k values selected from d i.e. $z_{1,} =< 1, 5, 7 >$ is a permutation for $d = 10, k = 3$.

This becomes computationally difficult as d increases and does not account for average stock returns; as discussed in Section 1.3, stocks with positive drift and high volatility are shown to improve the performance of the portfolio over time in the "buy and hold" model, or improve gains in the case of rebalancing. These two criteria may be used to minimize the space of stocks to select from. Drift over time is simplified as positive return averaged over time. We let p_d and p_v denote the threshold values for drift and volatility respectively. These values will be dependent on the size of the dataset as we wish to preselect p_d and p_v such that $k < d < 2k$ to reduce the computational burden. The selection strategy then becomes

Algorithm 8 Adjusted stock selection strategy

procedure (k is the number of stocks to select and A is the adjacency matrix of $G = (V, E)$. Let D be the vector of sample average returns and V be the vector of sample standard deviation.)
    function DRIFT
        $d \leftarrow \text{len}(A)$
        $rmv = [\,]$
        loop from $i = 1$ to d:
            If $D_i \leq p_d$ **then** rmv.append(i)
        return rmv
    loop from $i = 1$ to len(rmv):
        Delete row rmv_i and column rmv_i from A
    function VOLATILITY
        $d \leftarrow \text{len}(A)$
        $rmv = [\,]$
        loop from $i = 1$ to d:
            If $V_i \leq p_v$ **then** rmv.append(i)
        return rmv
    loop from $i = 1$ to len(rmv):
        Delete row rmv_i and column rmv_i from A
    function SELECTION
        $d \leftarrow \text{len}(A)$
        $z \leftarrow \text{Perm}(d, k)$*
        $min = \infty$
        $index = 0$
        loop from $i = 1$ **to** $\binom{d}{k}$:
            $c = 0$
            loop from $j = 1$ **to** k:
                loop from $l = 1$ **to** k:
                    $c \leftarrow c + A_{z_{i,j}, z_{i,l}}$
            If $c < min$ **then**
                $min = c$
                $index = i$
        return $z_{index,}$

*Perm(d, k) is a function that returns all possible permutations of k values selected from d i.e. $z_{1,} =< 1, 5, 7 >$ is a permutation for $d = 10, k = 3$.

5.3 Healthcare stock data

We select 43 stocks from the healthcare industry to apply the system to. There are 4113 observations of the stocks from 1998 to 2014. We have cleaned the data to get

rid of the dependency on time so that each observation of a stock may better simulate an independent draw from some distribution. This is done by fitting a regression to the values and using the residuals as the new dataset.

For $i \in \{1, ..., 4\}$, we employ a "buy and hold" tactic to compare the performance of each portfolio P_i selected utilizing the methodology described in Section 5.2), which correspond to correlation graphs $G_i^{\text{num}} = (V, E)$. While a rebalancing method performs better in practice [11], the "buy and hold" is better-suited for observing the portfolio performances over time, providing a more concrete basis of comparison for the selected correlation coefficient.

5.4 Results

Here, we have a comparative time plots of the "buy and hold" performance (yearly returns) of the S&P 500, stocks selected from the final numerical correlation graph $G_i^{\text{num}} = (V, E)$, and other correlation graphs $G_j^{\text{num}} = (V, E)$ where $j \neq i$. We hope to find that correlation graph i, which was selected from the usage of the VS, is a top performer.

We may also analyze the VS graph output $G = (V, E)$ against the final selection G_i^{num} by asking the following questions: What new links were formed? Which links were deleted? Did we actually find the plots of those links interesting/not interesting? If it is possible (if the graphs are not too dense), then I would also include a visualization of G and G_i^{num}.

Chapter 6

Conclusion

The proliferation of high-dimensional datasets has led analysts to rely on unverifiable numerical estimators. We are interested in visualizing high-dimensional correlation graphs as a way to verify numerical tests of dependence. This is important in asset management as it is desirable to select a portfolio of stocks that are as independent as possible. High-dimensional visualization is problematic because (1) there are too many potential plots to sort through manually (To be specific, there are $\binom{d}{2}$ plots where d is the number of variables e.g. stocks that we are interested in), which also means that (2) it is tedious to verify numerical results with visual results and vice versa (Section 1.1). In this work, we presented a visualization tool that actively learns user preferences, applies the fitted classifier to unlabeled data, provides visualization tools for the active learning output (a visual graph $G = (V, E)$), and outputs the difference among the numerical graph $G^{\text{num}} = (V, E)$ and G (Chapter 2).

