Tri-Plots: Scalable Tools for Multidimensional Data Mining

Agma Traina[1] Caetano Traina[1] Spiros Papadimitriou
Christos Faloutsos

CMU-CS-01-143

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We focus on the problem of finding patterns across two large, multidimensional datasets. For example, given feature vectors of healthy and of non-healthy patients, we want to answer the following questions: Are the two clouds of points separable? What is the smallest/largest pair-wise distance across the two datasets? Which of the two clouds does a new point (feature vector) come from?

We propose a new tool, the *tri-plot*, and its generalization, the *pq-plot*, which help us answer the above questions. We provide a set of rules on how to interpret a tri-plot, and we apply these rules on synthetic and real datasets. We also show how to use our tool for classification, when traditional methods (nearest neighbor, classification trees) may fail.

[1]Visiting from the Department of Computer Science and Statistics, University of S. Paulo at S. Carlos, Brazil.

This material is based upon work supported by the National Science Foundation under Grants No. DMS-9873442, IIS-9817496, IIS-9910606, IIS-9988876, LIS 9720374, IIS-0083148, IIS-0113089, and by the Defense Advanced Research Projects Agency under Contracts No. N66001-97-C-8517 and N66001-00-1-8936. Additional funding was provided by donations from Intel. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA, or other funding parties.

This research has been partially funded by FAPESP (São Paulo State Foundation for Research Support - Brazil), under grants 98/05556/5 and 98/0559-7), and CNPq (Brazilian National Council for Science and Technology Development) under grants 52.1685/98-6 and 860.068/00-7.

Keywords: data mining, multidimensional, box count

1 Introduction and motivation

The automatic discovery of meaningful patterns and relationships hidden in vast repositories of raw information has become an issue of great importance. Multimedia systems for satellite images, medical data and banking information are some examples of prolific data sources. Many of these data are inherently multi-dimensional. It is often difficult to summarize a large number of attributes by extracting a few essential features. Moreover, many methods proposed in the literature suffer from the *dimensionality curse* and are impractical to apply directly. Thus, dealing efficiently with high-dimensional data is a challenge for researchers in the database field [WSB98, BBK98]. Things become worse when more than one datasets are involved.

We propose a method for exploring the relationship between two multidimensional datasets, by summarizing the information about their relative position. Our method requires only a single pass on the data and scales *linearly* with the number of dimensions.

Problem definition Given two large multidimensional datasets, find rules about their relative placement in space:

Q1 Do the datasets come from the same distribution?

Q2 Do they repel each other?

Q3 Are they close or far away?

Q4 Are they separable?

Q5 For a given, unlabelled point, which of the two sets does it come from (if any)?

In the following section, we will briefly discuss the related work on data mining techniques and describe the datasets we used in our experiments. We then introduce the cross-cloud plots and explain their properties. Based on these, we present a set of practical rules which allow us to analyze two clouds of points. Finally, we describe the algorithm for generating the plots.

2 Related work

There has been a tremendous amount of work on data mining during the past years. Many techniques have been developed that have allowed the discovery of various trends, relations and characteristics with large amounts of data [JAG99, Cha98]. Detailed surveys can be found in [CHY96] and [GGR99]. Also, [Fay98] contains an insightful discussion of the overall process of knowledge discovery in databases (KDD) as well as a comprehensive overview of methods, problems, and their inherent characteristics.

In the field of spatial data mining [EKS99] much recent work has focused on clustering and the discovery of local trends and characterizations. Scalable algorithms for extracting clusters from large collections of spatial data are presented in [NH94] and [KN96]. The authors also combine this with the extraction of characteristics based on non-spatial attributes by using both spatial dominant and non-spatial dominant approaches (depending on whether the cluster discovery is performed initially or on subsets derived using non-spatial attributes). A general framework for discovering trends and characterizations among neighborhoods of data-points is presented in [EFKS98]. This framework is built on top of a spatial DBMS and utilizes neighborhood-relationship graphs which are traversed to perform a number of operations. Additionally, scalable clustering algorithms are included [AGGR98, TZ96, SCZ98, FRB98].

The work on fractals and box-counting plots is related: [BF95] used the correlation fractal dimension of a dataset to estimate the selectivity of nearest-neighbor queries; [FSJT00] gave formulas for the selectivity of spatial joins across two point-sets. [BBKK97] analyze the performance of nearest-neighbor queries, eventually using the fractal dimension. More remote work on fractals includes [PKF00], [JTWF00], [BC00].

Dataset	Description
Synthetic datasets	
Line	Points along a line segment, randomly chosen.
Circumference	Points along a circle, randomly chosen.
Sierpinsky	Randomly generated points from a Sierpinsky triangle (see fig. 7b).
Square	Points on a 2D manifold, randomly generated.
Cube	Points in a 3D manifold, randomly generated.
Super-cluster	256 uniformly distributed clusters, each with 7x7 points in a 2D manifold.
Real datasets	
California	Four two-dimensional sets of points (obtained from UCI) that refer to geographical coordinates in California [oC89]. Each set corresponds to a feature: 'streets' (62,933 points), 'railways' (31,059 points), 'political' borders (46,850 points), and natural 'water' systems (72,066 points).
Iris	Three sets describing properties of the flower species of genus Iris. The points are 4-dimensional (sepal length, sepal width, petal length, petal width); the species are 'virginica', 'versicolor' and 'setosa' (50 points each). This is a well-known dataset in the machine learning literature.
Galaxy	Datasets from the SLOAN telescope: (x, y) coordinates, plus the class label. There are 82,277 in the 'dev' class (deVaucouleurs), and 70,405 in the 'exp' class (exponential).
LC	Customer data from a large corporation (confidential). There were 20,000 records (belonging to two classes with 1,716 and 18,284 members each), each with 19 numerical/boolean attributes.
Votes	Two 16-dimensional datasets from the 1984 United States Congressional Voting Records Database: 'democrat' (267 entries) and 'republican' (168 entries).

Table 1: Description of datasets used for exposition and testing of our method.

Almost all of these papers use fast, linear (or $O(N \log N)$) algorithms, based on the *box-counting* method. We also use a similar approach for our tri-plots.

Visualization techniques for large amounts of multidimensional data have also been developed. The work described in [KK94] presents a visualization method which utilizes views of the data around reference points and effectively reduces the amount of information to be displayed in a way that affects various characteristics of the data (eg. shape and location of clusters, etc.) in a controlled manner.

There has also been significant work on data mining in non-spatial, multidimensional databases. Recent work on a general framework that incorporates a number of algorithms is presented in [iHLN99]. The authors introduce a general query language and demonstrate its application on the discovery of a large variety of association rules which satisfy the anti-monotonicity property.

However, none of the above methods can answer all the questions, Q1 to Q5, which we posed in the previous section. The method proposed in this paper can answer such questions. To find a solution for the given problem, we move away from association rules and focus on the spatial relationships between two multidimensional datasets.

2.1 Description of the data sets

We applied our method on several datasets, both synthetic and real. The former are used to build intuition, and the latter to validate our techniques. The synthetic datasets are always normalized to a unit hypercube and they may be translated, rotated and/or scaled in the experiments. The datasets are described in table 1.

Symbol	Definition
N_A (or N_B)	No. of points in dataset A (or B)
$Cross$	$Cross_{A,B}(r,1,1)$ plot between datasets A and B
$Self_A$	$Self_A(r,1,1)$ plot of dataset A
W_A	$Cross_{A,B}(r,10,1)$ cross-could plot weighted on dataset A
W_B	$Cross_{A,B}(r,1,10)$ cross-could plot weighted on dataset B
$C_{A,i}$ ($C_{B,i}$)	Count of type A (B) points in the i-th cell
n	No. of dimensions (embedding dimensionality)
D_2	Correlation fractal dimension
$\hat{r}_{min}$	Est. minimum distance between two points
$\hat{r}_{max}$	Est. maximum distance between two points

Table 2: Symbols and definitions

3 Proposed method: cross-cloud plots

Our approach relies on a novel method that allows fast summarization of the distribution of distances between points from two sets A and B. Table 2 presents the symbols used in this paper. Consider a grid with cells of side r and let $C_{A,i}$ ($C_{B,i}$) be the number of points from set A (B) in the i-th cell. The cell grid partitions the minimum bounding box of both datasets. The *cross-function* $Crossf_{A,B}(r,p,q)$ is defined as follows:

Definition 1 *Given two data sets A and B in the same n-dimensional space, we define the cross-function of order (p,q) as*

$$Crossf_{A,B}(r,p,q) = \sum_i C_{A,i}^p \cdot C_{B,i}^q$$

Typically, we plot the cross-function in log-log scales, after some normalization. The normalization factor scales the plot, maximizing the information presented:

Definition 2 *Given two data sets A and B (with N_A and N_B points) in the same n-dimensional space, we define the cross-cloud plot as the plot of*

$$Cross_{A,B}(r,p,q) = \frac{\log(N_A \cdot N_B)}{\log(N_A^p \cdot N_B^q)} \cdot \log\left(\sum_i C_{A,i}^p C_{B,i}^q\right)$$
$$\textit{versus } \log(r)$$

The cross-function has several desirable properties:

Property 1 *For $p = q = 1$, the cross-function is proportional to the count of A-B pairs within distance r. That is,*

$$Cross_{A,B}(r,1,1) \propto \textit{(\# of pairs of points within distance} \leq r\textit{)}$$

Proof Using Schuster's lemma [Sch88].

