MindReader: Querying databases through multiple examples

Yoshiharu Ishikawa[1] Ravishankar Subramanya[2]
Christos Faloutsos[3]

April 1998

CMU-CS-98-119

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1] Visiting from Nara Institute of Science and Technology, Nara, Japan
[2] Pittsburgh Supercomputing Center
[3] On leave from University of Maryland.

This work was supported by the National Science Foundation (NSF) under grant number IRI-9625428, and by NSF, ARPA and NASA under NSF Cooperative Agreement No. IRI-9411299.

Keywords: Databases, Information Retrieval, Access Methods, Multimedia

Abstract

Users often can not easily express their queries. For example, in a multimedia/image by content setting, the user might want photographs with sunsets; in current systems, like QBIC, the user has to give a sample query, and to specify the relative importance of color, shape and texture. Even worse, the user might want correlations between attributes, like, for example, in a traditional, medical record database, a medical researcher might want to find "mildly overweight patients", where the implied query would be "weight/height $\approx 4\,\mathrm{lb/inch}$".

Our goal is to provide a user-friendly, but theoretically solid method, to handle such queries. We allow the user to give several examples, and, optionally, their 'goodness' scores, and we propose a novel method to "guess" which attributes are important, which correlations are important, and with what weight.

Our contributions are twofold: (a) we formalize the problem as a minimization problem and show how to solve for the optimal solution, completely avoiding the ad-hoc heuristics of the past. (b) Moreover, we are the *first* that can handle 'diagonal' queries (like the 'overweight' query above). Experiments on synthetic and real datasets show that our method estimates quickly and accurately the 'hidden' distance function in the user's mind.

1 Introduction

In modern database applications, distance-based (or similarity-based) queries have gained importance, especially in the area of multimedia databases. However, it is not an easy task for a database user to express his or her queries appropriately in terms of the provided features (like color histogram, shape, etc.). There exist two problems. First, it would be difficult for the user to define an appropriate distance function; the user may not have a clear definition of the distance function he imperfectly has in mind. Second, the user's requirement for data changes time to time so that even if the user can define a good distance function for some cases, the function is not necessarily good for different cases.

As an example, consider content-based retrieval in image databases. Suppose that a user simply needs sunset images. Many existing image database system, such as QBIC (Query By Image Content) [FBF+94] and Virage [Vir], provide query mechanisms based on multiple image features. To ease user's query formulation, such image database systems support *Query by (Visual) Example* [HK92] queries and interfaces (e.g., sliding bars) to obtain user's preferences for the features. In this case, the user would provide sample sunset images or sketches and assign high importance on the color feature and medium importance on the shape feature by using sliding bars. For other features (e.g., texture), minimal importance would be assigned. Unfortunately, this query formulation scenario only holds for simple queries; for example, assume that the user wants to find "red circles in blue background." It is not clear how to assign relative importance to shape and color features.

A similar situation appears even in traditional databases. For example, the VAGUE system [Mot88] built upon a relational database system supports vague queries by incorporating the concept of data metrics on attribute domains and allowing "similar-to" operator to compare attribute values. If we have a method to derive the implied distance function, we can apply it to vague query processing. As an example, let us consider an example-based query in a traditional medical record database. Suppose that a user want to retrieve 'mildly overweight people' and gives some sample records to the database system. The implied condition would be, e.g., weight/height $\approx 4\,\mathrm{lb/inch}$. This is illustrated in Fig. 1, where the user has specified some desirable points with check-marks. Multiple check-marks for a single point indicate that this point is an extremely good example of what the user wants. Formally, single check marks and double check marks mean examples with importance ("goodness") level 1 and 2, respectively. Clearly, the desirable points are along a diagonal strip, with slope 4 (lb/inch). What we want is the system to "guess" the user's need from these example points and somehow approximate the implied query.

Figure 1 illustrates two important concepts that we want to introduce:

- "*diagonal queries*": these are queries like the "overweight" query, where the qualifying elements are along some diagonal in the address space.

- "*multi-level goodness scores*": the ideal system should allow the user to specify how good an example is for his needs.

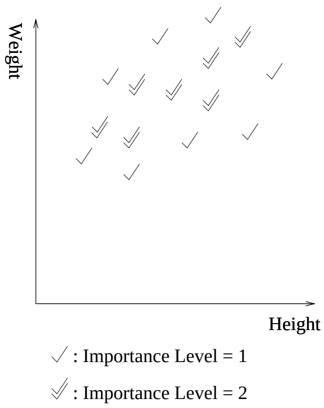


Figure 1: Example-based Query (e.g., "mildly overweight"): double check-marks indicate extremely good examples.

This is exactly the focus of this paper. We want a method that will "guess" the user's query, by combining multiple data examples and automatically determining the importance of each attribute (e.g., 'weight' and 'height' in the 'overweight example'), as well as the importance of *correlations* between attributes.

Such a method will be useful in many settings:

- multimedia systems and digital libraries that handle mixed media (e.g., Informedia [CKM+95, WKS96]): find videos with J.F. Kennedy (either by face matching or voice matching, or both).

- general approximate matching or nearest neighbor queries, in traditional databases. We mentioned the VAGUE system earlier. Nearest neighbor queries are attracting increasing attention not only by researchers but also by vendors: see, for example, the proposed 'STOP AFTER' keyword of SQL [CK97],

- time sequences: find stocks similar to, e.g., IBM's stock. Different users have different notions of distance/similarity: One user might be interested in the average over the year; another user might be interested in the trend; a third one might be interested in the standard deviation ($\approx$ volatily) as a measure of similarity. Although feasible to translate all these notions into distance functions that involve the DFT coefficients of the stock prices, it is very cumbersome for the average user to do so.

- spatial databases: find gas stations close to the I-270 interstate highway, between exit #15 and #20

Next we describe our solution, as well as experiments with the prototype ('MindReader') that we implemented. The structure of the paper is as follows: Section 2 surveys the related work. Section 3 describes the proposed method and the related theorems. Section 4 gives experimental results. Section 5 discusses implementation and user-interface issues. Section 6 gives the conclusions.

2 Related Work

Algorithms to guess the user's desires from a set of examples have attracted a lot of interest.

