Towards Systematic Design of Distance Functions for Data Mining Applications

Charu C. Aggarwal
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
charu@us.ibm.com

ABSTRACT

Distance function computation is a key subtask in many data mining algorithms and applications. The most effective form of the distance function can only be expressed in the context of a particular data domain. It is also often a challenging and non-trivial task to find the most effective form of the distance function. For example, in the text domain, distance function design has been considered such an important and complex issue that it has been the focus of intensive research over three decades. The final design of distance functions in this domain has been reached only by detailed empirical testing and consensus over the quality of results provided by the different variations. With the increasing ability to collect data in an automated way, the number of new kinds of data continues to increase rapidly. This makes it increasingly difficult to undertake such efforts for each and every new data type. The most important aspect of distance function design is that since a human is the end-user for any application, the design must satisfy the user requirements with regard to effectiveness. This creates the need for a systematic framework to design distance functions which are sensitive to the particular characteristics of the data domain. In this paper, we discuss such a framework. The goal is to create distance functions in an automated way while minimizing the work required from the user. We will show that this framework creates distance functions which are significantly more effective than popularly used functions such as the Euclidean metric.

1. INTRODUCTION

Distance function design remains at the core of many important data mining applications. Most applications such as clustering, classification and nearest neighbor search use distance functions as a key subroutine in their implementation. Clearly, the quality of the resulting distance function significantly affects the success of the corresponding application in finding meaningful results. For many data mining applications, the choice of the distance function is not pre-defined, but is chosen heuristically. There is not much literature on how distance functions should be designed for arbitrary applications. Many data mining algorithms and applications use the Euclidean distance metric as a natural extension of its use for spatial applications. The widespread use of distance functions such as the L_2-norm is perhaps rooted in the initial development of index structures for spatial applications. In such cases, the L_2 norm has special interpretability in 2 or 3-dimensions. However, this interpretability is not really relevant for arbitrary domains of data containing many dimensions.

The issue of distance function design becomes even more critical in the context of high dimensional applications. Recent work [13] has shown that in high dimensional space, the sparsity of the data may affect the quality of the distance function. In particular, it has been indicated in [4, 7] that distance functions such as the L_p-norm are often not very meaningful for high dimensional data because of the fact that the pairwise distances between the points are very similar to one another. In such cases, the poor contrast in the measurement of distances reduces the effectiveness of the measurements. The overall effect of high dimensionality on nearest neighbor search has been examined in some detail in [2, 11]. In [2], it has been shown that nearest neighbor search by visual interaction can often provide good quality results. However, this approach is only designed for the specific problem of nearest neighbor search and is not focussed on determining the most effective distance functions in closed form. In fact, the merits of user feedback for nearest neighbor search have been well documented for a number of data domains such as text and image retrieval [19, 20, 22]. However, these procedures cannot be directly used in data mining applications. For most real applications, an efficient computation of a closed form distance function is critical to the success of the underlying algorithm.

Some recent work [3] has discussed the effects of different kinds of closed-form distance functions in high dimensionality, but this method is not user centered in its approach. This technique only aims to increase the contrast in the distances of the different points from one another. It is important to understand that since the quality of a distance function may often be determined by the perceptual significance to an end-user, the user needs to be kept in the loop during distance function design. This would ensure that the final distance function can model what the user wants, rather than simply satisfy a particular criterion such

SIGKDD '03, August 24-27, 2003, Washington, DC, USA

as maximizing the contrast. Consider the following cases for distance function design:

- It has been shown in [8], that in categorical databases, the quality of similarity can be best measured in the context of the overall behavior of the data. Thus, a simple distance function such as the hamming distance does not work very well for this case. Instead, it is necessary to determine the statistical behavior of the interactions among different attributes in order to find the most effective form of the distance function. This fact cannot be determined easily a-priori without considerable human understanding and experience of the nature of categorical databases.

- In domains such as time series databases, similarity can be effectively measured while taking into account noise, scaling and translation of the data [5]. Such normalizations are usually heavily dependent in an indirect way upon the perceived user intent in measuring similarity.

- In an image database the similarity between two images may be quite subjective. For example, only particular visual aspects of the image may contribute to the definition of similarity in that domain. What makes the problem even more difficult is that the nature of the best possible distance function may vary considerably even across different image domains. It is often difficult to predict the best distance function in such cases a-priori without some kind of human intervention.

