UNIVERSITY OF CALIFORNIA, SAN DIEGO

Anthropocentric Data Analysis

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Cognitive Science

by

Joshua M. Lewis

Committee in charge:

Professor Virginia R. de Sa, Chair
Professor Serge J. Belongie
Professor Gedeon O. Deák
Professor Jeffrey L. Elman
Professor Lawrence K. Saul

2011

The dissertation of Joshua M. Lewis is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2011

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank Virginia de Sa and Gedeon Deák for advising me and guiding me through my graduate career from start to finish. I would also like to thank the other members of my committee, Serge Belongie, Jeff Elman and Lawrence Saul for being great teachers and giving valuable feedback. Finally, I would like to thank my undergraduate advisors Jay Atlas and James Marshall for introducing me to cognitive science research.

Chapter 2 is a reprint of material as it appears in J. M. Lewis, "Finding a better *k*: A psychophysical investigation of clustering," *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pp. 315-320, 2009.

Chapter 3 is coauthored with Margareta Ackerman and Virginia R. de Sa, Chapter 4 is coauthored with Laurens van der Maaten and Virginia R. de Sa, and Chapter 5 is coauthored with Virginia R. de Sa. The dissertation author is the primary author on these three chapters. Chapter 6 is coauthored with Eric Weiss, Cindy Zhang and Virginia de Sa. Eric Weiss is the primary author on this chapter, and the dissertation author is responsible for the CPU experiments and much of the writing. Chapters 3 - 6 are being prepared for submission.

During my graduate career the NSF supported my studies through NSF IGERT Grant #DGE-0333451 to GW Cottrell/VR de Sa and NSF Grant #SES-0963071 to VR de Sa.

VITA

2006	B. A. in Cognitive Science and B. A. in Philosophy *cum laude*, Pomona College
2006-2009	Graduate Teaching Assistant, University of California, San Diego
2008	M. S. in Cognitive Science, University of California, San Diego
2011	Ph. D. in Cognitive Science, University of California, San Diego

PUBLICATIONS

Joshua M. Lewis, Gedeon O. Deák, Hector Jasso, Jochen Triesch, "Building a Model of Infant Social Interaction," *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, pp. 278-283, 2010.

Joshua M. Lewis, Adam S. Fouse, Virginia R. de Sa, "Cross-Modal Influence on Binocular Rivalry," *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, pp. 718-723, 2010.

Virginia R. de Sa, Patrick M. Gallagher, Joshua M. Lewis, Vicente L. Malave, "Multi-View Kernel Construction," *Machine Learning*, 47-71, 2009.

Joshua M. Lewis, "Finding a Better *k*: A psychophysical investigation of clustering," *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pp. 315-320, 2009.

Joshua M. Lewis, Pincelli M. Hull, Kilian Q. Weinberger, Lawrence K. Saul, "Mapping Uncharted Waters: Exploratory Analysis, Visualization, and Clustering of Oceanographic Data," *Proceedings of the Seventh International Conference on Machine Learning and Applications*, pp. 388-395, 2008.

Douglas S. Blank, Joshua M. Lewis and James B. Marshall, "The multiple roles of anticipation in developmental robotics," *Proceedings of the 2005 AAAI Fall Symposium on Anticipatory Cognitive Embodied Systems*, pp. 8-14, 2005.

ABSTRACT OF THE DISSERTATION

Anthropocentric Data Analysis

by

Joshua M. Lewis

Doctor of Philosophy in Cognitive Science

University of California, San Diego, 2011

Professor Virginia R. de Sa, Chair

Machine learning techniques benefit science and industry primarily insofar as they enable data analysts to better understand their data, make valid conclusions, and gain insight into their domain of investigation. Studies in anthropocentric data analysis seek to understand how human judgment is applied in the data analysis process and to develop methods that provide explicit opportunities for human interaction and insight. We present three studies of human visual reasoning exploring the extent to which novice and expert subjects are able to judge the quality of and understand cluster analysis and dimensionality reduction stimuli. We then investigate whether humans are able to learn how cluster analysis algorithms function simply through interacting with them rapidly across diverse data sets. Finally we show how multicore processing on CPUs and GPUs can move human/algorithm interactions closer to real-time.

1 Introduction

Machine learning techniques benefit science and industry primarily insofar as they enable data analysts to better understand their data, make valid conclusions, and gain insight into their domain of investigation. Studies in anthropocentric data analysis seek to understand how human judgment is applied in the data analysis process and to develop methods that provide explicit opportunities for human interaction and insight. This thesis is composed of five papers that advance the anthropocentric data analysis agenda.

Chapters 2 - 4 are psychophysical studies of how humans perceive and reason about data analysis stimuli. If humans are able to make good visual judgments, such as whether a 2D reduction of a data set is preserving important structure, or whether a scatter plot contains five or seven clusters, then researchers should be able to leverage those judgments during the data analysis process. If novices with no formal training in data analysis and no knowledge of machine learning techniques can nevertheless accurately gauge the quality of a clustering, then human computation resources such as Mechanical Turk could be an aid to large scale data analysis. The papers in Chapters 2 - 4 investigate whether theses judgment capacities are present, and in which populations (experts vs. novices). In addition, they compare human judgments with automated quality measures and other algorithms to determine how much overlap there is between the two.

Just as a baseball player does not need an explicit model of physics in his or her head to field a ball on a short hop, researchers may not need an explicit mathematical understanding of how algorithms function in order to come to valid conclusions about their use. In Chapter 5, we explore whether giving naïve undergraduates informative, self directed and real time experience with two clustering algorithms allows them

to understand how the algorithms function and in which contexts they might be most profitably applied.

Finally, in Chapter 6 we describe research in parallel computation on graphics processors and traditional processors that will support real time interaction between researchers and algorithms for much larger data set sizes. These performance improvements support the active learning-style interaction described in Chapter 5.

2 Finding a Better k: A psychophysical investigation of clustering

2.1 Abstract

Finding the number of groups in a data set, k, is an important problem in the field of unsupervised machine learning with applications across many scientific domains. The problem is difficult however, because it is ambiguous and hierarchical, and current techniques for finding k often produce unsatisfying results. Humans are notoriously adept at navigating ambiguous and hierarchical situations, and this paper measures human performance on the problem of finding k across a wide variety of data sets. Two leading algorithms are compared to the human results and their relationship to observed human strategies is discussed. This paper also presents two new density based strategies to simplify the k-choosing problem, Density Regimes and Density Erosion, along with a means of integrating them into an existing algorithm.

2.2 Introduction

Within the field of unsupervised machine learning, clustering is a technique used to separate an arbitrary collection of data points into groups (commonly called clusters). Clustering is usually comprised of two steps. First, one must choose the number of groups, represented by the variable k, for which to look. Second, one must assign each

data point to one or more groups while ensuring that there are no empty groups. This second step has received the majority of attention from researchers, with techniques such as the venerable k-means and spectral clustering [2] focusing exclusively on assignment and leaving the task of choosing k up to other algorithms.

There are a few reasons why choosing k is a less attractive problem for researchers as compared to the assignment problem. In some application domains for clustering algorithms, researchers may approach their data with a particular value for k already in mind. In this case, they can simply enter the desired number of clusters into an algorithm like k-means and have a solution without bothering to find which values for k have statistical support. Beyond this practical matter, choosing k has all the hallmarks of a difficult computer science problem. For most data there is no one right answer for what k should be. In fact there may be many answers, some more likely than others. Thus k is an inherently ambiguous quantity, causing much algorithmic difficulty. Some of this ambiguity comes from multiple possible hierarchical interpretations of the data. For the data in Fig. 2.1, for example, the value of k depends on whether one wants to focus on the details (that there are seven or eight small groups) versus the broad trend (that there are two clear larger groups).

Beyond challenges with ambiguity and hierarchy, there is also the issue of profligacy. Naïvely, one might want to represent the effectiveness of a certain k by calculating an assignment based on that k and then measuring the sum squared distance between each data point and the centroid of the cluster that it is assigned to. This seems like a reasonable strategy, but imagine choosing a k equal to the number of points in one's data. In this situation, each cluster will have exactly one data point, which will be located at the cluster centroid. Obviously the sum squared distance in this case will be zero, indicating a good fit but providing an answer completely useless in terms of data analysis. In general, $k+1$ will always fit data better than k based on this simple measure. A solution to this limitation is to design a more sophisticated statistical test to determine when to stop increasing the number of clusters in order to better fit the data. Unfortunately, statistical tests are based on assumptions about the underlying distribution of the data and if these assumptions are incorrect the test will fail to provide a reasonable result.

The challenges presented in choosing k might lead one to wonder whether hu-

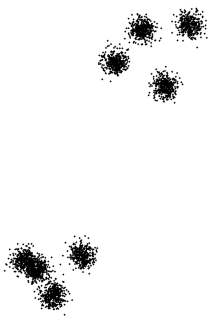

Figure 2.1: Are there 8, 7 or 2 groups?

mans are accomplished at the task. Humans are adept at navigating ambiguous and hierarchical situations, and we generally cringe at the thought of laboriously counting large numbers of objects, so perhaps we are k-choosing experts. There is a distinct (and often implicit) trend in the clustering literature to use the human visual system as a standard against which the performance of clustering algorithms should be judged. In one prominent spectral clustering paper, the authors state, "The results are surprisingly good... the algorithm reliably finds clusterings consistent with what a human would have chosen [2]." Given that our visual system is an adept and powerful data processing system (surprisingly resistant to myriad forms of algorithmic mimicry) it is reasonable to solicit its judgments on a thorny problem for which it seems particularly well-suited.

This paper takes inspiration from a computer vision project undertaken at UC Berkeley [3]. Faced with the challenging task of determining how to segment images such that objects are separated from one another by outlines, researchers enlisted human subjects to manually outline the objects in several hundred images of real-world scenes. The problem of image segmentation is very similar to choosing k in that ambiguity and hierarchy play a major role in determining reasonable answers. Detail oriented subjects might outline the leaves on a tree whereas others might just outline the branches. Through this effort researchers collected what is known as the Berkeley Segmentation Dataset, a large collection of human image segmentation data. These data have motivated and assisted several research projects and continue to be a valuable resource in the computer vision field. Studies explicitly measuring human clustering judgements are

rare, but at least one study exists that focuses on the developmental changes in human visual grouping of synthetic data sets [4].

This paper presents human judgements on a diverse set of clustering stimuli. The motivation for this undertaking is twofold. First and foremost, we hope to gain intuitions about the methods humans use to choose k and use those intuitions to develop better k-choosing algorithms. The results of this endeavor will be discussed later on. Second, we hope to create a comprehensive and detailed data set representing human clustering behavior that can be used as a standard against which to measure algorithmic performance, and to fuel innovation in this branch of machine learning.

2.3 Human Data

Eighteen undergraduate human subjects were recruited for this project, 11 female and 7 male, to determine the number of groups present in 50 distinct point light displays. Each point light display was presented at two different scales and two different rotations, for a total of four presentations per display and 200 trials per subject. Subjects were asked to determine the number of groups in each display and were encouraged to give more than one answer if appropriate. There was no time limit for response. Subjects were told to ignore answers above 20 and to focus on "the bigger picture" to find a reasonable answer less than 20. In addition to k judgments, response times and sequence information were recorded. The sequence of trial presentations was structured into four blocks of 50 randomly ordered trials each, with each block consisting of a unique permutation of every point light display. After the subjects completed all 200 trials, they were interviewed in order to gain insight into their techniques. The interview consisted of two questions:

- What strategies did you use for this task?
- Were any of the displays harder than the others?

While there are likely many interesting phenomena to investigate in the human data, such as consistency, reaction time, the relationship between reaction time and consistency, the relationship between reaction time and k, etc., this paper is mostly con-

cerned with the overall gist of the human responses, their relationship to state-of-the-art k-choosing algorithms, and the new k-choosing methods they inspire. To that end, the human data were analyzed and will be presented collapsed across subjects, scales and rotations. The results are presented in normalized bar plots meant to represent a probability distribution over k, based on the number of responses at each particular k. For each display there are at least 72 responses represented, assuming one answer per subject per trial. The actual number of responses might be larger if subjects were inclined to give multiple answers.

The 50 point light displays used in this experiment were chosen to provide a mixture of depth and breadth within the extremely large space of possible point light displays. Sixteen of the displays consisted of various riffs on mixtures of Gaussians, while another three were mixtures of Gaussians overlaid with uniformly distributed random noise. Nine displays consisted solely of uniformly distributed random noise (with differing number of samples between eight and 10,000). Three displays depicted two-dimensional embeddings of real data. Eight displays contained lines, circles or a combination of the two. The final 11 displays consisted of other synthetic data transformed by a variety of nonlinear distortions. See Fig. 2.2 for thumbnails of all the displays used. Subjects always saw the displays as white points on a black background, but for the sake of presentation the displays in this paper are black on white and the points have been increased in size.

We focused heavily on mixture of Gaussian data sets due to the prominence of the Gaussianity assumption in the machine learning literature [5][6][7]. We also used several data sets with uniform noise in order to investigate how subject responses varied with sample size and to what extent subjects saw patterns where none were justified by the underlying distribution. Our shape-based and distorted displays were included for breadth and represent a case where the data are drawn from no standard underlying distribution.

Though all of the data sets are two-dimensional, we anticipate that insights gained from this study will lead to algorithmic improvements even in high-dimensional spaces. Certain algorithms (such as the Eigengap algorithm discussed below) operate over affinity matrices that are insensitive to the underlying dimensionality of a data set.

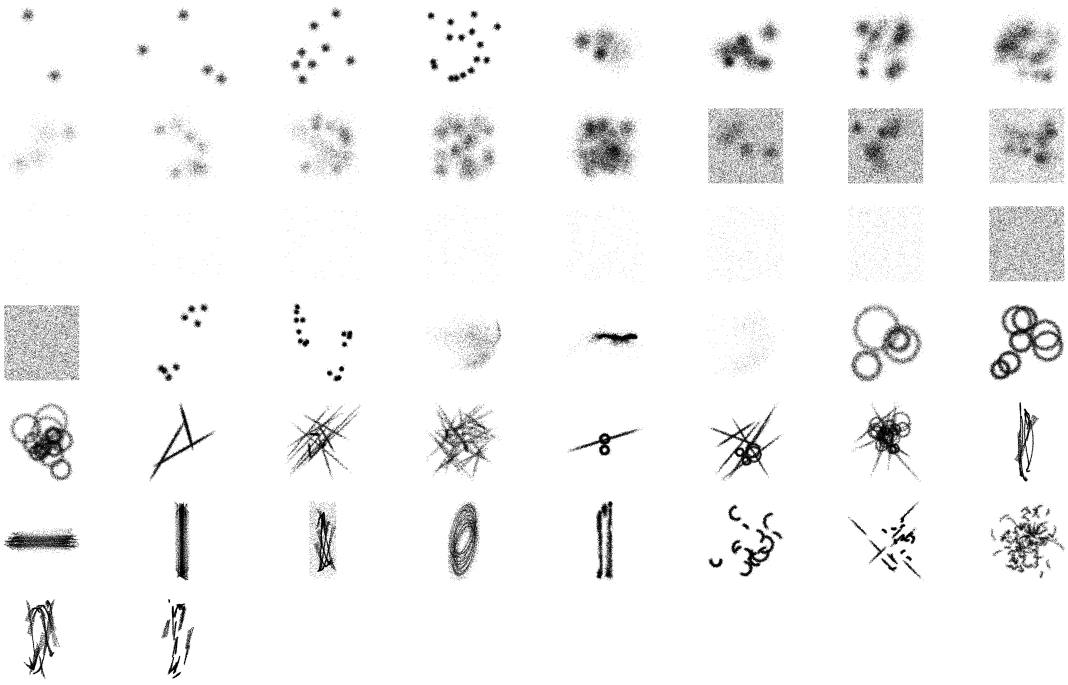

Figure 2.2: The stimuli.

Thus, improvements in these algorithms as measured by similarity to human performance in two dimensions will likely scale to high-dimensional data.

2.4 Results

Several interesting trends emerge in the human responses. In the interview section of the study, subjects predominantly report two central strategies: looking for areas of greatest density, perhaps separated by empty space ($N = 13$), and counting shapes or blobs ($N = 11$). Many of those subjects report using both strategies ($N = 9$). The latter strategy can be interpreted as a model fitting strategy, where subjects see a collection (mixture) of objects (e.g. arcs or Gaussians) and then explicitly count the number of those objects regardless of overlapping density. Rarer strategies include grouping by shape orientation ($N = 1$), and grouping by shape type (if there are both circles and lines in a display, there are two groups, $N = 1$). Finally, one subject explicitly mentions a hierarchical strategy, where he or she searches for small clusters first, and then groups

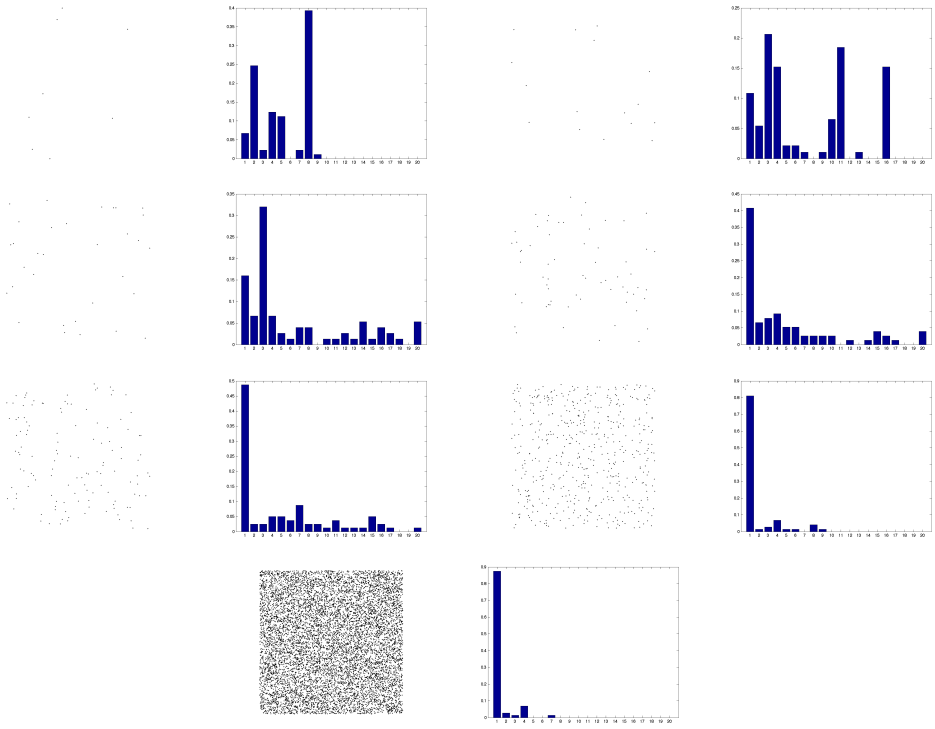

Figure 2.3: Responses to uniform noise. Displays with few samples present significant ambiguity. As sample size increases, entropy decreases.

them into larger clusters.