The underlying assumption is that the user has an ideal point $\vec{q}$ that he is looking for (e.g., the feature vector of his ideal sunset photograph); we try to guess the ideal point, as well as the importance of the axis (e.g., color versus shape versus texture, or, even more detailed: amount of red, versus blue, versus green).

The methods in the literature use one or both of the following ideas: (a) query-point movement and (b) axis re-weighting. Notice that the two ideas are orthogonal and can be combined, although not all past methods have done so.

Query-point movement: Here, a method tries to improve the estimate of the 'ideal query point' by by moving it towards 'good' example points (and away from 'bad' example points). This concept of example-based query refinement is often found in the information retrieval field as *relevance feedback* techniques [Har92, SL96]. For example, the Rocchio's formula [Roc71], that is based on the *vector space model*, is given as follows:

$$Q_1 = Q_0 + \beta \sum_{i=1}^{n_1} \frac{R_i}{n_1} - \gamma \sum_{i=1}^{n_2} \frac{S_i}{n_2} \tag{1}$$

(in [Har92] this formula is called *Standard Rocchio*), where Q_0 is the vector for the initial query, R_i is the vector for *relevant* document i, S_i is the vector for *nonrelevant* document i, n_1 is the number of relevant documents, and n_2 is the number of nonrelevant documents. In the vector space model, *cosine similarity* is typically used to calculate similarity values between documents. Query-point movement was also used in one of the methods proposed by MARS (Multimedia Analysis and Retrieval System) [RHM97, RHM98] for image retrieval. In the method called $tf \times idf$ ("term frequency – inverse document frequency"), they generated pseudo-document vectors from image feature vectors then directly applied the Rocchio's formula. Although the above techniques are based on similarity-based query processing, we can easily transform similarity values to straight Euclidean distances (see [FL95] or Eq. (13) (shown later)) and apply distance-based query processing.

Re-weighting: The second approach is based on *re-weighting*. For example, the MARS system mentioned above, proposes another refinement method based on re-weighting, which we call *standard deviation method*. The idea is very intuitive: if the variance of the good examples is high along, say, the j-th axis, apparently any value on the j-axis is acceptable to the user, and therefore the j-th axis should have a low weight w_j. Therefore, the inverse of the standard deviation of the j-th feature values in the feature matrix is used as the weight w_j for the feature j, namely, $w_j = 1/\sigma_j$. The resulting weights w_j $(1 \leq j \leq n)$ are used to calculate new similarity values for images. The paper gives no justification about this specific choice: any decreasing function of σ_j would be a good candidate for the weight, like $1/\sqrt{\sigma_j}$, or $1/\log(\sigma_j)$.

As we show later, our proposed method includes both the above types of query refinement as special cases. Moreover,

(a) it does not use ad-hoc heuristics (such as β and γ in the Rocchio's formula),

(b) it can handle multiple-level scores, and, most importantly

(c) it is the *only one* that can handle "diagonal queries".

Next, we give the details of our method.

3 Proposed Method

Intuitively, the problem is as follows: The user inspects some records (objects, e.g., images, video-clips, employee records)

- given the user's scores (0/1 or multi-level), on multiple examples
- "guess" the implied distance function, and issue the appropriate query

3.1 Basic Idea

Our major proposal is to use a distance function that allows not only for different weights of each attribute, but also for correlations. For example, see Figure 2: the straight Euclidean distance has circles for isosurfaces; a weighted Euclidean distance, like MARS, has ellipses, whose major axis is aligned with the coordinate axis. Our proposed distance functions result in ellipses that are *not* necessarily aligned with the coordinate axis.

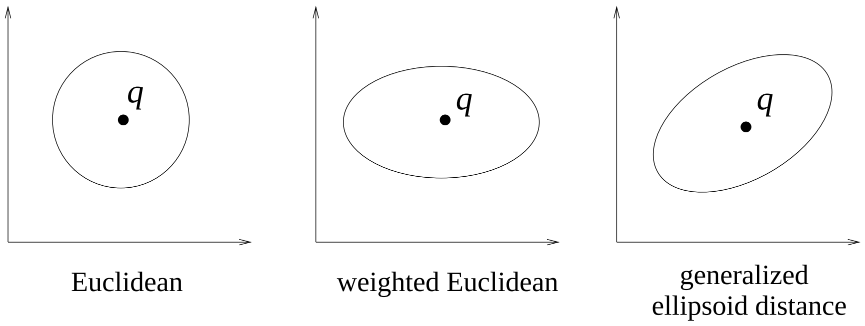


Figure 2: Isosurfaces for Distance Functions

3.2 Method

Table 1 gives a list of symbols used in the following discussion. The proposed distance function is

$$\mathcal{D}(\vec{x}, \vec{q}) = (\vec{x} - \vec{q})^T \mathbf{M} (\vec{x} - \vec{q}), \tag{2}$$

or, equivalently

$$\mathcal{D}(\vec{x}, \vec{q}) = \sum_{j}^{n} \sum_{k}^{n} m_{jk}(x_j - q_j)(x_k - q_k), \tag{3}$$

where $\vec{q} = [q_1, \ldots, q_n]^T$ is the "ideal" point, an n-d query vector and $\vec{x} = [x_1, \ldots, x_n]^T$ is a feature vector that corresponds to an entry in a database and 'T' indicates matrix transposition. All vectors are considered as column vectors. An $n \times n$ matrix $\mathbf{M} = [m_{jk}]$ defines a *generalized ellipsoid distance*. We request that $\mathbf{M}$ is a *symmetric matrix* ($m_{jk} = m_{kj}$). Obviously, it includes the straight and weighted Euclidean distance as special cases.

Our goal is to use the user-selected n-d points, to "guess" two things:

- coefficients of the implied distance function $\mathcal{D}()$, namely the distance matrix $\mathbf{M}$
- the best query point $\vec{q}$: we should use this query point $\vec{q}$ to find data points "similar" to it.