- The similarity between two different digital music data sets may be defined by commonalities in certain segments. These features can often be perceived by audio, but are difficult to predict manually using ad-hoc techniques. Again, the particular aspects of the stream which define similarity cannot be easily identified manually because of the lack of interpretability of the relevant features.

Which features of the data provide the greatest similarity? For example, in the image database, should the color histograms be given greater weight or do the texture and coarseness provide greater similarity? In most cases, these specific combination of weights one must assign to the different features cannot be easily specified only with domain knowledge. This makes distance function design by manual and ad-hoc techniques a somewhat impractical solution. In this paper, we will concentrate on the design of distance functions which are sensitive to the data behavior and serve the needs of the user.

We note that the process of designing distance functions even for a specific domain such as Information Retrieval (IR) has intrigued researchers over three decades [19]. Such a design in the IR community has been achieved by considerable testing and understanding of the particular characteristics of the data which are the most meaningful indicators of similarity. Unlike IR applications, we cannot always use such domain specific information about the "typical" nature of the data for arbitrary applications. Therefore, for these cases, designing distance functions is an even more difficult task. Furthermore, since advances in hardware technologies continue to increase the number of data collection domains at a rapid rate, it is impractical to undertake such an intensive research effort for each newly discovered data domain. On the other hand, given the importance and reusability of distance functions as a tool for many different algorithms in knowledge discovery, it may be acceptable to allow a few hours of human labor in modelling the distance function, provided that a systematic way of constructing the distance function is available. In this paper, we will provide just such a framework.

This paper is organized as follows. The remainder of this section will discuss important aspects and motivations of distance function design. In section 2, we will discuss how distance functions can be modelled and designed in a systematic way so as to model an end-user's perceptions. This section will primarily concentrate on the use of popular canonical forms such as the cosine model or Minkowski model for constructing distance functions. In section 3, we will discuss the use of purely learning strategies for distance function design. Techniques for combining different strategies in order to create the most effective distance function are discussed in section 4. The empirical results are presented in section 5. Finally, the conclusions and summary are contained in section 6.

1.1 Desiderata for Effective Distance Function Design

The design of distance functions for well studied data domains such as Information Retrieval provides some useful hints for the high dimensional case. We list some of the practical desiderata for effective design of distance functions below.

(1) User-Centered: Since the success of the data mining application ultimately rests with the perception of the user, it is clear that the distance function should model the corresponding requirements as closely as possible. While incorporation of domain knowledge in the design of the distance function is often effective, it lacks the direct systematic approach that is necessary in order to handle arbitrary applications. In many cases, for a new data domain, we have little experience or understanding of the features which are most indicative of similarity. Therefore, a necessary requirement is to find a way to solicit the feedback from the user which can be used to construct the distance function by a systematic modelling process.

(2) Contrasting: Since many data mining applications are inherently high dimensional, it is important to ensure that the distance functions provide sufficient discriminatory power. The results of [7] indicate that standard distance functions such as the L_p-norm are not very effective in this respect. This non-contrasting behavior is because even the most similar records are likely to have a few features which are well separated. For distance functions such as the L_p-norm, the results of [7] show that the averaging effects of the different dimensions start to dominate with increasing dimensionality. The utilization of user-feedback serves as a useful tool in resolving this problem of high dimensional distance functions. This is because it is easier for the user to guide the ultimate form of the distance function in such a way that the level of contrast is considerably increased.

(3) Statistically Sensitive: It is very rare that the data is uniformly distributed along any given dimension. Some of the values may be more sparsely populated than others. This behavior is commonly noted in Information Retrieval

applications in which a given document usually contains a small fraction of the available vocabulary. Distance functions such as dice, cosine or jaccard coefficients [19] do not treat the attribute values uniformly but normalize using the frequency of occurrence of the words in the documents. Empirical testing on large collections has established that the normalization process significantly affects the quality of the distance function. In this paper, we will try to achieve statistical sensitivity by analyzing the user requirements directly, rather than try to guess the importance of frequency distributions on the quality of similarity. In other words, we wish to develop *generic models* for distance functions which can directly incorporate user requirements.

(4) Compactness: In many applications, the computation of the distance function itself can be the most expensive operation because of its repeated use over a large number of iterations. Therefore, a distance function should be efficiently computable. In the event that the distance function uses a pre-stored model or statistical information, the storage requirements should be modest. We refer to this requirement as that of *compactness*. We note that a distance function can be represented either in closed form, or *algorithmically* on the feature sets of the two objects. In most cases, closed form distance functions have the advantage of being more efficiently computable.