As a specific response to the two aforementioned problems, we focused on the active learning and graph comparison components of the visualizer systems. We discussed two main approaches to active learning and provided algorithms for different active learning methods (Chapter 3). A simulation study indicated that $--$ would be the best learning algorithm for the financial application of the VS system.

Similarly, we discussed various measures of graph distance and performed a simulation study to determine which methodology to implement in the VS system for the financial application (Chapter 4).

We then ran the VS system with $----$ in stage 1 and used $----$ to compute the graph comparison of the output G with G_i^{num} for all $i \in \{1, ..., 4\}$, the correlation graphs for Pearson, Spearman's, Kendall's, and distance correlation coefficient respectively. $----$ most closely matched the visual graph. We then utilized the stock selection methodology described in Section 5.2 to select a portfolio of $k = 10$ stocks P_i for all $i \in \{1, ..., 4\}$ based on G_i^{num} based on data from year 1. Yearly returns were computed with the remainder of the observations in a simulation of the "buy and hold" strategy. As expected, the portfolio corresponding the correlation graph selected with the aid of the VS was one of the top performers (??).

The visualization tool presented in this work is an important step in streamlining the future of clean analysis. It provides a systematic way for confirming and/or suggesting dependencies among variables that match our visual concept of a dependence and produces an explicit decision tree that allow others to understand and replicate the data analysis process. This alleviates the problems associated with high-dimensional datasets and allows the user to quickly see ways in which the numerical model may have fallen short of the "true" relationship between variables. Nevertheless, there are several places to develop further work in order to refine the system and improve our concept of "clean analysis."

6.1 Further extensions

6.1.1 Estimator selection

Estimator selection involves actively fitting the best model as opposed to "checking" a numerical model thats been given. For instance, rather than requiring a numerical

model in order to check for graph distance (Chapter 4), the VS would select a numerical model without requiring the analyst to fit one beforehand. This problem is more difficult to define, and the value that the visualization system adds is not as concrete.

6.1.2 Outlier removal

Outliers are unavoidable in raw data and can skew results quite a bit. This can be seen in the analysis of Dataset 2 in Table 1.1 and Figure 1.1.

((When can the system remove outliers? What criteria should it use?))

6.1.3 Graphical models and improved stock selection

A single correlation (discussed in Section 1.2) can be thought of as a regression of the response variable against only one observed variable; it is a "local" property because it compares the behavior of only two random variables. On the other hand, a single link in a graphical model can be thought of as a regression of the response variable against all variables in the space. The general idea is that a graphical model has a "global" property because it takes all of the other variables into account. Although it may seem simple to numerically quantify the dependencies with correlation as conditional dependencies are less intuitive to compute, correlations tend to fall short of the desired result due to the property of transitivity.

Transitivity states that if X is correlated to Y, and Y to Z, then X is also correlated to Z. Although correlation is not always transitive, situations where the correlation is close to 1 or 0, then the transitivity of correlation can be recovered and observed in the relevant data [19]. Let us assume that the universe consists of Apple, Google, and Silicone (a manufacturer providing chips to both Apple and Google) stock. Suppose an analyst wants to model Google stock. Apple stock moves with Silicone stock as they depend on them for their chips. Similarly, Google stock (unbeknownst to the analyst) also moves with Silicone stock but not with Apple stock.

The correlation between observed prices of Google and Apple stock will clearly and erroneously be positive without considering the way the stocks are connected to the other observed variable (Silicone stock). Given that correlation tends to be transitive, a correlation graph can have too many edges. This goes against the concept of "sparsity" (Section 1.3) by cluttering the resulting space of observation variables that explain the response variable for the user, leaving the analyst with an uninformative and unaccountable numerical solution.

