Thesis Proposal: Designing Distance Functions

Leejay Wu

May 28, 2002

Contents

Abstract

Distance function design is a fundamental problem underlying much of data mining. Numerous methods designed to solve a variety of problems ranging from clustering to nearest-neighbor retrieval require some concept of distance with which to compare objects. Approaches can be classified as user-driven methods, in which complex distance functions are induced from user input or other sources of labels and feedback; and data-driven methods, which perform distance function design as a preprocessing step without such additional input. With regards to the first, the FALCON system provides example-driven query development via relevance-feedback; the second approach includes a scaling and selection system. Both are works in progress.

I will focus on the latter, with an eye towards adding reconstruction to the scaling and selection components as well as developing and testing a fair evaluation scheme to quantify how good a set of scales is. The former, FALCON, will also be extended but to a much lesser degree.

1 Introduction

Numerous data mining and machine learning tasks depend on good distance function design. Many, many algorithms need to consider pairs of objects and numerically evaluate either their distance or similarity. Clustering, outlier detection, neighborhood-based methods, similarity searches, and numerous other applications all depend on careful distance function design.

The problem can be split into two major situations, query-driven and data-driven, where "query-driven" means that distance functions are customized on a per-query basis while "data-driven" means that they are statically derived for the database as a whole.

Both have their uses. For instance, query-driven methods may be much more suitable for multimedia queries where even queries that seem simple to a user may be difficult or impossible. The latter would largely consist of cases where the query engine does not take into account information that is relevant to the actual query semantics. For instance, a complete search for photographs of former United States presidents in a database of non-annotated images would be effectively impossible unless either all former presidents are specified by example, or the distance function was already tuned to distinguish between presidents and non-presidents. With regards to query-driven systems, I will focus on feasible but non-trivial queries, as there are sound reasons to believe that no universal solution exists.

Data-driven models, on the other hand, can be applied before one even knows what the queries will be. Precomputing a distance function, or, isomorphically, transforming the data, can also improve efficacy and efficiency. For instance, if parts of the data prove valueless due to redundancy, they can be discarded. In addition, while computation performed during preprocessing may require significant amounts of time, none of it will be visible to the user at query time.

The next two sections describe past work on these two basic problem formulations. The core of the first section has previously appeared as [44], while the second has been submitted for publication.

2 Query-driven methods

Most, if not all, “reasonable” models for answering a user’s query require a method of computing the similarity or distance between two database objects. An important question is whether or not it is both useful and viable to create custom distance functions based on a specific query, or rather to standardize on one distance function for an entire database. This section describes query-driven approaches in which the distance function is in fact specialized on a per-query basis. In particular, it focuses on the relevance-feedback model, in which a user iteratively refines his query by supplying relevance judgements.

2.1 Motivation

Distance functions prove central to many query-handling engines, as many retrieval methods use distance or similarity to compare individual objects [28]. Trivial queries might even be reduced to simply computing distances between candidates and a single ideal point that summarizes each query. However, many queries will not be that simple.

As the size and diversity of multimedia databases increase, so does the potential complexity of queries. Query concepts such as, ”Return all video clips showing any US President speaking”, may be be easy to specify as far as a human being is concerned, but difficult for the database due to the complexity of the “US President” concept. Without annotations or a truly customized distance function, this query would most likely be impossible.

Even queries that deal only with readily extracted “in-object” information can be quite complicated. Consider a database of still images. “Pictures of sunsets” may actually be a possible query even without annotations, since they often include high amounts of red and orange, which would be reflected in both the image itself and in certain common image feature spaces such as color histograms. However, it would be unsafe to assume that *all* sunsets would be in one cluster reducible to a single ideal query point, or that an untuned distance function would necessarily perform well when comparing that ideal query point – if it even exists – to candidate objects.

Another complication is that, when good pairwise distance functions exist, what they actually are may be hidden. For instance, if one resorts to a commercial package for computing distances between images, the function used may well be a black-box which hides even the identity of features used, let alone how. Consequently, a general approach to query-based distance function design should avoid feature-level computations; hence, our emphasis on example-driven relevance feedback, where users specify and tune queries through selecting and weighting examples.

The FALCON system is such a flexible approach. It combines distances and incorporates user feedback in such a way as to "learn" the nature of reasonably complex queries. This *aggregate dissimilarity* model applies to metric data sets, since it does not use any information about the data itself aside from that returned by a pairwise distance metric.

2.2 Related work

A number of systems such as the following have been developed to handle example-based queries. Note they handle neither unusual disjoint queries, nor arbitrary metric spaces.

- Rocchio's relevance feedback mechanism [31], which generates hyper-spherical isosurfaces in feature space.

- MARS (Multimedia Analysis and Retrieval System) [29, 33, 34], which includes a query expansion model that can weight features from multiple objects. However, MARS limits the sums of weights, so that when using fixed thresholds it becomes difficult to specify a disjunctive query in which being similar to any one of the examples is sufficient to be considered good.

- MindReader, which uses the Mahalanobis distance to allow arbitrarily oriented ellipsoids [19]. The Mahalanobis distance $M(\vec{x}, \vec{y})$ is defined as $(\vec{x} - \vec{y})^T \times \mathbf{M} \times (\vec{x} - \vec{y})$ where $\vec{x}$ and $\vec{y}$ are n-dimensional column vectors and $\mathbf{M}$ is a $n \times n$ matrix. This corresponds to a weighted Euclidean distance, and permits effective rotation of the axes, but requires many examples to calculate the covariance matrix.

The original assumption behind the earliest systems is that there exists an ideal query vector. One can then try to determine both this vector and the optimal relative weights of the axes. This dependence on a vector space prevents these methods from generalizing to metric spaces.

Later systems, such as current versions of MARS, have query expansion models which permit actual elaboration by weighting the relative importance of different features of multiple positive examples [29]. However, one should note that MARS has been specialized for image databases with features, whereas MindReader generates isosurfaces consisting of single hyper-ellipsoids, and therefore does not handle disjunctive queries.

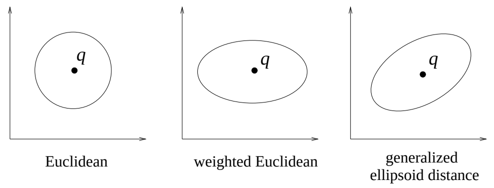


Figure 1: Generic isosurfaces for three previous methods.

See Figure 1 for an illustration of these types of isosurfaces.

Fox and Salton used the L_p metric to surpass fuzzy Boolean methods in the domain of text retrieval, replacing the use of minimum and maximum functions. FALCON's objective is similar, but more complex; it is to model arbitrarily disjunctive example-based queries in unbounded metric spaces using relevance feedback but lacking specific features [35].

FALCON does not rely on vector spaces, require user input beyond that of relevance feedback and examples, or sacrifice the ability to use disjunctive queries that correspond to arbitrary groupings in metric spaces. In addition, advanced indexing methods can be used to significantly speed up the search process. Spatial indexing methods such as R-trees [16] and R*-trees [3, 7] would serve if we were to limit ourselves to vector domains; M-trees [12] provide fast range-queries in general metric spaces.

2.3 System Design

This section describes the underlying mechanisms and assumptions made by FALCON. Table 1 lists the notation used in this section.

Let $\mathcal{X}$ be the set of objects in the metric data set. Then, let $d : \mathcal{X} \times \mathcal{X} \rightarrow \Re$ be the provided distance metric that defines the metric space.

To start each query, the user specifies at least one "desirable" example that is representative of the intended query. This set of "good" examples is denoted by $\mathcal{G}$. Thus:

Problem 1: Query by Multiple Examples	
Given	$\mathcal{X}$, a metric data set
	d, its pairwise distance metric
	$\mathcal{G} = \{g_i\}$, the set of "good objects"
Find	Other desirable objects in $\mathcal{X}$ that are similar to $\mathcal{G}$.

FALCON solves Problem 1 by determining a scoring function that models the user's query. Namely, it seeks a function $D_{\mathcal{G}} : \mathcal{X} \rightarrow \Re$ based on $\mathcal{G}$, such that this function varies inversely with the desirability of x.

Symbol	Description
$\mathcal{X}$	The set of objects in the data set.
x	Any single object from $\mathcal{X}$.
$\mathcal{G}$	The current set of user-specified "good points".
g_i	A member of $\mathcal{G}$.
$D_{\mathcal{G}}$	The aggregate dissimilarity function based on $\mathcal{G}$.
$D_{\mathcal{G}}(x)$	The aggregate dissimilarity value for object x to the current good set $\mathcal{G}$.
α	A constant that influences how $D_{\mathcal{G}}$ behaves.
d	The pairwise dissimilarity function.

Table 1: Notation used within this section.

2.3.1 The "Aggregate Dissimilarity" function

Given function that fulfills the main requirement – ranking objects inversely according to their apparent desirability as compared to $\mathcal{G}$, as listed below in Problem 2 – the problem of finding relevant objects reduces to that of sorting.

Problem 2:	**Aggregate Dissimilarity**
Given	$x \in \mathcal{X}$, a candidate
	$\mathcal{G} = \{g_i\}$, the set of user-selected "good objects"
	d, pairwise distance metric for $\mathcal{X}$
Determine	$D_{\mathcal{G}}(x)$, its aggregate dissimilarity

FALCON aggregate dissimilarity, $D_{\mathcal{G}}(x)$ is defined as the α^{th} root of the arithmetic mean of the α^{th} powers, of the pairwise distances, as expressed by

$$(D_{\mathcal{G}}(x))^{\alpha} = \begin{cases} 0 & \text{if } (\alpha < 0) \bigwedge \exists i \; d(x, g_i) = 0 \\ \frac{1}{k} \times \sum_{i=1}^{k} d(x, g_i)^{\alpha} & \text{otherwise} \end{cases} \tag{1}$$

If the value of α is very high, the highest distance will have the largest impact on $D_{\mathcal{G}}(x)$, while the reverse is true for very low values of α.

Note that Equation 1 mimics a fuzzy OR if $\alpha < 0$, and a fuzzy AND if $\alpha > 0$.

Since it is not obvious which values of α are the most suitable, we empirically compared results for various values of α. Experiments showed that $\alpha = -5$ is a reasonable choice for the queries and datasets selected.