To calculate the new matrix $\mathbf{M}$ and the query point $\vec{q}$, we need "goodness" information from the user. We assume that the user selects "good" items from the presented items. Let N be the number of the selected items and let $\vec{x}_i = [x_{i1}, \ldots, x_{in}]^T$ be a vector that represents i-th item ($i = 1, \ldots, N$). Let $\mathbf{X}$ denote an $N \times n$ matrix $\mathbf{X} = [\vec{x}_1 \ldots \vec{x}_N]^T$. Additionally, we incorporate user's 'goodness' scores for selected items: for each $\vec{x}_i$, the user can specify its goodness value v_i (the default is $v_i = 1$). Let $\vec{v}$ denote a vector $\vec{v} = [v_1, \ldots, v_N]^T$.

Symbol	Definition
$\vec{x} = [x_1, \ldots, x_n]$	a sample vector
$\vec{x}_i = [x_{i1}, \ldots, x_{in}]$	i-th sample vector
n	dimension of vectors
N	number of examples
$\vec{q} = [q_1, \ldots, q_n]$	ideal query point
$\mathbf{M} = [m_{jk}]$	matrix that gives a generalized ellipsoid distance function
$\mathcal{D}()$	distance function
$\mathbf{X} = [\vec{x}_1 \ldots \vec{x}_N]^T$	data point matrix
$\vec{v} = [v_1 \ldots \vec{v}_N]^T$	goodness values for examples
$\bar{x} = [\bar{x}_1, \ldots, \bar{x}_n]$	weighted average of data vectors
$\mathbf{C} = [c_{jk}]$	(weighted) covariance matrix of sample vectors
σ_j^2	(weighted) variance of j-th elements of sample vectors

Table 1: Symbols and Their Definitions

3.3 Theorems

We postulate that the user has an "ideal" vector $\vec{q}$ in mind, and that the distance of the sample vectors x_i from this ideal vector $\vec{q}$ is an generalized ellipsoid distance. Our goal is to "guess" $\vec{q}$ and $\mathbf{M}$ to minimize the penalties. Obviously important samples (i.e., samples with high goodness scores v_i) should have small distance from $\vec{q}$. Thus, the problem is mathematically formulated as follows:

$$\min_{\mathbf{M},\vec{q}} \sum_{i=1}^{N} v_i(\vec{x}_i - \vec{q})^T\mathbf{M}(\vec{x}_i - \vec{q}) \tag{4}$$

subject to the constraint

$$\det(\mathbf{M}) = 1, \tag{5}$$

where $\det(\mathbf{M})$ is the *determinant* of the matrix $\mathbf{M}$ (without any constraint, the zero matrix would give the minimum).

The minimization problem can be solved with the method of *Lagrange multipliers*. We give the major conclusions here:

Theorem 1 *Let the (weighted by v) average of data vectors*

$$\bar{x} = [\bar{x}_1, \ldots, \bar{x}_n] = \frac{\mathbf{X}^T\vec{v}}{\sum_{i=1}^{N} v_i},$$

namely
$$\bar{x}_j = \frac{\sum_{i=1}^N v_i x_{ij}}{\sum_{i=1}^N v_i} \qquad (j = 1, \ldots, n).$$
The optimal value of the new query point $\vec{q}$ is given by $\vec{q} = \bar{x}$.

Proof. See appendix A.

We define the *(weighted) covariance matrix* of data vectors by $\mathbf{C} = [c_{jk}]$ with

$$c_{jk} = \sum_{i=1}^{N} v_i (x_{ik} - \bar{x}_k)(x_{ij} - \bar{x}_j). \tag{6}$$

The following theorem gives the formula of the optimal matrix $\mathbf{M}$.

Theorem 2 *If $\mathbf{C}^{-1}$ exists, the matrix $\mathbf{M}$ that minimizes Eq. (4) is*

$$\mathbf{M} = (\det(\mathbf{C}))^{\frac{1}{n}} \mathbf{C}^{-1}. \tag{7}$$

Proof. See appendix B.

We define the *(weighted) variance* by

$$\sigma_j^2 = \sum_{i=1}^{N} v_i (x_{ij} - \bar{x}_j)^2. \tag{8}$$

Here we show that our model includes and improves the standard deviation (MARS) method if we restrict our methods.

Theorem 3 *If we restrict the matrix $\mathbf{M}$ to diagonal matrices only, then the solution is given by*

$$m_{jj} \propto \frac{1}{\sigma_j^2}. \tag{9}$$

Proof. See appendix C.

That is, we have proved that the weighting scheme of MARS is optimal, if we restrict $\mathbf{M}$ to a diagonal matrix. As we show in the experiments, MARS is unable to "guess" generalized ellipsoid distances.

3.4 The Case for Singular Covariance Matrix

Now we have a seemingly difficult question: what do we do if the covariance matrix is singular and non-invertible? In our system, this situation happens when the number of feedback points is less than the number of the feature dimensions ($N < n$). It turns out that we can easily solve the problem by using the *Moore-Penrose inverse matrix* (or *pseudo-inverse* matrix) [GV96]. The details are in appendix D.

4 Experiments

Here we describe our experiments on synthetic and real data. The questions we want to ask are the following:

- How quickly and how well can we learn 'diagonal' queries? How much better than the 'standard deviation' method is it?
- Query movement: How quickly does our method estimate the correct ideal query point?
- How does the method perform on real data?

Before we answer these questions, we describe the measures we used to judge the quality of our approximation to the hidden distance function. Then, we describe our experiments and observations.

4.1 Definitions and Preliminaries

In order to evaluate how well our method can find the hidden distance function, we need some kind of metrics.

The most obvious metric, the *CD-k metric*, is defined by the *sum of the actual distance of top k neighbors.* In the experiments, we used $k = 20$. Specifically, we retrieve the top k elements according to our *estimate* of the distance (as defined our estimate of $\mathbf{M}$ and $\vec{q}$), and then we compute the sum of their *actual* distances, according to the actual matrix $\mathbf{M}_{hidden}$ and the actual query point.