Graphical models alleviate some of the problems associated with correlation graphs, but they have their own set of problems, as well. Let $G = (V, E)$ be an undirected graph with vertices $V_1, ..., V_d$ (a d-dimensional distribution) and edges $E_{i,j} \in \{0, 1\}$. We set $E_{i,j} = 1$ when there is an edge between V_i and V_j, and 0 otherwise. Do not draw an edge between V_i and V_j iff $V_i \perp V_j$ given V_k where $k \in \{1, ..., d\} \setminus \{i, j\}$. In other words, do not draw an edge if

$$\mathrm{P}(V_i, V_j | V_k) = \mathrm{P}(V_i | V_k)\mathrm{P}(V_j | V_k)$$

This result is known as a graphical model. The drawback is the difficulty in empirically computing conditional distributions and the problems associated with fitting distributions to real data. While there are simplifications that can be made for plotting, the solution is not always so clear. However, the conditional independence of graphical models is more of a global property than correlation is because, for every pair of variables, it conditions on all the remaining variables. Returning to the universe of Google, Apple, and Silicone stocks, conditioning Google on Silicone and Apple on Silicone makes the relationship between the two clearly uncorrelated. Thus, although conditional independence tends to be more difficult to determine, it will tend to give a sparser network that is more interpretable for the analyst.

Correlation graphs and graphical models each have their own niche to fill, though a graphical model rebalancing has been shown to strongly outperform traditional portfolio management methods [11]. It is important, therefore, for the VS to support both models. The system would need to be modified to allow for plotting V_i against V_j conditional on V_k for all $k \in \{1, ..., d\} \backslash \{i, j\}$. One method to consider would be to control for the behavior of all the other explanatory variables while making this scatter plot, similar to regression.

6.1.4 Ordering of queried plots

As people interact with graphs, they maintain a "mental map" of the graph; when users label a new graph, they remember the previous plots that they have labeled [7]. The importance of the mental map depends on various factors such as the user preferences and tasks that they must complete [7]. Suppose that several active learning queries (scatter plots) are selected at once. Given a gradation of graphs (showing graphs that are most alike one after the other), users are less able to distinguish between differences than if they are shown graphs from different ends of the spectrum at different times [7]. To put it concisely, the scatter plot display itself (discussed in Section 2.1.1) is not the only thing that matters. The ordering of the display matters, as well, and it is best to show the plots in an order that allows users to distinguish the differences among graphs that they have already seen. By improving their understanding of the plots, careful display ordering advances the accuracy of user responses. User responses can be thought of as observations of the users true preferences, and the ordering of plots as a way to fine-tune the precision of the classifier.

6.1.5 Line-up tests

One of the pitfalls of data visualization is "apophenia," a phenomenon where the user sees patterns in random noise. Part of the reason for this is due to the vague-

ness of defining "independence" on a non-uniform domain and range (Section 2.1.1). Wickham *et al.* propose a line-up protocol that is similar to the Rorscach test where subjects are asked to interpret abstract blots of ink [21]. In the line-up test, users are asked to identify the real data from a set of n plots where $n1$ plots are synthetically generated (Figure6.1). Identification of the raw data against the synthetic data is then an indicator that the current fitted model is a poor fit of the user's preferences. In the context of the classification problem, the learner may generatae four "uninteresting" plots (following the proposed decision tree) with one "interesting" plot and asks the user if he/she can identify the interesting plot. If the user is able to consistently identify the interesting plot, it is an indication that the current decision tree is a close fit of the users preferences.

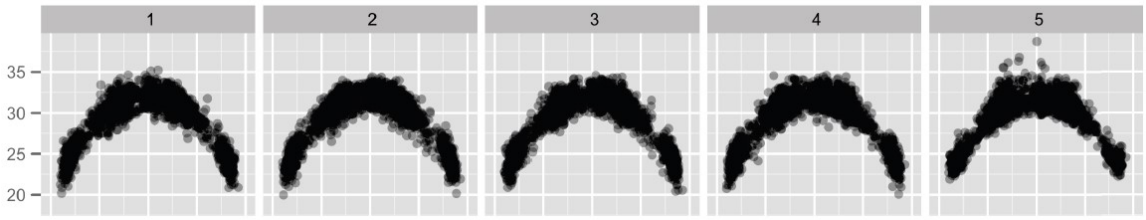

Figure 6.1: A line-up test for $n = 5$. Consistently identifying the raw data against the synthetic data indicates that the fitted model may not be good enough. Figure from Wickham *et al.* [21]

6.1.6 Edge-weighted graphs

Edge-weighted graphs also stray from the black and white classification of correlation graphs (Section 1.2) and graphical models (Section 6.1.3) which have been discussed earlier. In an edge-weighted graph, edges can be weighted depending on the type of conditional dependence or correlation. Negative or positive values are assigned 1 or -1, respectively. A value of 0 (no edge) still implies conditional independence or uncorrelated variables. This problem is more difficult as the number of classification

labels changes from two to three, and the graph has become more information but complex. Active learning methods may need to be more generalized as in the case of uncertainty sampling (Section 3.2.1).