Were a user to select parrot photos via query-by-example, the following might ensue.

1. The user chooses a data set, $\mathcal{X}$, consisting of bird photos. This data set is paired with a pairwise distance metric d, which relies primarily on coloration when comparing images.

2. The user first chooses the image of a popular green-and-red parrot. This becomes the first member of $\mathcal{G}$.

3. FALCON returns the best matches, according to $D_{\mathcal{G}}$; note that grey parrots might be ranked below cardinals – which are mostly red – and peacocks – which tend to be green.

4. The user adds a picture of a grey parrot to the good set, which alters the aggregate dissimilarity function $D_{\mathcal{G}}$. Other images of grey parrots will now be considered more relevant to the query.

5. Repeat as desired; depending upon the data and the metric, convergence may be fairly rapid.

One useful generalization of the FALCON distance is to allow for weights. Unlike a distance "combination" function that only uses the minimum distance, the FALCON distance is easily modified to accept a positive w_i term for numerical feedback from a user. The proposed extension takes the form:

$$(D_{\mathcal{G}}(x))^{\alpha} = \frac{1}{\sum_{i=1}^{k} w_i} \cdot \sum_{i=1}^{k} w_i (d(x, g_i))^{\alpha} \tag{2}$$

This has the effect that distance to the favored objects is penalized less if α is negative. FALCON does *not* raise the weights to the α^{th} power, as this would have the opposite effect and might be be far less intuitive. In addition, FALCON retains the influence of all pairwise distances between candidate and examples in all cases except that of an exact match, unlike a pure minimization function.

2.3.2 Speed and completeness

There is the obvious question of speed. Sequential scanning would entail computing aggregate dissimilarity via Equation 1 or 2 for each element of the database, which would require $\Theta(nk)$ time given n objects in $\mathcal{X}$ and k in $\mathcal{G}$. The objective is to return the same results as a sequential scan, with less work on average per search.

Indexing structures which support fast range queries can be used to achieve significant speed-ups *without* modification of the indices. First, shift focus to the aggregate dissimilarity version of the range query:

Problem 3: Range Query by Multiple Example	
Given	$\mathcal{X}$, the database
	$\mathcal{G} = \{g_i\}$, the set of "good objects"
	ϵ, a threshold
	d, pairwise distance metric for $\mathcal{X}$
Find	Q, such that $Q = \{x : x \in \mathcal{X} \wedge D_{\mathcal{G}}(x) < \epsilon\}$ quickly

Theorem 1 *Consider Problem 3 above. Executing $k = |\mathcal{G}|$ separate range queries with threshold ϵ, will yield k sets, the union of which forms a superset of the actual answer. No object will be falsely discarded as a candidate by such a procedure.*

The proof is omitted for the sake of brevity, but may be found in [45].

This theorem shows that FALCON can use existing indexing methods without fear of false dismissals. A post-processing step can then check every member of the union and discard any whose actual aggregate dissimilarity exceeds the threshold. Faster algorithms may well be possible if, rather than rely on standard k-NN and range-query APIs for an indexing structure, custom search methods are implemented.

The question remains of selecting a suitable ϵ. One method would be to allow the user to flag examples that they consider to be about as dissimilar as they would accept; then, an ϵ can be generated as the maximum $D_{\mathcal{G}}$ for these borderline cases.

2.4 Experimental setup

The core parts of FALCON were implemented in C, C++ and Perl, and tested on Intel Pentium II^{TM} workstations under Linux.

FALCON was tested with the intent of answering five central questions.

(a) Does FALCON "learn" to model concave and disjunctive queries?
(b) Does FALCON provide satisfactory precision/recall?
(c) Does FALCON's distance function rapidly converge?
(d) What is a suitable value of α?
(e) How fast is FALCON?

Four data sets were used during the experiments, each with exactly one query. Two of the data sets were synthetic, and two consisted of real data. FALCON used the standard Euclidean distance as the pairwise distance metric; with the $2D\_20K$ data set, a few experiments were also run with L_∞.

Vector data sets were used primarily due to ease of generating consistent and objective feedback results. The FALCON system never examined the vectors themselves, as it relied solely on the results from the pairwise distance function.

2D_50K: This synthetic data set consists of 50,000 points in 2-dimensional Cartesian space, randomly distributed approximately uniformly within the axis-aligned square (-2,-2) - (2,2).

2D_20K: This synthetic data set was generated using identical rules to that of the 2D_50K data set, but consists only of 20,000 points.

PEN: This database was obtained from the UCI repository [26], and consists of objects that correspond to handwritten digits, with features being the classification of each and the coordinates of spatially re-sampled points. The existing split of 3498 objects in the training set and 7494 in the test set was retained.

STOCKS: Daily closing prices for up to five year periods were collected for 51 stocks from Yahoo's online quote server. They were split into 1856 non-overlapping vectors of length 32, which were then processed via the discrete wavelet transform (DWT). All 32 coefficients for each vector were used for L_2 distance computations.

2.4.1 Queries

One implicit query was associated with each data set. Queries were reflected via a seed set of five initial "good" objects, and appropriate feedback for each data item. A subset of each data set was selected as training sets, used for query refinement; the rest of the data served only for evaluation. It may in fact be feasible to implement a hierarchical view of a database – perhaps traversing something like an M-tree – in order to allow the user to easily select examples without wading through an entire database at once [12]. In addition, tests varied α as there was no *a priori* reason to believe that one particular value would be optimal.

RING: Vectors within the 2D_50K data set were marked as positive examples if and only if they were between 0.5 and 1.5 units from the null vector, inclusive. The training set consisted of 1,000 vectors. 431 in the subset and 19,734 in the full set met this criterion. The five seeds were vectors of magnitude 1.4, with counter-clockwise angles from the positive X axis of 0, 72, 144, 216, and 288 degrees.

This query to tests performance on a contiguous, but non-convex set.

TWO_CIRCLES: Vectors within the 2D_20K data set were marked as positive examples if and only if they were within 0.5 units of either (-1,-1) or (1,1); 1899 points qualified. The subset consisted of 1000 points randomly drawn from the full set, including 99 positive examples. The seeds were randomly generated from within the two circles.

This query tests performance on a disjunctive set.

PEN: Vectors within the PEN data set that were classified as 4's were flagged positive. The existing training and test partition was used. 364 vectors in the training set and 780 in the full set were positive instances. The five seeds were drawn randomly from the positive instances in the training set.

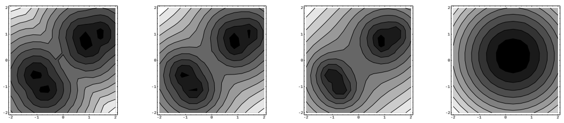
This query tests performance on real data.

STOCKS: Vectors within the STOCKS data set were marked as positive examples if and only if the slopes of their least-squares linear approximations were within the range -0.02 to 0.02. The training set consisted of 500 examples, of which 153 were positive. 540 in the full set were flagged positive. The five seeds were the vectors of DWT coefficients of five perfectly flat lines with y-intercepts of 8, 40, 80, 200 and 400.

This query also tests performance on real data.

2.4.2 Evaluation methodology

For each data set and its paired query, tests were run the following values of α: $-\infty$, which scores by minimum distance; -100, which approximates that; -10; -5; -2; 2; and 5.



$\alpha = -\infty$ $\quad$ $\alpha = -5$ $\quad$ $\alpha = -2$ $\quad$ $\alpha = 2$

Figure 2: Contour plots for the seeds of the TWO_CIRCLES query.

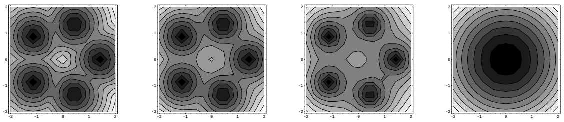
Figure 2 shows contour plots for four values of α in the TWO_CIRCLES. Note that $\alpha = 2$ reflects only the center of the "good set", even on the TWO_CIRCLES query, and hence is expected to fail on such a disjunctive query.

Figure 3 shows contour plots for four values of α in the RING. Again, $\alpha = 2$ stands out with contours that are not going to rank the points very well.

FALCON ranks objects by computed score, which permits evaluation via precision/recall. Define recall and precision as follows.

For arbitrary n, consider the n top-ranked objects. Let there be p positive examples among them, out of P positive examples total. Then recall is $\frac{p}{P}$, and precision is $\frac{p}{n}$. One can compute precision for any given level of recall for a full ranking by choosing n appropriately.

With each combination of a query and a value of α, the following procedure was used:



$\alpha = -\infty$ $\alpha = -5$ $\alpha = -2$ $\alpha = 2$

Figure 3: Contour plots for the seeds of the RING query.

- Start with the seeds as the "good set". Compute precision/recall values over the full set. This gives us a baseline that varies only with the contours generated by α.
- Repeat the following as needed:
  - Find the top twenty which have not yet been picked for feedback. Twenty is arbitrary, but plausible.
  - Add any newly-found positive examples into $\mathcal{G}$.
  - Repeat the precision/recall procedure on the full set.

2.4.3 Speed

Range-query speed tests were also performed. The query and data involved were that of TWO_CIRCLES as described previously. M-trees served as the indexing structure [12]. Sequential scanning provided a baseline with which to compare the method described in Theorem 1. Tests were then measured the individual effects of threshold and number of seeds on elapsed time and computational cost. Note that the elapsed time includes everything done on a per query basis, including all range queries as well as the time required for merging and verifying the results.

2.5 Results and discussion

This section presents an analysis of the results, organized with regard to the five central questions.