Subjects cite two main sources of difficulty: displays containing very few data points ($N = 9$) and displays with lots of (often overlapping) shapes ($N = 12$). A few subjects consider displays with random noise to be difficult ($N = 3$).

We find echoes of these subjective measures in the choices of k that humans make. Insofar as the distribution over k is less peaked (has higher entropy) for a particular data set, one might interpret that data set as more difficult. Conforming with interview responses indicating that small sample sizes cause difficulty, we can see in Fig. 2.3 that entropy decreases as sample size increases for displays of uniform noise.

In concordance with interview responses indicating two primary strategies, we find several examples of bimodal responses for displays where these two strategies would diverge. Some examples are shown in Fig. 2.4.

In all of the mixture of Gaussian cases, humans perform very consistently. In cases where the Gaussians have low variance and well separated means, almost all sub-

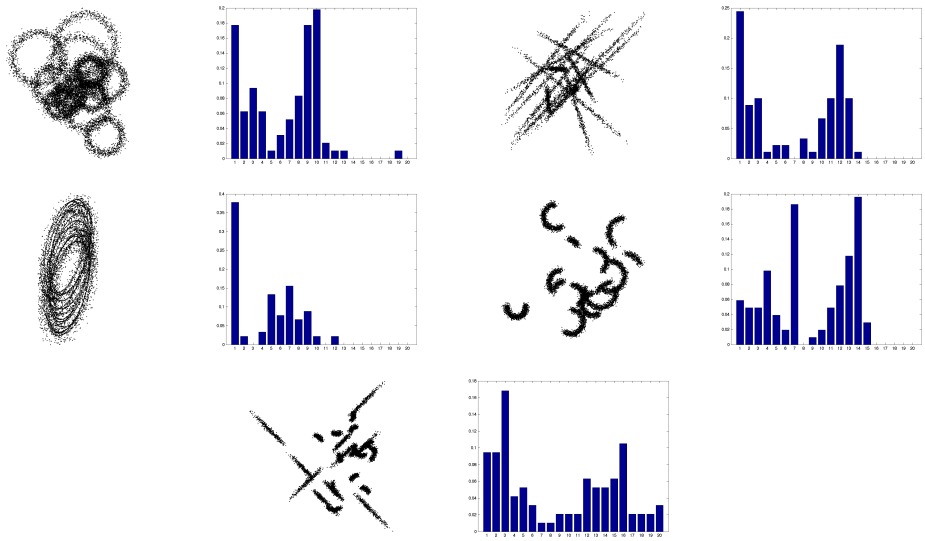

Figure 2.4: A sample of displays that elicited bimodal responses from subjects.

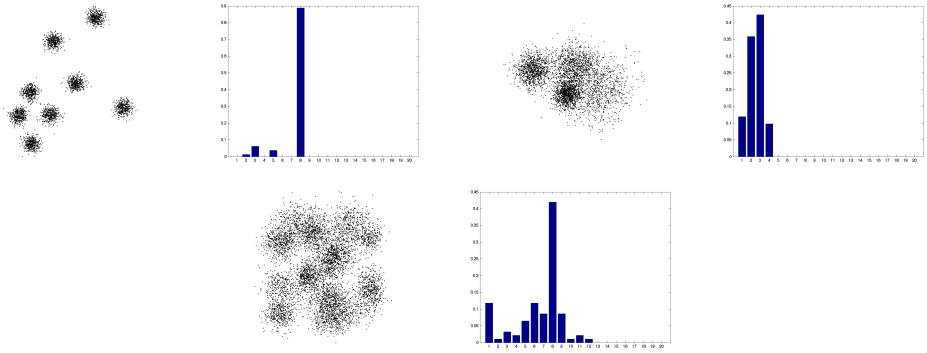

Figure 2.5: Human responses are generally consistent for mixture of Gaussians data sets.

jects indicate the correct number of Gaussians. Where the Gaussians have high variance and close means, humans generally agree on a tight range of values for k that corresponds to the number of "blobs" in the display. See Fig 2.5 for some examples of these results.

2.5 Strategies

Based on the observations discussed above, humans follow at least two broad strategies when choosing k, density strategies and model fitting strategies. In this section, two algorithms from recent work in the field that represent these two strategies will be briefly described and their performance compared to the human data.

2.5.1 Density Strategies

Density strategies discover clusters by looking for regions of low density between groups of points, following density within groups to find all the points that belong to them, and attempting to ignore low density noise. Several algorithms have endeavored to formalize these strategies, notably [8].

A more recent algorithm [9], which this paper will refer to as the Eigengap algorithm, brings similar strategies for finding k under the spectral clustering umbrella. The Eigengap algorithm treats each data point as a node on a graph, and then performs a random walk between the nodes, with the probability of transitioning between any two nodes weighted by the distance between them. If two nodes are close together then the probability of transitioning from one to the other will be high and if two nodes are far apart then the probability of transitioning from one to the other will be low. Thus, if a group of points is separated by a large distance from the rest of the data, a random walk will be unlikely to transition across that gap. In this case, all the points within the group will have a high probability of ending up on other points in the group and little probability of ending up outside the group.

A matrix, P, representing the probability of any point ending up at any other point in the data set will therefore be block diagonal if there are distinct groups within the data set that are separated by sufficient distance. This block diagonal structure is represented by the n largest eigenvalues of P, and eigenvalues greater than the nth will generally be much smaller than the first n eigenvalues. By finding the largest difference between neighboring eigenvalues sorted in descending order, one can find a useful estimate of the number of groups in the data. For example, if the difference between the third and fourth eigenvalues is 0.4 and that distance is greater than the distance between

all other adjacent eigenvalues, then there are likely to be three groups in the data.

As the random walk progresses the Eigengap algorithm naturally finds groups of coarser and coarser structure. Over an increasing number of steps, a random walk will become more and more likely to cross over low density sections of the data set, and thus two groups that initially might be separated will over time merge and lower values for k will be discovered. In this way the Eigengap algorithm can respond well to hierarchical data given a sufficiently long random walk.

The implementation of the Eigengap algorithm in this paper uses a small tweak as compared to [9]. Given a data set with N points, the authors of [9] suggest searching over N possible values for σ, a parameter used in generating the transition probability matrix, between the minimum and maximum pairwise distances in the data set. The algorithm used in this paper searched over 10 possible values for σ in order to drastically reduce computation time while still investigating a reasonable range of values. Also, given the large (over 10,000) number of points in some of the data sets, a sparse implementation of the Eigengap algorithm was used, with pairwise distances only calculated between nearest neighbors (and the number of nearest neighbors equal to one percent of the total number of points in the data set).

2.5.2 Model Fitting Strategies

Several model fitting strategies based on an assumption of mixture of Gaussian distributed data have been proposed in the past [5] [6]. This section describes a recent variant called PG-means [7]. PG-means searches for Gaussian clusters in a data set using an iterative process. The algorithm is initialized with $k = 1$ and it attempts to find an appropriate centroid and covariance matrix for a single Gaussian cluster given the data using the Expectation-Maximization (EM) algorithm. PG-means then randomly projects the data set and the Gaussian model down to one dimension n times (we used $n = 10$). The Kolmogorov-Smirnov (KS) test is applied to each projection and if every KS test indicates a sufficiently good fit (as measured by a parameter α that was set to 0.001) then the current value for k is accepted. Otherwise, k is incremented by one and the entire process is repeated.

If PG-means did not find an answer less than $k = 20$, the algorithm was halted

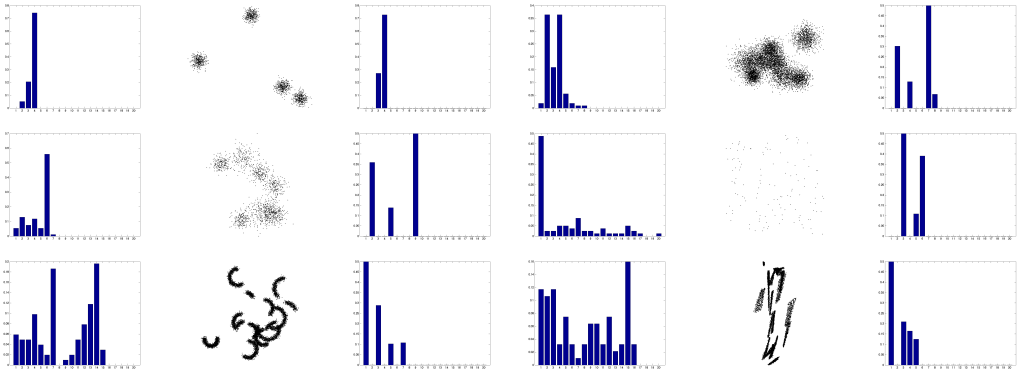

Figure 2.6: Sample human (left) versus combined Eigengap and PG-means (right) probability distributions over k.

and its response considered to be $k = 1$. Note that unlike the Eigengap algorithm, PG-means will only give one possible value for k.

2.5.3 Comparison with Human Data

To broadly compare Eigengap and PG-means performance with human performance, both the human results and the algorithmic results are interpreted as probability distributions over k. The sum Kullback-Leibler (KL) divergence is then calculated between the human results and both Eigengap and PG-means over all 50 data sets. The human results are considered the true distribution and the algorithmic results are considered the model distributions for purposes of calculating KL.

Unsurprisingly, given its ability to return multiple values of k and discover hierarchical organization, Eigengap outperforms PG-means with a sum KL divergence of 269.1 compared to 316.2 for PG-means. A simple unweighted combination of the two, however, performs better than either algorithm on its own with a sum KL divergence of 245.8 (an improvement of 8.7 percent over the Eigengap algorithm). See Fig. 2.6 for some sample comparisons of this combined result to human responses.

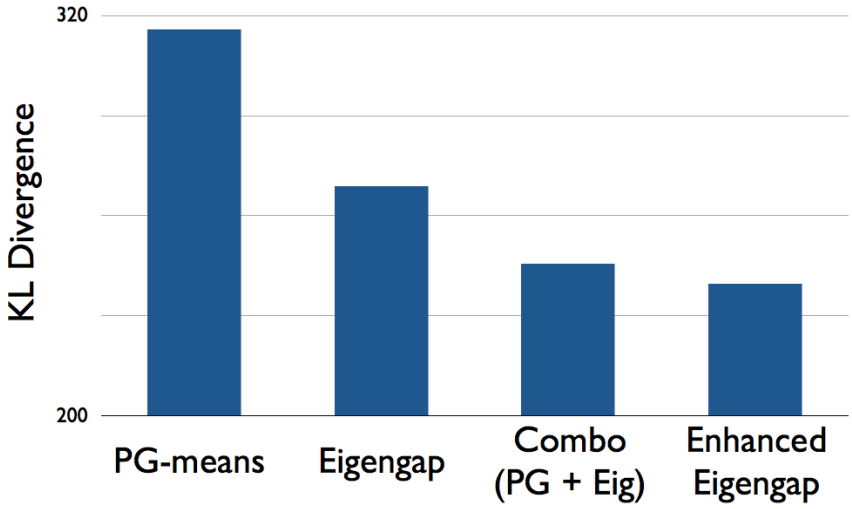

Figure 2.7: The KL divergence scores for PG-means, Eigengap, their combination, and the enhanced version of Eigengap. (Note that the Y-axis is scaled to make the distinctions more visible.)

2.6 New Density Strategies

While the Eigengap algorithm performs its function of following density well, this paper proposes two novel strategies that use density in other ways. Both of the techniques proposed are based on leveraging higher order density information than traditional pairwise distances or affinities.

First, we are developing an algorithm that intelligently culls uninformative samples from a dataset in order to increase the accuracy and decrease the computational complexity of k-choosing algorithms. These uninformative samples should correspond to the less dense or "bright" samples that humans tend to ignore. Consider a sample, $\vec{x}_i$, and the set of its κ (kappa) nearest neighbors, ν_i. Define the neighborhood variance of $\vec{x}_i$ as $\sigma_i^2 = \sum_{n \in \nu_i} ||\vec{x}_i - \vec{x}_n||^2 / \kappa$ and define the normalized neighborhood variance of $\vec{x}_i$ as $norm(\sigma_i^2) = \sigma_i^2 / \min_{n \in \nu_i}(\sigma_n^2)$. Remove all samples from the dataset whose $norm(\sigma_i^2)$ measure is above threshold. Both this threshold and κ can be reasonably set based on the number of samples in the dataset and the potential range of k. In a mixture of Gaus-

sians setting, samples with high $norm(\sigma_i^2)$ will be far from the mean of their underlying distribution, and thus less prototypical than points with lower $norm(\sigma_i^2)$.

For an intuition on how this might help a density based approach to choosing k, consider a simple mixture of two Gaussians whose means are well separated but whose samples overlap in some part of the space. An Eigengap-style algorithm will be able to traverse points across both Gaussians quite easily due to this overlap, leaving little indication that there are two clear clusters. By culling all but a small number of points with the lowest normalized neighborhood variance this "bridge" between the two Gaussians is removed.

Second, we propose an extension to spectral clustering based on the observation that human subjects consider samples with large differences in neighborhood variance to be likely drawn from different clusters (though this is not the case when the variance changes smoothly across space). σ_i^2 can be interpreted as the projection of $\vec{x}_i$ into a one-dimensional space, and a new pairwise affinity matrix between samples can be created based on distances in this space. By adding this new affinity matrix as a second view to spectral clustering one might expect to obtain results more similar to human judgments. Early data from applying this technique to the algorithm in [9] are promising and support this expectation. A version of this technique improves Eigengap performance as compared to human performance by approximately 11 percent with a sum KL divergence of 239.7 (compared to 269.1 for the standard Eigengap algorithm). Interestingly, combining this modified version of Eigengap with PG-means nets only a 2.1 percent improvement over the modified Eigengap alone (compared to the 8.7 percent improvement when combining PG-means with standard Eigengap), indicating that sensitivity to density changes might be part of what drives model fitting strategies in humans. See Fig. 2.7 for a comparison across all algorithms.

2.7 Conclusion

Finding reasonable values for k is an important and difficult problem in unsupervised machine learning. As one can see from the samples in Fig. 2.6, current algorithms do well in certain situations and very poorly in others. By further investigating hu-

man performance and attempting to apply the insights garnered from such investigation, substantial progress can be made in developing new algorithms to tackle this thorny problem.

2.8 Acknowledgments

The author would like to thank Virginia de Sa, Gedeon Deák and Marta Kutas for valuable feedback on this project. This work was supported by NSF IGERT Grant #DGE-0333451 to GW Cottrell/VR de Sa. This chapter is a reprint of material as it appears in J. M. Lewis, "Finding a better *k*: A psychophysical investigation of clustering," *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pp. 315-320, 2009.

3 Human Cluster Evaluation and Formal Quality Measures: A Comparative Study

3.1 Abstract

Clustering quality evaluation is an essential component of cluster analysis. Given the plethora of clustering techniques and their possible parameter settings, data analysts require sound means of comparing alternate partitions of the same data. When proposing a novel technique, researchers commonly apply two means of clustering quality evaluation. First, they apply formal Clustering Quality Measures (CQMs) to compare the results of the novel technique with those of previous algorithms. Second, they visually present the resultant partitions of the novel method and invite readers to see for themselves that it uncovers the correct partition. These two approaches are viewed as disjoint and complementary.

Our study compares formal CQMs with human evaluations. For this study, we select a versatile set of measures based on a novel theoretical property-based taxonomy. We find that some highly natural CQMs are in sharp contrast with human evaluations while others correlate well.

Through the comparison of clustering experts and novices, as well as a consistency analysis, we support the hypothesis that clustering evaluation skill is present in the general population.

3.2 Introduction

Clustering is a fundamental data analysis tool that aims to group similar objects. It has been applied to a wide range of disciplines such as astronomy, bioinformatics, psychology, and marketing. Successful clustering often requires using a number of different clustering techniques and then comparing their output. The evaluation of clusterings is an integral part of the clustering process, needed not only to compare clusterings to each other, but also to determine whether *any* of the clusterings obtained are sufficiently good.[1]

Unfortunately, there is no consensus on a formal definition of clustering. As a result, there are a wide variety of formal Clustering Quality Measures (CQMS), aka. internal validity indices, that aim to evaluate the quality of clusterings. To compare clusterings, researchers often select a CQM, which assigns a numerical value to a partition representing its quality.

Researches rarely rely on CQMs alone. There is a deep implicit assumption running through the clustering literature that human judgment of clustering quality is quite good. Authors visually present the resultant partitions and invite the reader to see for themselves that the new method performs well. To take one example, in their influential paper on spectral clustering Ng, Jordan and Weiss write, "The results are surprisingly good... the algorithm reliably finds clusterings consistent with what a human would have chosen [2]." Up until now, clustering quality measures and human judgment were considered complementary approaches to clustering evaluation. Most papers that present novel clustering algorithms include these two types of evaluations separately.

Our study compares formal CQMs with human evaluations to determine how often they agree, and whether certain CQMs correlate better with human judgments than others. We also evaluate the consistency of human responses–if humans are very inconsistent, then it is unlikely that they are good judges of cluster quality (an ideal measure is stable on the same partition). Further, we separate our human subjects into expert and non-expert groups to determine whether clustering evaluation requires experience, and

[1] If no good clusterings have been found, it could either indicate that further exploration is necessary, or that the underlying dataset has no good clustering (the data is not "clusterable", see [10] for more on clusterability).

identify divergent strategies between the groups.

In order to select a representative set of CQMs for our study, we construct a property-based taxonomy of CQMs that distinguishes them on grounds beyond their particular mathematical formulations. The CQMs selected for the study are versatile in that they each satisfy a distinct set of these properties. The CQM properties are detailed in Section 3.7.

Previous studies have investigated how humans choose the number of groups [11] and partition data [4] in a clustering setting, but these approaches only show what humans think are the optimal partitions rather than how they judge partition quality in general. Our study uses a set of non-optimal partitions that humans partially order by quality, giving us more detailed quality judgments than in past work. Intuitively, in [11] and [12] subjects took on the role of a k-choosing algorithm and a clustering algorithm (respectively), whereas in this study subjects are in the role of clustering evaluators.

Our main findings are as follows. We find that some highly natural CQMs disagree with human evaluations. CQMs with natural mathematical formalizations do not always evaluate clusterings in ways that seem natural to humans. On the other hand, we identify CQMs whose evaluations are well correlated with those of humans. In particular, we find that Silhouette [13] and Dunn's index [14] are highly correlated with human evaluations.

Our findings also indicate that there is sufficient similarity between the evaluations of novices and experts to suggest that clustering evaluation is a task that does not require specific training (though it may benefit from training). This opens the door for using human computation [15] resources such as Amazon's Mechanical Turk to quickly solicit a large number of clustering quality judgments from novices as part of the data analysis process.[2]. Nevertheless, experts show much less sensitivity to the number of clusters and relate more closely to a greater range of clustering quality measures than novices, indicating a nuanced approach to the evaluation problem. Regarding consistency, we find that even novices are more consistent in their evaluations than our set of CQMs.