6.1.7 Rejection classification

So far, classification has been discussed in terms of black and white, interesting versus non-interesting. While interacting with the data visually, the user's concept of what's interesting in the specific dataset may evolve over time; at the beginning, they have no idea what the data looks like and where to set the bar for their own standards of dependence. There are several ways to take this into consideration.

- The visualization system could assign a weight to the analyst's responses by trial number where the last few plots are more valuable than the first few. However, this may destabilize cases where the user's preferences dont end up changing.
- The system can include an alternative option that allows the user to refuse to label a plot when it's too close to their decision boundary. This is welcome for the user who is not forced into making a decision he/she is uncertain about, but it is problematic for the learner as it causes the hypothesis space to remain unchanged rather than shrink. The point of the active learning in stage 1 (Section 2.2, Chapter 3) is to have the oracle provide labels in order to better understand user preferences. By allowing for this option, the active learner may run for too long and/or return a poorly-defined tree.
- The system can contain a third option that permits the user to "recycle" the plot. This allows to user to return to the plot later after learning more about what the data looks like and understanding his/her own preferences better, and it ensures that the active learner will eventually receive data on the ambiguous plots that it has queried. The main concern is that this could potentially de-balance an ordering procedure (Section 6.1.4), but the system can strategically insert the recycled plot between two plots it differs from with the constraint that the insertion location is after the current plot.

Appendix A

Implementation

A.1 Code for figure 1.1, left

```
#generate a reproducible dataset and scale to [0,1]
set.seed(10)
x <- seq(0, 1, length.out = 100)
y <- rnorm(100)
y <- (y-min(y))/(max(y)-min(y))

#sort the noise
y <- sort(y)
y <- y[c(seq(1,99,length.out=50), seq(100,2,length.out=50))]

#local swapping
for(i in 4:96){
        y[(i-3):(i+3)] <- y[sample((i-3):(i+3))]
}

idx <- sample(1:100)
x <- x[idx]; y <- y[idx]

######################
#numerical feedback

## fit linear regression
fitlm <- lm(y ~ x)
anova(fitlm)

## see if any coefficients are significant
summary(fitlm)

## see if residuals are normally-distributed
shapiro.test(fitlm$residuals)
```

```
## correlation is not significantly different from zero
cor.test(x, y)

####################
#visual feedback
plot(x, y, pch = 16, cex = 2)
```

A.2 Code for figure 1.1, right

```
## generate a reproducible dataset
set.seed(10)
n <- 50
x <- sort(rnorm(n))
sd.vec <- c(seq(1, 1.5, length.out = 50), seq(1.5, 1, length.out =
   50))
y <- -x + 0.5*rnorm(n, sd = sd.vec)
y <- scale(y)

y[c(1,5,10)] <- min(y)
y[c(n-10, n-5, n)] <- max(y)

######################
#numerical feedback

## fit linear regression
fitlm <- lm(y ~ x)
anova(fitlm)

## see if any coefficients are significant
summary(fitlm)

## see if residuals are normally-distributed
shapiro.test(fitlm$residuals)

## correlation is not significantly different from zero
cor.test(x, y)

####################
#visual feedback
plot(x,y, pch = 16, cex = 2)
```

A.3 Code for figure 2.2

```
## generate the dataset
set.seed(10)
n <- 500
x <- rnorm(n)
y <- rnorm(n)

par(mfrow=c(1,2))
```

```
plot(x,y, pch = 16, cex = 2)
## apply the cdf
plot(pnorm(x),pnorm(y),pch = 16, cex = 2)
```