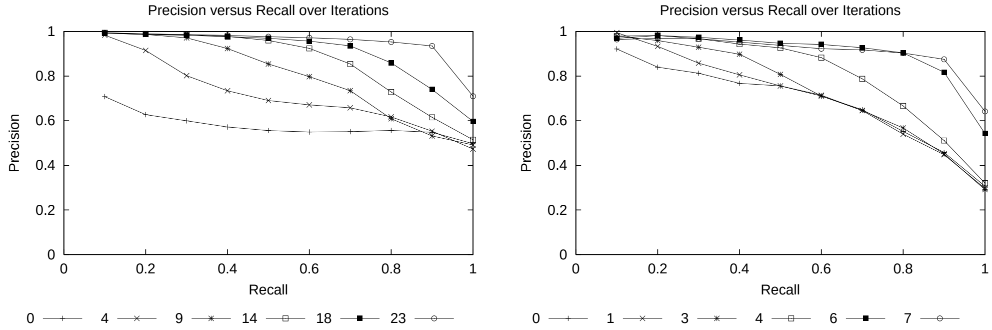
The following notes apply to the various graphs. For all the figures marked "Precision versus Recall", such as Figures 4, and 6, one line is plotted per charted iteration. Not all iterations were charted, for purposes of readability. Each line is drawn with ten points, each of which shows precision at one level of recall from 10% to 100% at 10% increments. Thus, the general trend of the lines tracks the progress of FALCON as feedback is provided on increasing numbers of points.

Any figure that tracks "Precision versus Iterations", such as Figure 7 instead focuses on precision at one level of recall, 40%, for multiple values of α. Positive slopes indicate positive progress. 40% was arbitrarily chosen as a level at which it is non-trivial, but also not extremely difficult, to provide good precision.

Figures 8 and 9, labelled "Precision at Multiple Levels of α" also track precision, but after 5 and 20 iterations, respectively. The first shows precision at 40% recall; the second, average precision based on all 10 levels of recall.

2.5.1 Concave and disjunctive queries

The RING query is concave, but contiguous; the TWO_CIRCLES query is disjunctive. As one sees in Figure 4, FALCON can handle either data set with high levels of precision versus recall. This is a substantial improvement over existing methods such as MARS and MindReader, which would have difficulty with these two.



RING TWO_CIRCLES

Figure 4: Precision versus recall for $\alpha = -5$, for RING and TWO_CIRCLES using Euclidean pairwise distance.

The numbers in the legend indicate the number of feedback iterations to achieve that level of progress. Observe that in both cases, precision largely stabilizes after the first several iterations – especially for TWO_CIRCLES.

The success on the TWO_CIRCLES query is particularly notable because that set is completely disjunctive; there are two distinct regions of points that deserve high rankings, separated by points that do not. To further test the system, the same query was also run with substituting L_∞ for the Euclidean distance as pairwise metric d. Figure 5 shows the resulting precision-recall.

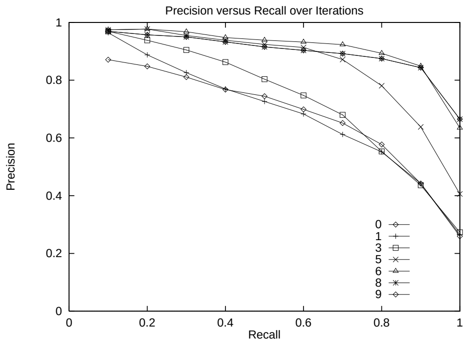


Figure 5: Precision versus recall for $\alpha = -5$, for TWO_CIRCLES with L_∞ as distance metric d

2.5.2 Quality of results on real data

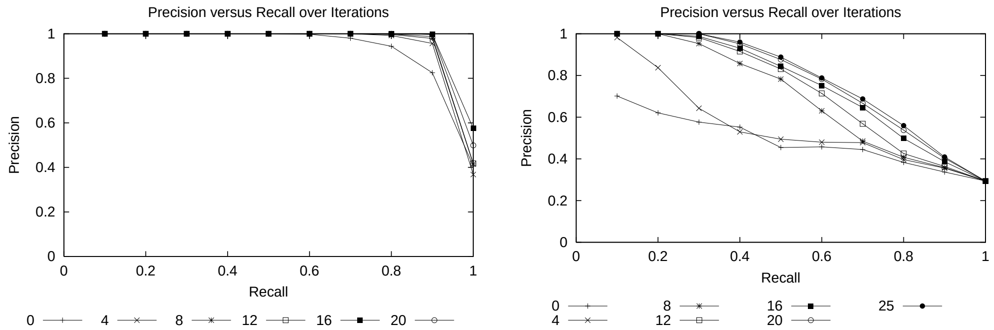
Both the PEN and STOCKS queries are reasonable queries on real data. Consequently, FALCON's performance on these, as shown in Figure 6, is relevant to any question as to whether FALCON "works" on real data.

Note that, for the PEN query, convergence at all levels of recall below 100% is rapid; the rest of the iterations after the fourth iteration do not add precision except at the highest level of recall. The pattern with the STOCKS query is more interesting, with large gaps between the lines. One plausible explanation is that the positive examples were not evenly distributed in vector space, but instead clustered. The first member of a cluster to be added to $\mathcal{G}$ would immediately cause everything nearby to move upwards in the rankings. This is particularly plausible given that many of the stock vectors were consecutive slivers from the same stock over time, and therefore may have had similar properties.

Figure 6 shows that FALCON provides good precision at almost all levels of recall. In particular, it provides perfect precision at most levels of recall for the PEN data set and query, identifying objects that correspond to 4's. Note also that FALCON provides good precision on the RING and TWO_CIRCLES queries, as shown in Figure 4, as cited above.

2.5.3 Speed of convergence

Figure 7 show precision at 40% recall versus the number of iterations for the PEN and STOCKS queries; corresponding results for the synthetic RING and TWO_CIRCLES queries are similar, and may be found in [44]. FALCON can quickly attain high levels of precision over high levels of recall, even with an early $\mathcal{G}$ that has not yet grown to include many of the "good points" in even the training set.



PEN STOCKS

Figure 6: Precision versus recall for $\alpha = -5$, for PEN and STOCKS.

The STOCKS query is again unlike other queries. Here, there are very wide gaps in performance until 12 iterations have elapsed; the others show more continuous gains.

Depending upon the complexity of the query and the data set, progress can be very fast (such as in the PEN data set), or slower (as in the RING data set). For 40% recall, precision exceeds 90% after only 4 to 11 iterations for $\alpha = -5$ on all four queries.

There can be overfitting, as seen in the TWO_CIRCLES data set. The law of diminishing returns applies to increasing $|\mathcal{G}|$, suggesting that one does not need to maximize $|\mathcal{G}|$ to attain near-optimal levels of precision, and therefore sampling might be a reasonable approach.

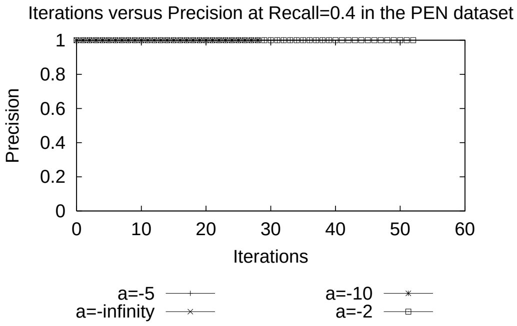
2.5.4 Optimal value of α

An initial hypothesis was that $\alpha = -5$ would be a reasonable choice from the set of possible α's. $\alpha = -100$ generally performed significantly worse due to numerical precision problems, and therefore will not be discussed further.

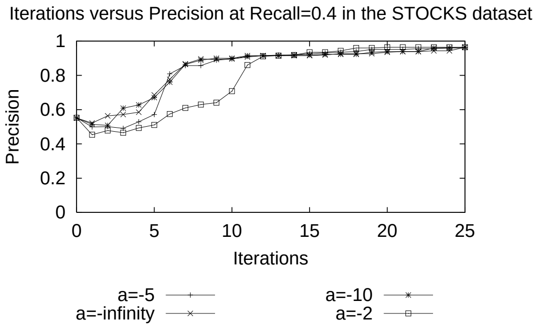
Precision was generally similar for all negative values of α, especially once $\mathcal{G}$ had reached its maximum size. With positive values of α, precision was quite low at most levels of recall; see Figures 8 and 9 for some results.

The first figure shows precision at 40% recall for each level of tested α; each line tracks the precisions yielded by one particular value of α for the different queries. This allows us to compare α both early and late in the process. In the case of TWO_CIRCLES, whose $\mathcal{G}$ stabilized in fewer than 20 iterations, final results were used. Observe that the differences among the negative values of α largely disappear by the 20^{th} iteration of feedback.

As one can see in the Figure 9 difference in average precision over all levels of recall after 20 iterations are a bit more marked than at 40%, but again the negative values of α yield



PEN



STOCKS

Figure 7: Precision versus iterations at recall=40%.

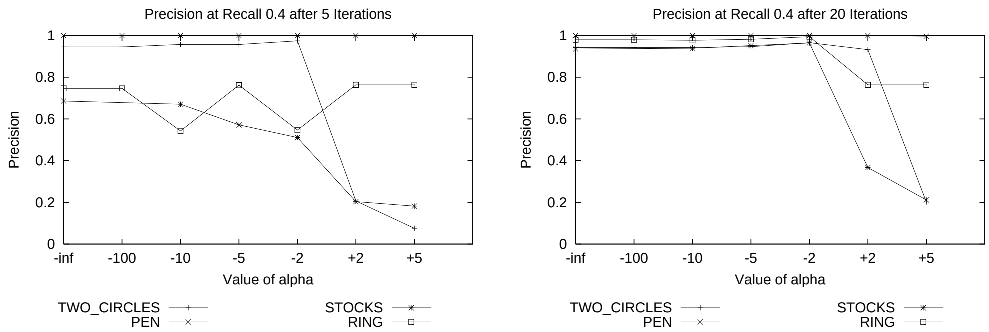
fairly similar average precision.

$\alpha = -5$ still seems reasonable, as it provided good performance as empirically observed, and in no case is it significantly inferior. Similar values appear to yield similar results. This may be due to the limited range of values tested, but also reflects the fact that this value permits a significant amount of flexibility, in that this corresponds to a quite fuzzy OR.