In some cases, two variations of our method gave the identical set of top k neighbors. In order to still distinguish their relative merits, we use a second metric. The *MN metric* is based on the matrix norm concept and evaluates how small the difference between the derived matrix $\mathbf{M}$ and the hidden distance matrix $\mathbf{M}_{\text{hidden}}$.

$$||\mathbf{M} - \mathbf{M}_{\text{hidden}}||_2 \tag{10}$$

We also need to describe our experimental set up. Goodness score values were automatically generated as follows: Let $d = \mathcal{D}(\vec{x}, \vec{q})$ be the actual (=hidden = implied) distance of a data vector $\vec{x}$ from the ideal vector $\vec{q}$. Then, the goodness score v is given by Eq. (13) and Eq. (15). That is, first we turn the distance d into a similarity score $s = \exp(-\frac{d^2}{2})$ and this into "goodness" score: $v = \log \frac{s}{1-s}$. The choices of the specific functions are justified in Appendix E. We refer to this method as '*infinite levels*' of score values.

4.2 Diagonal Queries

As a first experiment, we used a two-dimensional Gaussian distribution of points shown in Fig. 3(a). The data points has the mean (0, 0) and standard deviation 1. The purpose of this experiment is (i) to examine how fast our method can find the hidden distance function and (ii) to show the difference from the "standard deviation" (MARS) method. In this experiment, we used 0/1 scores. Figure 3(b) shows an isosurface of the hidden distance function (shown by a dotted ellipse) along with top 10 nearest neighbors for our method (marked by '`*`') and standard deviation method (marked by '`o`'). The "ideal" query point $\vec{q} = (0, 0)$ is marked with '`x`'. The result is shown in Fig. 3(c). This figure plots the values of the CD-k metric (namely, cumulative distance of top $k = 20$ points) of our method and of the standard deviation method, for each iteration. The horizontal line in the bottom of the figure represents the global minima of CD metric value for the target query, and the upper dotted line (marked by '`o`') shows the standard deviation method. Our method is given by '`*`'. After five iterations, our method converged into the hidden distance function. On the other hand, the standard deviation method cannot find the distance function.

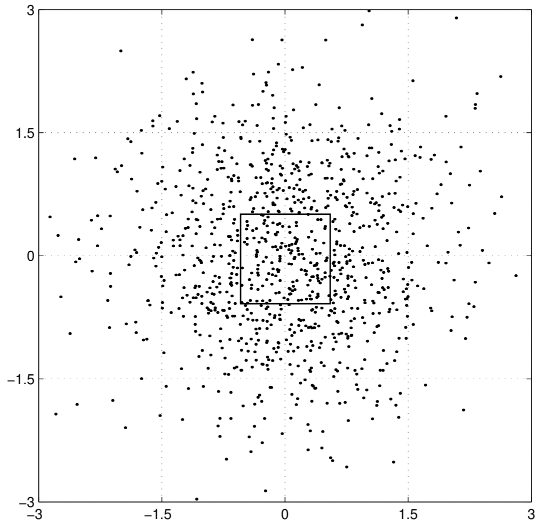
Figure 4 shows iterative refinement of the queries. The figure shows the isosurfaces for the "hidden" distance function (dotted), for our method (solid), and for the standard deviation method (dashed). The initial state after the query is issued (iteration #0) is shown in Fig. 4(a). Since only one example point was given, both methods start with the Euclidean distance (that is, with circles for iso-surfaces). However, after the goodness scores of two iteration, our method starts capturing the hidden function. After six iterations, the agreement is visually almost perfect. On the other hand, the standard deviation method is unable to make any improvements, because of its nature.

4.3 Moving Query Center

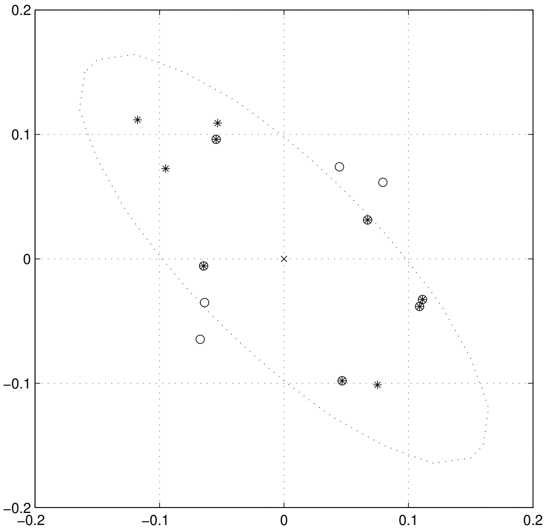
In all our experiments, the "ideal" query point was fixed at (0, 0), and the examples were nearby. Figure 5 illustrates the ability of our method to "move" its estimate of the query point, according to Theorem 1. Our first example was the (0.5, 0.5) point. Despite its large distance from the "ideal" query point, our method managed to locate the correct center in three iterations and then to align itself to the target ellipsoid distance function (dotted ellipse).

4.4 Real Data

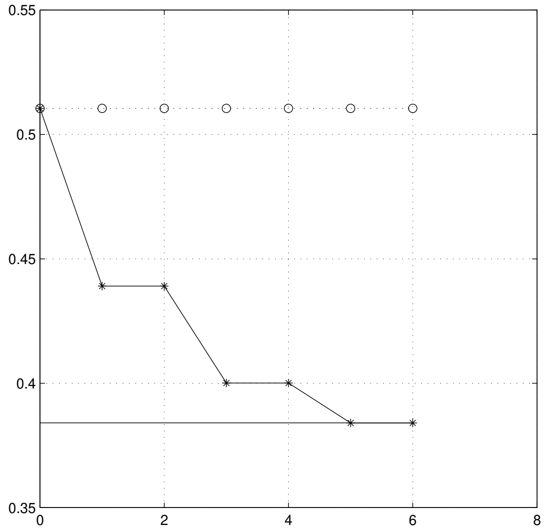
We used the Montgomery County dataset [FK94] with end-points of road segments from the Montgomery County of Maryland (Fig. 6). The solid box shown in the plot is the region of interest. The dataset is normalized to the $[-1, 1] \times [-1, 1]$ square.