This is an important property. For $p = q = 1$, the cross-cloud plot gives the cumulative distribution function of the pairwise distances between the two "clouds" A and B [FSJT00]. Because of its importance, we will use $p = q = 1$ as the default values. We will also omit the subscripts A, B from the cross-cloud plot when it is clear which datasets are involved. That is,

$$Cross(r) \triangleq Cross_{A,B}(r) \triangleq Cross_{A,B}(r, 1, 1)$$

Property 2 *The cross-function includes the correlation integral as a special case when we apply it to the same dataset (i.e., $A \equiv B$).*

Proof From the definition of correlation integral [Sch91].

The correlation integral gives the correlation fractal dimension D_2 of a dataset A, if it is self-similar. Since the above property is very important, we shall give the self cross-cloud plots a special name:

Definition 3 *The self-plot of a given dataset A is the plot of*

$$Self_A(r) = \log\left(\frac{\sum_i C_{A,i} \cdot (C_{A,i} - 1)}{2}\right) \textit{ versus } \log(r)$$

In order to avoid artifacts that self-pairs generate, self-plots do not count self-pairs, by definition. Moreover, minor pairs ($\langle p_1, p_2\rangle$ and $\langle p_2, p_1\rangle$) are counted only once.

Property 3 *If A is self similar, then the self-plot of A is linear and its slope is its intrinsic dimensionality (correlation fractal dimension, D_2).*

Proof See [BF95].

We are now ready to define our two main tools, the tri-plot and the pq-plot.

Definition 4 *The tri-plot of two datasets, A and B, is the graph which contains the cross-plot $Cross(r)$ and the normalized self-plots for each dataset ($Self_A(r) + \log(N_A/N_B)$ and $Self_B(r) + \log(N_B/N_A)$).*

The normalization factors, $\log(N_A/N_B)$ and $\log(N_B/N_A)$, perform only translation, preserving the steepness of the graphs. In this paper, for every tri-plot we present the three graphs with the same color pattern: the cross-plot is presented in blue lines with diamonds, $Self_A$ in green lines with crosses and $Self_B$ in red lines with squares. We also show the slope (or *steepness*) of the fitted lines.

Definition 5 *The pq-plot of two datasets, A and B, is the graph of the three cross-cloud plots: $Cross_{A,B}(r)$, $Cross_{A,B}(r, 1, k)$, and $Cross_{A,B}(r, k, 1)$ for large values of k ($k \gg 1$).*

Fig. 1 shows the tri-plot and *pq*-plot for the Line and Sierpinsky datasets. Notice that, although the $Cross()$ is almost always linear (fig. 1a), this is not necessarily true for the $Cross(r, 1, k)$ and $Cross(r, k, 1)$ (in fig. 1b, $k = 10$).

Definition 6 *The steepness of a plot is its slope, as determined by fitting a line with least-squares regression.*

The tri-plots allow us to determine the relationship between the two datasets. If they are self-similar (ie. both their self-plots are linear for a meaningful range of radii), their slopes can be used in the comparisons that follow. However, the proposed analysis can be applied even to datasets which are not self-similar (ie. do not have linear self-plots). Thus, we will in use the terms *steepness* and *similarity* (as defined above). The pq-plot is used in a further analysis step. Its use is more subtle and is discussed in section 4.3.

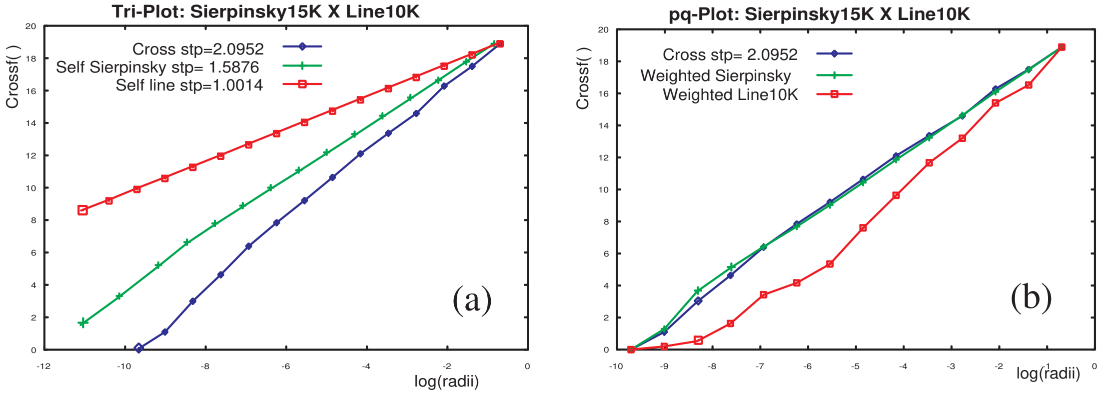

Figure 1: Sierpinsky and Line datasets: (a) the tri-plot, (b) the *pq*-plot. The cross-plots are presented in blue with diamonds, the self- and weighted-Sierpinsky plots in green with crosses, and the self- and weighted-Line in red with squares.

3.1 Anatomy of the proposed plots

This section shows how to “read” the cross-cloud plots and take advantage of the tri- and *pq*-plots, without any extra calculations on the datasets.

3.1.1 Properties of the self-plots

Property 4 *The first radius for which the count-of-pairs is not zero in the self-plot provides an accurate estimate, $\hat{r}_{min}$, of the minimum distance between any two points.*

Property 5 *Similarly, the radius up to which the count-of-pair increases (being constant for larger radii) provides an accurate estimate, $\hat{r}_{max}$, of the maximum distance between any two points. We also refer to this distance as the* dataset diameter.

Fig. 2 illustrates the above properties. The lower row of fig. 2a shows a line with 15,000 points. Its self-plot is linear. The slope, which is D_2, is equal to 1, as expected (since this is the intrinsic dimensionality of a line). The $\hat{r}_{min}$ and $\hat{r}_{max}$ estimates are also indicated.

Property 6 *If the dataset consists of clusters, the self-plot has a plateau from radius $\hat{r}_{min}$ to $\hat{r}_{max}$ (see fig. 2).*

Whenever the self-plot is piecewise linear, the dataset has characteristic scales. Plateaus are of particular interest; these occur when the dataset is not homogeneous. From the endpoints of the plateau, we can accurately estimate the maximum cluster diameter, $\hat{r}_{cdmax}$, and characteristic separation between clusters, $\hat{r}_{sepc}$. This occurs in the self-plot of the Super-cluster dataset (fig. 2b).

3.1.2 Properties of the cross-cloud plot

Fig. 3 presents an example of a tri-plot, where dataset A is a randomly generated set of 6,000 points from a line ($y = x0/x, y \in [0, 1]$), and dataset B is a Sierpinsky triangle with 6,561 points. These two datasets where chosen to highlight some interesting plot properties. These are discussed in the following (see also fig. 3).