(5) Interpretability: The interpretability of a distance function is very valuable for effective use in many applications. For example, one would like to know which features of the data are most influential in affecting the distance function. This can provide considerable insight and understanding in a large number of applications. In this respect, closed-form distance functions are usually more desirable.

In the next section, we will discuss some useful models for distance function computation. Specifically we will discuss two generalizations of widely used distance functions such as the L_p-norm and the cosine distance function. Specifically, we will propose the *Parametric Minkowski model* and the *Parametric Cosine model* for distance function computation. We will show how to use these models in conjunction with human interaction in order to create techniques which are significantly more friendly to the needs of a given user.

2. PARAMETRIC MODELING OF DISTANCE FUNCTIONS

The essential idea behind a parametric model for distance function computation is that we do not specify the distance function exactly, but specify it incompletely along with some parameters. Then, we utilize user interaction and feedback to fill in these incomplete parameters. In order to formalize this concept, we will introduce some additional notations and definitions.

We assume that we have a data set $\mathcal{D}$ containing a set of N records, each with d dimensions. Let us assume that the records $\overline{X}$ and $\overline{Y}$ are drawn from the database $\mathcal{D}$. We define the *parametric distance function* $f(\overline{X}, \overline{Y}, \lambda_1, \ldots \lambda_k)$ as the function of its arguments $\overline{X}$ and $\overline{Y}$ and the k parameters $\lambda_1 \ldots \lambda_k$. We will denote the vector $(\lambda_1 \ldots \lambda_k)$ by λ. Thus, we note that this definition of the distance function is *incomplete* since it only assumes a basic form of the distance function and leaves several parameters unspecified. These parameters are then *learned* by a series of user interactions in which the user feedback is utilized in order to solicit the perceptual similarity between different pairs of objects. If desired, multiple users can be used in order to increase the robustness of the data collection process. These similarity values are then used in conjunction with a canonical form in order to train the data for a distance function which is most reflective of user behavior. We note that the use of a canonical form has the advantage that it is able to incorporate some degree of domain knowledge about the data, while leaving the function sufficiently incomplete, so that its final form is sufficiently reflective of user behavior.

Let us assume that the data pairs for which the distance function needs to be defined are given by $(\overline{X_1}, \overline{Y_1}) \ldots (\overline{X_N}, \overline{Y_N})$. The interactive process assumes that for each pair of objects $(\overline{X_i}, \overline{Y_i})$, the user provides the perceptual similarity value α_i. We assume that the user-defined similarity values for these different data pairs are given by $\alpha_1 \ldots \alpha_N$. We note that the nature of each of these objects and the user feedback may depend upon the particular domain at hand. For example, in an image database, this perceptual similarity could correspond to the similarity in the patterns between the different images. In this case, the visual perception of the user may be leveraged in order to define the final similarity values.

Thus, at the end of the process, the values $\alpha_1 \ldots \alpha_N$ are the user responses to the similarities between the corresponding pairs of objects $(\overline{X_1}, \overline{Y_1}) \ldots (\overline{X_N}, \overline{Y_N})$. We would like to choose values of $\lambda_1 \ldots \lambda_k$, so that the error in prediction is as low as possible. This error can be defined in several ways; some of which are algorithmically more desirable. For example, one way of defining the error $E(\lambda_1 \ldots \lambda_k)$ is the *square prediction error*:

$$E(\lambda_1 \ldots \lambda_k) = \sum_{i=1}^{N} (f(\overline{X_i}, \overline{Y_i}) - \alpha_i)^2 \quad (1)$$

A slightly different way of defining the error is the *mean p-norm-squared error*.

$$E(\lambda_1 \ldots \lambda_k) = \sum_{i=1}^{N} (f(\overline{X_i}, \overline{Y_i})^p - \alpha_i^p)^2 \quad (2)$$

We note that the use of the p-norm squared error is a matter of algorithmic convenience, since the final objective function often takes on a simpler form for some of the models that we will propose.