A.4 Uncertainty sampling (Section 3.2.1) implementation

Refer to Algorithm 1.

```
#' Uncertainty Sampling with bivariate labels
#'
#' @param X the full data matrix, n x d, including all unlabeled
    data
#' @param y a factor vector with 2 levels and NAs for unlabeled data
#' @param unlabel_index_c is a vector of n pre-selected (pooled)
    indices
#' method may choose from
#' @param classifier the classifier name
#' @param ... additional parameters for the active learning method
#'
#' @return a vector of indices to query
#' @export

uncertainty_sample <- function(X, y, unlabel_index_c, classifier,
    ...){
        if (length(classifier) > 1 || missing(classifier) || is.null
            (classifier) ||
        is.na(classifier)) {
                stop("A␣single␣classifier␣is␣required␣for␣
                    uncertainty␣sampling")
        }

        # Check that the classifier is compatible with uncertainty
            sampling
        c <- try(caret::modelLookup(classifier))
        if (!any(c$probModel)) {
                stop(classifier,"␣must␣return␣posterior␣
                    probabilities")
        }

        # Split x and y to retrieve labeled and unlabeled pairs
        unlabel_index <- which(is.na(y))
        x_lab <- X[-unlabel_index,]
        y_lab <- y[-unlabel_index]
        x_ulab <- X[unlabel_index_c,]

        tout <- caret::train(x_lab,y_lab,classifier)
        p <- as.matrix(stats::predict(tout, newdata=x_ulab, type="
            prob"))

        # Return corresponding X index of posterior closest to 0.5
```

```
        p <- apply(p, 1, function(x) abs(x[1]-0.5))
        unlabel_index_c[which(p == min(p))]
}
```

A.5 Query by committee implementation

Refer to Algorithm 3. This implementation contains the functions for query selection and pruning. These functions are called by the main simulation engine for the QBC method. The main simulation engine acts as the entire algorithm in Section 3.2.2. The simulation engine code may be found in Appendix A.9.

A.5.1 Query selection

```
#' Query by Committee
#'
#' @param X the full data matrix, n x d, including all unlabeled
   data
#' @param y a factor vector with 2 levels and NAs for unlabeled data
#' @param unlabel_index_c is a vector of n pre-selected (pooled)
   indices
#' @param committee the list of committee classifiers
#' @param dis is the disagreement measure between committee
   classifications
#' @param ... additional parameters for the active learning method
#'
#' @return a vector of indices to query, the committee predictions
   of this round
#' @export

qbc_sample <- function(X,y,unlabel_index_c,committee,dis = "vote_
   entropy",...){
        if (missing(committee) || is.null(committee)) {
                stop("A␣committee␣is␣required␣for␣QBC")
        }

        unlabel_index <- which(is.na(y))
        x_lab <- X[-unlabel_index,]
        y_lab <- y[-unlabel_index]
        x_ulab <- X[unlabel_index_c,]

        p <- vector("list",length(committee))

        for (i in 1:length(committee)) {
                tout <- caret::train(x_lab,y_lab,committee[i])
                p[[i]] <- predict(tout, newdata=x_ulab)
```

```
        }

        # Compute disagreement (from activelearning package)
        d <- switch(dis,
                vote_entropy=vote_entropy(p),
                post_entropy=post_entropy(p),
                kullback=kullback(p)
                )

        index <- unlabel_index_c[which(d == max(d))]
        if (length(index) > 1) index <- sample(index,1)

        # Gather each committee's prediction
        pre <- rep(0,length(committee))
        for (i in 1:length(committee)) {
                # Predict function returns a factor
                pre[i] <-as.numeric(as.character(p[[i]][which(
                    unlabel_index_c==index)]))
        }

        list(index, pre)
}
```