Property 7 *The minimum distance between the datasets can be accurately estimated as the smallest radius which has a non-zero value in the cross-cloud function.*

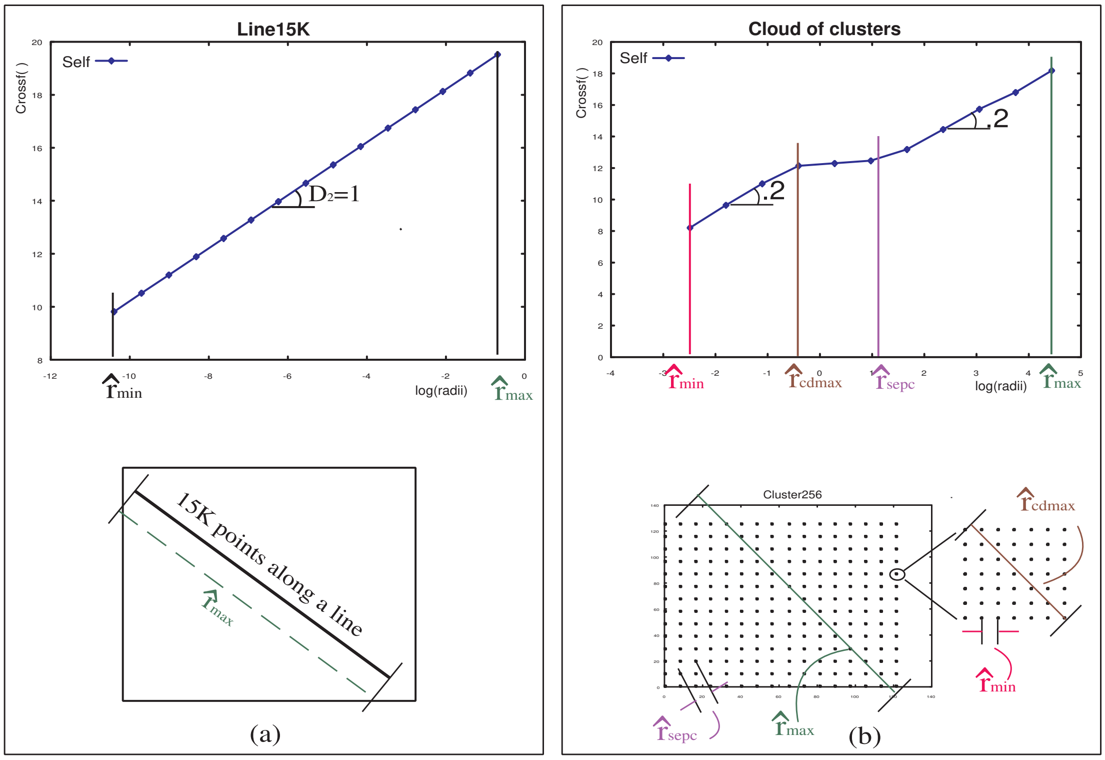

Figure 2: Measurements obtained from self-plots: (a) Line, and (b) Super-cluster datasets.

Property 8 *Similarly, the maximum distance between the datasets (or, the maximum surrounding diameter) can be accurately estimated as the greatest radius before the plot turns flat.*

Property 9 *Whenever the cross-cloud plot has a flat part for very small radii, there are duplicate points across both datasets.*

All the previous estimates can be obtained with a *single* processing pass over both datasets to count grid occupancies, *without* explicitly computing any distances.

Property 10 *The steepness of the cross-cloud plot is always greater than or equal to that of the steepest self-plot.*

4 Practical usage – Cloud mining

Before presenting our main analysis process, we need to define some terms:

Definition 7 *The* shape *of a dataset refers to its formation law (eg. "line," "square," "sierpinsky").*

Definition 8 *Two datasets are* collocated *if they have (highly) overlapping minimum bounding boxes.*

Definition 9 *The* placement *of a dataset refers to its position and orientation.*

We use these three terms when comparing two datasets. Two datasets can have the same shape but different placement (eg. two non-collinear lines). Two datasets have the same shape but different placement, if the one can be obtained from the other through *affine* transformations. Also, two datasets with the same intrinsic dimensionality can have different shapes (eg. a line and a circle – both have $D_2 = 1$).

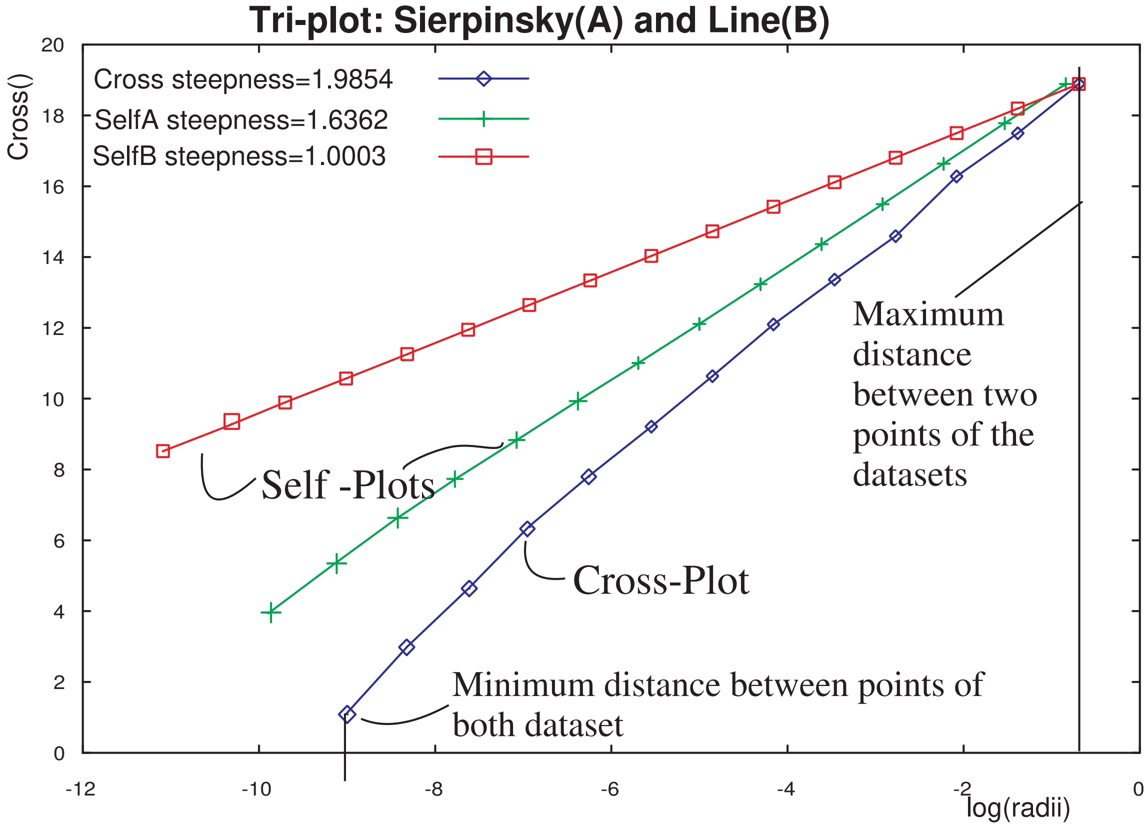

Figure 3: Example of a tri-plot indicating where to find meaningful information. The cross-plot is always in blue with diamonds, $Self_A$ in green with crosses and $Self_B$ in red with squares.

4.1 Rules for tri-plot analysis

In this section we present rules (see table 3 for a summary) to analyze and classify the relationship between two datasets. From the tri-plots we can get information about the intrinsic structure and the global relationship between the datasets.

Rule 1 (identical) If both datasets are identical, then all plots of a tri-plot are similar ($Self_A \approx Self_B \approx Cross$). In this case, the three graphs will be on top of each other. This means that the intrinsic dimensionality, shape as well as placement of both datasets are the same. This may be because one dataset is a subset of the other, or both are samples from a bigger one. Fig. 4 shows the tri-plots for (a) two lines with different number of objects, (b) two Sierpinsky triangles, and (c) two coplanar squares in 3D. All datasets in fig. 4 are in a 2D manifold. In all these examples, both datasets have the same shape and placement but different number of points.

Rule 2 (same shape, different placement) If both datasets have the same intrinsic dimensionality, but different placement, then their steepness is similar ($Self_A \approx Self_B$), but $Cross$ is only moderately steeper than both. Further analysis using the pq-plot can indicate whether the datasets are separable or not and, if separable, to what extent. Examples are intersecting lines, intersecting planes, or two Sierpinsky datasets with one rotated over the other (see fig. 5a, 5b and 5c, respectively).

Rule	Situation	Condition	Example
		A and B are similar ($Self_A$ and $Self_B$ have same steepness), **and**	
1	Datasets A and B are statistically identical	$Cross$, $Self_A$ and $Self_B$ have the same steepness	Figure 4
2	Both datasets have the same intrinsic dimensionality	$Cross$ has steepness comparable to that of $Self_A$ and $Self_B$	Figure 5
3	The datasets are disjoint	$Cross$ is much steeper than both $Self_A$ and $Self_B$	Figure 6
		A and B are not similar ($Self_A$ and $Self_B$ have different steepness), **and**	
4	The (less steep) dataset is a proper subset of the other	$Cross$ and $Self_A$ or $Self_B$ have the same steepness	Figure 7
5	The datasets are collocated	$Cross$ has steepness comparable to that of $Self_A$ and $Self_B$	Figure 8
3	$Cross$ is much steeper than both $Self_A$ and $Self_B$	The datasets are disjoint	Figure 6

Table 3: Conditions and rules used in tri-plot analysis.