The paper is organized as follows. In Section 3.3 we introduce the formal CQMs

[2]This tactic has been applied successfully in topic modeling [16] and image segmentation [3]

selected for our study, and in Section 3.4 we review our experimental methods. In Section 4.5 we present our results, followed by a discussion in Section 4.6. Our theoretical taxonomy of CQMs is in Section 3.7, and conclusions in Section 3.8.

3.3 Clustering quality measures

In this section we introduce the CQMs selected for our study. Each CQM satisfies a distinct set of properties in the taxonomy we present in Section 3.7.

Let X be a finite domain set. A *distance function* is a symmetric function $d : X \times X \rightarrow R^+$, such that $d(x,x) = 0$ for all $x \in X$. A *k-clustering* $C = \{C_1, C_2, \ldots, C_k\}$ of dataset X is a partition of X into k disjoint subsets (so, $\cup_i C_i = X$). A *clustering* of X is a k-clustering of X for some $1 \leq k \leq |X|$. Let $|C|$ denote the number of clusters in clustering C. For $x, y \in X$ and clustering C of X, we write $x \sim_C y$ if x and y belong to the same cluster in C and $x \not\sim_C y$, otherwise. Finally, a CQM is a function that maps clusterings to real numbers.

Gamma: This measure was proposed as a CQM by [17] and it is the best performing measure in [18]. Let d^+ denote the number of times that a pair of points that was clustered together has distance smaller than two points that belong to different cluster, whereas d^- denotes the opposite result.

Formally, let $d^+(C) = |\{\{x,y,x',y'\} \mid x \sim_C y, x' \not\sim_C y', d(x,y) \leq d(x',y')\}|$, and $d^-(C) = |\{\{x,y,x',y'\} \mid x \sim_C y, x' \not\sim_C y', d(x,y) \geq d(x',y')\}|$. The *Gamma* measure of C is $\frac{d^+(C) - d^-(C)}{d^+(C) + d^-(C)}$.

Silhouette: The Silhouette measure was defined by [13]. The measure is useful in that it enables graphical representation of clusterings on data that otherwise may be difficult to visualize. Silhouette is also the default clustering quality measure in MATLAB.

Let $dist(x, C_i) = avg_{y \in C_i} d(x,y)$. The *silhouette of a point x* with respect to clustering C is $S(x,C) = \frac{\min_{j \neq i} dist(x,C_j) - dist(x,C_i)}{\max(\min_{j \neq i} dist(x,C_j), dist(x,C_i))}$ where $x \in C_i$. The *silhouette of a clustering C* is $sum_{x \in X} S(x,C)$.

Dunn's measure: Dunn's measure [14] compares the maximum within-cluster distance to the minimum between-cluster distances. *Dunn's measure* of C is $\frac{\min_{x \not\sim_C y} d(x,y)}{\max_{x \sim_C y} d(x,y)}$.

Average Between and Average Within: The Average Between and Average Within measures evaluate the between-cluster separation and within cluster homogeneity, respectively. The *average between* of C is $avg_{x \not\sim_C y} d(x,y)$. The *average within* of C is $avg_{x \sim_C y} d(x,y)$.

Calinski-Harabasz: The Calinski-Harabasz measure [19] makes use of cluster centers. Let $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} |x|$ denote the center-of-mass of cluster C_i, and $\bar{x}$ the center-of-mass of X. Let $B(C) = \sum_{C_i} |C_i||c_i - \bar{x}|^2$ and $W(C) = \sum_{C_i} \sum_{x \in C_i} |x - c_i|^2$. The *Calinski-Harabasz* of C is $\frac{n-k}{k-1} \cdot \frac{B(C)}{W(C)}$.

Weighted inter-intra: The weighted inter-intra measure is proposed by [20]. It compares the homogeneity of the data to its separation. Let $intra(C_i) = avg_{x,y \in C_i} d(x,y)$ and $inter(C_i, C_j) = avg_{x \in C_i, y \in C_i} d(x,y)$. The *Weighted inter-intra* of a clustering C is $(1 - \frac{2k}{n}) \cdot (1 - \frac{\sum_i \frac{1}{n-|C_i|} \sum_{j \neq i} inter(C_i, C_j)}{\sum_i \frac{2}{|C_i|-1} intra(C_i)})$, where n is the number of points in the dataset.

3.4 Methods

We ran two groups of human subjects and a group of clustering quality measures on a partition evaluation task. Our human subjects were divided into a novice group with little or no knowledge of clustering methods and an expert group with detailed knowledge of clustering methods. In this section we will first detail our human protocol and stimuli, and then review the components of our analysis, the results of which are presented in the next section.

3.4.1 Human subjects and stimuli

Twelve human subjects were recruited for this project as the novice group, 9 female and 3 male, with an average age of 20.3 years. The novice subjects have no previous exposure to clustering. The expert group consists of 5 people and includes the authors of this paper. All experts have studied clustering in an academic setting, and 4 have done research on the subject.

We used 19 different two dimensional datasets to generate our clustering stimuli, drawn from [11], and chosen to represent a range of dataset types including mixtures of Gaussians and datasets with hierarchical structure. In order to maintain responsiveness

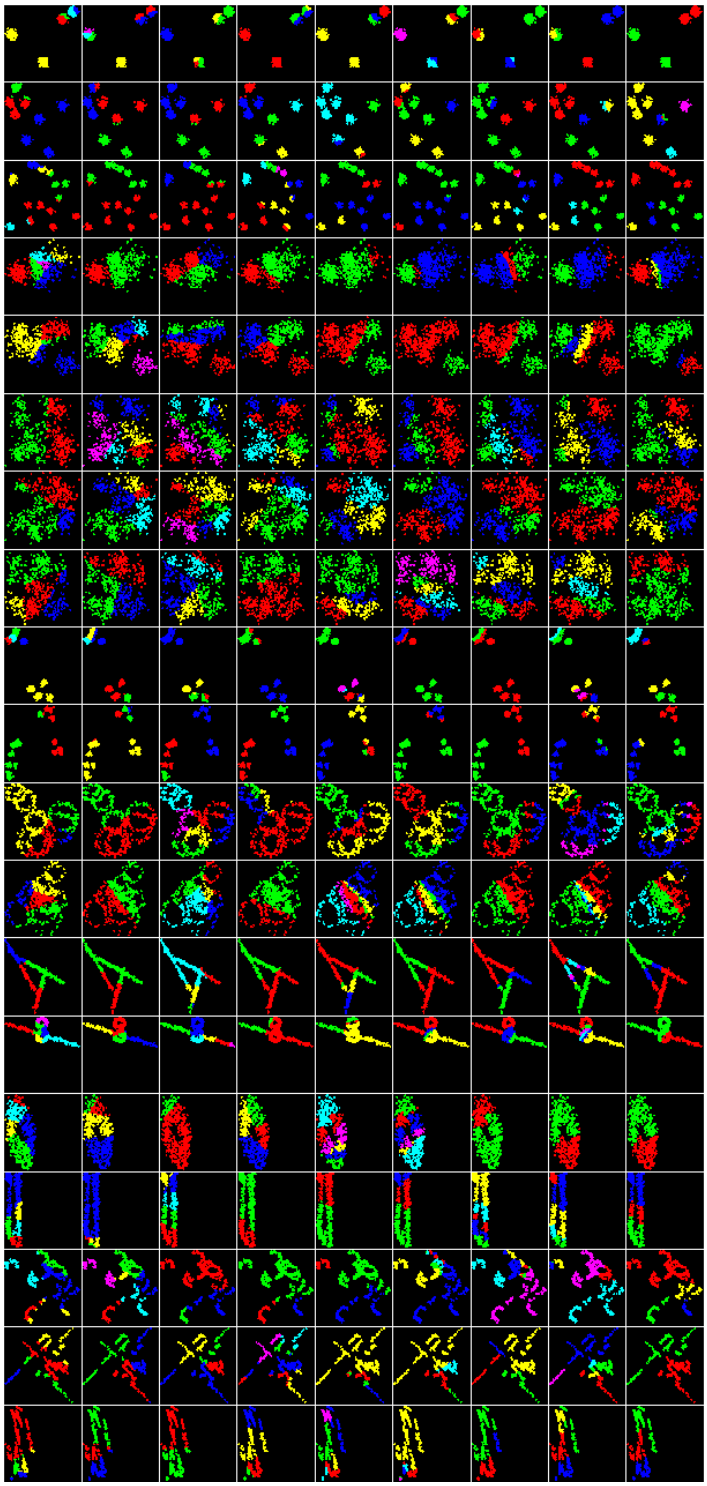

Figure 3.1: All stimuli. Datasets are in rows; partitions are in columns. Colors represent different clusters, and we asked subjects to provide a ranking of the partitions across every row.

Table 3.1: Correlation coefficients between human responses and CQMs with k factored out (except for the k column). Text in bold (excluding k column) if $p < .0025$ after Bonferroni correction for $n = 20$ comparisons per subject group and $\alpha = .05$.

ρ	Expert Positive	Expert Negative	Novice Positive	Novice Negative	Gamma	Silhouette	Dunn	Avg Within	Avg Btw	CH	W-Inter/Intra	k
Expert Pos	1	**-.35**	**.56**	-.19	-.15	**.46**	**.40**	**-.39**	**.34**	**.44**	.19	-.43
Expert Neg		1	-.13	**.44**	.09	**-.27**	-.12	**.44**	-.18	**-.36**	**-.30**	.32
Novice Pos			1	-.04	-.13	**.39**	**.40**	-.20	.23	**.30**	.04	-.73
Novice Neg				1	.08	**-.27**	.01	**.30**	-.07	**-.25**	**-.27**	.71

of the stimulus presentation interface, we subsampled 500 points randomly from each dataset. We use synthetic datasets in order to better generate a wide range of stimuli, and our datasets are 2D to facilitate visualization.

Each dataset is randomly clustered nine times in the following manner. For each of the nine clusterings, we first draw the number of partitions, k, from a uniform distribution over the integers 2 to 6. Second we choose cluster centroids using two strategies: for four of the clusterings we randomly select k centroids from the original dataset, and for five of the clusterings we select k centroids from a Laplacian Eigenmap embedding of the data. Finally we color points based on the identity of their nearest centroid in the appropriate space. The goal of this approach is to create stimuli with varied clustering quality.

Each trial consisted of all nine different partitions of the same dataset randomly arranged per trial in a 3 by 3 grid (see Figure 3.1 for a visualization of all the stimuli and the supplementary material for a more detailed view). The datasets were shown as scatter plots with colored points on a black background to reduce brightness-related eye strain. For novice subjects, trials were organized into three blocks of 19, where each dataset appeared once per block and the order of the datasets within each block was randomized. Expert subjects were tested on one block of non-randomized datasets. We instructed subjects to choose the two best partitioned displays and the one worst partitioned display from the nine available on every trial.

3.4.2 Analysis

We analyzed our novice subjects for internal consistency of their positive and negative ratings across blocks and found that even our least consistent subject performed well above chance. We did not exclude any subjects due to inconsistency and we did not analyze internal consistency for experts as they were only tested on one block.

To analyze consistency across subjects we use Fleiss' κ [21] and include neutral responses. Fleiss' κ measures the deviation between observed agreement and the agreement attributable to chance given the relative frequency of ratings and normalized for the number of raters. Neutral ratings are twice as frequent as non-neutral, and positive ratings are twice as frequent as negative ratings, so the compensation for relative frequency in Fleiss' κ makes it well-suited to our data. In addition, we perform a consistency analysis on the clustering quality measures by discretizing their ratings in a manner similar to the human data.

We analyze the relationship between novice ratings, expert ratings and clustering quality measures by calculating the Pearson's correlation coefficient, ρ, between ratings. To make the responses as comparable as possible we normalize response vectors to a length of one within each dataset. Human subjects have to rank two positive and one negative partition per dataset, even if every partition is quite bad, so by normalizing within dataset we make the CQM responses similar in structure—partitions are judged only relative to other partitions within a dataset.

Because cluster centroids are chosen randomly, increasing k is likely to increase the chance of getting an undesirable partition (e.g. a partition with very few data points). Additionally, partitions with higher k require more effort to interpret, and therefore we might expect novice subjects to be biased towards a lower k (see Section 4.5 for more details). For these reasons the correlations presented below control for k by partialing out a vector of k values for each partition. Geometrically this is equivalent to projecting each response vector onto the hyperplane orthogonal to the vector of k values.

3.5 Results

3.5.1 Correlation

Table 5.2 shows correlation coefficients between all measures for both expert and novice responses, with k factored out. The correlation between expert and novice human positive ratings is higher than the correlation between any CQM and either human positive rating. Similarly for the negative ratings (though the correlation is tied with that between Avg Within and novice negative ratings). The absolute values of the correlation coefficients between CQMs and expert ratings are strictly greater than or equal to those between CQMs and novice ratings, indicating a closer relationship between expert strategies and the dataset characteristics summarized by the CQMs when k is factored out. k itself correlates very strongly with the novices and less so with the experts. Silhouette provides the best overall correlation with expert ratings, and Avg Within provides the best overall correlation with novice ratings (save k).

3.5.2 Consistency

For our purposes the most disturbing form of inconsistency is a response to the same stimulus in both the positive and negative columns. For experts, stimuli with a number of positive responses 3 or higher never receive a negative rating, and only once does this occur for stimuli with 2 positive responses. In contrast the CQMs exhibit much more disagreement and novices seem to fall somewhere in between. The quantitative measure κ bears this out: CQMs score 0.128, novices score 0.183 and experts score 0.213. κ ranges from -1 to 1, with -1 representing complete disagreement, 1 representing complete agreement and 0 representing the amount of agreement expected by chance. While there is no standard significance test for differences in κ, the rating scale suggested by Landis and Koch [22] would classify the CQM and novice rater groups each as in slight agreement, and the expert raters as in fair agreement. To test whether any one measure was significantly harming CQM consistency we left each out in turn from the analysis and found values ranging from 0.098 to 0.172, which is in line with the CQM consistency with no measure left out, and in every case less consistent than the novice subjects.

Table 3.2: A summary of the number of partitions for which a high degree of agreement was achieved by the raters. If a partition is classified as negative or positive by 90% - 100% of raters, it would be added to the top row, and similarly for the other buckets.

Percent Response for Majority Rating	Experts	Novices	CQMs
90% - 100%	4	0	0
80% - 90%	15	3	1
70% - 80%	0	2	7
60% - 70%	20	9	0
50% - 60%	0	20	19
Sum >= 50%	39	34	27

Table 3.3: Summary of comparison between CQMs and human evaluations. The CQMs partition into three categories: high correlation with both subject groups, high correlation only with experts, and low correlation with both subject groups. See Table 5.2 for the correlation scores.

High correlation with human evaluations	Silhouette and Dunn
High correlation with expert evaluations only	Avg Within and CH
Weak correlation with human evaluations	Gamma, Avg Between, and W-Inter/Intra

In Table 3.2 we summarize the consistency of experts, novices and cluster quality measures. It shows how often certain percentages of raters are able to agree on negative or positive ratings for particular stimuli. Experts agree over 50% of the time on more samples (39), than do novices (34) or CQMs (27).

3.6 Discussion

3.6.1 Comparing human evaluations with CQMs

Some natural quality measures have low correlation with human evaluations. Most notably, gamma has low correlation with both positive and negative human ratings for both novices and experts. W-Inter/Intra has low correlation with the positive ratings of both subject groups. This shows that a natural mathematical formalization does not suffice to guarantee that the evaluations of clusterings produced using the CQM will seem natural to humans.

On the other hand, we identify CQMs that correlate well with human evaluations. Of these the most notable are Dunn's index and silhouette. These two popular measures correlate well with both expert and novice evaluations, on both the positive and negative ratings.

3.6.2 Comparing experts with novices

Evaluations of experts and novices have a correlation score of 0.56, higher than the correlation of any CQM with any of the two subject groups. This suggests that a cluster evaluation skill is present in the general population.

On the other hand, we observe some interesting differences between the two groups of subjects. One of the most notable differences between experts and novices is that, while both groups prefer clusterings with a fewer number of clusters, novices rely much more heavily on this heuristic.

Experts seem to use more, and more complex strategies than novices. Positive expert ratings correlate well with two more measures than positive novice ratings. No measure considered correlates better with novice ratings than with expert ratings, and in the great majority of cases the correlation is higher with expert ratings.

With a cover of at most six domain elements on any input dataset (see Section 3.7.2, Definition 5), Dunn's measure is (according to this measure of complexity) the simplest measure that we explore. While positive expert evaluations correlate well with five distinct measures, Dunn's measure is one of three measures that correlate well with novice evaluations. This further illustrates that novices rely on fewer simpler strategies, which indicates that expert evaluations may be more sophisticated and reliable. Since Dunn's measure always has a constant cover, it is also highly susceptible to noise. Indeed, manual examination of novice evaluations confirms that their evaluations are noise sensitive.

3.6.3 Consistency

Given the difficulty of knowing whether humans or CQMs do a reasonable job of evaluating clustering quality, one might hope that at least they are consistent across

individuals (or measures). Consistency indicates that some repeatable process is at work and that its repeatability is minimally affected by changes in input. Of course CQMs are perfectly consistent on a within measure basis—given the same partition they will always report the same quality–and one is tempted to suggest that between measure consistency is an unfair point of comparison; aren't all the measures using quite different evaluative procedures, and didn't we select them to be distinct? We did, but CQMs purport to evaluate clustering quality in general. Insofar as they evaluate this more nebulous property they should be consistent, even if their methods differ. As it turns out, they are somewhat consistent with each other, just not as consistent as humans. Further, the consistency story did not vary when we tested all the leave-one-out subsets of CQMs, indicating that CQM consistency is not being ruined by just one divergent measure.

Human experts are the most consistent group in this study. This lends empirical support to the common practice of seeking human visual evaluations of partition quality. Novices are less consistent, and as discussed above there is evidence that the evaluations they provide are less sophisticated. Despite the unfavorable comparison to experts, it is notable that subjects with no formal knowledge of cluster analysis are able to respond more consistently than a set of CQMs. This lends credence to the notion that our ability to evaluate partitions is acquired in the natural course of visual development.

3.7 A Property-Based Taxonomy of CQMs

In the absence of formal guidelines for CQM selection[3], in particular for selecting a versatile set of CQMs, we develop a property-based framework for distinguishing CQMs based on such a framework for clustering algorithms discussed in [24] (also see [25] and [26]). The framework consists of identifying natural properties of CQMs and classifying measures based on the properties that they satisfy. For the purposes of our study we use this framework to select meaningfully versatile CQMs. This taxonomy may have independent interest for choosing CQMs in other settings.