2.5.5 Speed

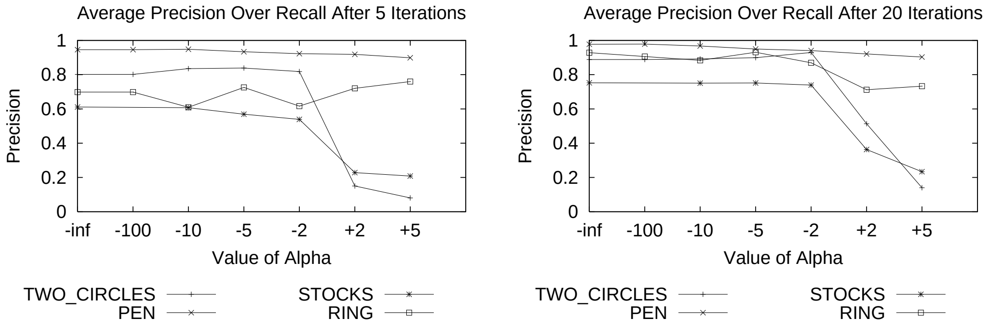
Experiments showed that merging the results of k separate range queries as per Theorem 1 is satisfactory in terms of both the number of distance computations, and the elapsed time. Some of these results may be noted in Figures 10 and 11.

The two graphs in Figure 10 compare the elapsed (real) time and the pairwise distance computation costs incurred in searching the TWO_CIRCLES training data set for points within a variable aggregate dissimilarity threshold from the seeds. These results demonstrate



After 5 Iterations　　After 20 Iterations

Figure 8: Precision at 40% recall



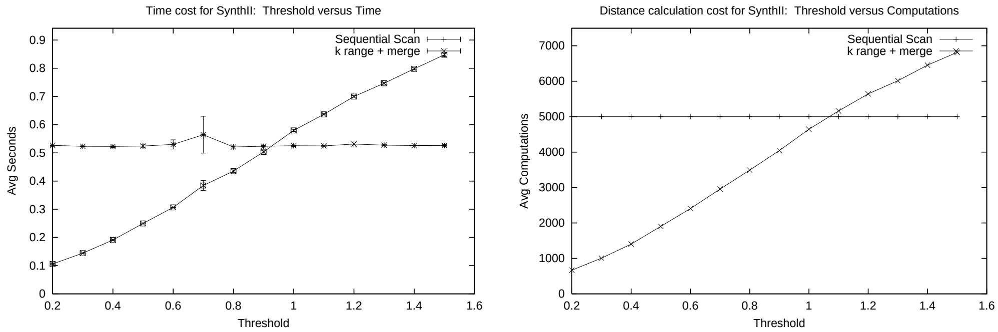
After 5 Iterations　　After 20 Iterations

Figure 9: Average precision at multiple values of α.

that the k range queries can be merged into one.

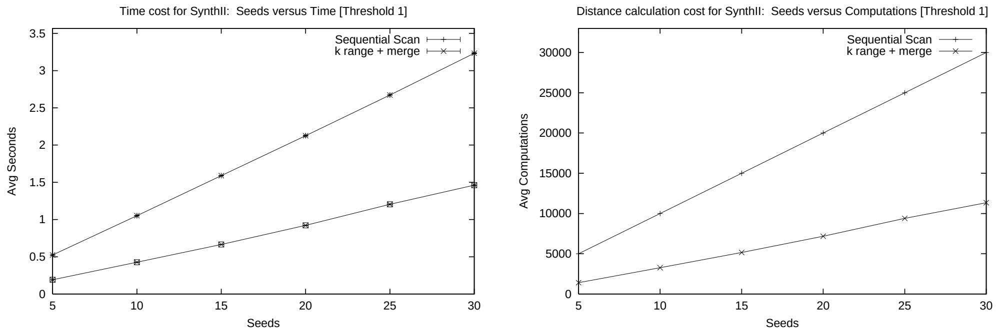
Note that the cross-over point where sequential scan performs as quickly as merging occurs at a threshold of approximately 0.7, which accepts approximately 20% of the database. It seems reasonable that users of large interactive databases will rarely be interested in queries of such scope, and thus the merging method is worthwhile.

As shown in Figure 11, both measurements of performance cost appear to scale with the number of seeds. These results are taken without caching the distance computations from previous searches; these searches were all independent. These two graphs compare the elapsed (real) time and the pairwise distance computation costs incurred in searching the TWO_CIRCLES training data set for points with aggregate dissimilarity less than or equal



Elapsed Time Distance Computations

Figure 10: Time and computational cost for TWO_CIRCLES, varying threshold.



Elapsed Time Distance Computations

Figure 11: Time and computational cost for TWO_CIRCLES, varying seeds.

to 1, varying the number of seeds.

The results for the same test, but with the underlying distance function $d = L_\infty$, are very similar and are not presented here [45].

No direct experimental comparisons are shown with previous methods. This is because the queries were specifically designed to include queries of disjunctive and other highly non-convex behavior in general metric spaces, and thus previous methods simply do not apply.

2.6 Conclusions

The FALCON method applies to general metric spaces, as the distance combination function depends on only the pairwise distances and not the actual nature of the data. In addition, it handles disjunctive and concave queries that are fundamentally *impossible* for traditional relevance feedback methods. Using this method, a user could, without any domain-specific query language, specify a disjunctive query merely by labelling as "good" objects representative of the different classes. This combination of general applicability, power and ease-of-use makes this method more valuable than other existing systems today.

The heart of the method is the FALCON aggregate dissimilarity, or $D_{\mathcal{G}}$, which is able to "learn" disjunctive queries via relevance feedback. Additional contributions include the following:

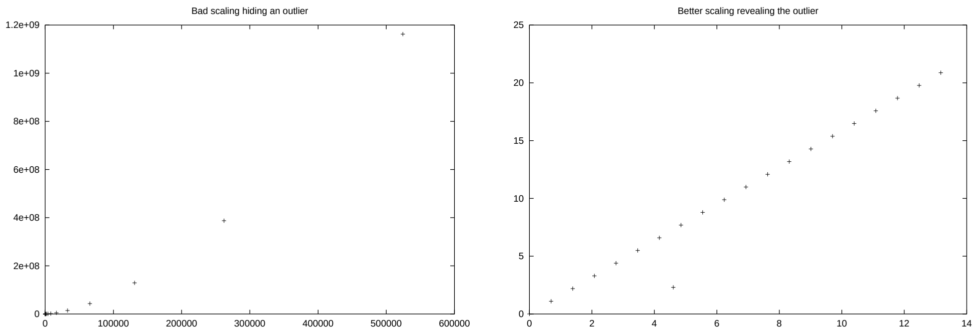
- Theorem 1, which shows that one can use indexing structures that support range queries, to speed up the search, guaranteeing zero false dismissals.
- Experiments on real and synthetic data, that show that the proposed method ("FALCON") achieves good precision and recall. For instance, with all queries, $\alpha = -5$ yielded at least 80% precision at 50% recall with 10 iterations.
- Experiments that show that FALCON needs at most 10 feedback iterations to reach high precision/recall, and will reach a "steady state" for all levels of recall below 100% within approximately 20 iterations. Beyond 20 iterations, minor progress is made for 100% recall.
- Experiments that show that query performance does not seem especially sensitive to α.
- Experiments that show that the method described by Theorem 1 yields gains of up to 200% observed performance improvements versus sequential scanning.

3 Data-driven methods

Data-driven methods consider distance function design without direct intervention of a user. In many situations, this would allow algorithms to operate during a preprocessing phase, instead of online, and thus can devote substantial amounts of CPU time to this goal. On the other hand, they cannot rely on any form of user feedback, and any distance function they return needs to be broadly applicable within that database.

3.1 Motivation

The assumption that the Euclidean distance is already acceptable is inherent in most other scaling problems. The work described here instead focuses on determining what is, in fact, a good distance function for data; isomorphically, what space is appropriate for any given data. Motivating this problem are situations such as that shown in Figure 12. In (a), the common concept of distance-based outliers fails completely, as the distances between points are exponential. Remove the furthest point as an outlier, and the new furthest point also looks like an outlier, and so forth. Version (b) shows that, in fact, the distances between points are quite regular – except for a now-revealed outlier.



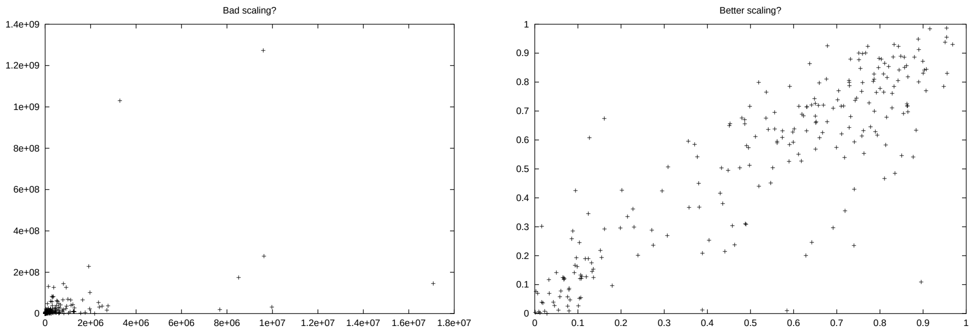
(a) Original scaling (b) Post-logarithm

Figure 12: Two versions of the same artificial data, where most of the points are of the form $(2^i, 3^i)$ for $i \in [0, 19]$. Version (b) simply presents the same data as in (a), but after applying a natural logarithm to both axes.

Also consider Figure 13, which shows one data set presented in two different sets of scales. The exact scaling methods used need not be specified; suffice it to say that in both cases, the original data could be reconstructed without error given the methods, and that any mathematical rules learned in either space could be transformed to rules in the original space.

One might suggest that version (b) is generally superior. Even a cursory visual examination suggests the existence of a linear rule between the axes, and the space is no longer dominated by outliers. One would expect that this version would be much more readily processed by data mining algorithms.

Notice that (b) has a vastly different structure than (a); it is much more linear, the outliers have been brought closer, and the data is much more uniform on each axis. Traditional scaling methods would have preserved the structure of (a), thus retaining a bad distance function. Distance functions, in turn, are critical to many problems such as detecting outliers, determining nearest neighbors, clustering, and others – almost everything that can be done with vectors requires making judgements as to how far two vectors are from each other. Thus,



(a) Bad scaling? (b) Better?

Figure 13: Two versions of the same data. Which is better? More importantly and less obviously, *why*?

trustworthy, well-grounded distance functions are absolutely *vital* – and most algorithms simply assume that this problem has already been solved.