Rule 3 (disjoint datasets) If the datasets are disjoint, then $Cross$ is much steeper than both $Self_A$ and $Self_B$ (does not matter whether the latter are similar or not). For two intersecting datasets, the $Cross$ steepness will not be so far from the steepness of their self-plots. However, if the $Cross$ is much steeper than both $Self_A$ and $Self_B$, it means that the minimum distance between points from the datasets is bigger than the average distance of the nearest neighbors of points in both datasets, so the datasets are disjoint. In fact, this case leads to the conclusion that both datasets are well-defined clusters, hence they should be separable by traditional clustering techniques. Examples of this situation are non-intersecting lines, squares far apart, or a Sierpinsky triangle and a plane which is not coplanar with the Sierpinsky's supporting plane (see fig. 6a to 6c). All datasets are in 3D space. Notice that the self-plots have the expected slopes, but the cross-plots have very high steepness (18, 13 and 26 respectively).

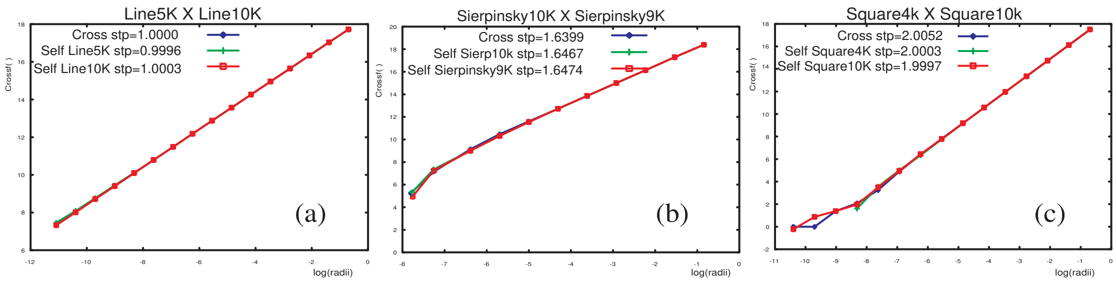

Figure 4: Rule 1 – The two datasets have the same shape and placement: (a) Two superimposed lines (all plots have slope ≈ 1, (b) Two superimposed Sierpinsky triangles (all plots have slopes $\approx 1.64 \approx \log 3/\log 2$, (c) Two superimposed squares (all plots have slopes ≈ 2. All datasets are in 2D space, and the axes of all tri-plots are in log-log scale.

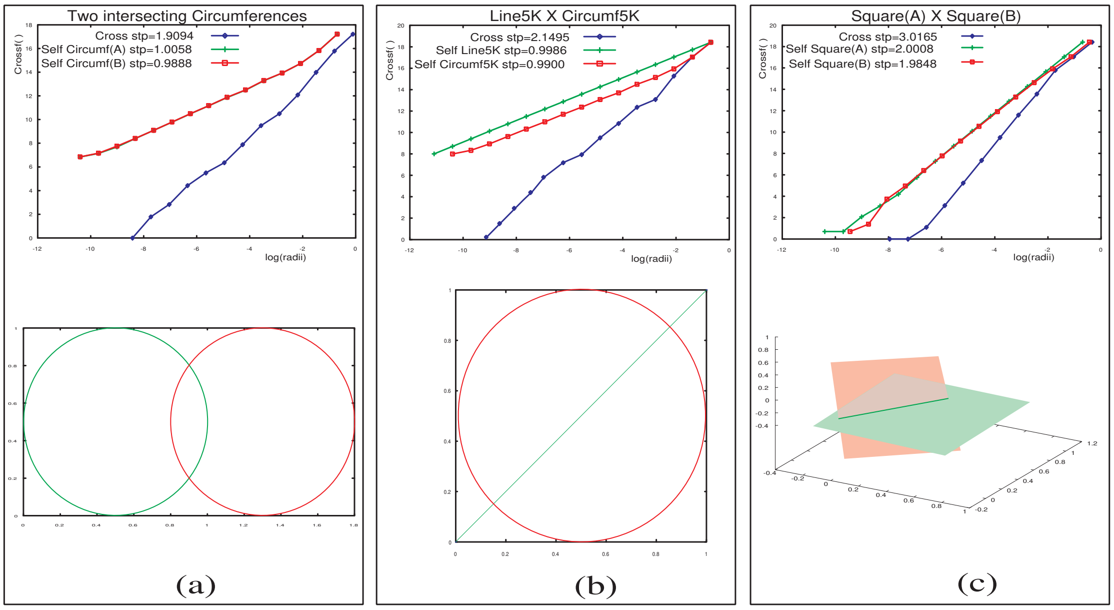

Figure 5: Rule 2 – The two datasets have the same intrinsic dimensionality, but different placements: (a) Two intersecting circumferences in 2D space, (b) A line crossing a circumference in 2D space, (c) Two piercing planes in 3D space. The upper row shows the tri-plots with the axes in log-log scale. The lower row shows the corresponding datasets in their respective spaces.

Rule 4 (sub-manifold) Without loss of generality, let $Self_A$ be the steepest of $Self_A$ and $Self_B$. If dataset B is a sub-manifold of dataset A, the self-plots do not have similar steepness ($Self_A \not\approx Self_B$) and the $Cross$ is equal to $Self_A$. Remember that the steepness of the $Cross$ cannot be smaller than the steepness of $Self_A$ or $Self_B$. Therefore, if the steepness of the $Cross$ is similar to one of the self steepnesses (eg. $Cross \approx Self_A$), then the other graph (in this case $Self_B$) will be less steep than $Cross$. This means that the points in dataset B have a stronger correlation than the points in dataset A. Rule 1 deals with the situation where both datasets are subsets of a larger one, or one is a subset of another, but there is no rule to extract the subsets. Rule 4 deals with the same case of occurrence of subsets, but here there are rules to choose points that pertain to the dataset with a smoother self-plot. Examples of this case are a line embedded in a plane, a Sierpinsky dataset and its supporting plane, and a square embedded in a volume (see fig. 7a, 7b and 7c, respectively).

Rule 5 (collocated) If both datasets have different shape, placement and intrinsic dimensionality, then $Self_A \not\approx Self_B$ and the $Cross$ is only moderately steeper than $Self_A$ and $Self_B$. In this case, the datasets are not related to each other. They are, however, collocated, or at least intersecting. This means that although part of the datasets may be separable, this would not be true for the entire dataset, or for both datasets. Whenever this situation occurs, it should be further analyzed, for example, using the pq-plot. These are the cases of a line with a Sierpinsky triangle, a line piercing a square, and a Sierpinsky intersecting a square, as fig. 8 shows.

4.2 Application to real datasets

In the previous section we described the rules, using synthetic datasets to build intuition. Here we apply them to real datasets (see fig. 9).

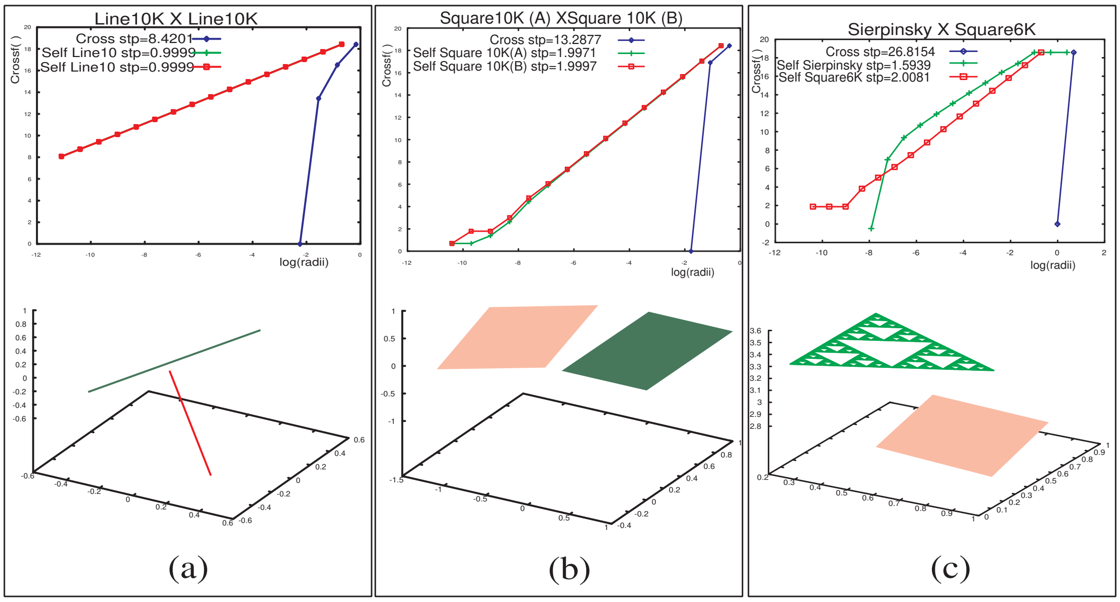

Figure 6: Rule 3 – The two datasets are disjoint: (a) two non-intersecting lines, (b) two non-intersecting squares, (c) a square and a Sierpinsky triangle. The upper row shows the tri-plots with the axes in log-log scale. The lower row shows the corresponding datasets in 3D space.