In order to determine the value of this objective function we use the gradient search method. In the gradient search method, we start off with an initial value of the vector $\overline{\lambda} = \overline{\lambda^0}$ and successively update to $\overline{\lambda^1}$, $\overline{\lambda^2}$, ..., $\overline{\lambda^m}$ using the *gradient vector* $\overline{\Delta E}$. This gradient vector is defined as follows:

$$\overline{\Delta E(\overline{\lambda})} = (\delta E(\lambda_1 \ldots \lambda_d)/\delta\lambda_1 \ldots \delta E(\lambda_1 \ldots \lambda_d)/\delta\lambda^d) \quad (3)$$

For each value of $\overline{\lambda} = \overline{\lambda^m}$, we substitute its corresponding value in the above equation in order to determine the gradient. The value of $\overline{\lambda^{m+1}}$ is obtained from $\overline{\lambda^m}$ by utilizing the steepest descent method. In this method, the greatest reduction of the objective function value of $\overline{\Delta E(\overline{\lambda^m})}$ occurs along the gradient direction given by $\overline{d^m} = -\overline{\Delta E(\overline{\lambda^m})}$. Therefore, the value of $\overline{\lambda^m}$ is updated as follows:

$$\overline{\lambda^{m+1}} = \overline{\lambda^m} + \theta^m \cdot \overline{d^m} \quad (4)$$

Here, the choice of θ^m determines the convergence behavior of this iterative mechanism. Picking θ^m effectively ensures

a rapid convergence rate for the algorithm. To this effect, we use the line search method [6] in order to perform the iterations. In the line search technique we would like to pick θ^m such that the value of $\gamma(\theta^m) = E(\overline{\lambda^{\mathbf{m}}} + \theta^m \cdot \overline{d^m})$ is minimized. In order to achieve this, we should have $\gamma'(\theta^m) = 0$. This value of $\theta^m = \beta$ can be determined using a bracket bisection technique. It is elementary to show that $\gamma'(\theta^m) = -\Delta E(\lambda^{\mathbf{m}} + \theta^m \cdot \overline{d^m})^t \cdot \overline{d^m}$. We note that since the direction $\overline{d^m}$ is a descent gradient at λ^m, we have $\gamma'(0) < 0$. In the event that we are able to determine at least one value of θ^m such that $\gamma'(\theta^m) \geq 0$, we know that the required value of θ^m must lie in this range $(0, \beta)$. Finding the value of θ^m exactly can often be time consuming; therefore we estimate this value within a factor of two. To do so, we first pick an arbitrary value of $\theta^m = 1$. If $\gamma'(1) < 0$, we keep doubling θ^m until we obtain the largest value for which $\gamma'(\theta^m) < 0$. Otherwise, if $\gamma'(1) > 0$, we keep halving θ^m until we obtain the first value of θ^m for which $\gamma'(\theta^m) < 0$. The final value of θ^m at the end of this two-pronged search method is used for the update step of Equation 3. We note that this technique is somewhat similar to the Armijo rule [6] used for line search in gradient methods.

2.1 The Parametric Minkowski Model

The parametric minkowski model used in this paper is basically a generalization of the L_p norm. In this model, the distance function is defined by a weighted version of the L_p norm. The weights on the different dimensions are the parameters for this model. Let us assume that the weights for the d dimensions are given by $\lambda_1 \ldots \lambda_d$. Therefore, for a pair of objects $\overline{X} = (x_1 \ldots x_d)$ and $\overline{Y} = (y_1 \ldots y_d)$, the value of the parametric distance is defined as follows:

$$f(\overline{X}, \overline{Y}, \lambda_1 \ldots \lambda_d) = \left(\sum_{i=1}^{d} \lambda_i ||(x_i - y_i)||^p \right)^{1/p} \quad (5)$$

Note that this distance function is easily interpretable in terms of the different dimensions. A higher value of λ_i indicates a greater significance for a particular dimension. In this case, the use of the p-norm mean square error results in a linear set of equations for $\lambda_1 \ldots \lambda_k$. Let us assume that the coordinates for each individual pair of points X^i and Y^i are given by $(x_{i1} \ldots x_{iN})$ and $(y_{i1} \ldots y_{iN})$ respectively. The error function in this case is of the following form:

$$E(\lambda_1 \ldots \lambda_k) = \sum_{i=1}^{N} \left((\sum_{j=1}^{d} a_{ij} \lambda_j) - \alpha_i^p \right)^2 \quad (6)$$

Here the value of a_{ij} is given by $||x_{ij} - y_{ij}||^p$. On using the derivative with respect to each λ_j, we obtain the following:

$$\delta E(\lambda_1 \ldots \lambda_d)/\delta \lambda_j = \sum_{i=1}^{N} a_{ij} \cdot \left((\sum_{k=1}^{d} a_{ik} \lambda_k) - \alpha_i^p \right) \quad (7)$$

Now the gradient vector $\overline{\mathbf{\Delta E}}$ can be defined directly using Equation 3. This gradient vector is used in order to update the vector $\overline{\lambda}$ as indicated in Equation 4.