3.1.1 Intuition

Intuition leads us to point out three major aspects to bad scaling – skew, or lack of uniformity, along an axis; badly matched ranges between attributes; and redundant expression of latent variables in the attribute set.

One issue is skew, the inverse of uniformity. The more uniform a distribution, the less influence outliers will have, and the easier it will be to deduce rules. For instance, linear regression is a simpler technique than polynomial regression, and it'll perform much better if the nonlinearities in rules have been transformed away.

In addition, it may well be desirable for attributes to have similar ranges. Radically different ranges, such as those resulting from heterogeneous attributes, may result in simple distance functions like the Euclidean distance essentially ignoring the lesser in favor of the former. In medical data, for instance, patient body temperatures would be expected to have a far smaller range, in degrees Celsius, than would body mass in kilograms. Treating them with equal weight would be a serious mistake, as minor differences in mass would completely overwhelm major differences in body temperature as far as a Euclidean distance would be concerned.

Redundancy is another issue. The impact of redundant, highly correlated attributes on performance and storage is obvious. Its impact on correctness may not be. Consider what will happen if one latent variable is strongly represented in ten attributes – a Euclidean distance would take this variable into account ten times, thus giving it an undue impact.

Of these three issues – skew, mismatched ranges, and redundancy, the second is the easiest to deal with. Affine normalization schemes based on the standard deviation, the mean absolute deviation, and other metrics are ubiquitous. However, these ignore the other two problems. Redundancy is a difficult problem, since ultimately it involves determining the true number of latent variables; and skew is usually ignored or preserved instead of corrected.

3.1.2 Informal specification

Generally, one could specify the problem as: Given a set of n vectors in $\mathcal{R}^m$, produce through invertible transformations a second set of n vectors in $\mathcal{R}^m$, with increased uniformity, well-matched ranges, and minimal redundancy. Compound this with dimensionality reduction, and then instead of $\mathcal{R}^m$, the new set should be in $\mathcal{R}^{m'}$ where $m' \leq m$. In experiments performed to date, m' is specified implicitly as the user provides a stopping criteria for attribute selection. Deducing an "optimal" m' is beyond the scope of current work.

Intuitively, a good scaling and dimensionality reduction algorithm should scale the retained attributes in such a way that the attributes which are not retained could be reconstructed with minimal error, while not using such highly parametric models as to unduly increase total storage required.

In addition, not all transformations are reasonable choices. Limiting transformations to those that preserves spatial ordering seems reasonable, as the intent is not to obliterate structure, but merely to adjust it. If one visualizes a space as an elastic solid, uneven stretching is permitted, but the solid may not be folded back upon itself.

However, before any questions regarding transformations or models can be relevant, we must first determine how to measure uniformity and redundancy so that we can assess how transformations change them.

3.2 The importance of scaling

Herein are presented concrete examples as to the benefits of good nonlinear scaling. Consider the logarithmic transformation as that is a well-known, ubiquitious method for dealing with nonlinear distributions.

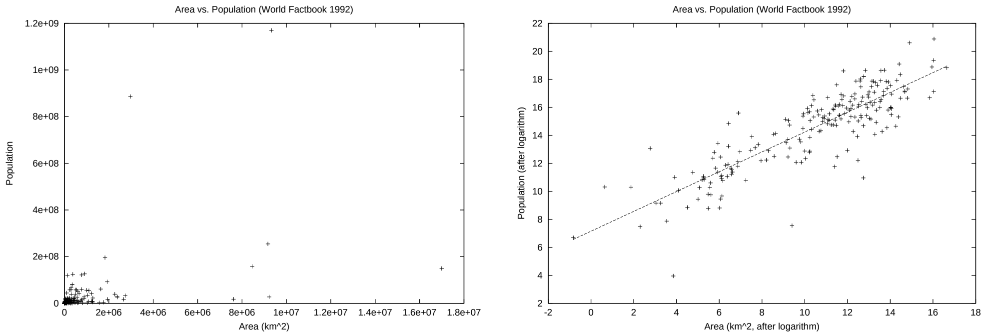
Consider Figures 14 and 15, which chart the area and population of many nations in 1992 and 2001, respectively. In both figures, (a) shows the raw data and (b) shows the data after discarding any vector with a zero, and then applying a natural logarithm. In both 1992 and 2001, least-squares linear regression on the raw data would be pointless, as the correlation coefficients are only 0.4776 and 0.4741, respectively. After the logarithms, linear regression yields, for population p and area a,

$$\ln p_{1992} = 0.7080 * \ln a_{1992} + 7.1501 \tag{3}$$

and

$$\ln p_{2001} = 0.7237 * \ln a_{2001} + 7.0277 \tag{4}$$

with correlation coefficient 0.8555 and 0.8495, respectively. Interestingly, the slopes are both reasonably similar, and supralinear. While one might expect population and area to be linearly related, this is evidently not the case – although, to be fair, this rule is only evident *after* nonlinear scaling.



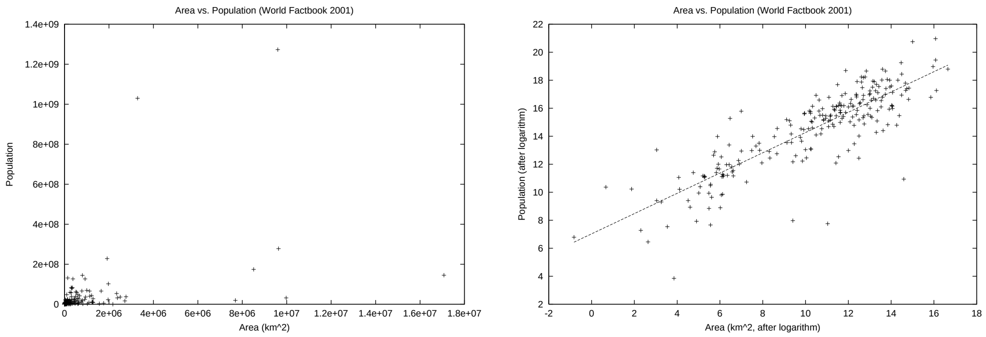
(a) Raw data. (b) Logarithmic axes.

Figure 14: National area and population in 1992, (a) raw and (b) scaled, with least-squares linear regression.

Figure 16 shows corresponding raw and logarithmic plots for two player statistics for the NBA 1991-1992 season, the number of games played versus the number of field goals made. Linear regression might be considered only somewhat dubious initially, with a correlation coefficient of 0.7569 on the raw data, but after discarding eight vectors corresponding to players who scored zero in either statistic and applying the natural logarithm to the rest, the following rule holds with a greatly improved correlation coefficient of 0.9109:

$$\ln f = 1.5080 * \ln g - 1.0032 \tag{5}$$

or $f \sim g^{1.5}$, where f is the number of field goals and g the number of games. Nonlinear scaling has thus revealed a superlinear rule. In fact, if one relied only on linear scaling, one

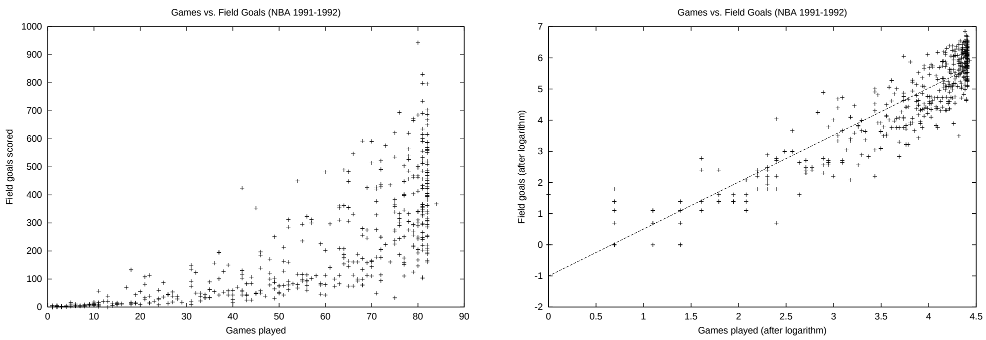


(a) Raw data.

(b) Logarithmic axes.

Figure 15: National area versus population in 2001, (a) raw and (b) scaled, with least-squares linear regression.

might have been misled by the 0.7569 correlation on the raw data, and accepted a much less correct linear rule.



(a) Raw data.

(b) Logarithmic axes.

Figure 16: Games played versus field goals scored by players in the NBA 1991-1992 season, (a) raw and (b) scaled, with least-squares linear regression.

Thus, nonlinear scaling can be vital for revealing rules otherwise hidden by bad scaling. Other nonlinear transformations, however, exist besides the logarithm. Blindly applying it without reason is little better than trying to preserve distances without justification. Later sections will describe what transformations were selected as candidates.

3.3 Related Work

Principal components analysis (PCA), and specifically the singular-value decomposition, is perhaps one of the most-used algorithms for scaling. It performs optimally among the space of linear transformation methods when used for minimizing second-order projection error [21, 27]. It also analytically determines rotated axes. However, it still performs only linear scaling. Many approximation methods [14] that share the same objective of preserving Euclidean distances also limit themselves to linear projections, and thus share the same problem; likewise for independent component analysis based on matrix multiplication [17, 18]. Nonlinear transformations are also beyond the Donoho-Stahel estimator, which adjusts the Euclidean distance to account for scatter [23]. This last work acknowledges that the Euclidean distance is not perfect, in that it attempts to avoid outliers. Thus, it uses a more robust measure.

The above methods are global scaling and dimensionality reduction methods, in that they apply the same transformations on the entire space. There are also local methods designed to apply the same principles to multiple regions with diverse properties [10, 24, 32, 36, 41].

All of the aforementioned models, however, share one particular philosophical goal: the preservation of structure, whether it be global or local. In most cases, this means preserving already-existing Euclidean distances. Thus, they only apply when the distances are *already* trustworthy, which is a dangerous assumption to make.

The SPARTAN algorithm can compress data using feature selection by modeling discarded attributes using CART trees with the retained attributes as possible inputs [1]. Neural network models can also deal with redundancy by encoding data into a lower-dimensional set, and then penalizing reconstruction error [13, 22, 25, 40]. Other nonlinear dimensionality reduction methods include principal curves [11] and kernel PCA [37].