A.5.2 Committee pruning

```
#' Query by Committee (Pruning function)
#'
#' @param X the full data matrix, n x d, including all unlabeled
    data
#' @param y a factor vector with 2 levels and NAs for unlabeled data
#' @param index is the classification of X[index,] which was queried
#' @param committee_pred is the list of committee predictions for
    index
#' @param k is the current iteration number that the AL_engine is on
#' @param pt is the pruning threshold (any error value above it is
    pruned)
#' @param err in (0 best,1 worst) is the committee's error-to-
    iteration ratio
#' @param is_prune is TRUE when pruning is desired, FALSE when not
#' @param ... additional parameters for the active learning method
#'
#' @return a list with updated error, indices to delete from the
    committee
#' @export
qbc_prune <- function(X, y, index, committee_pred, k, pt = 0.5, err,
    is_prune,
...){
        if (missing(err) || is.null(err) || is.na(err)) {
                stop("Committee␣error␣ratio␣is␣required␣for␣QBC␣
                    pruning")
        }
        prune <- vector() # Do not know how long prune will be until
            the end
```

```
        # Do not prune if committee size is 1 or it's the first
            round
        if (length(committee_pred) == 1 | k == 1) {
                list(err, prune)
        } else {
                # Update error value
                for (i in 1:length(committee_pred)) {
                        if (committee_pred[i] == y[index]) iv <- 0
                            else iv <- 1
                        err[i] <- err[i] + (iv - err[i])/k
                        if (err[i] > pt & is_prune) {
                                prune <- c(prune,i)
                        }
                }
                list(err, prune)
        }
}
```

A.6 Vote entropy implementation

Refer to Section 3.2.2.

```
#' Disagreement method (From activelearning package)
#' @importFrom itertools2 izip
#' @importFrom entropy entropy

vote_entropy <- function(x, type='class', entropy_method='ML'){
        it <- do.call(itertools2::izip, x)
        disagreement <- sapply(it, function(obs) {
                entropy::entropy(table(unlist(obs)), method=entropy_
                    method)
        })
        disagreement
}
```

A.7 Query by bagging implementation

Refer to Algorithm 4.

```
#' Query by Bagging
#'
#' @param X the full data matrix, n x d, including all unlabeled
    data
#' @param y a factor vector with 2 levels and NAs for unlabeled data
#' @param unlabel_index_c is a vector of n pre-selected (pooled)
    indices
#' @param classifier the name of a classification model
#' @param dis is the disagreement measure between committee
    classifications
#' @param num_class is the number of desired committee members
```

```
#' @param r in (0,1). r*(labeled set) = training set for each num_
   class round
#' @param ... additional parameters for the active learning method
#'
#' @return an index to query
#' @export

qbb_sample <- function(X, y, unlabel_index_c, classifier, dis = "
   vote_entropy",
num_class, r, ...){
        if(r<=0 || r>=1) stop("r must be in (0,1)")

        x_ulab <- X[unlabel_index_c,]

        # Randomly sample from the labeled set to create a
           classifier
        label_index <- which(!is.na(y))
        committee <- vector("list",num_class)
        for (i in 1:num_class) {
                idx <- sample(label_index,round(length(label_index)*
                   r,0))
                committee[[i]] <- caret::train(X[idx,],y[idx],
                   classifier)
        }

        # Utilize the resulting classifiers as a committee
        p <- vector("list",length(committee))
        for (i in 1:length(committee)) {
                p[[i]] <- stats::predict(committee[[i]], x_ulab)
        }

        # Compute disagreement (utilizing the functions from the
           activelearning
        package)
        d <- switch(dis,
                vote_entropy=vote_entropy(p),
                post_entropy=post_entropy(p),
                kullback=kullback(p)
                )

        index <- unlabel_index_c[which(d == max(d))]
        if (length(index) > 1) index <- sample(index,1)
        index
}
```

A.8 Min-max clustering implementation

Refer to Algorithm 5.

```
#' Query by Min-Max Clustering
#'
```

```
#' @param X the full data matrix, n x d, including all unlabeled
    data
#' @param y a factor vector with 2 levels and NAs for unlabeled data
#' @param unlabel_index_c is a vector of n pre-selected (pooled)
    indices
#' @param dis is the distance measure between data
#' @param ... additional parameters for the active learning method
#'
#' @return a vector of indices to query
#' @export

cluster_sample <- function(X, y, unlabel_index_c, dis = "euclidean",
    ...){
        label_index <- which(!is.na(y))
        x_lab <- X[label_index,]
        y_lab <- y[label_index]
        x_ulab <- X[unlabel_index_c,]
        y_ulab <- y[unlabel_index_c]

        # Select the point furthest from points that are already
            labeled
        q <- rep(0,length(y_ulab))
        for (i in 1:length(y_ulab)) {
                min <- Inf
                for (j in 1:length(y_lab)) {
                        temp <- cs_distance(X[unlabel_index_c[i],],X
                            [label_index[j],],dis)
                        if (min > temp) min <- temp
                }
                q[i] <- min
        }
        unlabel_index_c[which(q==max(q))]
}

# General Support Function for Distance Computation
cs_distance <- function(a,b,dis = "euclidean"){
        d <- switch(dis,
                euclidean=cs_euclidean_distance(a,b)
                )
}

# Euclidean Distance Computation
cs_euclidean_distance <- function(a,b) {
        sqrt( sum( mapply( function(x,y) (x-y)^2, a, b)))
}
```