Rule 1 (identical) There are four pairs of datasets which conform this rule: two different subsets of California-political (fig. 9a), the two galaxy datasets (for $\log r \in [-4, 4]$ – fig. 9b), Iris-versicolor and Iris-virginica (fig. 9c), and two different subsets of California-water (fig. 9d).

Rule 3 (disjoint datasets) The Iris-Versicolor and Iris-Setosa pair (fig. 9e), and the Democrat and Republican pair (fig. 9f) conform to this rule. Their cross-plot is much steeper than their self-plots. Versicolor and Setosa species are indeed apart. Also, the Democrat and Republican parties have distinct behavior, which allows separation of their members. Thus, we conclude that these dataset pairs can be separated and we can estimate the minimum distance between them (see property 7).

Rule 4 (sub-manifold) Fig. 9g shows the tri-plot of California-water and California-political. Recall that the dataset with smaller steepness is probably a proper sub-manifold of the one with larger steepness (or of the superset from which both are samples). We thus conclude that California-political is a subset of California-water. This makes sense, since many political divisions are along water paths.

Rule 5 (collocated) According to fig. 9h, California-railroad and California-political agree with Rule 5. This is reasonable, since railroads are built with objectives irrelevant to political divisions. Also, the LC datasets agree with Rule 5 and require further analysis. The flat parts in fig. 9i and in the political self-plot (fig. 9h) indicate that these datasets possibly have duplicate (or near-duplicate) points. The Galaxy datasets (fig. 9bb) demonstrate the case of clusters, which are present at two characteristic distances. Also, the datasets repel each other for radii close to the cluster diameter. After analyzing the relationship between two datasets using tri-plots, more information can be obtained from the pq-plots.

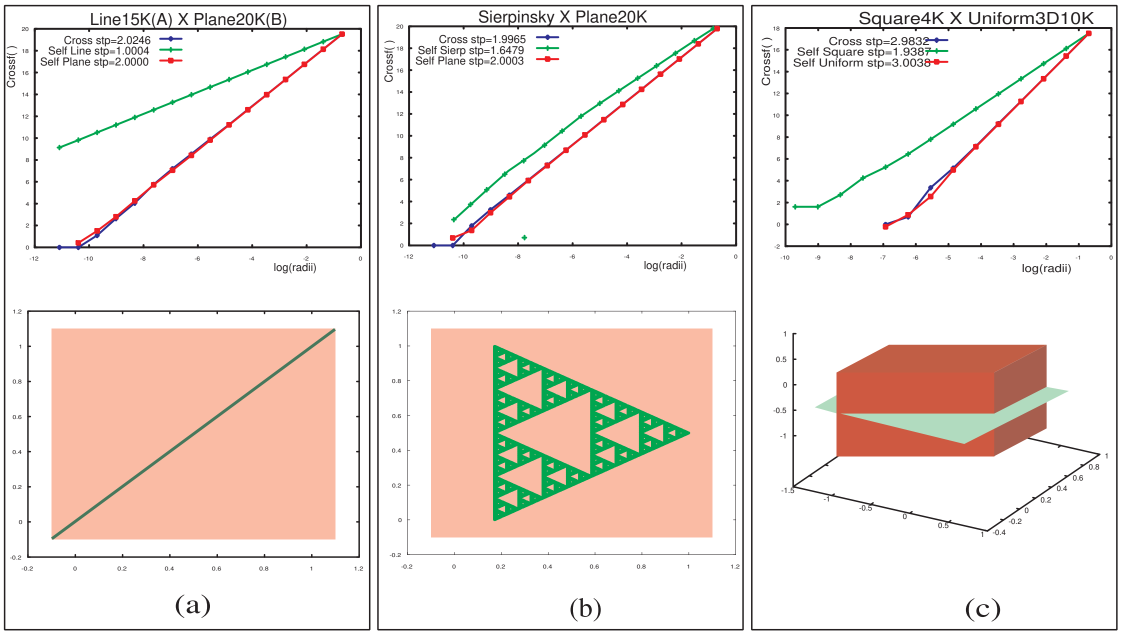

Figure 7: Rule 4 – One dataset is a proper subset of the other dataset: (a) a square overlapping a line in 2D space, (b) a Sierpinsky triangle and its supporting plane in 2D space, (c) a volume travesed by a plane in 3D space. The upper row shows the tri-plots in log-log scale. The lower row shows the datasets in their respective spaces.

4.3 Analysis of the pq-plot

The pq-plot allows us to further examine the relationship between two datasets, by weighting one dataset when comparing its distance distribution with that of the other dataset. The analysis of the pq-plots is directed to specific ranges of the cross-cloud plots, in contrast to the more global analysis of the tri-plots.

Even if a $Cross_{A,B}(r, p, q)$ plot with $p \neq 1 \neq q$ happens to be a line, its slope has no meaning; only its overall shape has useful properties. Also, due to the normalization by $\log(N_A \cdot N_B)/\log(N_A^p \cdot N_B^p)$, both the leftmost and rightmost points in all pq-plots coincide. According to equation 1, if a particular $C_{A,i}$ (or $C_{B,i}$) in the calculation of $Cross_{A,B}(r, p, q)$ is zero for a given radius r in a given region of the space, the corresponding $C_{B,i}$ (or $C_{A,i}$) will not contribute to the total for this particular radius. The result will be a flat region in this part of the curve. Otherwise, if there is a regular distribution of distances over a continuous part of the curve, the resulting curve will exhibit a linear shape. Sudden rises in a plot indicate a large growth of counts starting at that radius. Hence, the two shapes in the curves of the cross-cloud plots that are worth looking for are: the linear parts, and the regions where the curves are flat.

The cross-cloud plots, $Cross_{A,B}(r, k, 1)$, and $Cross_{A,B}(r, 1, k)$ with $k \gg 1$ (which we have named W_A and W_B because they are 'weighted'), can be generated for any value of k. However, increasing k only increases the distortions on the plot, without giving any extra information. Thus, we picked $k = 10$. Each conclusion is valid for the range of radii which presents specific behavior. Next, we discuss two representative situations, using pairs of synthetic datasets and comparing the obtained tri-plots and pq-plots.

Fig. 10 compares two pairs of datasets: circumference-circumference and line-circumference. This illustrates the situation stated by Rule 2: the two datasets are similar ($Self_A \approx Self_B$ and $Cross$ steepness is less or equal than the steepness of $Self_A$ plus the steepness of $Self_B$). By looking only at the tri-plots in fig. 10a and 10d, it is not possible to say anything else about the datasets. However, in fig. 10b the three graphs are on top of each other. This means that both datasets have the same behavior under weighted cal-