2.1.1 Generalizations

In real applications, many attributes in the data are correlated with one another. Inter-attribute correlations have often been used for designing distance functions in categorical domains where there is no natural ordering of attribute values. In such cases, the use of inter-attribute summary information provides the only possible insight into the similarity of objects by examining whether commonly co-occurring inter-attribute values are present in the two objects [8]. This insight is equally relevant even for quantitative domains of data where a natural ordering of attribute values exists. The use of aggregate data behavior in order to measure similarity becomes more important for high dimensional data, where there may be considerable redundancies, dependencies, and relationships among the large number of attributes. Since a lot of the proximity information may be hidden in the aggregate summary behavior of the data, the use of the linearly separable parametric model may miss many of the critical characteristics. We also note that some data domains such as text factor the correlation behavior indirectly into the distance function by using data transformation techniques such as Latent Semantic Indexing [12].

Fortunately, the parametric Minkowski model can be directly generalized in a way so as to make it more sensitive to the correlation behavior of the data. In order to do so, we use the *generalized parametric Minkowski model* in order to define distances. In this case, we use a symmetric $d * d$ matrix Θ containing the parametric values. In this case, the distance between the point vectors $\overline{X}$ and $\overline{Y}$ is given by:

$$f(\overline{X}, \overline{Y}, \Theta) = (\overline{X} - \overline{Y})^T \cdot \Theta \cdot (\overline{X} - \overline{Y}) \quad (8)$$

We note that when the matrix Θ has non-zero entries only on the diagonal, the model reduces to the simple Minkowski case. Thus, the non-zero values on the non-diagonal entries provide the magnitudes of the correlations between the corresponding dimensions. As in the previous case, the gradient descent method can be used to determine the parameter values.

2.2 The Parametric Cosine Model

The parametric cosine model is actually a similarity function rather than a distance function, since higher values imply greater similarity. The basic idea behind the parametric cosine model is drawn from the cosine function used in text databases [19, 21]. We will propose this function for the case of boolean data sets, though it can be directly extended to quantitative data sets by applying a simple preprocessing phase of discretization. For two boolean data points $X = (x_1 \ldots x_d)$ and $Y = (y_1 \ldots y_d)$, the cosine distance between these points is given by:

$$Cosine(X, Y) = \sum_{i=1}^{d} \frac{x_i \cdot y_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \cdot \sqrt{\sum_{i=1}^{d} y_i^2}} \quad (9)$$

In many cases, the attributes are appropriately weighted and normalized in order to improve the quality of the distance function. An example of such normalization is the simple inverse document frequency normalization used in Information Retrieval applications. We note that such weighting or normalization is usually done purely on the basis of the statistical properties of the data set, but may often deviate significantly from how similarity may be perceived by a given user. In order to achieve this, we introduce a parametric cosine function which is more sensitive to the process of actually *learning* similarity from user behavior. The parametric cosine similarity of X and Y is defined in terms of

the d parameters $\lambda = \lambda_1 \ldots \lambda_d$:

$$f(X, Y, \lambda_1 \ldots \lambda_d) = \sum_{i=1}^{d} \lambda_i \cdot x_i \cdot y_i \quad (10)$$

We note that this is a *similarity function* as opposed to a distance function, since higher numbers imply greater similarity. As before, let us assume that the coordinates for each individual pair of points X^i and Y^i are denoted by $(x_{i1} \ldots x_{iN})$ and $(y_{i1} \ldots y_{iN})$ respectively. In this case, we find that by using the mean square error function, we again obtain a similar form of the error function:

$$E(\lambda_1 \ldots \lambda_k) = \sum_{i=1}^{N} \left((\sum_{j=1}^{d} a_{ij}\lambda_j) - \alpha_i \right)^2 \quad (11)$$

In this case, the value of a_{ij} is equal to $x_{ij} \cdot y_{ij}$. The remaining analysis is exactly similar to that of the Minkowski model.

2.3 Interpretability Issues

We note that the above techniques for distance function design naturally lead to a high degree of interpretability of the distance function in a given data domain. For example, unlike simple Euclidean functions which treat all dimensions equally, the parametric Minkowski model provides a clear idea of which features are the most important based on the underlying data behavior. It is clear that any feature i for which the weight λ_i is small is less relevant in determining the similarity value.