(a)

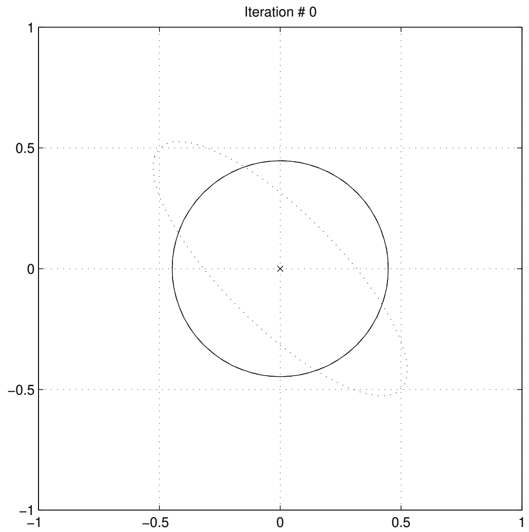


(b)

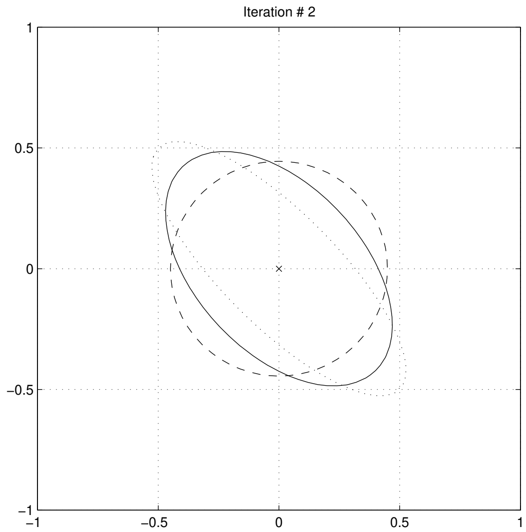


(c)

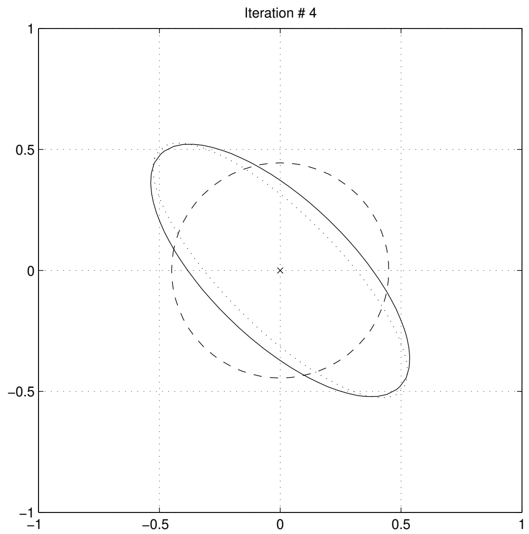
Figure 3: Experiment on Diagonal Query: (a) Data set, (b) Isosurface of "hidden" distance function, (c) Speed of convergence: CD metric value vs no. of iteration