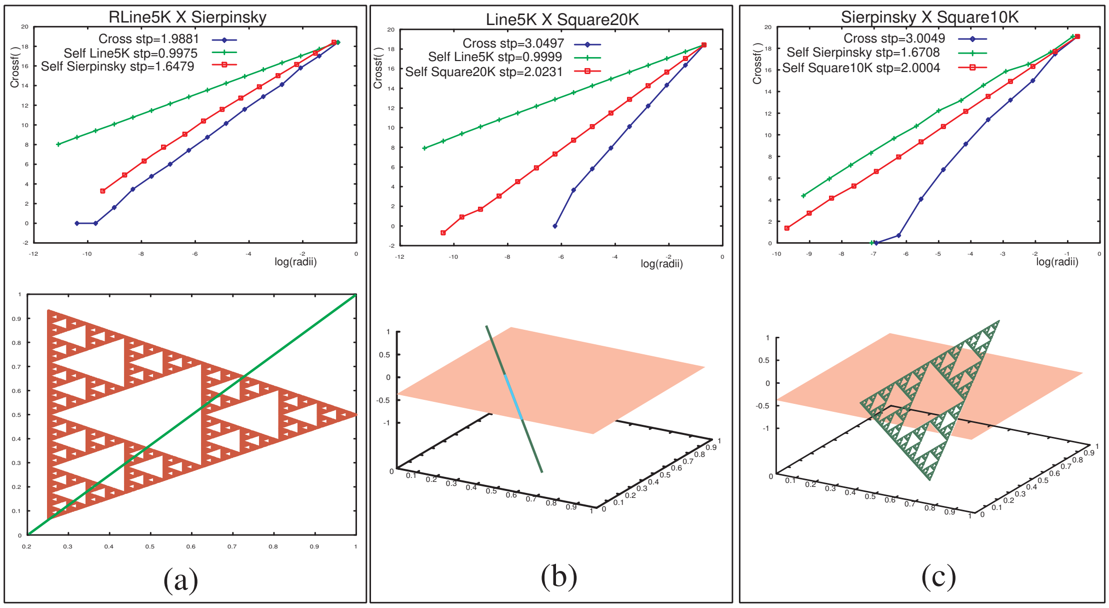

Figure 8: Rule 5 – The datasets come from different placements: (a) a line and a Sierpinsky triangle in 2D space, (b) a line piercing a square in 3D space, (c) a plane and an intersecting Sierpinsky triangle in 3D space. The upper row shows the tri-plots in log-log scale. The lower row shows the corresponding datasets in their respective space.

culation ($Cross(r, 1, 10)$ and $Cross(r, 10, 1)$). Thus, both datasets have the same shape. On the other hand, the behavior of the pq-plot in fig. 10e shows that the datasets have different shapes, as well as how they are correlated within specific radii ranges (Region I and II on the plots).

In this section we proposed the rules to analyze the tri-plots and the pq-plots using easily understandable synthetic datasets in 2D and 3D spaces. However, the same conclusions should apply for real datasets in any multi-dimensional space. In fact, for real datasets it is usually difficult to know how to describe the relationship between the attributes and to know if they are correlated. Nonetheless, our proposed analysis can indicate not only the existence of correlations, but also how "tight" they are. This analysis can also provide evidence of how separable the datasets are, as well as if it is possible to classify points as belonging to one or to the other dataset.

4.4 Using pq-plots to analyze datasets

Due to space limitations, we present pq-plots only for some of the real datasets(fig. 11). Fig. 11a shows the pq-plot for the Galaxy datasets. For the highlighted range, there is a distinct separation between the datasets. Besides confirming that the two galaxy types indeed repel each other, the pq-plots show that there are few clusters consisting only of 'exp' galaxies (although there are clusters including points of both datasets also only with 'dev' points). Outside the highlighted range, the sets are almost identical. As expected, fig. 11b confirms that the Democrat and Republican datasets are separable, since the weighted plots have completely opposite behaviors.

Fig. 11c shows the pq-plot of the California-water and California-political datasets. In this plot, there are four ranges with distinct behaviors. Range I corresponds to very small distances, so these distances are probably less than the resolution of the measurements; therefore they are not meaningful. Ranges II and III are where the real distances are meaningful. The sudden fall to the left of the wWater-plot in range II means that there are very few points in the political dataset at distances below this range from points in the water

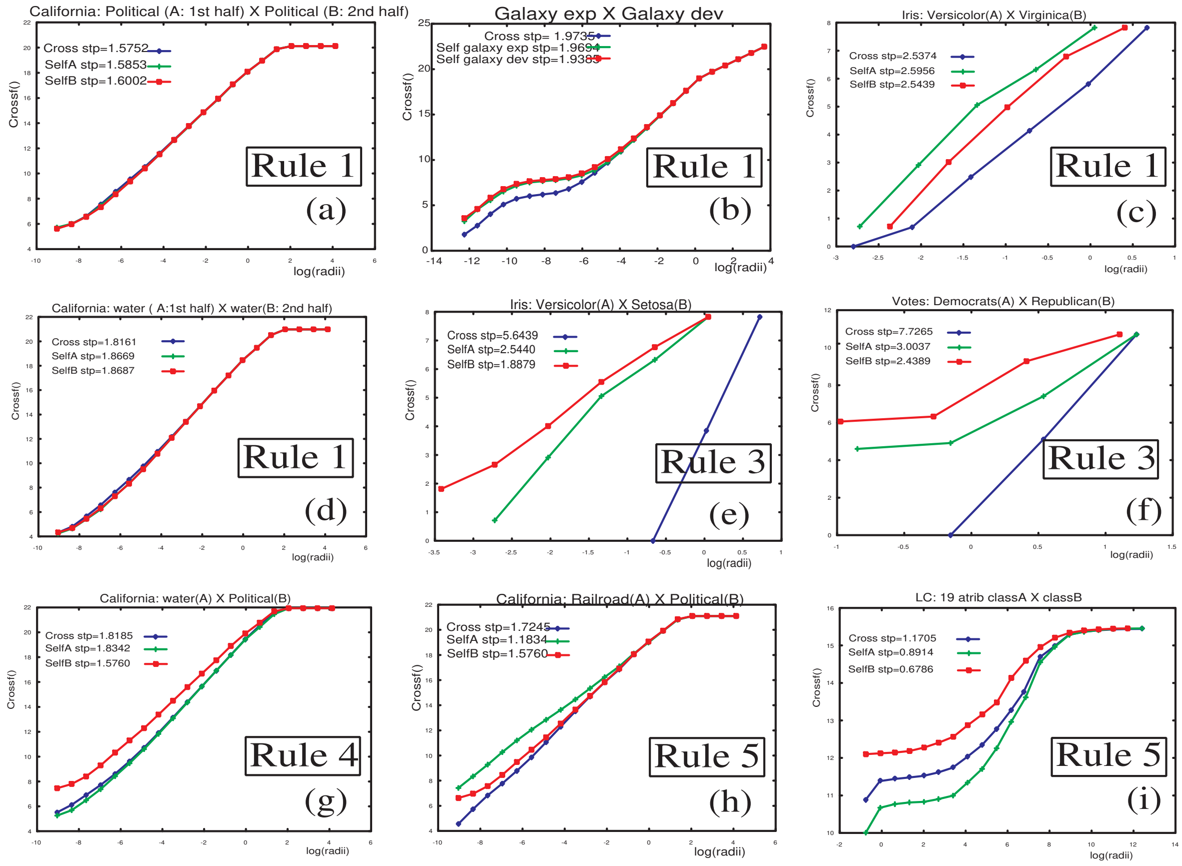

Figure 9: Tri-plots of real datasets and their classification as obtained from rules 1-5.

dataset. This indicates a kind of "repulsion" of points from both datasets for these small distances. In range III, both datasets have approximately the same behavior. Range IV is almost flat for all plots, meaning that there are almost no more pairs within this distance range. In fact, the "almost flat" part of the graph is due to a few outliers in the dataset.

4.5 Membership testing and classification

So far we have shown how to use the tri-plots to answer questions Q1-Q5. In this section we illustrate the power of cross-cloud plots in another setting: membership testing and classification (Q5). Fig. 12 illustrates the following situation: We have two datasets, A (20 points along a line) and B (900 points in a 'tight' square). A new point (indicated by '?') arrives. Which set, if any, does it belong to?

Visually, the new point ('?') should belong to the Line20 set. However, nearest neighbors or decision-tree classifiers would put it into the square: the new point has $\sim$ 900 'Square' neighbors, before even the first 'Line20' neighbor comes along!

We propose a method that exploits cross-cloud plots to correctly classify the new point ('?'). The new point is treated as a singleton dataset and its cross-plots are compared to the self-plots of each candidate set. In this particular case, we compare the steepness of $Cross_{\mathrm{Line,Point}}$ and $Cross_{\mathrm{Square,Point}}$ to the steepness of $Self_{\mathrm{Line}}$ and $Self_{\mathrm{Square}}$ and classify the new point accordingly. Notice that the plots in fig. 12b are more similar to each other (almost equal steepness), while the plots in fig. 12c are clearly not similar. Thus, we conclude that the new point ('?') belongs to the Line20 dataset, despite what k-nearset neighbor classification would say!