We focus on a set of seven CQMs defined in Section 3.3. We drew our imple-

[3] Although there are no formal guidelines for CQM selection, some interesting heuristics haven been proposed, see, for example, [23].

mentations from [27] save Gamma, which we implemented ourselves, and Silhouette, which is included in Matlab. We tested a larger set of CQMs (including the Davies-Boulid measure [28], Hartigan's measure [29], and the Krzanowski-Lai measure [30]), but some occupy the same branches of the taxonomy and therefore we focus on only the measures that satisfy distinct sets of properties.[4]

Our taxonomy of CQMs follows a line of work on theoretical foundations of clustering beginning with the famous impossibility result by [31], which showed that no clustering function can simultaneously satisfy three specific properties. [32] reformulate these properties in the setting of CQMs, and show that these properties are consistent and satisfied by many CQMs. We follow up on [32] by studying natural properties that can be used to distinguish between CQMs.

In Table 3.4, we present a taxonomy of our seven clustering quality measures. Each of the properties aims to capture some fundamental feature that is satisfied by some measures. In order to compare human evaluations against a versatile set of formal CQMs, we chose CQMs that satisfy different sets of properties. The properties are defined below. The proofs of the results presented in Table 3.4 are straightforward and omitted for brevity.

Table 3.4: A taxonomy of the seven quality measures used in the study.

	Gamma	Silhouette	Dunn	Avg Within	Avg Btw	CH	W-Inter/Intra
Order-consist.	✓	*X*	*X*	*X*	*X*	*X*	*X*
Sep-invariant	*X*	*X*	*X*	✓	*X*	*X*	*X*
Hom-invariant	*X*	*X*	*X*	*X*	✓	*X*	*X*
Bounded	✓	✓	*X*	*X*	*X*	*X*	*X*
Constant Cover	*X*	*X*	✓	*X*	*X*	*X*	*X*
Norm-based	*X*	*X*	*X*	*X*	*X*	✓	*X*

[4]CQMs that satisfied the same properties performed similarly in our initial analysis.

3.7.1 Normed clustering quality measures

A *clustering quality measure m* takes a domain set X, a distance function d over X, and a clustering C of X, and outputs a non-negative real number. Some quality measures are defined over normed vector spaces. *Normed CQMs* take a quadruple of the form $(V, X, C, \|\cdot\|)$, where V is a vector space, X a finite subset of V, and $\|\cdot\|$ is a norm over V. Normed CQMs can rely on centers-of-mass of clusters that are not necessarily in X, but are part of the vector-space V. Observe that the centers-of-mass are not defined for un-normed CQMs. We define the properties for CQMs in general, but one can apply any property to a normed CQM by using the norm to define the distance function. That is, set $d(x,y) = \|x-y\|$ for all $x, y \in X$.

3.7.2 Invariance and consistency properties

Invariance properties describe changes to the underlying data that do not affect the quality of a clustering. Consistency properties describe similarity conditions under which clusterings have similar quality. We propose two new invariance properties.

Definition 1 (Separation Invariance)**.** *A CQM m is* separation-invariant *if for all X and distance functions d and d' over X where $d(x,y) = d'(x,y)$ for all $x \sim_C y$, $m(C,X,d) = m(C,X,d')$.*

A separation invariant CQM is not affected by changes to between-cluster distances. Conversely, homogeneity invariant CQMs depend only on between-cluster distances, and are invariant to changes to within-cluster distances.

Definition 2 (Homogeneity Invariance)**.** *A CQM m is* homogeneity-invariant *if for all X and distance functions d and d' over X where $d(x,y) = d'(x,y)$ for all $x \not\sim_C y$, $m(C,X,d) = m(C,X,d')$.*

Observe that separation-invariance and homogeneity-invariance are consistency properties. An additional consistency property, order consistency, is an adaptation of an analogous property of clustering functions presented in [33]. Order consistency describes CQMs that depend only on the order of pairwise distances.

Definition 3. *A CQM m is* order consistent *if for all d and d' over X such that for all $p,q,r,s \in X$, $d(p,q) < d(r,s)$ if and only if $d'(p,q) < d'(r,s)$, $m(C,X,d) = m(C,X,d')$.*

3.7.3 Domain and range properties

A bounded range can aid in interpreting the results of a CQM, in particular if the bounds are attainable by some clusterings.

Definition 4 (Bounded)**.** *A CQM m is* bounded *if there exist datasets X_1 over d_1 and X_2 over d_2, and clusterings C_1 of X_1 and C_2 of X_2, so that $m(C_1,X_1,d_1) \leq m(C,X,d) \leq m(C_2,X_2,d_2)$ for all C,X, and d.*

Our next property describes the quantity of domain elements that effect the CQM. First, we introduce the notion of an m-cover of a clustering, a subset of the domain which has the same quality as the entire set. For clustering C of X, and $X' \subseteq X$, let C/X' denote the clustering C' of X' where for all $x,y \in X'$, $x \sim_{C'} y$ if and only if $x \sim_C y$.

An *m-cover* of clustering C of X is any set $R \subseteq X$, so that $m(X,k) = m(R,C/R)$. We define clustering quality measures that have a constant size cover for all clusterings.

Definition 5 (Bounded Cover)**.** *A CQM m has* bounded cover *if there exists a constant r so that for every data set X and clustering C of X, there exists an m-cover of C of size at most r.*

CQMs that have a bounded cover search the domain space for some local features, ignoring most of the information in the dataset.

3.8 Conclusions

We perform an empirical study comparing human evaluations of clustering with formal clustering quality measures. To select a versatile set of CQMs, we develop a theoretical property-based taxonomy of CQMs. Our study shows that some CQMs with seemingly natural mathematical formulations yield evaluations that disagree with human perception. On the other hand, we identify CQMs (Silhouette and Dunn's index) that have significant correlation with human evaluations.

Our consistency analysis reveals that even novices are at least as consistent a broad set of CQMs, and perhaps more consistent. We also find significant correlations between the evaluations of expert and novice subjects. This lends support to the common practice of seeking human visual evaluations of partition quality. If one needs to evaluate a very large number of partitions it may be reasonable to use human computation via a service such as Mechanical Turk to rank partitions efficiently (or at least throw out the really bad ones). Finally, experts appear to use more sophisticated strategies than novices, indicating that training can improve human clustering evaluation performance.

3.9 Acknowledgments

This work is coauthored with Margareta Ackerman and Virginia R. de Sa, and is currently being prepared for submission. The authors would like to thank Cindy Zhang for her valuable assistance on this project.

4 A Behavioral Investigation of Dimensionality Reduction

4.1 Abstract

A cornucopia of dimensionality reduction techniques have emerged over the past decade, leaving data analysts with a wide variety of choices for reducing their data. Means of evaluating and comparing low-dimensional embeddings useful for visualization, however, are very limited. When proposing a new technique it is common to simply show rival embeddings side-by-side and let human judgment determine which embedding is superior. This study investigates whether such human embedding evaluations are reliable, i.e., whether humans tend to agree on the quality of an embedding. We also investigate what types of embedding structures humans appreciate *a priori*. Our results reveal that, although experts are reasonably consistent in their evaluation of embeddings, novices generally disagree on the quality of an embedding. We discuss the impact of this result on the way dimensionality reduction researchers should present their results, and on applicability of dimensionality reduction outside of machine learning.

4.2 Introduction

There is an evaluative vacuum in the dimensionality reduction literature. In many other unsupervised machine learning fields, such as density modeling, evaluation may be performed by measuring likelihoods of held-out test data. Alternatively, in domains such as topic modeling, human computation [15] resources such as Amazon's Mechan-

ical Turk may be employed to exploit the fact that humans are phenoms in evaluating semantic structure [16]. Human evaluations have also been used to assess image segmentation techniques [34]. The field of dimensionality reduction, however, lacks a standard evaluation measure [35], and is not as obvious a target for human intuition. Two or three dimensional embeddings can be visualized as scatter plots, but on what intuitive basis can we judge a 200 to 2-dimensional reduction to be good? In addition, Gestalt effects or simple rotations may bias human evaluations of scatter plots. Nevertheless, with no broadly agreed upon embedding quality measure (though a few have been proposed, see below), human judgment is often explicitly and implicitly solicited in the literature. The most common form of this solicitation consists of placing a scatter plot of the preferred embedding next to those of rival embeddings and inviting the reader to conclude that the preferred embedding is superior (e.g., [36]). If one is interested in applying a dimensionality reduction algorithm to visualize a dataset, is this a valid way to select a technique from the wide range of dimensionality techniques?[1] To answer this question, we need to evaluate whether humans are good at evaluating embeddings. As there is no external authority we can appeal to, this is a daunting task. However, it is relatively easy to find out whether human data analysts are at least consistent in their evaluations, which is the first aim of this study. Consistency, across individuals and across a wide range of inputs, is a reasonable prerequisite for evaluation.

Beyond investigating whether human data analysts are consistent when they evaluate embeddings, the second aim of this study is to investigate what humans are doing when they are evaluating embeddings. Such information could be useful for determining whether humans are appropriate for an evaluation task with a known structure, or for developing techniques that are tailored towards producing results that humans will find helpful. Human strategies can be inferred to some extent from which algorithms humans appreciate the output of, but can also be investigated by correlating embedding characteristics with human evaluations.

Motivated by the two aims described above, we solicit embedding quality judgments from both novice and expert subjects in an effort to determine whether they are consistent in their ratings, and which embedding characteristics they find appealing. For

[1] Moreover, one should note that dimensionality reduction comprises only a small part of the "visualization zoo" [37].

the novice subjects, we manipulate dataset knowledge—half read a description and see samples from each dataset, and half do not. We hypothesize that providing dataset information will increase consistency, as it should if the evaluative process is principled. The study consists of two experiments. The first presents subjects with a selection of embeddings derived from nine distinct dimensionality reduction algorithms; the second uses embeddings from a single algorithm with several different parameter settings for a more controlled comparison between "clustered" and "gradual" embeddings.

This work falls under the broader umbrella of what we call *anthropocentric data analysis*. Machine learning techniques benefit science and industry primarily insofar as they enable data analysts to better understand their data, make valid conclusions, and gain insight into their domain of investigation. Anthropocentric data analysis seeks to understand how human judgment is applied in the data analysis process and to develop methods that provide explicit opportunities for human interaction and insight. By better understanding how those outside the machine learning field interpret and evaluate the dimensionality reduction task, we move closer to providing a genuine service to other scientific disciplines.

In Section 4.3 we review the dimensionality reduction techniques used in this paper. In Section 4.4 we review the methodology used in our two experiments, the results of which are presented in Section 4.5. Section 4.6 discusses our results and we conclude with the prospects for human evaluation of low-dimensional embeddings in Section 4.7.

4.3 Dimensionality reduction techniques

Dimensionality reduction techniques can be subdivided into several categories: linear or non-linear, convex or non-convex, parametric or non-parametric, etc. [38]. Whilst many new techniques have been proposed over the last decade, data analysts still often resort to a linear, convex, parametric techniques such as PCA to visualize their data. Non-linear, convex, non-parametric manifold learners such as Isomap [39], LLE [40], and MVU [41] are also frequently used for visualization purposes [42, 43, 44], even though it is unclear whether these techniques produce superior results [36].

As described in the introduction, one of the key problems of dimensionality reduction is that it lacks a widely agreed upon evaluation measure [35]. In fact, it is very unlikely that there will ever be such an evaluation measure, as it would imply the existence of a *free lunch* [45]: the "optimal" dimensionality reduction technique would be the technique that optimizes the measure. Also, there is a lot of debate within the field on what a good objective for dimensionality reduction is: for instance, a latent variable model approach to dimensionality reduction suggests one should focus on preserving *global* data structure [46], whereas a manifold learning viewpoint deems preservation of *local* data structure more important [40]. The lack of an evaluation measure and the ongoing debate within the field motivate the use of a more anthropocentric approach.

In our study, we investigated nine dimension reduction techniques: (1) PCA, (2) projection pursuit, (3) random projection, (4) Sammon mapping, (5) Isomap, (6) MVU, (7) LLE, (8) Laplacian Eigenmaps, and (9) t-SNE. The nine techniques were selected based on their importance in the literature. All nine techniques are briefly described below (we refer to points in the embedding as map points).

PCA and projection pursuit find a subspace of the original data space that has some desired characteristic. For PCA, this subspace is the one that maximizes the variance of the projected data. For projection pursuit [47], the subspace maximizes the non-Gaussianity of the projected data. Random projections are linear mappings that preserve pairwise distances in the data by exploiting the Johnson-Lindenstrauss lemma [48]. Sammon mapping constructs an embedding that minimizes a weighted sum of squared pairwise distance errors, in which the weights are inversely proportional to the original distances between the data points [49]. Isomap constructs an embedding by performing classical scaling on a geodesic distance matrix that is obtained by running a shortest-path algorithm on the nearest neighbor graph of the data [39]. MVU learns an embedding that maximizes data variance, while preserving the pairwise distances between each data point and its k nearest neighbors, by solving a semidefinite program [41]. LLE constructs an embedding that preserves local data structure by minimizing a sum of squared errors between each map point and its reconstruction from its k nearest neighbors in the original data [40]. Laplacian Eigenmaps try to minimize the squared Euclidean distances between each map point and the map points corresponding

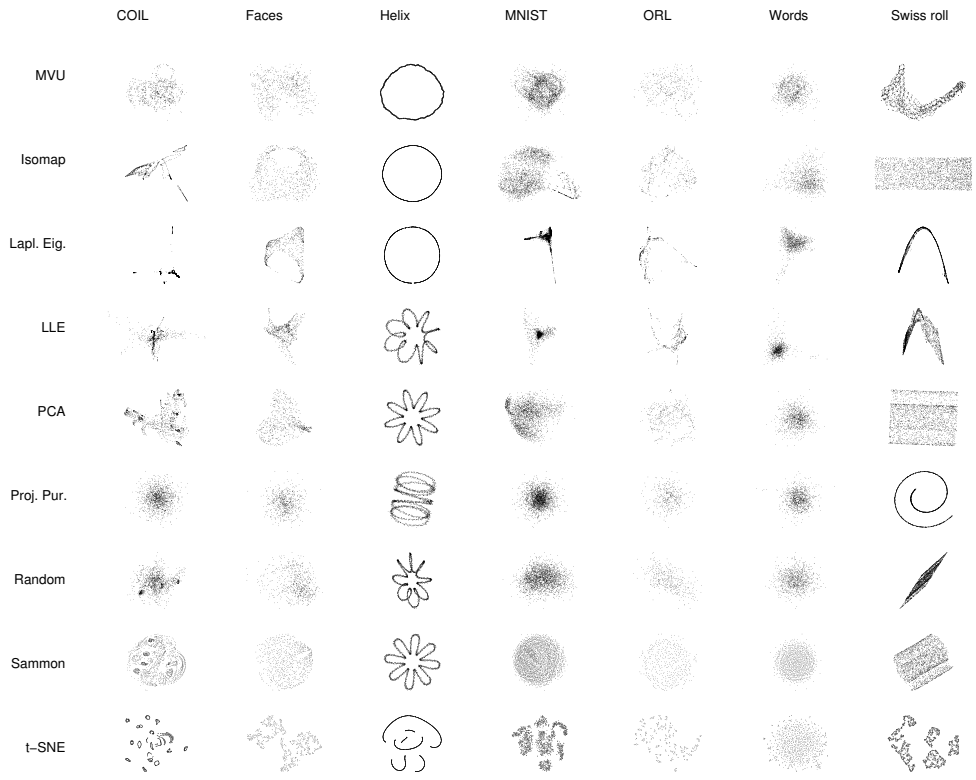

Figure 4.1: All stimuli from experiment 1. Methods are in rows; datasets are in columns.

to its k nearest neighbors in the original data, while enforcing a covariance constraint on the solution [50]. t-SNE embeds points by minimizing the divergence between two distributions over pairs of points, in which the probability of picking a pair of points decreases with their pairwise distance [36].

4.4 Experimental setup

We perform two experiments with our human subjects. The first experiment uses stimuli generated from the dimensionality reduction algorithms described above to determine whether humans are consistent in their evaluations when the embeddings are fairly distinct (the first aim of the study). The second experiment uses stimuli that are all generated by t-SNE, but with different parameter settings that affect how clustered the

resulting embedding appears. This helps us determine what type of structure humans generally prefer in embeddings (the second aim of our study).

4.4.1 Experiment 1

In the first experiment, we divided subjects into (1) an expert group with detailed knowledge of dimensionality reduction and information on the underlying datasets presented in written and pictorial form, (2) a novice group with no knowledge of dimensionality reduction and no information on the visualized data, and (3) a group of similar novices but with the same information on the underlying datasets given to the experts. The dataset information we presented to groups 1 and 3 constituted a intuitive description of the data, as well as images revealing the underlying data (e.g., the Swiss roll, or handwritten character images, see supplemental material).

Thirty one undergraduate human subjects were recruited for this study as the novice group, 16 female and 15 male, with an average age of 19.1 years. None of the novice subjects had any specific knowledge of dimensionality reduction techniques. Our expert group consisted of five subjects—three graduate students and two faculty members. The expert subjects were drawn from the same institution and represent two different departments. Amongst the five expert subjects there are four distinct academic backgrounds at the graduate level. The informed novice group had 15 subjects and the uninformed novice group 16. We generated two-dimensional point-light stimuli (see Figure 4.1 for a visualization of all the stimuli) by running the nine dimensionality reduction techniques discussed in Section 4.3 on seven different high-dimensional datasets, comprising a variety of domains. All techniques were run for a reasonable range of parameter settings, and we selected the embedding that was best in terms of the trustworthiness[2] [51] for presentation to the subjects.

Each trial consisted of nine different embeddings of the same dataset arranged randomly per trial in a 3×3 grid. The datasets were shown as scatter plots with white points on a black background to reduce brightness-related eye fatigue. For novice subjects, trials were organized into three blocks of seven, where each dataset appeared once

[2]The trustworthiness measures the ratio of k nearest neighbors in the data that is still among the k nearest neighbors in the maps.