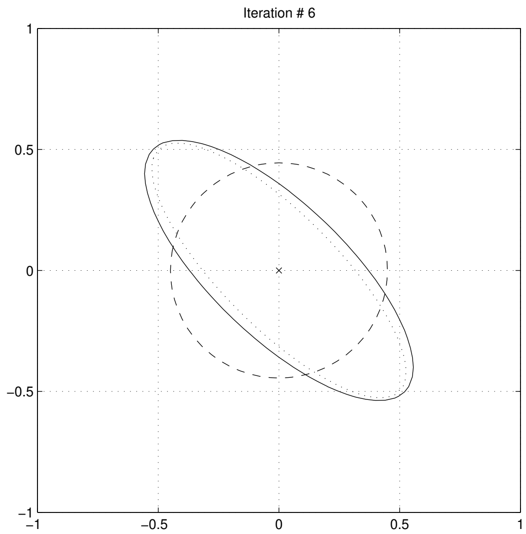
(a) Iteration #0



(b) Iteration #2



(c) Iteration #4



(d) Iteration #6

Figure 4: Iterative Refinement of Queries

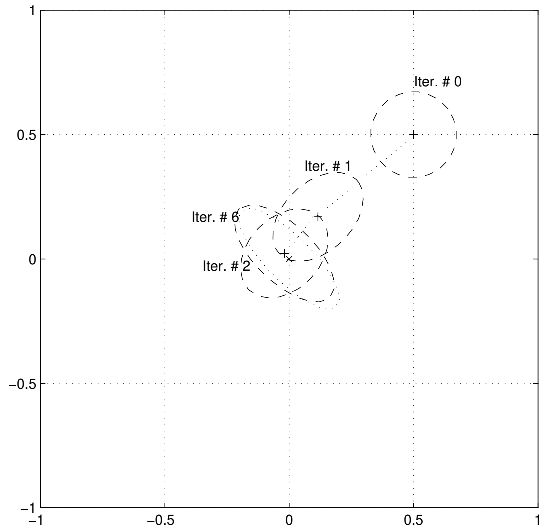


Figure 5: Moving query center

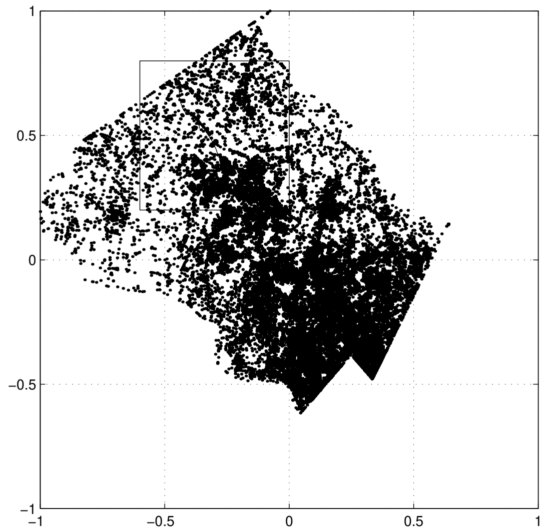
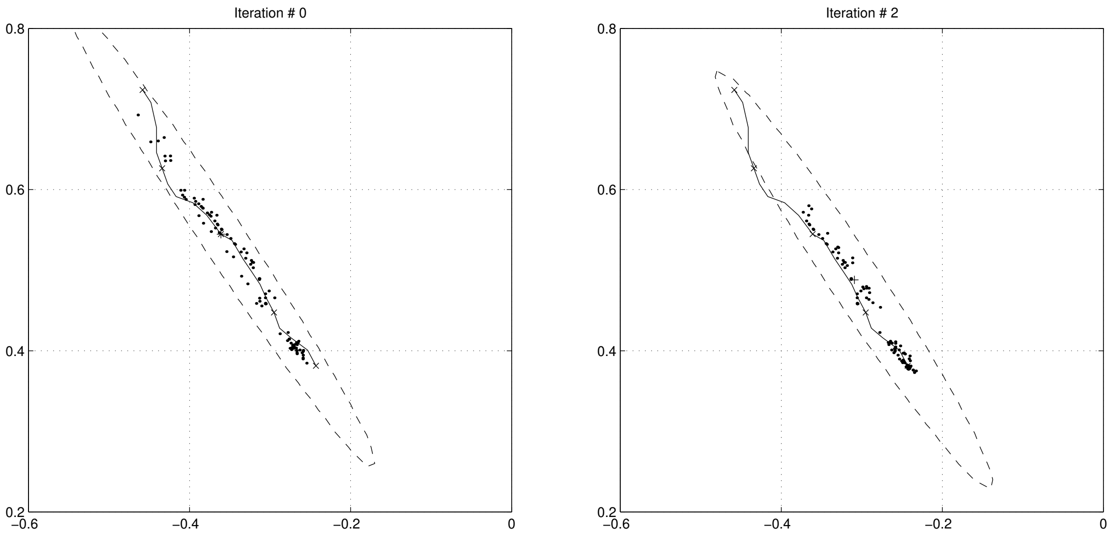


Figure 6: Montgomery County Dataset

The target query was to find points near the magnified road segment, namely a portion of the interstate I-270. We started with five samples shown as '**x**' on Fig. 7. Notice that even *without* any iterations, our guess for the distance function gives the isosurface shown in Fig. 7(a), which engulfs most of the desirable points. If we do want iterations, Fig. 7(b) shows the isosurface after two iterations.



(a) Iteration #0 (b) Iteration #2

Figure 7: Isosurfaces for Road Segment Query: (a) the 5 starting examples, shown with '**x**', and the resulting isosurface, without any iterations (b) the isosurface after 2 iterations.

5 Implementation Issues

User interface: We envision a very friendly user interface, as follows: the user starts browsing (e.g., the thumbnails of images, or the keyframes of video clips). He or she clicks on the desirable one(s), thus providing as many examples as he or she desires. Then, he submits the query to MindReader.

Notice that we can easily allow multi-level scores, by allowing the user to click multiple times on the desirable data item(s) — the score v_i for the i-th item is exactly the count of times that the user clicked on it.

Notice again that MindReader is *not necessarily* a relevance feedback system: The user may give enough examples, MindReader will respond, and no iteration need to take place. As we saw in the Montgomery County dataset, even at the 0-th iteration, MindReader had a very good guess about the distance function, just from the five samples.

Speed: As we mentioned, we expect our data items to be represented as points in some n-dimensional space (e.g., cities on a map, or feature vectors in a multimedia application e.t.c.). Typically, n-d data points will be stored in a Spatial Access Method (SAM) (e.g., X-tree [BKK96], SR-tree [KS97], R*-tree [BKSS90]). SAMs are carefully constructed and fine-tuned by decades of research, so that they can handle very efficiently spatial queries (range- and nearest neighbor queries) under the Euclidean distance. As it was recently discovered, all these indices can easily support any weighted Euclidean distance, as well as "diagonal" (= generalized ellipsoid) queries [SK97], without requiring any restructuring. That is, the user can give a generalized ellipsoid, and the SAMs will quickly retrieve the qualifying points.

Thus, MindReader can utilize any and all of the fast solutions in [SK97]. Therefore, we do not concern ourselves with the speed of searching in this paper any more.

6 Conclusion

We have focused on the problem of combining multiple examples, along with their "goodness" scores, to try to guess the distance function that the user has in mind. This is important, because the user does not need to articulate his desires in terms of predicates, nor to specify the relative importance of the data attributes or their correlations. Our proposed method does all that *automatically* for him!

The major contribution that solved all these issues is the introduction of quadratic forms as candidate distance functions. Specifically

- we formulated the problem rigorously and showed how to find the optimal solution
- we showed that our formulation is the *first* one that can guess "diagonal queries"
- we also showed that it includes as special cases some older methods, like the method by Rocchio and Salton [Roc71] and the method of standard deviation [RHM97, RHM98]. Moreover, our method does *not* need any of the ad-hoc heuristics and constants that the above two methods need.

Additional benefits of our approach include:

- the ability to handle multiple levels of scores (in contrast to the majority of older methods)
- a very friendly user interface (the user just clicks on each good example, one, or more times, in proportion to the "goodness" of the specific example)

- fast searching, thanks to recent developments on "generalized ellipsoid queries" in spatial access methods technology [SK97]

We implemented the proposed approach in an operational system ("MindReader") and we ran experiments on real and synthetic data. The conclusions are the following:

- MindReader can easily guess "diagonal queries", while *no* competitor can do so
- the number of levels of goodness scores can be low, and still give good results
- Intuitively, the more examples we give, the sooner MindReader will converge to the correct distance function. In our experiments on the real data (the 'I-270' query on the Montgomery dataset), just 5 examples are enough to lead to an excellent first guess, which was only mildly modified by subsequent iterations. Thus, MindReader can also be used even in "batch-mode", in addition to an interactive mode.

Directions for future work include the use of MindReader to search in mixed-media, such as video clips with audio tracks, where we want to find video clips with, e.g., John F. Kennedy (either his face, or his voice, or both).

Acknowledgements

We wish to thank Yihong Gong, Toshio Sato, and Ellen Hughes of the Informedia project for constructive feedback, and Dushyanth Narayanan for the help with the datasets.

References

[BKK96] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An index structure for high-dimensional data. In *Proceedings of Very Large Data Bases*, pages 28–39, Mumbai, India, September 1996.

[BKSS90] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of ACM SIGMOD*, pages 322–331, Atlantic City, NJ, May 1990.