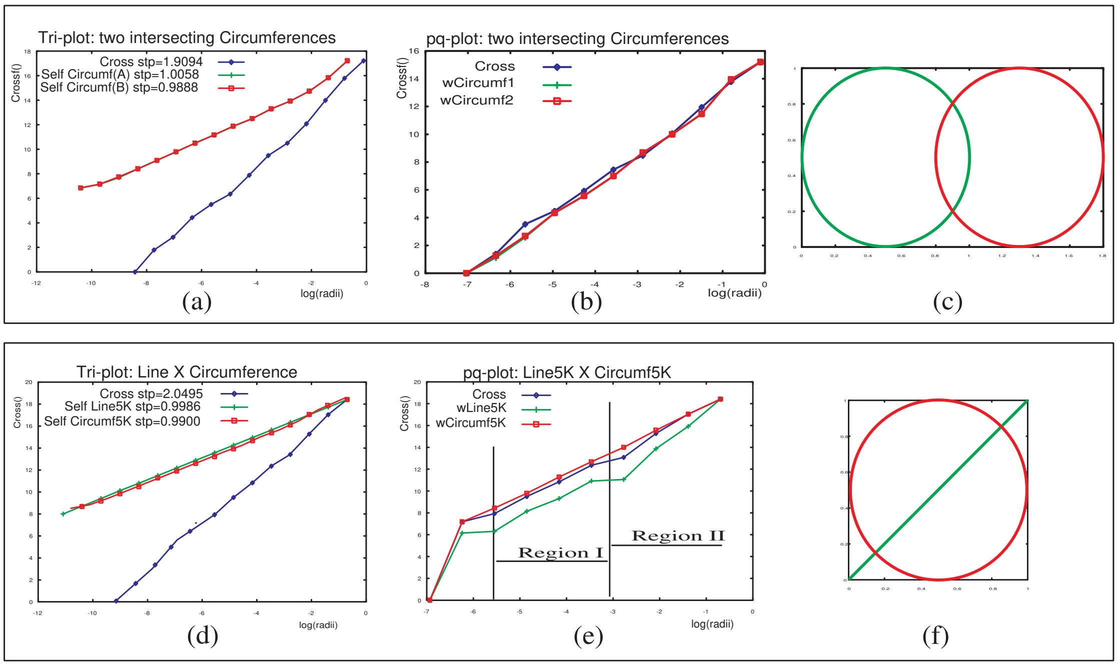

Figure 10: pq-plots for two pairs of datasets: (a) the tri-plot of two intersecting circumferences (as shown in (c)), (b) the pq-plot of the two circumferences, (d) the tri-plot of a line intersecting a circumference (as shown in (f)), and (e) the pq-plot of the line and the circumference.

The full details of the classification method are the topic of ongoing research. This is yet another application of the cross-cloud technique.

5 Implementation

To obtain the required tri-plots, we use the single-pass algorithm presented in appendix A. This is based on box-counting and is an extension of [BF95, FSJT00].

What is important is that this algorithm scales up for arbitrarily large datasets, and arbitrarily high dimensions. This is rarely true for other spatial data mining methods in the literature. The algorithm to generate the pq-plots is very similar to the algorithm in appendix A, except we construct W_A and W_B (instead of $Self_A$ and $Self_B$) plots.

5.1 Scalability

The algorithm is linear on the total number of points, ie. $O(N_A + N_B)$. If we want l points in each cross-cloud plot (ie. number of grid sizes), then the complexity of our algorithm is $O((N_B + N_A) \cdot l \cdot n)$, where n is the embedding dimensionality. Fig. 13 shows the wall-clock time required to process datasets on a Pentium II machine running NT4.0. The datasets on the left graph have varying numbers of points in 2, 8 and 16-dimensional spaces, and we used 20 grids for each dataset. For the right graph, we used datasets with 100,000, 200,000 and 300,000 points and dimensions 2 to 40. The execution time is indeed linear on the total number of points, as well as on the dimensionality of the datasets. The algorithm does not suffer from the *dimensionality curse*.

Notice that steps 1 and 2 of the algorithm read the datasets and maintain counts of each non-empty grid cell. These counts can be kept in any data structure (hash tables, quadtrees, etc).

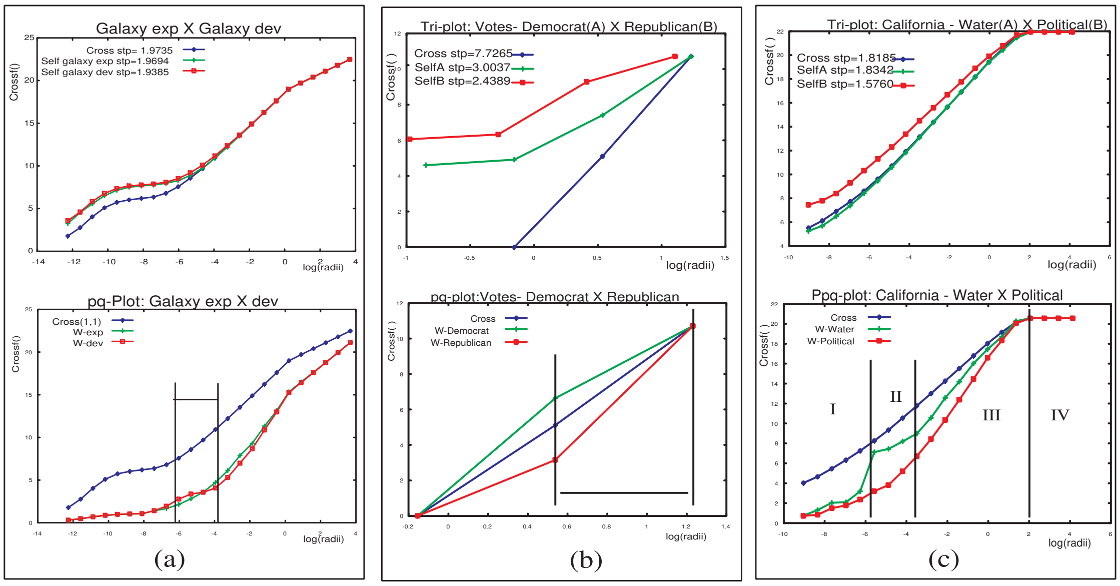

Figure 11: pq-plots for real datasets: (a) Galaxy, (b) Democrat and Republican, (c) California-water and California-political. The upper row shows the tri-plots and the lower row the corresponding pq-plots. The axes are in log-log scales.

6 Conclusions

We have proposed the cross-cloud plot, a new tool for spatial data mining across two n-dimensional datasets. We have shown that our tool has all the necessary properties:

- It can spot whether two clouds are disjoint (separable), statistically identical, repelling, or in-between. That is, it can answer questions Q1 to Q4 from section 1.
- It can be used for classification and is capable of "learning" a shape/cloud, where traditional classifiers fail to do so (ie. it can answer question Q5).
- It is very fast and scalable: We use a box-counting algorithm, which requires a single pass over each dataset, and the memory requirement is proportional to the number F of non-empty grid cells and to the number l of grid sizes requested ($1 \leq F \leq N_A + N_B$, and clearly not exploding exponentially).

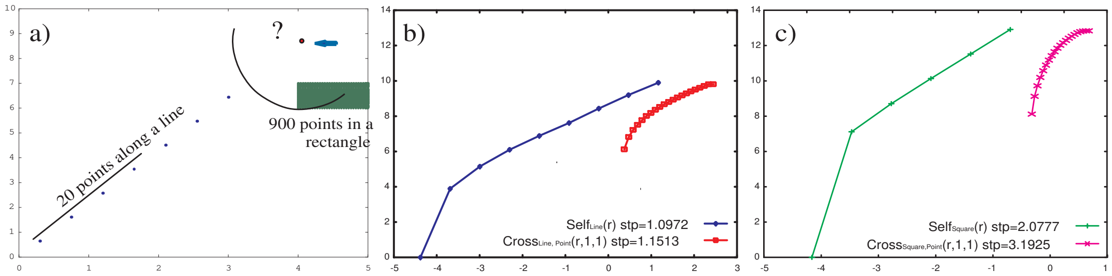

Figure 12: Classifying a point as either belonging to a sparse line or to a dense square, using the cross-cloud method: (a) spatial placement of the incoming point and the datasets, (b) $Self_{\text{Line}}$ and $Cross_{\text{Line,Point}}$ plot, (c) $Self_{\text{Square}}$ and $Cross_{\text{Square,Point}}$ plot.