A.9 Simulation (Section 3.3) implementation

A.9.1 MNIST data

A.9.2 Simulation engine

A.9.3 AL algorithm engine

A.9.4 Simulator (Main)

Bibliography

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

[2] A. Buja, A. Krieger, and E. George. A Visualization Tool for Mining Large Correlation Tables: The Association Navigator. `http://stat.wharton.upenn.edu/~buja/PAPERS/Buja-et-al-Association-Navigator.pdf`, 2016.

[3] A. Cutler. Random forests for regression and classification. `http://www.math.usu.edu/adele/RandomForests/Ovronnaz.pdf`, 2010.

[4] I. Dagan and S. Engelson. Committee-based sampling for training probabilistic classifiers. *Proceedings of the International Conference on Machine Learning*, pages 150–157, 1995.

[5] Dasgupta, S. Two faces of active learning. `http://cseweb.ucsd.edu/~dasgupta/papers/twoface.pdf`, 2011.

[6] B. Efron. Why Isn't' Everyone a Bayesian? *The American Statistician*, pages 1–5, February 1986.

[7] P. Federico and W. Oldford. Evaluation of two interaction techniques for visualization of dynamic graphs. *arXiv preprint arXiv:1608.08936*, pages 1–15, August 2016.

[8] M. Hofert and W. Oldford. Visualizing Dependence in High-Dimensional Data: An Application to S&P 500 Constituent Data. *arXiv preprint arXiv:1609.09429*, pages 1–33, September 2016.

[9] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[10] R. J. Little. In Praise of Simplicity not Mathematistry! Ten Simple Powerful Ideas for the Statistical Scientist. *Journal of the American Statistical Association*, pages 359–369, July 2013.

[11] H. Liu, J. Mulvey, and T. Zhao. A semiparametric graphical modelling approach for large-scale equity selection. *Quantitative Finance*, pages 1053–1067, 2016.

[12] S. Liu, D. Maljovec, B. Wang, P. T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics*, pages 1249–1268, December 2016.

[13] A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. *Proceedings of the International Conference on Machine Learning*, pages 359–367, 1998.

[14] B. O'Connor. load the MNIST data set in R. `https://gist.github.com/brendano/39760`, 2008. Github repository.

[15] J. Ramey. Active learning in r. `https://github.com/ramhiser/activelearning`, 2015. Github repository.

[16] S. Santos, D. Y. Takahashi, A. Nakata, and A. Fujita. A comparative study of statistical methods used to identify dependencies between gene expression signals. *Briefings in Bioinformatics*, pages 1–13, August 2013.

[17] Settles, B. Active Learning Literature Survey. `http://burrsettles.com/pub/settles.activelearning.pdf`, 2010.

[18] G. Szekely, M. Rizzo, and N. Bakirov. Measuring and testing independence by correlation of distances. *The Annals of Statistics*, pages 2769–2794, March 2007.

[19] Tao, T. When is correlation transitive? `https://terrytao.wordpress.com/2014/06/05/when-is-correlation-transitive/`, 2014.

[20] V. Vu, N. Labroche, and B. Bouchon-Meunier. Active Learning for Semi-Supervised K-Means Clustering. *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, pages 12–15, October 2010.

[21] H. Wickham, D. Cook, H. Hofmann, and A. Buja. Graphical inference for infovis. *IEEE Transactions on Visualization and Computer Graphics*, pages 973–979, December 2010.

[22] Y. Zhou and G. Hooker. Interpreting Models via Single Tree Approximation. *arXiv prepint arXiv:1610.09036v1*, pages 1–15, October 2016.