3.4 Shannon information

Shannon information [39], also known as entropy, provides a way to quantify the intuitive concept of information. The formulas

$$H(x) = -\sum_i p_i \log_2 p_i \tag{6}$$

where p_i corresponds to a probability of a discrete outcome i, and

$$H(x) = \int -f(x) \log_2 f(x) dx \tag{7}$$

where $f(x)$ is the probability density function (PDF) evaluated at x, measure the entropy of a random variable x. The second form is also known as *differential entropy*.

In each case, entropy corresponds to the theoretical minimum expected number of bits required to transmit individual instances of the random variable.

3.5 Proposed method

While entropy may seem a logical choice, there are multiple problems with using either of Equations 6 or 7 as a heuristic for determining which scaling method is preferable.

1. Much raw data is continuous, and the continuous form of entropy computation requires density estimation.
2. If one discretizes the data, it's still high-dimensional. The number of buckets will scale exponentially with the dimensionality, and with real data sets this can be quite high. If the number of vectors is not large compared to the number of buckets, the buckets might be very thinly populated, and even minimal noise will be treated as information.
3. If one estimates the density or discretizes the data, it's adding more parameters, and it risks instability; what if a slight shift in parameters would result in a different number? Using too many intervals exaggerates noise, while too few smoothes data excessively.

Let us consider not one but a series of discretizations of the same data set, where each additional level of granularity halves the discretization interval along each dimension. Using multiple levels of discretization provides some measure of robustness, and a sufficiently large number of vectors would allow for discarding of the less reliable estimates at either extreme. Within a reasonable range, one might expect this progression to be linear – more precision, more information. Therefore, we can compute “the” derivative of information with respect to precision in general instead of fixing exact value of precision. Thus, the following definition:

Marginal information content: Define the marginal information content, or MIC, as the derivative of information with respect to precision. This corresponds to the amount of information gained for each bit of precision in each attribute.

Essentially, a MIC of x implies that within a reasonable range, adding an additional bit of precision in each attribute yields x additional bits of entropy. For instance, Figure 17 shows the precision-versus-information plot for a point-set consisting of 2,048 points evenly distributed along a line in a plane. The derivative of information, or MIC, is 1, meaning that each bit of additional precision along both axes yields another bit of information – within limits. Increasing the precision does not increase information if every vector is already uniquely distinguished, and likewise reducing precision does not reduce information if all vectors are already indistinguishable.

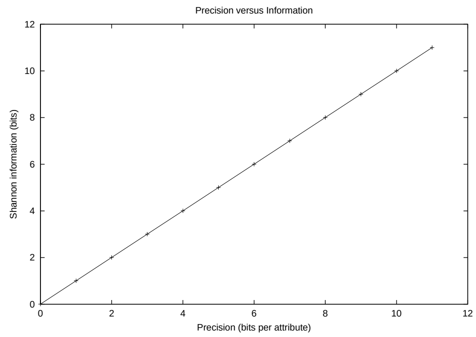


Figure 17: Entropy plot for a linear point-set.

This metric is *critical* because it allows us to discard the assumption that previously given distances are good, or that they would be if there were no outliers – and outliers are usually defined in terms of distances. It enables objective comparison of different spaces in a theoretically well-grounded manner.

3.5.1 Complete problem definition

Version 1: Axis Scaling Given a set of n vectors in $\mathcal{R}^m$, produce through invertible transformations a second set of n vectors in $\mathcal{R}^m$, with maximal MIC.

Version 2: Version 1 + Dimensionality Reduction Identical to Version 1, except that the new space is $\mathcal{R}^{m'}$ where $m' \leq m$. This means that need to retain only independent, high-information attributes.

There are two variations of this, depending on how m' is determined.

1. m' is supplied by the user, either explicitly – for instance, if the user only wants two attributes because he intends to plot the results – or implicitly, such as the user specifying other termination conditions that have the effect of limiting m'.

2. m' is based only upon the data. This problem is considerably more difficult, as "optimal" values of m' may vary based on the choice of criterion.

This work focuses on the second version, axis scaling and dimensionality reduction, with implicitly specified m'. A minimum-description-length (MDL) approach may be possible for determining m', but even MDL approaches require parameters such as how to penalize errors [30].

3.5.2 Fractal dimension

This concept is closely related to the fractal dimension. Specifically, the D_1 fractal dimension [4, 42] is

$$D_1 = \frac{\partial \sum_i p_{i,r} \log_2 p_{i,r}}{\partial \log_2 r} \tag{8}$$

where $p_{i,r}$ is the fraction of pairs of vectors within distance r of each other, which bears more than a passing resemblance to Equation 6. Suppose one normalizes continuous data so it fits within a unit hypercube. Then, for any given radius r we may discretize it by dividing that unit hypercube into smaller hypercubes of side length r along each dimension. Let H_r be the entropy of the discretized version with respect to side length r. Then,

$$H_r(x) = -\sum_i p_{i,r} \log_2 p_{i,r} \tag{9}$$

As per [38], we can replace Equation 8 with:

$$D_1 = \frac{-\partial H_r(x)}{\partial \log_2 r} \tag{10}$$

In addition, with such a discretization, $\log_2 r$ is precisely the number of bits of precision per attribute. Hence, this is the derivative of information with respect to precision, or the marginal information content.

The D_1 fractal dimension is but one of a series of generalized fractal dimensions [4], including the more famous D_0 Hausdorff fractal dimension. They have been used as estimators for intrinsic dimensionality [2], another concept which intuitively corresponds to the number of degrees of freedom present within data. That a *single* number corresponds to both marginal information and to degrees of freedom reinforces the belief that this is a suitable measure for the goodness of a given scale.

The MIC is superior to raw entropy as a measure of the goodness of scaling. It retains the vulnerability with regards to high dimensional data, which is unsurprising given the difficulties that the curse of dimensionality poses to even a simple Euclidean distance [5]. However, if and when this happens, it is trivial to notice since very few meaningful points will be generated on the line that tracks precision versus entropy.

The MIC imposes a self-similarity requirement, in that the curve will not be linear nor the slope be meaningful otherwise. This may be less of a limitation than it seems, in that self-similar data is not unusual, and discarding unreliable estimates may not have much of a negative impact on an algorithm.

As for computational complexity, the box-counting algorithm suffices for estimating the MIC in time linear with both dimensionality and cardinality [4].

Another direction from which an information-theoretic viewpoint makes sense is that of maximizing uniformity. When the range is constant, uniformity maximizes information. Therefore, scaling methods that increase entropy should also increase uniformity.

3.5.3 Transformations

Now that MIC has been established as the logical metric, it is now reasonable to describe how to create attribute spaces to compare. For computational and implementation reasons, the search is constrained to the space of transformation methods in which transformations are applied to individual attributes, and no rotation is used. The MIC will thus be used to decide which versions of which attributes are accepted or rejected.

What transformations should one choose? Recall that we wish to address skew, mismatched ranges, and redundancy. The first of these problems dictates that we use nonlinear transformations, while the second can be dealt with by using affine transformations as need be. In addition, we require that transformations be monotonic, so that ordering is preserved.

See [43] for the list of transformations chosen. To summarize, almost all of the transformations chosen were cumulative distribution functions of distributions with common, actual real-world applications [20], with the one exception being the simple application of a natural logarithm, with a shift if necessary.

3.5.4 Selection

Suppose the data set is $A = \{\hat{a}_i\}$ where each $\hat{a}_i$ is a continuous attribute. In addition, define the transformation set as $T = \{t_j\}$, where $t_j : \mathcal{R} \mapsto \mathcal{R}$. Then, each attribute $\hat{a}$ be retained in exactly one transformed version $t_j(\hat{a})$, or it may be dropped entirely as a candidate for reconstruction from retained attributes. Since the current system requires exactly one transformation per retained attribute, the search space scales linearly with both cardinality and dimensionality.

The prototype used a forwards-selection search, specifically a more flexible variation of what was used before [42]. The algorithm starts with no attributes retained, and at any iteration the search may perform one of the following three steps:

1. Add a transformed version of an attribute.

2. Swap a transformed version of an attribute for another. The latter may or may not be another version of the same underlying original attribute.

3. Stop.

The first is constrained so that no two versions of the same attribute are ever simultaneously considered "retained", as additional information should be minimal and checking is a waste of time; these states can be discarded immediately. The "goodness" of such a step is measured as the increase in the MIC criterion; higher is better.

The second shares the same constraint and "goodness" measure. Bias was added to the algorithm in that if an addition and an exchange produce identical gains in MIC, it prefers the exchange as it does not increase the number of attributes. Determination of which pairs are involved is itself greedy; it picks the attribute whose removal produces minimum MIC penalty, followed by an addition to greedily maximize MIC gain.

If neither additions nor exchanges produces a reasonable gain, then the algorithm terminates. In all tests, the code required a minimum MIC gain of 0.1 for iteration to continue.

As for complexity, exchanges can largely be ignored as they rarely happen. On the i^{th} iteration, $i-1$ attributes have already been selected. Of the others, $(t-1)(i-1)$ are simply other transformed versions of these attributes, and are therefore disqualified. Each of the remaining $tm-(t-1)(i-1)$ candidates will result in a fractal dimension computation of order $O(in)$. There are k iterations that involve selections, and a $k+1$ iteration that checks selections but finds that none are acceptable. Then, the total elapsed time $T(t,m,n,k)$ should be

$$T(t,m,n,k) = \sum_{i=1}^{k} \left(tm-(t-1)(i-1)\right) O(in) \tag{11}$$

or, overall, $O(tmnk^2)$. Thus, the algorithm should scale *linearly* with cardinality. Empirical verification is provided later in this paper.

Computationally, backwards-elimination would be a far worse approach. Without specific domain knowledge, the start state would require a full mt attributes, which will increase the computational cost of early iterations dramatically. In addition, if $k \ll m$, there will be many more iterations than forwards-search. More and longer iterations, plus a more difficult search space disqualify that approach as impractical.

3.6 Experiments

Experiments were conducted with a focus on answering three questions.

1. What is the impact of MIC-based selection method on MIC and overall dimensionality?
2. What about the quality of the chosen attributes?