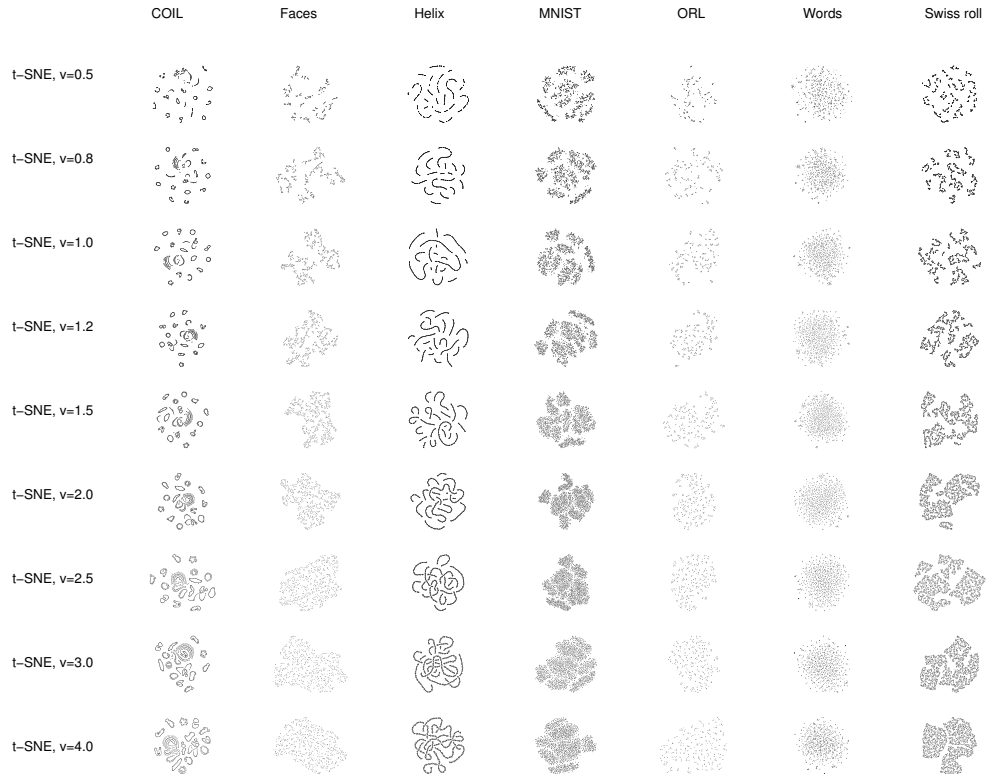

Figure 4.2: All stimuli from experiment 2. Parameter values are in rows; datasets are in columns.

per block and the order of the datasets within each block was randomized. Expert subjects were tested on one block. We instructed subjects to choose the two most useful displays and the one least useful display from the nine available on every trial. From the subject instructions, after describing what a scatter plot is and emphasizing that each set of nine plots is a different perspective on the same dataset: *For each trial, please examine all the scatter plots and choose the two that you find most useful and the one that you find least useful. The task in the second part of this experiment will be much faster and easier if you choose useful scatter plots. Do the best you can to choose useful plots based on whatever criteria you deem appropriate.* We intentionally left the task ambiguous so as not to bias subjects towards particular evaluation criteria[3], and the fictitious

[3]For instance, defining a classification task would bias subjects to embeddings that show separated clusters.

"second part" of the experiment existed solely for increasing subject motivation.

We analyzed our novice subjects for internal consistency of their positive and negative ratings across blocks and found that even our least consistent subject was more consistent than expected by chance. Hence, we did not exclude any subjects due to internal inconsistency. To analyze consistency across subjects (the first aim of this study) we use Fleiss' κ [21] and include neutral responses. Fleiss' κ measures the deviation between observed agreement and the agreement attributable to chance given the relative frequency of ratings, and normalizes for the number of raters. Neutral ratings are twice as frequent as non-neutral, and positive ratings are twice as frequent as negative ratings, so the compensation for relative frequency in Fleiss' κ makes it well-suited to our data.

We also measured the following six characteristics of our embedding stimuli: (1) variance, (2) skewness, (3) kurtosis, (4) clusteredness, (5) visual span, and (6) Gaussianity. The variance, skewness, and kurtosis were measured per dimension in a map that was scale-normalized such that $\mathbf{y}_i \in [0,1]^d$ (preserving the aspect ratio of the maps), and averaged over the d dimensions of the map. We measured clusteredness by constructing k-nearest neighbor graphs in the map with $k = 3, \ldots, 12$, and measuring the maximum clustering coefficient of the resulting graphs [52]. The clustering coefficient computes the ratio of connections between the adjacent vertices of map point i, averaged over all map points. The visual span of each map was measured by fitting a Parzen kernel density estimator with Gaussian kernels on the map (the variance σ of the Gaussians was optimized on a small validation set). Subsequently, we measure the ratio of the map surface that has a density of at least 10% of the maximum density of the density estimate. The Gaussianity of the maps was determined by averaging the results of Lilliefors tests [53] performed on $5,000$ one-dimensional random projections of the map[4]. We analyze the relationships between novice informed ratings, novice uninformed ratings, expert ratings, and the six quantitative measures by calculating the Pearson's correlation coefficient ρ between ratings and measures (after normalization within trial).

[4]Note that if the distribution of points in the embedding is Gaussian, the point distribution in each of the random projections has to be Gaussian as well.

4.4.2 Experiment 2

The second experiment was run directly following experiment 1 on the same subject pool using the same methods, save stimulus design. In experiment 2, the nine stimuli in each trial are obtained by running t-SNE with nine different degrees of freedom v (viz. $v = 0.5, 0.8, 1.0, 1.2, 1.5, 2.0, 2.5, 3.0, 4.0$) on the seven datasets. The degrees of freedom in t-SNE determine to what extent the visualizations are "clustered" [54]. This allows us to investigate whether people appreciate "clusteredness" or "graduality" in embeddings. All stimuli are shown in Figure 4.2.

4.5 Results

We present the results of our experiments separately below in Section 4.5.1 and 4.5.2.

4.5.1 Experiment 1

For the first experiment, the Fleiss' kappa consistency measure κ for experts is 0.39, for uninformed novices is -0.28, and for informed novices is -0.40. Fleiss' kappa κ ranges from -1 to $+1$, with -1 representing complete disagreement, $+1$ representing complete agreement and 0 representing the amount of agreement expected by chance. Though there is no standard significance test for Fliess' kappa, based on the Landis and Koch scale [22], experts exhibited fair to moderate agreement, while both groups of novices exhibited poor agreement. Hence, the consistency measures reveal that, whereas experts tend to agree with each other on the quality of an embedding, novices strongly disagree with each other in their evaluations (they disagree more than if the evaluation was done randomly). Surprisingly, novices who received information on the underlying data disagree more strongly with each other than novices who had no information about the underlying data (counter to our hypothesis but interpretable, see below).

In Figure 4.3, we depict the raw ratings (averaged over each group) as a collection of Hinton diagrams. In the figure, a large square indicates that a relatively large number of subjects gave a positive or negative evaluation of the embedding of the corresponding dataset, constructed by the corresponding technique. The top row of diagrams

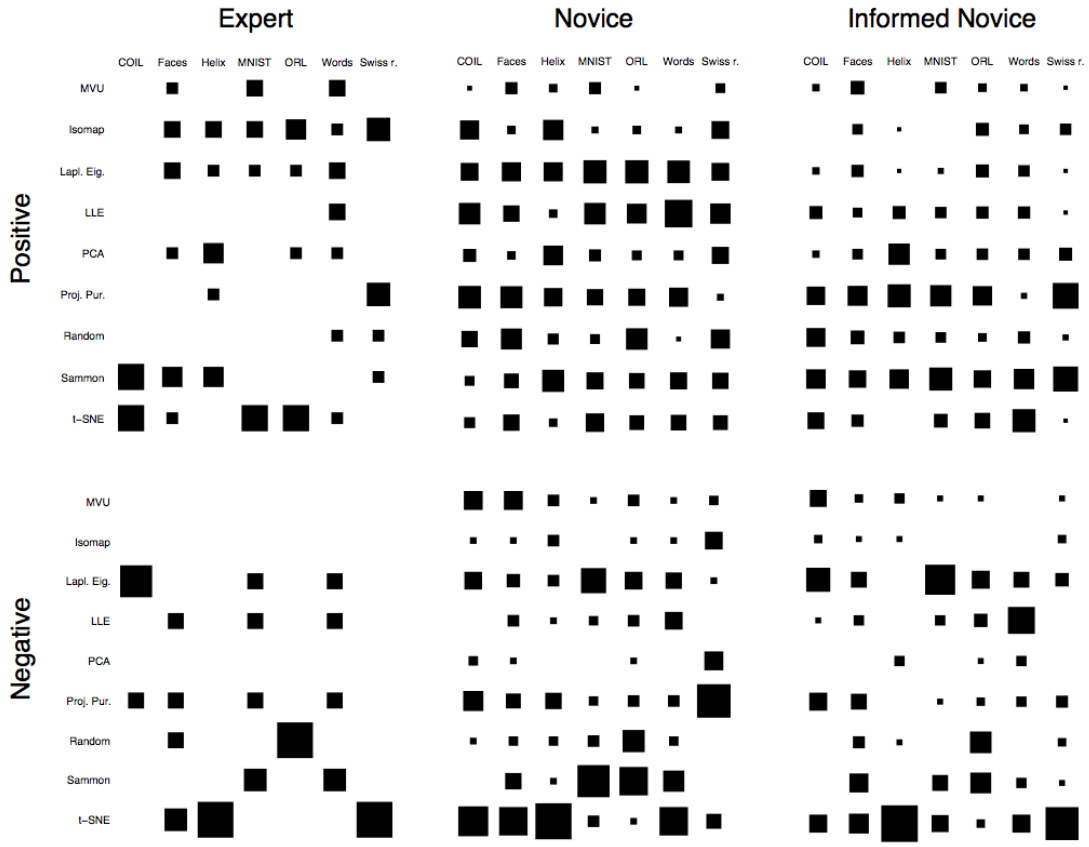

Figure 4.3: Human responses to the embeddings in experiment 1. Positive responses in the first row, negative in the second row. Experts (left), novices (center) and informed novices (right) by column.

represent positive responses and the bottom negative, so if subjects are in disagreement about a stimulus, there will be a large box in its corresponding location in both rows. The diagrams reveal that informed novices exploit dataset knowledge in specific instances to differ significantly from uninformed novices. For example, the t-SNE embedding of the Swiss roll dataset (a relatively clustered embedding) is rated much more negatively by novices when they know that the data are gradual. Experts tend to rate t-SNE positively or negatively depending on the dataset and show a relatively consistent rating for Isomap. Informed novices consistently rated Sammon mapping and projection pursuit positively while generally rating manifold learners such as Isomap and LLE negatively. Uninformed novices are all over the map with the exception of (like all other subjects) rating MVU as not notable in either a positive or negative sense.

Table 4.1: Correlation coefficients between human responses and dataset characteristics. Text in bold if $p < .0036$ after Bonferroni correction for $n = 14$ comparisons per subject group and $\alpha = .05$.

ρ	Lilliefors	Skewness	Kurtosis	Variance	Visual Span	Clusteredness	Trustworthiness
Expert Positive	.26	-.01	-.19	.34	.17	.22	**.41**
Expert Negative	-.08	.17	.19	-.14	-.17	.08	-.08
Novice Positive	.07	-.03	**.50**	-.18	-.29	.01	-.08
Novice Negative	.00	.17	-.10	.22	.10	-.03	.24
Informed Novice Positive	-.02	-.16	-.10	-.11	**.44**	**-.45**	-.09
Informed Novice Negative	.03	.31	.19	.10	-.19	.20	.19

Table 4.1 shows correlation coefficients between the six embedding characteristics and the evaluations by the three human groups. We also present the correlation between the evaluations and the trustworthiness, which gives an indication of the quality of the embedding (in terms of local neighborhood preservations). The significant correlations are in bold type, after a Bonferroni correction for multiple comparisons (14 comparisons per subject group, $\alpha = .05$). Notably, expert positive ratings are the only ratings that correlate significantly and in the correct direction with trustworthiness. Another correlation that stands out is visual span: it appears to play a substantial role in informed novice ratings (they apparently surmise an embedding should fill up the available space), whereas it plays little role in expert ratings.

4.5.2 Experiment 2

For the second experiment, the consistency measure κ for experts is 0.35, for uninformed novices is -0.32, and for informed novices is -0.26. The results of the second experiment thus reveal a similar trend: experts have fair agreement on the quality of embeddings, whereas novices give ratings have poor agreement. In Figure 4.4, we present the raw ratings obtained in experiment 2 as Hinton diagrams. The raw ratings reveal that, whereas experts selectively rate more clustered or more continuous embeddings

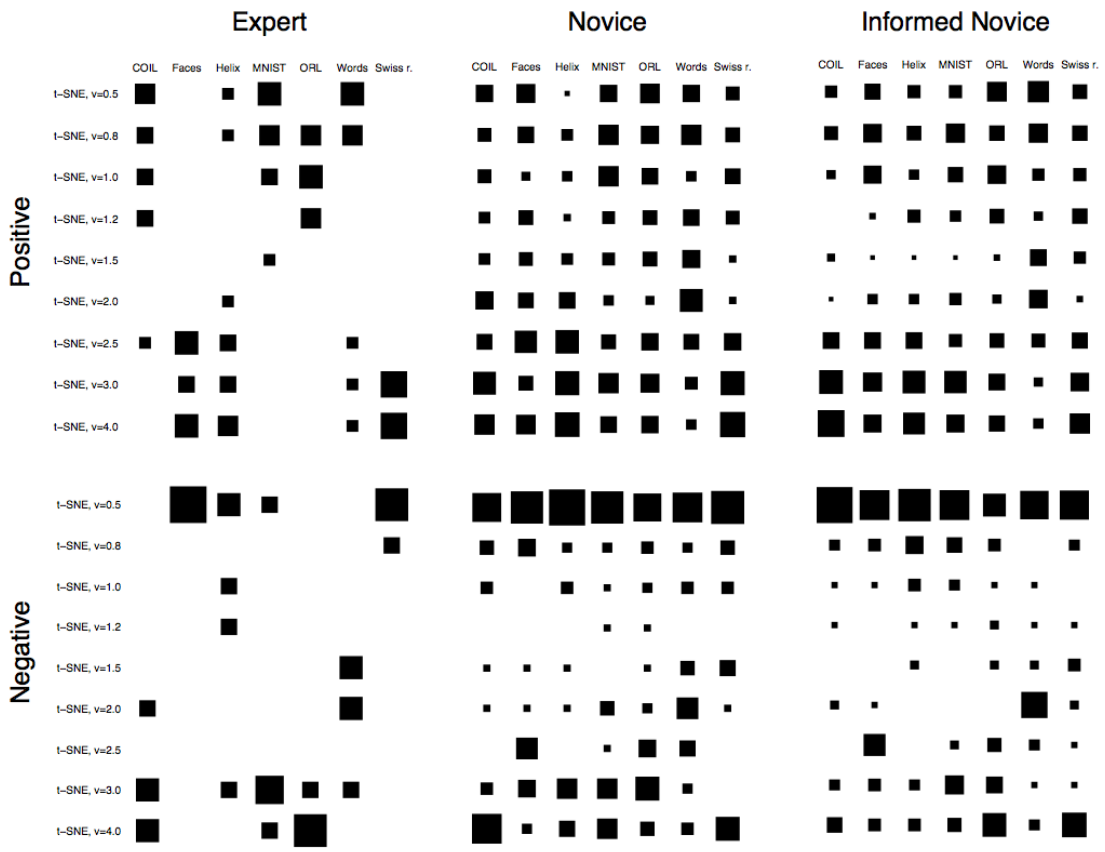

Figure 4.4: Human responses to the embeddings in experiment 2. Positive responses in the first row, negative in the second row. Experts (left), novices (center) and informed novices (right) by column.

positively depending on the dataset, novices overwhelmingly rate the more clustered embeddings as negative. On the other hand, for positive ratings the novices tend to choose embeddings at either end of the spectrum while avoiding the moderate values of v. Moderate values of v might be avoided since subjects want to classify displays closest to the prototypical clustered or graded display [55].

Table 4.2 shows correlation coefficients between all embedding characteristics and the human groups. The correlations show that experts ratings do not strongly correlate with any of the characteristics (including trustworthiness), but both groups of novices show a correlation between negative ratings and those stimuli with low kurtosis and high clusteredness (as expected given Figure 4.4).

Table 4.2: Correlation coefficients between human responses and dataset characteristics. Text in bold if $p < .0036$ after Bonferroni correction for $n = 14$ comparisons per subject group and $\alpha = .05$.

ρ	Lilliefors	Skewness	Kurtosis	Variance	Visual Span	Clusteredness	Trustworthiness
Expert Positive	.10	.23	-.05	.11	-.03	.21	.07
Expert Negative	-.10	-.10	-.27	.09	-.04	.13	-.01
Novice Positive	.18	.05	-.04	-.16	.26	.02	0.17
Novice Negative	.11	-.05	**-.51**	**.39**	-.32	**.49**	.09
Informed Novice Positive	.14	.08	.00	-.13	**.40**	-.13	.20
Informed Novice Negative	.17	-.04	**-.49**	.29	-.23	**.47**	.07

4.6 Discussion

In both experiments, experts show themselves to be more consistent than chance and much more consistent than novices in either condition. This is reassuring, and indicates that the idea of having experts evaluating embeddings is not flawed to begin with. In the first experiment, novice subjects actually get less consistent with each other if they are informed. While this seems troubling at first, it actually makes some sense after closer consideration. Comparing the Hinton diagrams between novices and informed novices, one can plainly see that informed novices are converging on a smaller selection of techniques for both positive and negative ratings. The issue for the informed novices, however, is that they are not sure whether these stimuli should be rated as positive or negative. As a result, there is often energy for the same cell in both diagrams. Since the base rate of positive and negative ratings is low compared to the neutral ratings, the κ consistency measure interprets this as substantial disagreement and thus the negative numbers. Importantly, the informed novice κ is further from chance level than the novice κ. In the t-SNE experiment, uninformed novices actually differ more from chance but the effect is about half the size. Experts remain consistent in experiment 2.

Expert ratings are laudable in that they correlate in the correct direction with trustworthiness and have a context-dependent appreciation of clusteredness. Both novice

groups rate clusteredness negatively regardless of context and are more influenced by elementary characteristics such as visual span. The substantial difference in strategy between novices and experts indicates that novices could really benefit from training on the task of evaluating embeddings (unlike in evaluating results from topic modeling, image segmentation, or object recognition).

4.7 Conclusion

With respect to the first aim of our study (determining whether humans are consistent in rating embeddings), we conclude that dimensionality reduction experts are indeed reasonably consistent judges of embedding quality. This supports the practice of soliciting expert judgment for embedding evaluations, as nowadays is common in the literature on dimensionality reduction. However, we also conclude that novices are very inconsistent with one another in terms of their rating of an embedding, even when they have detailed information on the dataset the embedding is visualizing. In fact, novices even correlate negatively with a measure of embedding quality.

With respect to the second aim of our study (determining what types of structure humans appreciate in embeddings), we conclude that humans do not appear to have overwhelmingly strong *a priori* preferences, although novices appear to appreciate gradual embeddings that span a large portion of the space. Experts can adapt their preference for gradual vs. clustered depending on the dataset.

Overall, our results discourage free-form solicitation of human computation approaches á la [16] and [34] to the evaluation of dimensionality reduction techniques. Moreover, the novices' lack of consistency lends worry to the prospect of naïve dimensionality reduction-based analysis. Most data analysts seeking to apply dimensionality reduction are not very familiar with the field. As a result, they are likely to be susceptible to the favorable visualizations presented in many dimensionality reduction papers. To ensure that dimensionality reduction techniques are applied wisely, authors should strive to explicate the dataset characteristics that favor their algorithms (e.g., t-SNE is useful if the data is expected to have cluster structure, Isomap if the data lie on a convex manifold). Authors could also cover usage scenarios appropriate to their algorithm

(e.g., if a researcher is interested only in visualizing points that are most different then PCA would suffice and other techniques would be overkill), including guidelines for interpreting the relationship between the high and low dimensional spaces (sometimes this relationship will be very clear, as in PCA; other times, as in MVU, there is not a clear relationship). In addition, data analysts should be encouraged to use sanity checks such as the trustworthiness measure in order to prevent them from basing analysis on interesting, but flawed, embeddings.