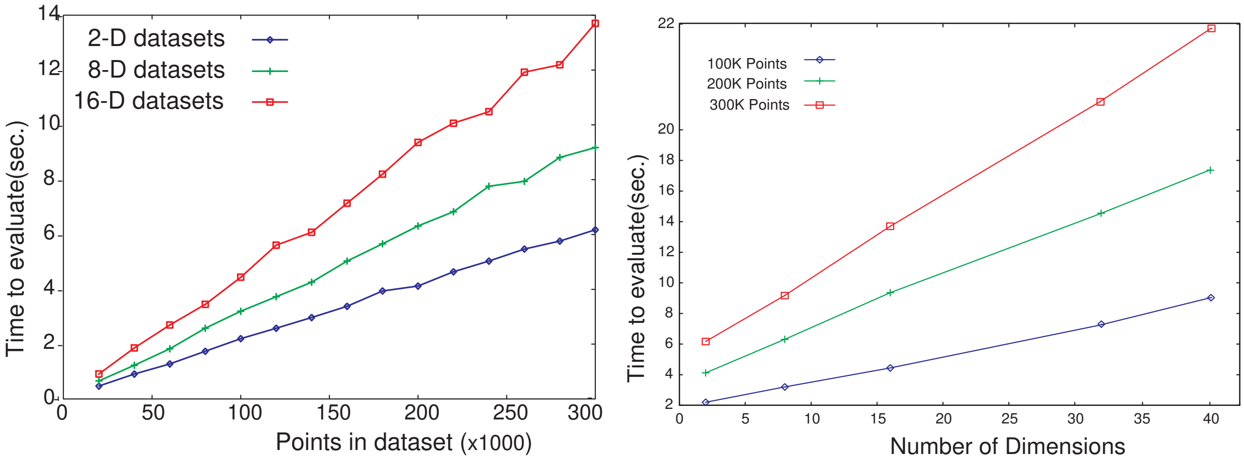

Figure 13: Left – Wall-clock time (in seconds) needed to generate the tri-plots for varyingly sized datasets. The blue graph represents the time for 2D datasets, the green graph for 8D datasets and the red graph for 16D datasets. Right – Wall-clock time (in seconds) needed to generate the Tri-plots versus the dimensionality of the datasets, for three different dataset sizes (100,000, 200,000 and 300,000).

- Tri-plots can be applied to high-dimensional datasets easily, because the algorithms scale linearly with the number of dimensions.

The experiments on real datasets show that our tool finds patterns that no other known method can. We believe that our cross-cloud plot is a powerful tool for spatial data mining and that we have just seen only the beginning of its potential uses.

References

[AGGR98] Rakesh Agrawal, Johannes Gherke, Dimitrios Gunopoulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications, 1998.

[BBK98] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. The pyramid-tree: Breaking the curse of dimensionality. In *Proc. ACM SIGMOD Conf. on Management of Data*, pages 142–153, 1998.

[BBKK97] Stefan Berchtold, Christian Böhm, Daniel A. Keim, and Hans-Peter Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *Proc. of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 78–86, 1997.

[BC00] Daniel Barbará and Ping Chen. Using the fractal dimension to cluster datasets. In *Proc. of the 6th International Conference on Knowledge Discovery and Data Mining (KDD-200)*, pages 260–264, 2000.

[BF95] A Belussi and Christos Faloutsos. Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In *Proc. of VLDB - Very Large Data Bases*, pages 299–310, 1995.

[Cha98] Surajit Chaudhuri. Data mining and database systems: Where is the intersection? *Data Engineering Bulletin*, 21(1):4–8, 1998.

[CHY96] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining - an overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, 1996.

[EFKS98] Martin Ester, Alexander Frommelt, Hans-Peter Kriegel, and Jörg Sander. Algorithms for characterization and trend detection in spatial databases. In *Proc. of the 4th International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 44–50, 1998.

[EKS99] Martin Ester, Hans-Peter Kriegel, and Jörg Sander. Spatial data mining: A database approach. In *LNCS 1262: Proc. of 5th Intl. Symposium on Spatial Databases (SSD'97)*, pages 47–66, 1999.

[Fay98] Usama M. Fayyad. Mining databases - towards algorithms for knowledge discovery. *Data Engineering Bulletin*, 21(1):39–48, 1998.

[FRB98] Usama M. Fayyad, Cory Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proc. of the 4th International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 194–198, 1998.

[FSJT00] Christos Faloutsos, Bernhard Seeger, Caetano Traina Jr., and Agma Traina. Spatial join selectivity using power laws. In *Proc. ACM SIGMOD 2000 Conf. on Management of Data*, pages 177–188, 2000.

[GGR99] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. Mining very large databases. *IEEE Computer*, 32(8):38–45, 1999.

[iHLN99] iawei Han, Laks V. S. Lakshmanan, and Raymond T. Ng. Constraint-based multidimensional data mining. *IEEE Computer*, 32(8):46–50, 1999.

[JAG99] Roberto J. Bayardo Jr., Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. In *Proc. of IEEE Intl. Conference on Data Engineering*, pages 188–197, 1999.

[JTWF00] Caetano Traina Jr., Agma Traina, Leejay Wu, and Christos Faloutsos. Fast feature selection using the fractal dimension. In *XV Brazilian Symposium on Databases (SBBD)*, 2000.

[KK94] Daniel A. Keim and Hans-Peter Kriegel. Possibilities and limits in visualizing large amounts of multidimensional data. In *Perceptual Issues in Visualization*. Springer, 1994.

[KN96] Edwin M. Knorr and Raymond T. Ng. Finding aggregate proximity relationships and commonalities in spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):884–897, 1996.

[NH94] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proc. of VLDB - Very Large Data Bases*, pages 144–155, 1994.

[oC89] Bureau of Census. Tiger/line precensus files: 1990 technical documentation. Bureau of the Census. Washington, DC, 1989.

[PKF00] Bernd-Uwe Pagel, Flip Korn, and Christos Faloutsos. Deflating the dimensionality curse using multiple fractal dimensions. In *Proc. of IEEE Intl. Conference on Data Engineering*, pages 589–598, 2000.

[Sch88] Heinz Georg Schuster. *Deterministic Chaos*. VCH Publisher, Weinheim, Basel, Cambridge, New York, 1988.

[Sch91] Manfred Schroeder. *Fractals, Chaos, Power Laws*. W.H. Freeman and Company, New York, 1991.

Algorithm: Fast tri-plot
Inputs: Two datasets, A and B (with N_A and N_B points respectively) normalized to the unit hyper-cube, and the number l of desired points in each plot.
Output: Tri-plot

Begin
1 - For each point a of dataset A:
  For each grid size $r = 1/2^j, j = 1, 2, \ldots, l$:
    Decide which grid cell it falls in (say, the i-th cell)
    Increment the count $C_{A,i}$

2 - For each point b of dataset B:
  For each grid size $r = 1/2^j, j = 1, 2, \ldots, l$:
    Decide which grid cell it falls in (say, the i-th cell)
    Increment the count $C_{B,i}$

3 - Compute the sum of product occupancies for the functions:
  $\mathit{Self}_A(r) = \log\left(\frac{1}{2}\sum_i C_{A,i} \cdot (C_{A,i} - 1)\right)$,
  $\mathit{Self}_B(r) = \log\left(\frac{1}{2}\sum_i C_{B,i} \cdot (C_{B,i} - 1)\right)$,
  $\mathit{Cross}_{A,B}(r) = \log\left(\sum_i C_{A,i} \cdot C_{B,i}\right)$

4 - Print the tri-plot:
  for $r = 1/2^j, j = 1, 2, \ldots, l$:
    Print $\mathit{Cross}_{A,B}(r)$
    Print Self_A normalized: $\mathit{Self}_A(r) + \log(N_B/N_A)$
    Print Self_B normalized: $\mathit{Self}_B(r) + \log(N_A/N_B)$
End

Figure 14: Algorithm

[SCZ98] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proc. of VLDB - Very Large Data Bases*, pages 428–439, 1998.

[TZ96] Miron Livny Tian Zhang, Raghu Ramakrishnan. Birch: An efficient data clustering method for very large databases. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 103–114, 1996.

[WSB98] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. of VLDB - Very Large Data Bases*, pages 194–205, 1998.

A Algorithm

Given two datasets A and B (with cardinalities N_A and N_B) in a n-dimensional space, we generate the tri-plot (ie. $\mathit{Cross}_{A,B}$, Self_A and Self_B plots) using the algorithm shown in Figure 14. Note that the number F of non-empty cells in each grid does not depend on the dimensionality n. In fact, $1 \le F \le N_A + N_B$.