[CK97] Michael J. Carey and Donald Kossman. Processing top n and bottom n queries. *IEEE Data Engineering Bulletin*, 20(3):12–19, September 1997.

[CKM+95] M. Christel, T. Kanade, M. Mauldin, R. Reddy, M. Sirbu, S. Stevens, and H. Wactlar. Informedia digital video library. *Communication of the ACM*, 38(4):57–58, April 1995.

[FBF+94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, July 1994.

[FK94] Christos Faloutsos and Ibrahim Kamel. Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension. In *Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 4–13, Minneapolis, MN, May 1994.

[FL95] Christos Faloutsos and King-Ip (David) Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of ACM SIGMOD*, pages 163–174, San Jose, CA, May 1995.

[GV96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, third edition edition, 1996.

[Har92] Donna Harman. Relevance feedback and other query modification techniques. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, chapter 11, pages 241–263. Prentice-Hall, 1992.

[HK92] Kyoji Hirata and Toshikazu Kato. Query by visual example – content based image retrieval –. In A. Pirotte, C. Delobel, and G. Gottlob, editors, *Proceedings of 3rd International Conference on Extending Database Technology (Advances in Database Technology – EDBT'92)*, volume 580 of *Lecture Notes in Computer Science*, pages 56–71, Vienna, Austria, March 1992. Springer-Verlag.

[KS97] Norio Katayama and Shin'ichi Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of ACM SIGMOD*, pages 369–380, Tucson, Arizona, May 1997.

[Mot88] Amihai Motro. VAGUE: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems*, 6(3):187–214, July 1988.

[RHM97] 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.

[RHM98] 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.

[Roc71] 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.

[SK97] Thomas Seidl and Hans-Peter Kriegel. Efficient user-adaptable similarity search in large multimedia databases. In *Proceedings of VLDB*, pages 506–515, Athens, Greece, August 1997.

[SL96] Amanda Spink and Robert M. Losee. Feedback in information retrieval. In Martha E. Williams, editor, *Annual Review of Information Science and Technology (ARIST)*, volume 31, chapter 2, pages 33–78. 1996.

[Vir] Virage inc. home page. `http://www.virage.com/`.

[WKS96] Howard D. Wactlar, Takeo Kanade, and Michael A. Smith. Intelligent access to digital video: Informedia project. *IEEE Computer*, 29(5):45–52, May 1996.

A Proof of Theorem 1

From the summation formula in Eq. (4), we get

$$\sum_{i=1}^{N} v_i(\vec{x}_i - \vec{q})^T \mathbf{M}(\vec{x}_i - \vec{q}) = \sum_{i=1}^{N} v_i \left(\sum_{j=1}^{n} \sum_{k=1}^{n} (x_{ij} - q_j) m_{jk} (x_{ik} - q_k) \right).$$

From the constraint (Eq. (5)), for all k

$$\sum_{j=1}^{n} (-1)^{j+k} m_{jk} \det(\mathbf{M}_{jk}) = 1$$

holds. Therefore,

$$\sum_{j=1}^{n} \sum_{k=1}^{n} (-1)^{j+k} m_{jk} \det(\mathbf{M}_{jk}) = n.$$

To solve the minimization problem, we use the method of *Lagrange multipliers*. Defining

$$F = \sum_{i=1}^{N} v_i \left(\sum_{j=1}^{n} \sum_{k=1}^{n} (x_{ij} - q_j) m_{jk} (x_{ik} - q_k) \right) - \lambda \left(\sum_{j=1}^{n} \sum_{k=1}^{n} (-1)^{j+k} m_{jk} \det(\mathbf{M}_{jk}) - n \right), \quad (11)$$

we obtain

$$\frac{\partial F}{\partial q_t} = -\sum_{i=1}^{N} v_i \left(2m_{tt}(x_{it} - q_t) + \sum_{\substack{k=1 \\ k \neq t}}^{n} m_{tk}(x_{ik} - q_k) + \sum_{\substack{j=1 \\ j \neq t}}^{n} (x_{ij} - q_j) m_{jt} \right).$$

Since $\mathbf{M}$ is a symmetric matrix,

$$\begin{aligned} \frac{\partial F}{\partial q_t} &= -2 \sum_{i=1}^{N} v_i \left(m_{tt}(x_{it} - q_t) + \sum_{\substack{k=1 \\ k \neq t}}^{n} m_{tk}(x_{ik} - q_k) \right) \\ &= -2 \sum_{i=1}^{N} v_i \sum_{k=1}^{n} m_{tk}(x_{ik} - q_k). \end{aligned}$$

Let $\partial F / \partial q_t = 0$. Then we get

$$\sum_{i=1}^{N} v_i \sum_{k=1}^{n} m_{tk} x_{ik} = \sum_{i=1}^{N} v_i \sum_{k=1}^{n} m_{tk} q_k.$$

This can be translated as

$$\frac{[m_{t1}, \ldots, m_{tn}] \mathbf{X}^T \vec{v}}{\sum_{i=1}^{N} v_i} = [m_{t1}, \ldots, m_{tn}] \vec{q}.$$

Therefore,

$$\frac{\mathbf{M} \mathbf{X}^T \vec{v}}{\sum_{i=1}^{N} v_i} = \mathbf{M} \vec{q}.$$

Thus, if $\mathbf{M}^{-1}$ exists,

$$\vec{q} = \frac{\mathbf{X}^T \vec{v}}{\sum_{i=1}^N v_i}$$

holds. The k-th element of $\vec{q}$ is

$$q_k = \frac{\sum_{i=1}^N v_i x_{ik}}{\sum_{i=1}^N v_i}.$$

Namely, q_k is a weighted average of k-th elements of $\vec{x}_i$'s. As a special case, if $v_i = 1$ ($i = 1, \ldots, N$), q_k becomes an average:

$$q_k = \frac{\sum_{i=1}^N x_{ik}}{N}.$$

□

B Proof of Theorem 2

From Eq. (11), for some fixed r and s ($1 \le r,\, s \le n$), we have

$$\frac{\partial F}{\partial m_{rs}} = \sum_{i=1}^N v_i(x_{ir} - q_r)(x_{is} - q_s) - \lambda(-1)^{r+s}\det(\mathbf{M}_{rs}),$$

where $\mathbf{M}_{rs}$ is an $(n-1) \times (n-1)$ matrix obtained by deleting r-th row and s-th column of $\mathbf{M}$. Let $\partial F/\partial m_{rs} = 0$. Then we get

$$\sum_{i=1}^N v_i(x_{ir} - q_r)(x_{is} - q_s) = \lambda(-1)^{r+s}\det(\mathbf{M}_{rs}).$$