4.8 Acknowledgments

This work is coauthored with Laurens van der Maaten and Virginia R. de Sa, and is currently being prepared for submission. The authors would like to thank Cindy Zhang for her valuable assistance on this project. Laurens van der Maaten is supported by the Netherlands Organization for Scientific Research (NWO; grant 680.50.0908), and by the EU-FP7 NoE on Social Signal Processing (SSPNet).

5 Learning Cluster Analysis through Experience

5.1 Abstract

The field of machine learning is constantly developing useful new techniques for data analysis, but they are often ignored by researchers outside the field due to unfamiliarity and the difficulty of keeping up with a large body of work. We propose a methodology for training researchers how algorithms work through experience, such that they gain an implicit, rather than explicit, understanding of their function. This implicit knowledge might still lead to good data analysis decisions. We find that undergraduate subjects are generally able to learn machine learning concepts through experience, though they have only partial success in applying them.

5.2 Introduction

Machine learning has a PR problem. The field has developed many techniques that cluster, classify, or reduce the dimensionality of data, and most techniques could be profitably applied to scientific data sets. Researchers that are not machine learning experts face a daunting question, however—which techniques should I use to analyze my data? Authors proposing a new technique will focus on its strengths over its weaknesses, and most researchers do not want to spend a year reading math papers and becoming a machine learning expert in order to best analyze their data. So too often the analysis technique used is the convenient one (freely available online or as part of a soft-

ware package), or the traditional one. Researchers miss out on the advances in machine learning, and the machine learning field is not as valuable as it could be to the broader scientific community.

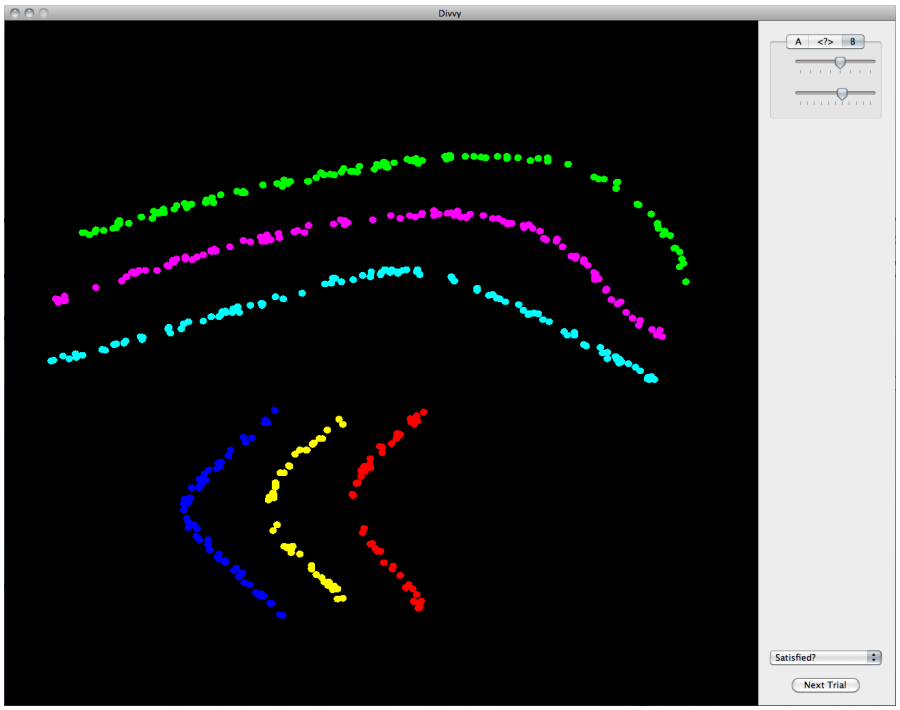

Figure 5.1: The Divvy UI used in this experiment. The tabs at the top right select method A (*k*-means) or B (single linkage), and the sliders below control the number of clusters and the relative weighting of the horizontal and vertical axes.

There are two fundamental problems: access and expertise. Access issues can be ameliorated by author-provided reference implementations of new techniques, the integration of techniques into data analysis platforms such as Matlab or R, or heroic compilation efforts such as Laurens van der Maaten's Dimensionality Reduction Toolbox [56]. This paper is concerned with the second problem, expertise. If a researcher wants to find the right technique for the job, but is unwilling to engage in the time-consuming process of learning the details of every technique, how can they be trained to apply the best one?

Baseball players have an excellent idea of how baseballs behave. A baseball's behavior is, of course, governed by the laws of physics and an explicit description of

that behavior might be quite complex when spin, deformation, wind and field texture are taken into account. Nevertheless, through extensive experience baseball players acquire an excellent pragmatic understanding of how baseballs behave, an understanding that one might guess is founded on an implicit learned model of baseball behavior rather than the explicit model a physicist would give. We believe that interactive experience with machine learning techniques can give rise to a similar sort of practical and implicit model of algorithm behavior, and that researchers can use such a model to make informed decisions during data analysis.

Providing the right kind of interactive experience requires a software tool that is both intuitive and extremely responsive. We are developing a tool called Divvy with the goal of fulfilling those requirements, and it is the experimental platform for this study. Divvy is described in detail in the next section.

In this paper we give subjects interactive experience with two clustering techniques, *k*-means and single linkage [57], labeled simply as method A and method B and without any explicit instruction as to their differences. We find that after training almost every subject learns a few relevant facts about A or B or their parameters, and that some subjects appear to be able to apply this knowledge to new analysis contexts.

5.3 Divvy

Data analysis is often a laborious process. A researcher collects data, and then loads it into a software package such as Matlab or R. To apply an algorithm to his or her data, the researcher has to write a command or fill out a dialog box and then wait for processing to finish. Finally, the researcher will use other commands to visualize the algorithm's output. To change a parameter and see the impact it has, this process must be repeated. Clever researchers might write a script that runs a set of different parameters and visualizations, and then go out for a coffee and come back to see if the whole endeavor bore any fruit.

This is a tenuous kind of interaction. A baseball, by virtue of being in the real world, provides instantaneous feedback to those interacting with it. In the above process the algorithm does not, and the goal of the Divvy project is to close that gap and provide

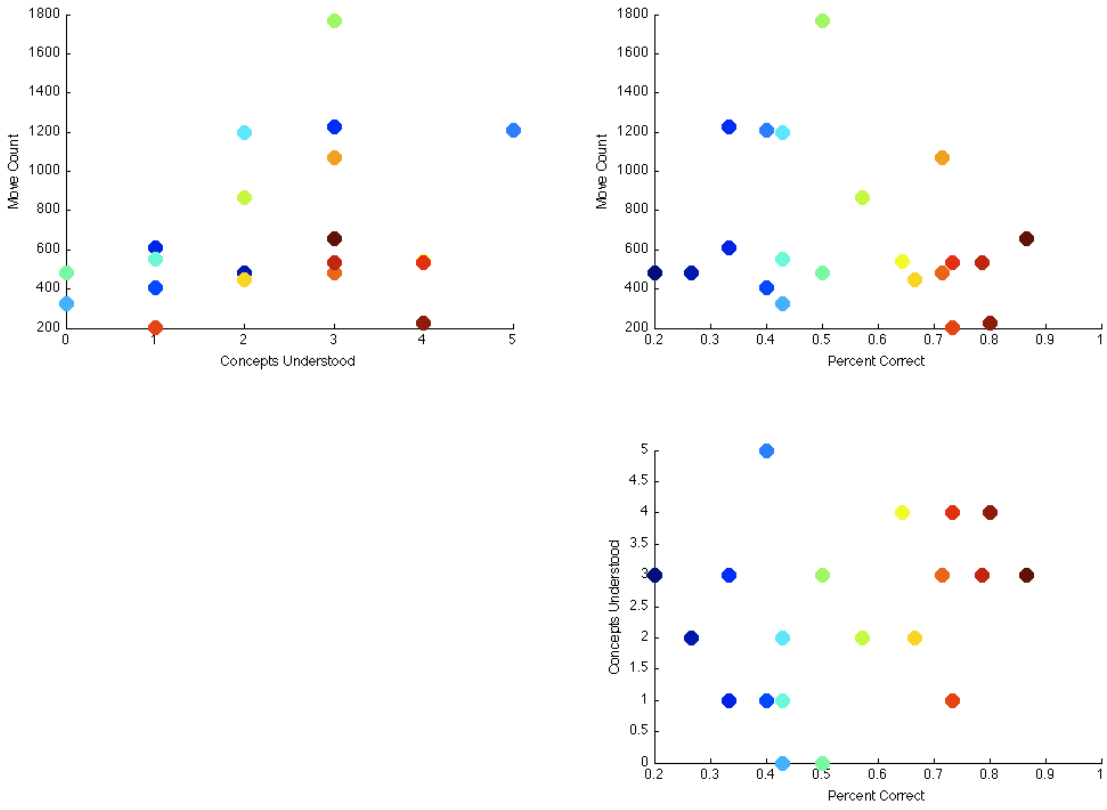

Figure 5.2: Scatter plots of the three main variables. The points are colored from dark blue to dark red based on percent correct.

an interface where visualization happens instantaneously and researchers can tweak parameters and see their effect in real time. In a way, Divvy is providing the human analog to active learning [58], where learning algorithms choose which training samples to get based on what they predict to be the most informative. Divvy is similar in spirit to a tool called GGobi [59], which brings cutting edge methods in high-dimensional data visualization to a user friendly graphical interface but without a strong machine learning component.

Divvy achieves this feat through a combination of parallel computing and UI design. Many personal computers (and all Macs) ship with multi-core processors (CPUs), as well as graphics processors (GPUs) that can be used for general purpose computation. High performance computing research has so far focused on how these hardware resources can make very large problems tractable [60]. With Divvy, we are using these

technologies to make medium problems very fast—fast enough to feel real time, and to invite the exploratory interaction that we believe leads to learning. Our UI design puts a focus on visualization, using OpenGL to render up to millions of points quickly in a large swath of the application window. Algorithm parameters are controlled with standard UI elements (such as sliders or check boxes) rather than having to be specified with time-consuming key strokes. See Figure 5.1 for the simplified version of the UI we used in this experiment.

5.4 Methods

We recruited 22 undergraduate subjects for this experiment. Subjects received course credit for participation. None of the subjects were familiar with cluster analysis. One subject was excluded from the study after he indicated at the end during the interview segment that he must not have understood the instructions, and so we analyzed the data from a grand total of 21 subjects.

Each subject performed 36 trials, which were split into two 18 trial blocks, a training block and a testing block. In both blocks, subjects use the sliders to change the number of clusters, k, and the relative weighting of the horizontal and vertical axes in order to best group the points in each stimulus (one stimulus per trial) and then indicate their satisfaction with the result. In the training block, subjects use both A and B (k-means and single linkage, respectively) to group the points, and are required to arrive at a solution for each method. In the testing block, neither A nor B are initially selected and subjects must choose which method they want to use for that trial. Once the choice is made they cannot switch. We divided subjects into two groups of 10 and 11. One group's training set was the other's testing set, and vice versa. At the end of the two blocks, subjects filled out an interview form that assessed their knowledge. There were eight interview questions were as follows:

1. What did you feel like method A was doing?

2. What organizations of circles was method A good for grouping?

3. What did you feel like method B was doing?

4. What organizations of circles was method B good for grouping?

5. Did you have a preference between A and B?

6. Why or why not?

7. What did the first (top) slider do?

8. What did the second (bottom) slider do?

We instructed subjects to do their best to learn what A, B and the sliders were doing in the first half of the experiment, as they would need to use that knowledge during the second half. We also made clear that not every stimulus could be ideally grouped with both A and B, and that if they did not like a solution they could just indicate dissatisfaction using the dropdown above the next trial button. We provided two helper images along with the instructions. One showed a well-separated mixture of Gaussians where each Gaussian had its own color. This was held up as a positive example. The second showed two circular groups split in half with color, which was considered a negative example. Beyond these very simple prompts (show in Figure 5.3) we did not bias the subjects as to what a group should be.

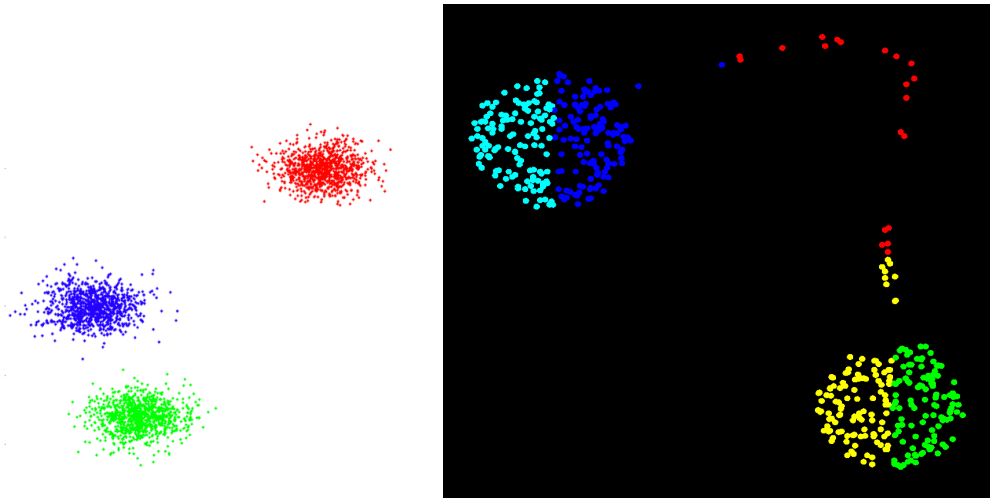

Figure 5.3: Sample images to give subjects basic guidance on good groups (top) versus bad groups (bottom).

The 36 stimuli fall into three categories, those where A is most effective (14), those where B is most effective (15), and those where A and B are similarly effective (7). We created all 36 stimuli by hand in order to ensure that the first two categories had sufficient membership. Stimuli ranged from collections of non-convex lines, rings and spirals to connected and disconnected blobs to uniform noise. While these are not real data, so to speak, they provide us with a solid foundation on which to train and judge our subjects that real data would not necessarily provide. Additionally, most meaningful real data are more than two dimensional, and while the Divvy project proposes to use dimensionality reduction techniques and multiple views to visualize such data, those techniques raise additional issues that are outside the scope of this experiment.

Divvy records every method and parameter combination subjects try over the course of the experiment, including their final grouping and satisfaction. We use these data in concert with interview responses to determine what subjects were able to learn from their experience. From the Divvy records we extract two variables per subject, the total number of different algorithm and parameter settings queried in the training period (the number of "moves"), and the percent of correct method choices in the testing period. From the interviews we extract one variable, the number of concepts learned and reported from a list of seven possible concepts. The seven possible concepts are as follows:

1. The first slider controls the number of colors (i.e. clusters).
2. The second slider controls the orientation of the boundary between clusters.
3. *k*-means works well on blobs of points (convex regions).
4. Single linkage works well on shapes like lines or rings (non-convex regions).
5. *k*-means can work when there is no space separating clusters.
6. Single linkage works best when there is lots of space between clusters.
7. *k*-means tends to divide the points into evenly sized groups, whereas single linkage can make large and small groups.

We measure correlations between these three variables with the hypothesis that they should all be positive, and report in detail the concepts learned on a per-subject basis. Finally, we compare subject satisfaction, ranging from 1 (Not Satisfied) to 7 (Satisfied), when using the correct method on a stimulus versus the incorrect method. This test indicates whether subjects recognize when the partitions are not ideal.

5.5 Results

In Table 5.1 we summarize the contents of each subject's interview, using the seven concepts described above. Nineteen of the subjects learned at least one concept, and 15 of the subjects learned at least one concept excluding the simplest one (the function of the first slider). On average subjects learned 2.4 concepts over the course of the study.

The number of concepts learned correlates positively with both percent correct ($\rho = .29$, $p < .10$) and number of moves ($\rho = .34$, $p < .07$). Surprisingly, percent correct and number of moves are negatively correlated ($\rho = -.22$, $p < .84$). In Figure 5.2 we show scatter plots of the pairwise comparisons between these variables.

For stimuli with a correct answer where the subject used the correct method, we had 470 satisfaction ratings with $\mu = 5.88, std = 1.37$. For stimuli with a correct answer where the subject used the incorrect method, we had 444 satisfaction ratings with $\mu = 4,94, std = 1.77$. A t-test indicated a significant $p < .01$ effect of correct versus incorrect method on satisfaction.

5.6 Discussion

While all but two subjects were able to learn something about cluster analysis through their experience, and over half learned three concepts or more, some subjects appeared to have difficulty using that knowledge to make good data analysis decisions. In addition, though subjects might have explored quite a bit during the training phase (an activity that showed a trending correlation with concept learning) they did not parlay that experience into better data analysis decisions. So while we are pleased that subjects

Table 5.1: A summary of the concepts subjects learned. Subjects in bold chose the correct method for over 70% of stimuli in the test block.

Subjects	1st Slider	2nd Slider	*k*-means Blobs	Single Linkage Shapes	*k*-means No Separation	Single Linkage Separation	*k* Even vs SL Uneven	Sum
1								0
2								0
3	✓				✓	✓	✓	4
4	✓	✓		✓				3
5	✓				✓			2
6	✓	✓		✓				**3**
7	✓		✓	✓				**3**
8	✓			✓			✓	**3**
9	✓							1
10	✓						✓	2
11							✓	1
12	✓		✓	✓				3
13	✓		✓	✓				**3**
14	✓							**1**
15						✓	✓	2
16	✓				✓	✓	✓	**4**
17	✓							1
18	✓		✓	✓	✓	✓		5
19	✓		✓	✓			✓	**4**
20	✓	✓						2
21	✓				✓		✓	3
Sum	17	3	5	9	5	4	7	

learned concepts, we are curious why that knowledge did not translate into better performance.

The subjects were overall less satisfied when using the incorrect method, which indicates that there was not substantial confusion in evaluating the partitions.

We performed a quick follow up study with three graduate students (unfamiliar

with cluster analysis) in an attempt to determine whether motivation or understanding of the experiment might have been a factor in performance.[1] The graduate student data did not seem to differ substantially in character; they all learned at least two concepts, but only one was able to consistently choose the appropriate method for the test stimuli.

The process of crystallizing the implicit knowledge gained during the experiment in the interview might help subjects make better decisions. To test this, the experiment could be modified to place the interview between the training and test blocks. If this results in better concept application, it would indicate that having to articulate knowledge assists application, and that the subjects are in a sense still learning when they fill out the interview.

Beyond fiddling with the existing experiment, we would like to compare our results with simply showing subjects a set of partitions and their associated methods and parameter values. This approach would be similar to traditional machine learning approaches where the training data are fixed (as opposed to the active learning paradigm mentioned earlier). It would also correspond to writing a script to run through a set of parameter settings and visualizations while one goes out for coffee, and then interpreting when one returns.