3. Does the method scale well?

The first question will be easy to answer; one need simply present before-and-after comparisons of the attribute counts and MIC values. Likewise, the third presents no particular difficulties where cardinality is concerned; elapsed time will suffice. The measurements used for answering the second are sufficiently complicated to require a detailed explanation that will be omitted here for brevity, but may be found in [43]. On that question, it would be accurate to state that in general, the selected attributes provided more novelty – additonal information – with respect to previously selected attributes than did attributes which were not selected.

3.6.1 Data

The data sets used are listed in Table 2. Of these, only `synthia` was specifically generated by us subsequent to the design of the algorithm tested.

3.6.2 Impact on dimensionality and MIC

Table 3 lists the effects of greedily selecting attributes from the transformed versions as previously described. In *all* cases except for the two-attribute sets – the 1992 and 2001 versions of national area and population from the CIA World Factbooks – the number of attributes retained was significantly lower than the number of original attributes. In addition, in *each case* the MIC increased, in most cases again significantly. This is possible because rather than preserving untrustworthy structure, the system improves it – in most cases, greatly.

3.6.3 Scalability

Both quality and scalability are important enough to empirically assess. For this, I used the `synthia` set, which was generated with three "real" attributes, each independent identically-distributed Gaussians, and 25 derivative attributes, each of which is a linear combination of three cubic polynomials of the respective "real" attributes.

This permits us to easily and fairly test for scalability with respect to cardinality. Specifically, the test generated variations of `synthia` with cardinalities of 1000 to 10,000. Figure 18 shows time required for the transformation phase, the selection phase, and the total of those two. The entire algorithm thus scales linearly with cardinality, as confirmed by a correlation coefficient of 0.9958.

Name	Source	Attrs.	Vecs.	Notes
baseball	MLB '96	17	365	
basketball	NBA '91-92	45	459	1
CIA-1992	CIA	2	215	2,3
CIA-2001	CIA	2	235	2,4
machine	UCI/ML	8	209	5,6
page-blocks	UCI/ML	11	5473	5
synthia	synthetic	28	5000	1,8
wine	UCI/ML	13	178	5,7

Note	Meaning
1	Multiple obviously related attributes.
2	Area versus population, in km^2.
3	CIA World Factbook 1992[8].
4	CIA World Factbook 2001[9].
5	From the UCI Machine Learning Repository [6]
6	`cpu-performance` database, minus nominal attributes.
7	Minus the nominal classification attribute.
8	Generated specifically for this work. Consists of three independent identically-distributed Gaussian variables, and 25 linear combinations of cubic polynomials of the Gaussians.

Table 2: Summary of the data used.

3.7 Discussion

The algorithm clearly reduces the number of attributes required, while increasing their value and rendering them more uniform – scalably. The obvious question is how close the algorithm gets to optimal. At this point, consider a concept from algorithmic complexity.

The Kolmogorov complexity of a binary string is the length of the shortest Universal Turing Machine completely self-contained program which outputs the exact binary string. No algorithm can exactly compute the Kolmogorov complexity over the space of all possible inputs [15]. The scaling problem is similar but not isomorphic to the Kolmogorov problem, as the reconstruction methods correspond to a UTM program with a size that depends upon how easily the retained attributes can be used to model the dropped attributes. In any event, the set of possible axis scalings, let alone more general data space transformations, is clearly infinite. In light of this, it may well be impossible to create an algorithm that can

	Before		After	
Name	Attrs.	MIC	Attrs.	MIC
`baseball`	17	2.0540	6	3.6987
`basketball`	45	1.6310	8	4.8211
`CIA-1992`	2	0.3160	2	1.5840
`CIA-2001`	2	0.2591	2	1.5724
`machine`	8	0.5858	4	2.1718
`page-blocks`	11	1.6726	5	3.2933
`synthia`	28	2.3918	4	3.0966
`wine`	13	0.9909	4	3.0425

Table 3: Summary of the dimensionality and MIC changes.

always determine what the optimal is.

That said, consider how the algorithm performed. Experiments were performed to answer questions regarding the algorithm's effect on MIC and dimensionality; whether or not the algorithm chose informative attributes, and dropped redundant or low-content ones; and its overall scalability.

First, as shown in Table 3, reducing the embedding dimensionality is quite compatible with substantial increases in marginal information content.

Second, using MIC as an attribute selection criterion does appear to prefer attributes that provide more novel, non-mutual information. Intuitively, this is desirable in order to capture as much information as one can without succumbing to noise or inefficiently accepting overly redundant data.

Third, the algorithm scales well. Explicit scalability testing on `synthia` variants further showed linear scalability with respect to cardinality. In addition, in real cases it may be possible to use domain knowledge to trim the transformation set, which would increase performance significantly.

3.8 Conclusions

Preserving distances and structure is a common theme in principal components analysis, random projection, FASTMAP, and many other methods. However, these and many other tasks make a potentially fatal assumption: that the distances and structure are largely correct in the first place. When data is distributed in a skewed manner, when axes are badly weighted, when the distance function no longer makes sense – they fail. This is implicit not only in many dimensionality reduction methods, but also other problems – *distance*-based

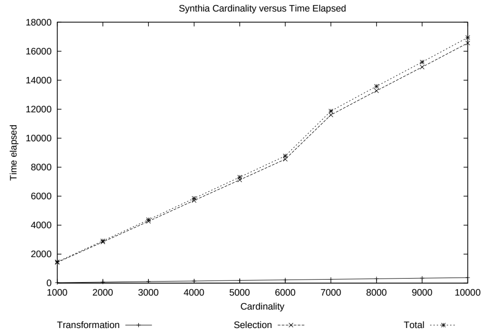


Figure 18: Performance results. Both the transformation and scaling phases scale linearly with cardinality.

outlier detection, *nearest-neighbor* methods, and so forth.

Instead of ignoring this problem, this work presents effective, new approach. Instead of preserving distances that one may have little *a priori* reason to trust, we can instead aim at maximizing information via reversible transformations, while reducing embedding dimensionality by discarding redundant attributes. Assessing how close any given solution is to optimal, however, may be well be impossible; instead, the current system relies on marginal information content (MIC), intrinsic dimensionality, and estimates of attribute worth based on entropy. The system simply focuses on *making every bit count*, by improving instead of merely preserving.

4 Proposed Work

This section discusses potential and proposed avenues for future research along the lines of extending the work described in the previous two. In addition, a basic schedule will be described.

First, we describe various outstanding issues. Not all of these will be major components of the final thesis; the presence of those that will not be is merely an acknowledgement of their existence and relevance. Some of these may be more of interest as engineering and testing issues instead of theoretical research.

4.1 Two-Phase Query Iteration

One issue with FALCON as it currently exists is that there is no distinction between iterations which are used for relevance feedback, and iterations which result in the final query answer. For this reason,the system always attempts to return objects that are most similar to the query objects – which, counterintuitively, is in some respects a suboptimal approach.

It may be better to separate the two. Iterations meant solely for further describing the query, rather than extracting full results, could be constrained to returning distinct objects, each representing some possible "region" of the object space. Given a standard hierarchical index, the search might correspond to an iterative tree traversal with pruning of rejected regions and descent – expansion – into accepted ones. The overall objective would be to reduce the number of iterations required by more rapidly determining the structure of the underlying query concept. Once the user is sufficiently happy with the tentative results, he could then run the query "for real" and receive a full list of results.

4.2 Negative Feedback

Another unresolved issue is negative feedback. At present, weights are constrained to be strictly non-negative, as it is unclear as to how to "properly" handle feedback labelling something as having negative value.

To extend the weighting system beyond "not a good object" and "a good object with weight w", one could use a more elaborate system where a user might specify one of the following options.

1. This is the sort of object I'm looking for, but it's a fluke and avoid giving me anything similar to this.

2. This is the sort of object I'm looking for, and give me more like this.

3. This is not the sort of object I'm looking for, and avoid giving me anything similar to this.

4. This is not the sort of object I'm looking for, but it's not that far off.

In addition, the user would of course have the default option of providing no feedback on particular objects.

The first case should be very rare; if it is not, then the user may have an exceedingly complicated query beyond the ability of distance-based models.

The second case should be fairly common, and is already handled.

The third case might be implementable via penalizing objects for similarity to all objects thusly labeled as objects to avoid.

The fourth case requires some care. I suspect that it would be acceptable to mark the objects as "consider as worst on later iterations unless the rating is changed", but without having any effect on scoring other objects.

Determining or not any particular system would "make sense" would require a substantial amount of user testing with actual human subjects.

The rest of the issues in this section will focus on the scaling and selection work, as I believe that more interesting, unanswered theoretical questions lie there than with FALCON.

4.3 Kolmogorov Complexity and Minimum Description Length

As previously mentioned, the minimum length of all self-contained Universal Turing Machine (UTM) programs that exactly reproduce a given bit string is the Kolmogorov complexity of that string. This is known to be incomputable. However, one can still measure how many bits are actually required by a particular algorithm when operating on specific data, and in practice having "low" total costs instead of optimal would still be useful.

The total cost of the general scaling problem could be analyzed as follows.

1. Cost of storing retained data.
2. Cost of reconstructing the remaining data.
   (a) Cost of identifying which reconstruction model is being used for which data.
   (b) Cost of defining all relevant parameters for chosen models, including identifying input sources.
   (c) Cost of storing any error correction data.

All of these costs are parameterized to some degree. Notably, data storage costs will vary depending upon the desired amount of precision.

The set scaling and reconstruction problem, however, deviates from the Kolmogorov formulation in two major aspects. First, when storing unordered point-sets order does not matter, which means that we need to estimate the complexity of a set instead of any single binary string. All scaling and reconstruction methods that result in any order are thus acceptable. The second major difference is that the UTM programs are constrained; by insisting on transformations that preserve ordering along axes, we have limited the still-infinite space of candidate programs.

In addition, formulating a reasonable MDL cost model may contribute to the next open question.

4.4 Finding m'

As the described scaling method currently works, there exists a parameter m' that dictates the final dimensionality. m' is specified either explicitly, if the user has some particular requirements that would limit the acceptable dimensionality; or implicitly, when the user specifies a minimum MIC contribution for accepting an attribute.