Therefore,

$$\det(\mathbf{M}_{rs}) = \frac{\sum_{i=1}^N v_i(x_{ir} - q_r)(x_{is} - q_s)}{\lambda(-1)^{r+s}}.$$

The inverse matrix $\mathbf{M}^{-1} = [m_{jk}^{-1}]$ can be represented as

$$\begin{aligned} m_{jk}^{-1} &= \frac{(-1)^{j+k}\det(\mathbf{M}_{kj})}{\det(\mathbf{M})} \\ &= (-1)^{j+k}\det(\mathbf{M}_{kj}) \\ &= (-1)^{j+k}\frac{\sum_{i=1}^N v_i(x_{ik} - q_k)(x_{ij} - q_j)}{\lambda(-1)^{k+j}} \\ &= \frac{\sum_{i=1}^N v_i(x_{ik} - q_k)(x_{ij} - q_j)}{\lambda}. \end{aligned}$$

Let $\mathbf{C} = [c_{jk}]$ be a matrix $c_{jk} = \sum_{i=1}^N v_i(x_{ik} - q_k)(x_{ij} - q_j)$, namely, $\mathbf{C} = \lambda\mathbf{M}^{-1}$. Since

$$\det(\mathbf{C}) = \lambda^n \det(\mathbf{M}^{-1}) = \lambda^n \cdot 1 = \lambda^n,$$

we get $\lambda = (\det(\mathbf{C}))^{\frac{1}{n}}$. Therefore $\mathbf{M} = \lambda\mathbf{C}^{-1} = (\det(\mathbf{C}))^{\frac{1}{n}}\mathbf{C}^{-1}$.

According to Theorem 1, the new optimal query point is given by $\vec{q} = \bar{x}$. Therefore, the optimal matrix $\mathbf{C} = [c_{jk}]$ in terms of the new query point is given by

$$c_{jk} = \sum_{i=1}^{N} v_i (x_{ik} - \bar{x}_k)(x_{ij} - \bar{x}_j).$$

□

C Proof of Theorem 3

Since this is a special case of Theorem 2, we can easily prove that

$$\mathbf{M} = (\det(\mathbf{C}))^{\frac{1}{n}} \mathbf{C}^{-1}$$

where $\mathbf{C} = [c_{jk}]$ is

$$c_{jk} = \begin{cases} \sum_{i=1}^{N} v_i (x_{ij} - \bar{x}_j)^2 & (\text{if } j = k) \\ 0 & (\text{if } j \neq k). \end{cases}$$

Now we define

$$\sigma_j^2 = \sum_{i=1}^{N} v_i (x_{ij} - \bar{x}_j)^2 = c_{jj}.$$

Since $\mathbf{C}$ is a diagonal matrix,

$$\begin{aligned} \det(\mathbf{C}) &= \prod_{j=1}^{n} \sigma_j^2 \\ \mathbf{C}^{-1} &= \operatorname{diag}(1/\sigma_1^2, \ldots, 1/\sigma_n^2), \end{aligned}$$

and we get

$$\mathbf{M} = \prod_{j=1}^{n} \sigma_j^2 \cdot \operatorname{diag}(1/\sigma_1^2, \ldots, 1/\sigma_n^2).$$

Therefore, we can say that

$$m_{jj} \propto \frac{1}{\sigma_j^2}.$$

□

D The Case for Singular Covariance Matrix

First we calculate the SVD decomposition of the covariance matrix $\mathbf{C}$ ($\mathbf{C}$ is defined by Eq. (6)). Since $\mathbf{C}$ is a symmetric matrix, we get

$$\mathbf{C} = \mathbf{U} \mathbf{\Sigma} \mathbf{U}^T,$$

where $\mathbf{U}$ is a orthonormal $n \times n$ matrix and $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$ is a diagonal $n \times n$ matrix. Then define

$$\mathbf{C}^+ = \mathbf{U}\mathbf{\Sigma}^+\mathbf{U}^T,$$

where $\mathbf{\Sigma}^+ = \mathrm{diag}(1/\sigma_1, \ldots, 1/\sigma_r, 0, \ldots, 0)$. The matrix $\mathbf{C}^+$ is called the *Moore-Penrose inverse matrix* (or *pseudo-inverse matrix*) of $\mathbf{C}$ [GV96]. Then we use instead of Eq. (7) (i.e., $\mathbf{M} = (\det(\mathbf{C}))^{\frac{1}{n}}\mathbf{C}^{-1}$),

$$\mathbf{M} = \alpha\mathbf{C}^+, \tag{12}$$

where $\alpha = (\sigma_1\sigma_2\cdots\sigma_r)^{1/r}$.

E Conversion from Distance Values to Goodness Values

We formalize the concepts of distance, similarity and "goodness" scores, as follows:

- similarity s: ranges from 0 to 1 (1 is the best match) — typically, cosine similarity
- goodness g: ranges from $-\infty$ to $+\infty$ (the latter is the best match), e.g., user scores
- distance d: ranges from 0 to $+\infty$ (0 is the best)

We propose the following formulas to translate distances to similarities etc:

$$s = \exp\left(-\frac{d^2}{2}\right) \tag{13}$$

Thus the similarity corresponds roughly to the likelihood (pdf) of a Gaussian distribution. Earlier [FL95], we used $s = \cos\theta = 1 - d^2/2$. These two forms approximately similar for $d \ll 1$. For the goodness, we propose the sigmoid function of neural networks:

$$s = \frac{\exp(g)}{1 + \exp(g)}, \tag{14}$$

whose inverse is

$$g = \log\frac{s}{1-s}. \tag{15}$$

The above formulas are used, whenever we need to translate 'goodness' scores into distances, similarities etc.

Contents