These results provide compelling evidence that undergraduate subjects can learn useful concepts about machine learning algorithms just by interacting with them. This leads one to suspect that the target population for this work, practicing researchers, will be able to do so as well. Unfortunately subjects do not reliably apply these concepts when tested, and additional study is required to determine why this is, and how to better support the effective application of algorithm knowledge.

5.7 Acknowledgments

This work is funded by NSF Grant #SES-0963071 (PI Virginia de Sa, who is also a coauthor), and is currently being prepared for submission. Thanks to Cindy Zhang for valuable code contributions.

[1] Anecdotally, some of the undergraduates seemed to just click through the second half of the experiment without making much effort to find good groupings.

6 Pairwise Distance Matrix Calculation: A Comparative Study

6.1 Abstract

Many data analysis techniques in bioinformatics and machine learning require one to calculate a pairwise distance matrix as an initial step. This order n-squared procedure is time consuming, but readily accelerated through parallel processing. In this paper we propose two novel GPU implementations of pairwise distance matrix calculation that both improve performance and flexibility compared to existing GPU implementations across two GPU architectures. In addition, we describe a parallel CPU algorithm that is a better basis for comparison to GPU approaches than existing single-threaded examples, and performs up to two orders of magnitude faster than previous efforts. Our results represent both a substantial algorithmic innovation, and a useful guide for practitioners looking to optimize a foundational part of the data analysis process.

6.2 Introduction

Pairwise distance matrices underlie many popular data analysis techniques, such as linkage-based clustering, spectral clustering, nearest neighbor-based dimensionality reduction, and multiple sequence alignment. Calculating these matrices is an order n-squared problem, and thus can represent a substantial amount of the computation time for each algorithm. Since the distance between any two points is independent from

the distance between any other two points[1], the problem is naturally parallelizable, and stands to gain substantial performance benefits from recent computer hardware trends, such as multi-core central processing units (CPUs) and general purpose graphics processing unit (GPU) computation.

GPUs, along with other high-performance computing platforms such as the Cell microprocessor, have parallel architectures with many processing cores, allowing them to perform many computations simultaneously. While traditionally CPUs have been single core, they too now incorporate multiple cores. Nevertheless, GPUs maintain a substantial advantage in core count, with cores numbering in the hundreds compared to CPUs, which typically have fewer than ten. On problems that are readily parallelized, GPUs can outperform CPUs operating at a much higher frequency. Hardware vendors are encouraging GPU computation by developing general purpose GPU coding libraries such as Nvidia's CUDA [61] and the Khronos Group's OpenCL [62].

In this paper we develop two new GPU algorithms for Euclidean pairwise distance matrix computation that provide improvements in execution time, memory footprint and memory access patterns compared to previous efforts [63, 1]. We also illustrate the tradeoffs encountered in GPU algorithm design that have relevance to any GPU computation practitioner. On the CPU side, we describe an optimized parallel implementation of distance matrix computation that performs much better than a naïve approach. Finally, we explore the performance characteristics of these algorithms across a representative sample of hardware platforms, in order to give readers a useful set of performance expectations for their own endeavors.

6.2.1 Performance Concerns in Parallelization

The performance improvements one can acheive by implementing an algorithm on a parallel processor depend heavily on the structure of that algorithm. Algorithms that contain a low degree of intra-data dependency, or that consist of a large number of subroutines that may be executed in any order (e.g., finding the square root of each number in an array) are ripe for parallelization—such problems are sometimes dubbed "embarrassingly parallel." On GPUs, where threads cannot easily be synchronized glob-

[1] Barring inferences one can make using, e.g., the triangle inequality.

ally within a function call, this level of parallelism is highly desirable. On the other hand, it is often the case that an algorithm must complete a series of lengthy computations in a very specific order, or access many different pieces of data in a complex way. This kind of algorithm suffers two issues with parallel implementation. First, if some of the sub-problems in the algorithm must receive a result from some prior computation before running, then they must sit idle until that prior computation completes. If this takes a long time, or occurs frequently, then much of the run time will be wasted and little will be gained with a parallel implementation. Second, most parallel architectures suffer from high data latency—that is, each processing unit owns only a very small piece of fast-access memory, while most of the data resides in much larger, slow-access memory. Code with frequent, erratic data access patterns will often have to halt while waiting for the hardware to fetch data from slow memory, hampering performance. Because of this memory limitation, one must design code that accesses data sparingly and efficiently, performing as much computation between memory reads or writes as is possible. The ratio of compute operations (such as add or multiply) to memory operations (such a read or write) is called computational intensity, and maximizing this ratio is a core component parallel design. As we will see computational intensity plays a critical role in our improvements on existing GPU algorithms for computing pairwise distance matrices.

6.2.2 Previous Work

As mentioned above, memory access latency is the primary bottleneck in GPU computation. Groups of processing cores on the GPU, called Streaming Multiprocessors (SMs), incur long delays while reading or writing to global memory. One way to circumvent this issue involves the use of relatively small (16KB on Tesla, 48KB on Fermi) caches that the SMs can access quickly, which we will refer to as local memory. By loading all of the data relevant to a small subset of the problem into local memory, the SM can avoid making slow, repetitive reads from global memory. As long as the data that have been loaded into local memory can be reused while computing the sub-problem, this pre-loading scheme will give a performance boost.

Another way to improve performance is to hide memory latency by launching hundreds of threads within a work group. While only a small number of threads can be

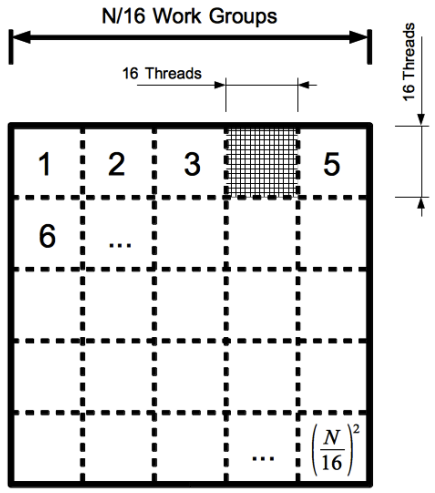

Figure 6.1: A diagram of work group and thread allocation in Chang et al.'s algorithm. Each block in the pairwise distance matrix is represented by a work group, with each work group containing 256 (16 by 16) threads. Each thread computes a single element of the result matrix.

active on an SM at any given time, the GPU thread scheduler is able to rapidly swap threads into and out of the processing cores. This makes it possible for some threads to do computation while other threads are waiting for their memory requests to complete. Once the SM fetches the necessary data, the idle threads begin executing again while the others wait for their own memory transactions. In this way, the processing cores are kept busy, leading to good performance. Thread swapping is performed automatically by the GPU hardware, so the programmer must only ensure that there are enough threads in the work group to take up the time in between memory accesses.

Chang et al.[63] devised a GPU algorithm that makes use of local memory and a high thread count to speed up distance matrix computation (other similar efforts include [1, 64, 65], though Wirawan et al. are focused on multiple sequence alignment). Their method divides the result matrix into a grid of blocks, each consisting of a 16 by 16 array of pairwise distances, and assigns one work group to compute each of those blocks (see Figure 6.1). To compute each 16 by 16 block, the work group needs access to two groups of 16 data vectors.[2] The threads in each work group begin by loading two

[2]For blocks on the diagonal only one group is required.

such groups into local memory. One of the arrays is loaded in row major format, while the other is loaded column major, to allow the hardware to perform very fast coalesced memory accesses.[3] Then, each of the 256 threads, having a unique i, j coordinate within the block, computes the distance between a single pair of vectors according to its coordinates, i.e., each thread computes the distance between vector i in group 1 and vector j in group 2. Each thread then writes that distance to the appropriate location in global memory.

Chang et al. describe an efficient approach, as it utilizes the speed of local memory and allows for coalesced memory access. However, there are a few aspects that could be improved upon. First, it is evident that their algorithm is performing a large amount of redundant global memory access. Consider that the i^{th} sample is involved in computing every element of the i^{th} row of the distance matrix. Therefore, when computing a block of rows in the distance matrix, one should be able to reuse one block of data vectors while iterating over the columns. This saves $(N/b) - 1$ loads from global memory, where N is the number of data vectors and b is the number of columns each work group computes between memory loads. Second, their algorithm computes an entire N by N result matrix, half of which is redundant, because any pairwise distance matrix is symmetric. By storing only the upper-triangular region of the result matrix, we halve the memory footprint, and can therefore tackle larger problem sizes for a given global memory capacity. Third, due to its block size the algorithm is unable to realize performance gains when reducing dimensionality below 16.

6.3 Methods

6.3.1 Hardware

We test our algorithms across a broad range of hardware, summarized in Table 6.1. The three GPUs we test represent a good sample of the spectrum of GPU hardware: the Tesla C1060 is a dedicated GPU compute card[4] with a large amount of

[3] Coalesced memory operations are rapid blocked transfers that can take place when many threads need to access a contiguous region of memory simultaneously. For more information, see [61].

[4] It does not drive a display and cannot be used for games.

Table 6.1: Summary specifications of hardware used in the study.

CPU/GPU	Local Memory	Global Memory	Clock	Cores
Tesla C1060	16KB/SM	4GB	602MHz	240
GeForce GTX460 (Fermi)	48KB/SM	1GB	675MHz	336
GeForce 8600M GT	16KB/SM	512MB	475MHz	32
2 x Intel X5650	12MB L3/processor	12GB	2.66GHz	2 x 6
Intel T9300	6MB L2	4GB	2.53GHz	2

RAM, the GTX460 is a mid-range consumer GPU, and the 8600M GT is a laptop GPU. Similarly, we test our CPU algorithm on dual Intel X5650s, six core workstation-class processors, and an Intel T9300, a standard dual core laptop processor. In general, CPUs will have a substantial clock speed advantage (and at least on the desktop a substiantial maximum allocatable RAM advantage) while the GPUs will have many more cores. We also show the CPU times from [1][5] in the results as a point of comparison. The CPU they used was a 3.0GHz Pentium D with 2GB of RAM, but since there were multiple versions of the 3.0GHz Pentium D, we cannot be sure of its model.

6.3.2 GPU Distance Matrix Computation

In this section we describe two GPU algorithms, A1 and A2, that improve upon Chang et al.'s (which we will call A0). The first algorithm reuses memory across multiple blocks and efficiently allocates memory for dimensions below 16, but suffers a reduction in thread occupancy at high dimensionality. The second algorithm reuses memory without a reduction in thread occupancy, but is not as efficient at low dimensionalities. We implement A0, A1 and A2 using CUDA. We tested OpenCL implementations, including on AMD's competing Radeon graphics cards, but found OpenCL to be slower; we focus on CUDA for the remainder of the paper.

Both algorithms follow the same flow, shown in Figure 6.2. A work group is assigned a row of blocks (at zero iterations all rows are unassigned) and loads the group of data vectors that are involved in computing that row into local memory. This group of vectors will be kept in local memory until the work group has computed the entire row of blocks. Starting with the rightmost sub-section of the distance matrix, the work group

[5]Using their own algorithm, see Table 1 in their paper.

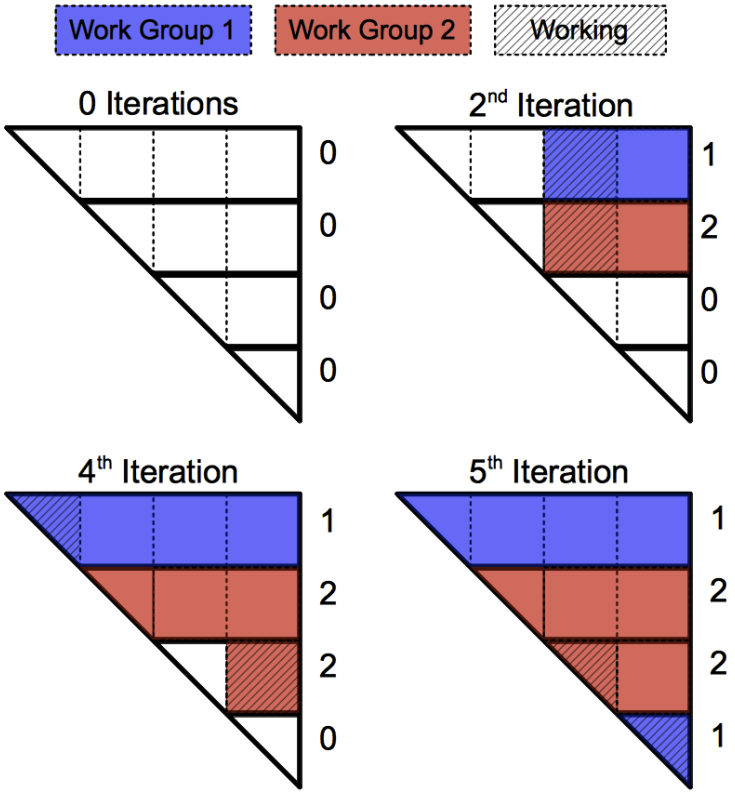

Figure 6.2: A1 & A2 Flow diagram.

loads one more group of vectors into local memory, and then computes the pairwise distances between each vector in local memory, filling in every element of the sub-section. Then the thread block moves to the next sub-section and repeats the process (see second iteration), continuing until the entire row is complete (work group 1 has completed its row by the fourth iteration). The work group then finds the next available row and begins computing.

A1: Optimizing for Low Dimensionality

In A1, each thread represents one data vector, and is responsible for computing the distance between that vector and every other vector in the data set with a greater index.[6] A work group represents a row of blocks in the distance matrix, and computes b columns of the distance matrix at a time, so there are b threads in a work group. At any given time, there are two sets of b data vectors stored in local memory. Because there is a limited amount of local memory available to each work group, we set b to the maximum possible value for each dimensionality, while staying within local memory bounds. This both optimizes local memory usage and maximizes the number of threads that occupy each work group. The threads are arranged in a one-dimensional list, each thread having a unique index ranging from 0 to b, rather than a two-dimensional grid, as is the case in A0 and A2. This helps us exercise greater control over coalesced memory transfers, which behave more erratically for two-dimensional thread grids. Further, as dimensionality decreases we can increase the block size and thus increase memory reuse.

This approach faces a significant challenge—each row of blocks in the upper triangular distance matrix is of a different size. If we simply create a number of work groups equal to the number of block rows the workload will be imbalanced between work groups (one work group will get N/b blocks to compute while another will get only one, and everything in between). To avoid this issue, we create a number of work groups approximately equal to the number of SMs and use an integer stored in global memory to assign block rows. It is initialized to zero. When a work group is free to

[6] A thread need not calculate the distance to a vector with a lower index because it will already have been calculated by the other vector's thread.

start computing a new row, a single thread in that work group reads and increments the counter using atomic operations.[7] In this way, we implicitly load balance the work groups. Those assigned rows with shorter computation times will finish more rows, while those with assigned rows with longer computation times will finish fewer.

The algorithm has as input an $M \times N$ data matrix (dimensions by samples) and a number of blocks, b, and it outputs a upper triangular distance matrix, D. Each work group repeats the following steps until all rows are complete:

1. Read the next available row index from global memory and increment it as an atomic operation.
2. Load all samples in the current row block, i, into local memory.
3. Loop through every block of columns, $j \neq i$, in the result matrix.
   (a) For each pair of samples, s_k and s_l, in blocks i and j, compute their distance and write the result to global memory at $D_{k,l}$.
4. For each pair of samples, s_k and s_l, where $l > k$ in block i, compute their distance and write the result to global memory at $D_{k,l}$. (This step fills in the diagonal block of D.)

In the above, k and l are assumed to be the global sample coordinates (not the within-block coordinates).

A2: Optimizing Memory and Hiding Latency

In A2, we replicate the work group structure of A0, assigning 256 threads to compute the 256 elements of each 16 by 16 block of the distance matrix at once. Unlike A0, which stores in local memory only 16 elements of 32 $(16+16)$ data vectors, A2 stores two sets of 16 entire data vectors (that is, two arrays of size $16*D$). If D is greater than 16, A0 must initialize a series of separate blocked loads from global memory when

[7] When a thread accesses a variable atomically, other threads must wait until that thread finishes using the variable before they can access it, and only one thread gets control over the variable at a time. This prevents multiple work groups from simultaneously reading or incrementing the counter, which could result in two work groups receiving the same row index, or in some rows being skipped.

computing each block of the distance matrix. In A2, those loads are grouped into one big transfer, in which all the information needed to compute the block is fetched at once. This reduces memory access overhead, but it is in fact required if the algorithm is to reuse memory efficiently. If we wish to load only one group of vectors from global memory before computing each block of one row of the distance matrix (instead of loading two groups per block, as in A0), we keep one group of vectors in local memory throughout the duration of the row calculation. Then, during each block calculation, we assign one thread to compute each of the 256 pairwise distances defined by the 16 vectors in the persistent group and the 16 vectors in the transient group. A2 always has 256 threads per work group and is therefore able to hide memory latency quite well. A2 uses the same load balancing technique as A1 to determine which work groups are assigned which rows of blocks.

6.3.3 CPU Distance Matrix Computation

For the purposes of this problem, there are three key distinctions between the CPU and the GPU. First, the CPU has an established high performance matrix/vector operation library, the Basic Linear Algebra Subroutines [66] (BLAS), which we use for low level arithmetic.[8] Second, global synchronization of all CPU threads is possible (whereas GPU threads can only synchronize within a thread group). Third, we do not explicitly manage CPU caches, and therefore all reads and writes are to shared memory.

In previous studies touting GPU-based distance matrix calculation, authors report performance improvements of over two orders of magnitude compared to CPU implementations. We believe that these performance improvements are exaggerated due to unoptimized CPU implementations, and here we seek to provide a fair and comprehensive comparison. Our CPU implementation uses OpenMP [68] to manage threads across multiple cores, and the platform-optimized version of BLAS provided in Apple's Accelerate library.

The algorithm has as input an $M \times N$ data matrix (dimensions by samples) and a number of blocks, b, and it outputs a upper triangular distance matrix, D. The algorithm

[8] Similar libraries to BLAS (and LAPACK), such as MAGMA [67], are available in non-final forms for the GPU.

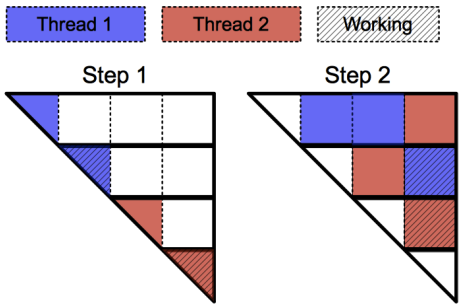

Figure 6.3: The CPU algorithm calculates the diagonal blocks of the pairwise distance matrix in Step 1. In Step 2, the algorithm dynamically assigns off-diagonal blocks to each thread to fill out the rest of the matrix.

follows a two-step process:

1. Loop through each $M \times N/b$ sample block of the data matrix.
   (a) Multiply the block with its transpose using BLAS.
   (b) For all samples i in the block, put the dot product of sample i, $s_i \cdot s_i$, in the diagonal term vector, d, at d_i.
   (c) For all samples i, j in the block where $j > i$, put $d_i + d_j - 2 * s_i \cdot s_j$, in the upper triangular distance matrix, D, at $D_{i,j}$. (See Step 1 in Figure 6.3.)
2. Loop through each non-identical pair of blocks from the data matrix.
   (a) Multiply the blocks together (transposing one) using BLAS.
   (b) For all samples i, j in the product, put $d_i + d_j - 2 * s_i \cdot s_j$, in the upper triangular distance matrix, D, at $D_{i,j}$. (See Step 2 in Figure 6.3.)