An MDL model could be a "fair" way to numerically compare spaces across a wide range of m' values; since it takes into account both reconstruction cost and data storage cost, it would not necessarily be biased towards low or high m'. Applying the MDL model, however, requires having something to measure. This brings up the next point, reconstruction.

4.5 Reconstruction

Actually performing reconstruction is non-trivial, as there are a fair number of methods with assorted drawbacks. For instance, one could use regression. Simple analytical solutions exist for least-squares linear regression, while nonlinear regression models can be tweaked via the Gauss-Newton method. Regression trees could be used as well.

At the extreme end, there are neural networks. Given an infinite amount of time and two hidden layers of extreme size, and an infinite number of random restarts, then a neural network should eventually converge to the optimal solution as constrained by choice of inputs. In practice, finite neural networks have been applied with some success. Five-layer neural networks, for instance, have been used for dimensionality reduction by themselves via an "hourglass" model in which the middle layer corresponds to the "compressed" outputs, and the layers before and after correspond to encoding and decoding. For any situation in which inputs have already been decided, a three- or four-layer network should suffice. Determining the number of nodes in each layer is both important – since it determines flexibility and number of parameters – and difficult, as there are few guidelines as to how to do this.

No "optimal" solution is expected here. The construction of a reasonable assortment of approaches, however, combined with studies of their effectiveness or lack thereof, would reaffirm the utility of information-based scaling and selection.

4.6 Competing Methods

Additional evidence might be found in empirical comparisons versus competing methods. The SPARTAN system comes very close, in that it performs both selection and reconstruction, but not nonlinear scaling [1]. Traditional scaling and dimensionality reduction methods would need to be paired with reconstruction techniques for MDL-based evaluation to be fair.

For the methods that attempt to find a linear, orthonormal basis, reconstruction is easy, as missing attributes are interpreted as linear combinations of the basis attributes. Then, all that are needed are the linear combination coefficients and any error reduction deltas.

Auto-associative feed-forward neural networks, with the traditional five-layer model, also provide an obvious reconstruction method – the fourth layer, which maps from reduced to original space. Other nonlinear methods may require more work in this regard.

4.7 Schedule

More questions have been enumerated above than are reasonable objectives for a single thesis. Of greatest interest are the issues regarding Kolmogorov complexity and MDL as they pertain to attribute scaling and selection. Therefore, I propose to

1. Formulate a minimum description length-based (MDL) method for evaluating the total space requirements of the scaling, selection and reconstruction model. This should be quite quick, compared to actually using it.

2. Determine and empirically assess a reconstruction toolkit. Evaluation might consist of parameters and time required versus error. This could take a considerable amount of time – perhaps two or three months or more.

3. Assess a possible switch to genetic programming. Encoding schemes could be quite simple to design, but I first need a standardized metric such as MDL for use in the fitness function. Again, this would be require significant programming and testing time, perhaps a few months to half a year depending on achievable performance.

4. Given an MDL cost model, use its objective nature to estimate m' for sets. Without experiments, one may conjecture about expectations – it might seem reasonable that a graph of m' versus MDL cost would be a "well-behaved" graph with few points of inflection, but neither theory nor data yet exists to support that. This will be highly dependent on observed performance; if the values are "well-behaved", then estimating this should be relatively simple as a nearly natural consequence of the algorithm. If not, then it will be necessary to consider whether there is, in fact, such a thing as reasonable generic grounds for determining m' automatically instead of leaving it either implicitly or explicitly to the user.

5. Clarify the relationship between Kolmogorov complexity and the minimum possible length of the scaled, selected set plus reconstruction data, if possible. If no strong relationship can be readily determined, this fact itself may be interesting.

6. Given an MDL cost model, compare the presented method versus competing methods. Arbitrary amounts of time could be spent on this, obviously, but a few months would probably suffice for a respectable survey of possible methods.

Work with respect to FALCON and example-driven relevance feedback will continue, but will primarily involve engineering issues as well as substantial work by those other than the proposer. For instance, the current method of selecting examples requires the user to page through a flat list of database objects, when clearly a semantically organized hierarchy would be superior. Implementing the latter, however, will be left to others.

References

[1] Shivnath Babu, Minos Garofalakis, Rajeev Rastogi, and Avi Silberschatz. Model-based semantic compression for network-data tables. In *Proc. of NRDM 2001*, May 2001.

[2] S. De Backer, A. Naud, and P. Scheunders. Nonlinear dimensionality reduction techniques for unsupervised feature extraction. *Pattern Recognition Letters*, 19:711–720, 1998.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. *ACM SIGMOD*, pages 322–331, May 23-25 1990.

[4] Alberto Belussi and Christos Faloutsos. Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In *Proc. of VLDB*, pages 299–310, Zurich, Switzerland, September 1995.

[5] Kevin Beyer, Jonathan Goldstein, Raghu Ramkrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? ICDT, 1998.

[6] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[7] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient processing of spatial joins using R-trees. In *Proc. of ACM SIGMOD*, pages 237–246, Washington, D.C., May 26-28 1993.

[8] Central Intelligence Agency, editor. *The World Factbook*. U.S. Government Printing Office, 1992. http://www.cia.gov/cia/publications/factbook/.

[9] Central Intelligence Agency, editor. *The World Factbook*. U.S. Government Printing Office, 2001. http://www.cia.gov/cia/publications/factbook/.

[10] Kaushik Chakrabarti and Sharad Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *Proc. of 26th International Conference on Very Large Data Bases*, pages 89–100. Morgan Kaufmann, September 2000.

[11] Kuiyu Chang and Joydeep Ghosh. Principal curves for nonlinear feature extraction and classification. *SPIE Applications of Artificial Neural Networks in Image Processing III*, 3307:120–129, 1998.

[12] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. *VLDB*, pages 426–435, 1997.

[13] David DeMers and Garrison Cottrell. Non-linear dimensionality reduction. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 580–587. Morgan Kaufmann, San Mateo, CA, 1993.

[14] Christos Faloutsos and King-Ip (David) Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *ACM SIGMOD*, pages 163–174, May 23-25 1995.

[15] Alexander Gammerman and Vladimir Vovk. Kolmogorov complexity: Sources, theory and applications. *Computer Journal*, 42(4):252–255, 1999.

[16] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD*, pages 47–57, Boston, Mass, June 1984.

[17] Aapo Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[18] Aapo Hyvärinen, Juha Karunen, and Erkki Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[19] Yoshiharu Ishikawa, Ravishankar Subramanya, and Christos Faloutsos. Mindreader: Querying databases through multiple examples. Technical report, 1998.

[20] Norman Johnson and Samuel Kotz. *Continuous univariate distributions*. Houghton Mifflin, 1970.

[21] I. T. Jolliffe. *Principal Components Analysis.* Springer-Verlag, New York, 1986.

[22] M. Jones. Using recurrent networks for dimensionality reduction. AI Tech Report 1396, MIT, 1992.

[23] Edwin M. Knorr, Raymond T. Ng, and Ruben Zamar. Robust space transformations for distance-based operations. In *Proc. of 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2001.

[24] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume 78, 1990.

[25] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *American Institute of Chemical Engineers J.*, 37(2):233–243, 1991.

[26] P.M. Murphy and D.W. Aha. UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA. [http://www.ics.uci.edu/∼mlearn/MLRepository.html], 1994.

[27] M. Partridge and R. Calvo. Fast dimensionality reduction and simple PCA. *Intelligent Data Analysis*, 2(3), 1998.

[28] T.R. Payne. *Dimensionality Reduction and Representation for Nearest Neighbour Learning.* PhD thesis, The University of Aberdeen, Scotland, 1999.

[29] Kriengkrai Prokaew, Sharad Mehrotra, Michael Ortega, and Kaushik Chakrabarti. Similarity search using multiple examples in mars. In *1999 International Conference on Visual Information Systems*, June 1999.

[30] J. Rissanen. Minimum description length principle. In S. Kotz and N. L. Johnson, editors, *Encyclopedia of Statistical Sciences*, volume V, pages 523–527. John Wiley and Sons, New York, 1985.

[31] Joseph John Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System – Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Englewood Cliffs, N.J., 1971.

[32] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, December 2000.

[33] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Content-based image retrieval with relevance feedback in MARS. In *Proceedings of IEEE International Conference on Image Processing '97*, Santa Barbara, CA, October 1997.

[34] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Human perception subjectivity and relevance feedback in multimedia information retrieval. In *Proceedings of IS&T and SPIE Storage and Retrieval of Image and Video Databases VI*, San Jose, CA, January 1998.

[35] G. Salton, E.A. Fox, and H. Wu. Extended boolean information retrieval. *CACM*, 26(11):1022–1036, November 1983.

[36] J. W. Sammon. A nonlinear mapping for data analysis. *IEEE Transactions on Computers*, C-18:401–409, 1969.

[37] Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[38] G. Schuster. *Deterministic Chaos an Introduction*. Verlagsgesellschaft, Weinheim, Germany, 3rd edition, 1995.

[39] Claude Shannon. A mathematical theory of communcation. *Bell Systems Technical Journal*, 1948.

[40] Takashi Takahashi and Ryuji Tokunaga. Nonlinear dimensionality reduction by multi layer perceptron using superposed energy. In *Proc of. International Symposium on Nonlinear Theory and its Applications*, volume 2, pages 863–866, 1999.

[41] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. 290:2319–2322, December 2000.

[42] Caetano Traina Jr, Agma Traina, Leejay Wu, and Christos Faloutsos. Fast feature selection using fractal dimension. Simpósio Brasileiro de Banco de Dados, October 2000.

[43] Leejay Wu and Christos Faloutsos. Making every bit count: Fast nonlinear axis scaling. Submitted for publication, 2002.

[44] Leejay Wu, Christos Faloutsos, Katia Sycara, and Terry R. Payne. Falcon: Feedback adaptive loop for content-based retrieval. In *Proceedings of the 26th International Conference on Very Large Databases*, pages 297–306, Cairo, Egypt, September 2000. VLDB.

[45] Leejay Wu, Christos Faloutsos, Katia Sycara, and Terry R. Payne. Falcon: Feedback adaptive loop for content-based retrieval. Technical report, Carnegie Mellon University, 2000.