In the above, i and j are assumed to be the global sample coordinates (not the within-block coordinates). Since each iteration of both loops is completely independent, we use OpenMP with guided scheduling to farm out iterations to the number of simultaneously available threads in the system (24 for the X5650 machine, which can run two threads per core, and 2 for the T9300). Guided scheduling is a form of load balancing where threads are at first assigned many iterations of a loop, then upon completion assigned fewer and fewer. We perform a global thread sync after step one in order to

ensure that d is filled out in preparation for step two. Beyond the input/output memory, our algorithm only allocates $(b * b * numThreads + N) * 4$ bytes of working memory. For the X5650 machine we used a block size of $b = 256$ and for the T9300 machine we used $b = 128$.

In our experiments, we tested all three GPU distance algorithms, A0, A1 and A2, across our three pieces of GPU hardware. We tested the CPU algorithm above across our two pieces of CPU hardware.

6.4 Results

Our results are presented in detail in Table 6.2. Every time in the table is derived from ten runs, with the slowest and fastest run tossed out and the remaining eight averaged. A2 on the GTX460 is the fastest algorithm across every sample size and dimensionality, with the exception of the large sample sizes in four dimensions, where A1 on the C1060 prevails. The Fermi architecture employed by the GTX460 is better able to take advantage of coalesced reads and writes, and thus A2 outperforms A0 more consistently on that hardware. A0 is not run at four dimensions, since its design makes 16 dimensions the practical lower limit.[9] In addition, memory constraints prevent A0 from running at 16,384 samples on the GTX460, an issue that A1 and A2 solve by storing only the upper triangular distance matrix. Similarly, the 8600M is only able to run a subset of the tests due to memory constraints.

In Figure 6.4 we show the running times of A0, A1 and A2 on a log-scale plot for the C1060 and the GTX460. In Figure 6.5 we show speedup advantage of A1 and A2 over A0 for the C1060 and the GTX460.

CPU times on the X5650 are at most only a few tenths of a second slower than the fastest GPU times. This is in direct contrast to previous studies that report massive performance improvements moving to the GPU. For a more pedestrian CPU like the T9300 however, GPUs do offer an approximately 10x improvement. Still, the T9300 is

[9] To obtain a pairwise distance matrix from A0 with a 4D data matrix, one would pad the data matrix to 16D.

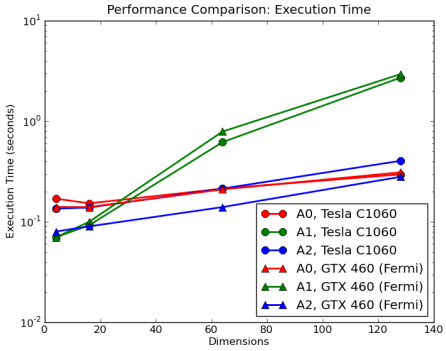

Figure 6.4: A log-scale runtime comparison of A0, A1 and A2 on the C1060 and GTX460.

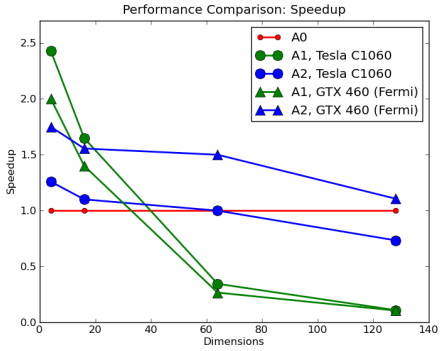

Figure 6.5: A speedup comparison of A1 and A2 to A0 on the C1060 and GTX460.

Table 6.2: Pairwise distance matrix computation time in seconds across hardware and algorithm implementations. In the last column we show the CPU results (which use a different algorithm) from [1] for comparison. Fastest times are in bold.

N	D	C1060			GTX460			8600M			X5650	T9300	CPU in
		A0	A1	A2	A0	A1	A2	A0	A1	A2			[1]
4K	4	x	0.011	0.013	x	0.014	**0.009**	x	0.051	0.065	0.036	0.234	x
8K	4	x	0.035	0.058	x	0.036	**0.034**	x	0.202	0.265	0.161	1.041	x
12K	4	x	**0.070**	0.135	x	0.073	0.076	x	x	x	0.343	2.645	x
16K	4	x	**0.121**	0.250	x	0.140	0.140	x	x	x	0.536	4.726	x
4K	16	0.018	0.014	0.014	0.016	0.011	**0.010**	0.110	0.080	0.074	0.044	0.283	1.550
8K	16	0.068	0.042	0.059	0.066	0.045	**0.040**	0.430	0.304	0.294	0.173	1.133	6.153
12K	16	0.153	0.093	0.139	0.141	0.101	**0.085**	x	x	x	0.351	2.856	13.869
16K	16	0.269	0.165	0.253	x	0.168	**0.148**	x	x	x	0.565	4.591	24.642
4K	64	0.025	0.073	0.024	0.025	0.093	**0.016**	0.203	1.078	0.277	0.057	0.385	5.172
8K	64	0.098	0.279	0.095	0.094	0.355	**0.063**	0.802	4.266	1.108	0.235	1.540	20.747
12K	64	0.214	0.620	0.214	0.210	0.788	**0.139**	x	x	x	0.430	3.756	46.746
16K	64	0.379	1.096	0.379	x	1.389	**0.249**	x	x	x	0.733	6.032	83.223
4K	128	0.035	0.306	0.045	0.036	0.330	**0.033**	x	x	x	0.068	0.483	10.237
8K	128	0.134	1.209	0.181	0.145	1.303	**0.126**	x	x	x	0.280	1.943	41.009
12K	128	0.296	2.715	0.404	0.311	2.943	**0.283**	x	x	x	0.495	4.625	92.386
16K	128	0.521	4.820	0.723	x	5.219	**0.508**	x	x	x	0.842	7.454	164.363

substantially faster than the CPU results in [1].

6.5 Discussion

A1 performs very well for low dimensional data sets, particularly on the Tesla. The Tesla architecture sometimes does a poor job orchestrating coalesced memory transfers, so it is likely that A1's one-dimensional thread layout overcomes this flaw. On the Tesla, A1 outperforms both A0 and A2 until dimensionality reaches 64, and across all hardware A1 outperforms A0 in the same range. When dimensionality is low, A1 achieves excellent memory reuse coupled with excellent thread occupancy. However, performance quickly drops off as the dimensionality increases. Since A1 loads two blocks of b data vectors into local memory, it requires that local memory can store at least $2 * D * b$ numbers. Since the total amount of local memory is a constant, when D increases, b must decrease. As we previously explained, A1 only has b threads in each work group, so the number of threads must also decrease with large dimensionality. On both of the GPUs we used, this number becomes catastrophically small; for $D = 128$,

the Tesla's 16KB of local memory only allows for 15 threads per work group, and only about twice that on the Fermi. In general, a work group must contain at least 128 threads in order to allow the GPU hardware to be effective in hiding memory latency. This is why we see a performance drop in A1 for larger dimensionality.

A2 does not suffer from this issue, as it never stores more than 32 data vectors, ensuring that each work group will always have 256 threads[10]. The Fermi architecture does a better job ensuring that coalesced memory transfers occur when they should, so A1 gains less of an advantage by controlling memory access patterns, and only outperforms A2 on the smallest dimensionality we tested.

It is evident that the efficient memory usage implemented in A2 leads to improved performance on nearly every problem size, but the gains fall off as dimensionality increases. With increased dimensionality, the amount of computation is large compared to the number of trips to memory (even if the chunks retrieved from memory are larger), so the gap between A0 and A2 shrinks as computation speed rather than memory efficiency becomes the limiting factor in performance.

Both A1 and A2 will be able to run problems with larger N than A0, due to their reduced memory footprint. This is critical because GPU global memory, while fast compared to CPU RAM, is generally quite constrianed (particularly on less expensive consumer hardware).

Compared to the numbers reported in [1] we see a substantial speedup in CPU computation time. Hardware differences account for some of this discrepancy, but much of it is likely due to our optimized code, including the addition of BLAS, parallelization and block decomposition. Since the authors' focus in [1] was on GPU implementations we feel that this is understandable, but it could give practitioners an inaccurate conception of the potential performance gains available from GPU computation. It bears noting, however, that at the time of this writing Intel X5650s cost approximately $1,000 USD apiece. Though the current Nvidia Tesla C2050 costs about $2,400 USD, consumer graphics hardware like the Nvidia GTX460 can deliver excellent perfomance at a sub-$200 USD price point, albeit with much less global memory.

[10]Except in the case of the Tesla for $D = 128$, where $32 * 128$ floating point numbers could not fit in local memory, so 30 vectors were stored instead, lowering the thread per work group count to 225.

6.6 Conclusions

Efficient algorithms for computing pairwise distance matrices can greatly speed up crucial components of the data analysis process in many fields. We have introduced two new GPU algorithms and a best-practices CPU implementation that improve upon previous techniques across a range of problem sizes and hardware configurations. Our results show the importance of memory access patterns and thread occupancy on the GPU, and block decomposition, BLAS and OpenMP on the CPU. We hope these results provide a useful point of reference for practitioners and inspire similar improvements to other parallelizable algorithms.

6.7 Acknowledgments

This work is coauthored with Eric Weiss, Cindy Zhang and Virginia R. de Sa, and is currently being prepared for submission. This work is supported by NSF Grant #SES-0963071, Divvy: Robust and Interactive Cluster Analysis (PI Virginia de Sa). Thanks to Patrick Gallagher for valuable assistance with this project.

7 Conclusion

In the preceding chapters we have seen that human visual judgment, even that of novices, is often sound in the context of data analysis. The conclusions humans reach are not identical to those of quality measures and algorithms with similar goals, however. We believe that this additional source of insight should be actively nurtured by software tools, such as the Divvy tool described in Chapter 5, that provide users with a seamless and instantaneous parameterize-compute-visualize loop when interacting with algorithms. By making data analysis techniques intuitive rather than mathematical and opaque, we believe that we can bring state of the art machine learning into new territories in science and industry. This expansion would be to the benefit of those fields, and would secure a greater impact for innovative machine learning research.

Bibliography

[1] Darjen Chang, Mehmed M. Kantardzic, and Ming Ouyang. Hierarchical clustering with CUDA/GPU. In James H. Graham and Anthony Skjellum, editors, *ISCA PDCCS*, pages 7–12. ISCA, 2009.

[2] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, Cambridge, MA, 2002.

[3] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[4] J.M. Santos and J. Marques de Sá. Human clustering on bi-dimensional data: An assessment. Technical Report 1, INEB ÑInstituto de Engenharia Biomédica, Porto, Portugal, 2005.

[5] Dan Pelleg and Andrew Moore. *X*-means: Extending *K*-means with efficient estimation of the number of clusters. In *Proc. 17th International Conf. on Machine Learning*, pages 727–734. Morgan Kaufmann, San Francisco, CA, 2000.

[6] Greg Hamerly and Charles Elkan. Learning the *k* in *k*-means. In *Advances in Neural Information Processing Systems*, volume 17, 2003.

[7] Yu Feng and Greg Hamerly. PG-means: learning the number of clusters in data. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 393–400. MIT Press, Cambridge, MA, 2007.

[8] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Evangelos Simoudis, Jiawei Han, and Usama Fayyad, editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996. AAAI Press.

[9] A. Azran and Z. Ghahramani. Spectral methods for automatic multiscale data clustering. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 1:190–197, 17-22 June 2006.

[10] M. Ackerman and S. Ben-David. Clusterability: A theoretical study. *Proceedings of AISTATS-09, JMLR: W&CP*, 5:1–8, 2009.

[11] J. M. Lewis. Finding a better k: A psychophysical investigation of clustering. In N. A. Taatgen and H. van Rijn, editors, *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pages 315–320, 2009.

[12] J.M. Santos and J. Marques de Sá. Human clustering on bi-dimensional data: An assessment. Technical Report 1, INEB ÑInstituto de Engenharia Biomédica, Porto, Portugal, 2005.

[13] P.J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[14] JC Dunn. Well-separated clusters and optimal fuzzy partitions. *Cybernetics and Systems*, 4(1):95–104, 1974.

[15] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321(5895):1465–1468, 2008.

[16] J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, and D.M. Blei. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, volume 21, 2009.

[17] F.B. Baker and L.J. Hubert. Measuring the power of hierarchical cluster analysis. *Journal of the American Statistical Association*, 70(349):31–38, 1975.

[18] G.W. Milligan. A Monte-Carlo study of 30 internal criterion measures for cluster-analysis. *Psychometrika*, 46:187–195, 1981.

[19] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-Simulation and Computation*, 3(1):1–27, 1974.

[20] A. Strehl. Relationship-based clustering and cluster ensembles for high-dimensional data mining. 2002.

[21] J.L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[22] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, March 1977.

[23] L. Vendramin, R.J.G.B. Campello, and E.R. Hruschka. On the comparison of relative clustering validity criteria, 2009.

[24] M. Ackerman, S. Ben-David, and D. Loker. Differentiating clustering paradigms: a property-based approach. In *Advances in Neural Information Processing Systems*, 2010.

[25] B. Bosagh-Zadeh and S. Ben-David. A uniqueness theorem for clustering. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, AUAI Press*. Citeseer, 2009.

[26] M. Ackerman, S. Ben-David, and D. Loker. Characterization of Linkage-based Clustering. In *Proceedings of COLT*, 2010.

[27] K. Wang, B. Wang, and L. Peng. CVAP: Validation for Cluster Analyses. *Data Science Journal*, 8(0):88–93, 2009.

[28] D.L. Davies and D.W. Bouldin. A cluster separation measure. *Trees*, 10, 1973.

[29] J.A. Hartigan. *Clustering algorithms*. Wiley New York, 1975.

[30] WJ Krzanowski and YT Lai. A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, 44(1):23–34, 1988.

[31] J. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, page 463. The MIT Press, 2003.

[32] M. Ackerman and S. Ben-David. Measures of clustering quality: A working set of axioms for clustering. In *Advances in Neural Information Processing Systems*. Citeseer, 2008.

[33] N. Jardine and R. Sibson. *Mathematical Taxonomy*. John Wiley and Sons, Inc., New York, 1971.

[34] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the* 8^{th} *International Conference on Computer Vision*, volume 2, pages 416–423, July 2001.

[35] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11(Feb):451–490, 2010.

[36] L.J.P. van der Maaten and G.E. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2431–2456, 2008.

[37] J. Heer, M. Bostock, and V. Ogievetsky. A tour through the visualization zoo. *ACM Queue*, 8(5), 2010.

[38] J.A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer, New York, NY, USA, 2007.

[39] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[40] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

[41] K.Q. Weinberger, F. Sha, Q. Zhu, and L.K. Saul. Graph Laplacian regularization for large-scale semidefinite programming. In *Advances in Neural Information Processing Systems*, volume 19, 2007.

[42] V. Jain and L.K. Saul. Exploratory analysis and visualization of speech and music by locally linear embedding. In *Proceedings of the International Conference of Speech, Acoustics, and Signal Processing*, volume 3, pages 984–987, 2004.

[43] J.M. Lewis, P. M. Hull, K.Q. Weinberger, and L.K. Saul. Mapping uncharted waters: exploratory analysis, visualization, and clustering of oceanographic data. In *Proceedings of the International Conference on Machine Learning and Applications*, pages 388–395, 2008.

[44] M.D. Mahecha, A. Martínez, G. Lischeid, and E. Beck. Nonlinear dimensionality reduction: Alternative ordination approaches for extracting and visualizing biodiversity patterns in tropical montane forest vegetation data. *Ecological Informatics*, 2:138–149, 2007.

[45] D. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8:1341–1390, 1996.

[46] N.D. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6(Nov):1783–1816, 2005.

[47] J.H. Friedman and J.W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23:881–890, 1974.

[48] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the* 7^{th} *ACM SIGKDD*, pages 245–250, 2001.

[49] J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409, 1969.

[50] M. Belkin and P. Niyogi. Laplacian Eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, volume 14, pages 585–591, Cambridge, MA, USA, 2002. The MIT Press.

[51] J. Venna and S. Kaski. Visualizing gene interaction graphs with local multidimensional scaling. In *Proceedings of the* 14^{th} *European Symposium on Artificial Neural Networks*, pages 557–562, 2006.

[52] D.J. Watts and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[53] H. Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62:399–402, 1967.

[54] L.J.P. van der Maaten. Learning a parametric embedding by preserving local structure. In *Proceedings of the* 12^{th} *International Conference on Artificial Intelligence and Statistics*, pages 384–391, 2009.

[55] Eleanor Rosch. Cognitive reference points. *Cognitive Psychology*, 7(4):532 – 547, 1975.

[56] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009.

[57] Stephen C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[58] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *CoRR*, cs.AI/9603104, 1996.

[59] GGobi data visualization system. http://www.ggobi.org.

[60] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 873–880, New York, NY, USA, 2009. ACM.

[61] NVIDIA Corporation. *NVIDIA CUDA Programming Guide Version 3.0*.

[62] Aaftab Munshi. *The OpenCL Specification Version: 1.0*.

[63] Darjen Chang, Nathaniel A. Jones, Dazhuo Li, Ming Ouyang, and Rammohan K. Ragade. Compute pairwise euclidean distances of data points with gpus. In *Proceeding of the IASTED International Symposium*, November 2008.

[64] Adrianto Wirawan, Bertil Schmidt, and Chee Kwoh. Pairwise distance matrix computation for multiple sequence alignment on the cell broadband engine. In Gabrielle Allen, Jaroslaw Nabrzyski, Edward Seidel, Geert van Albada, Jack Dongarra, and Peter Sloot, editors, *Computational Science Ð ICCS 2009*, volume 5544 of *Lecture Notes in Computer Science*, pages 954–963. Springer Berlin / Heidelberg, 2009.

[65] Adrianto Wirawan, Chee Keong Kwoh, and Bertil Schmidt. Multi-threaded vectorized distance matrix computation on the CELL/BE and x86/SSE2 architectures. *Bioinformatics*, 26(10):1368–1369, 2010.

[66] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28:135–151, 2001.

[67] Stanimire Tomov, Rajib Nath, Hatem Ltaief, and Jack Dongarra. Dense linear algebra solvers for multicore with GPU accelerators. *Proc. of IPDPS'10*, 2010.

[68] L. Dagum and R. Menon. OpenMP: an industry standard API for shared-memory programming. *Computational Science Engineering, IEEE*, 5(1):46 –55, 1998.