Similarly, in the case of the parametric cosine function, the importance of each attribute is decided by the corresponding weights. We note that in the case of the text domain, in which similarity functions have been extensively researched, such weights are determined by using term/inverse document frequency normalization [19]. While it is intuitively clear that such normalization gives greater weight to statistically infrequently occurring events, it does not incorporate the characteristics of the particular kind of data at hand. The ability to capture such relationships lies at the heart of a technique which incorporates such feedback into the learning process.

3. A DISCUSSION ON THE USE OF PURELY LEARNING STRATEGIES

In the previous section, we discussed a method which used purely parametric forms for learning a distance function with a particular canonical form. While such a technique has the advantage of providing a closed form representation to the distance function, a natural alternative is to transform the problem of distance function design to that of pure regression modelling. In order to illustrate this point, consider the case, when we have a set of N object pairs $(X_1, Y_1) \ldots (X_N, Y_N)$ along with the corresponding user responses $\alpha_1 \ldots \alpha_N$. In this case, we can recreate a new composite object containing $2 \cdot d$ attributes, by concatenating the attributes of X_i and Y_i for each value of i. Therefore, we can create the new object Z_i, by concatenating the d attributes each of X_i and Y_i. Now, we can use the set $Z_1 \ldots Z_N$ along with $\alpha_1 \ldots \alpha_N$ as the training data in order to model the distance function.

While the simplicity of the above model is tempting, it has some drawbacks as well. In most data domains, distance functions have some naturally desirable qualities in terms of the relationship of the quantitative class variable with the feature variables. Parametric techniques such as the Cosine or Minkowski models precisely try to encode such information within the training process. Just as the use of a fixed form such as the L_p-norm is inflexible, the absence of any canonical form ignores any kind of understanding of the inherent nature of distance functions. In order to create accurate models in cases where there are no guiding canonical forms, the amount of data required is likely to considerable. This is a significant drawback in a system in which the generation of each data point requires human intervention. However, it is possible to make some compromises in order to incorporate feature specific knowledge into the training process without at the same time using a fixed and inflexible canonical form. To do so, we propose to use feature transformations which capture the natural functional properties of many distance functions. Thus, for the set of N object pairs $(X_1, Y_1) \ldots (X_N, Y_N)$ along with the corresponding user responses $\alpha_1 \ldots \alpha_N$, one could create the set of functional features $g_1(X_1 \ldots X_N, Y_1, \ldots Y_N), \ldots g_q(X_1 \ldots X_N, Y_1, \ldots Y_N)$. Thus, a set of new features are created by the transformation. These new features are created by the user in order to help the learning process to some extent, while not designing precise functional forms for the distance function. While a regression model built on the original feature-response tuples (X_i, Y_i, α_i) ought to be the most flexible, the reality is that the constraints on training data availability make such a system difficult to implement. The use of functional features indirectly incorporates knowledge about the natural behavior of distance functions, which would otherwise have to be learned. In the next subsections, we will discuss some functional transformations which derive their motivations from the Minkowski and Cosine models respectively.

3.1 The Difference Functional Transformation

One example of such a system is the difference functional transformation. In this case, we have a total of $q = N$ functional features $g_1(\ldots) \ldots g_N(\ldots)$ which are defined by $g_i(\ldots) = ||X_i - Y_i||$. In this model, smaller values of α_i indicate greater similarity. We note that this model is influenced by the Minkowski model, since the latter is also determined completely by the values of $||X_i - Y_i||$. We note that such a transformation directly reduces the number of features by a factor of two, while adding additional knowledge in order to improve the efficiency of the knowledge discovery process.

3.2 The Product Functional Transformation

This model assume that each of the values of (X_i, Y_i) are boolean. However, quantitative values can easily be converted to boolean by the use of data discretization. As is the case of the cosine functional representation, this model uses only the products of corresponding values of (X_i, Y_i) in order to define the new features on which the distance function model is constructed. In this case $q = N$ new features $g_1(\ldots) \ldots g_N(\ldots)$ are defined. Specifically, we have $g_i(\ldots) = X_i \cdot Y_i$. In this model, larger values of α_i indicate greater similarity. This model derives its motivation from the cosine function in which the (normalized) product sum defines the value of the similarity function. However, a product functional transformation is somewhat more flexible than a parametric cosine model because it recognizes that the actual similarity value may be a more complex combina-

tion of feature interactions than is allowed by the parametric closed form.

3.3 Use of Functional Features for Distance Function Learning

Once the functional features have been created, we use them directly for distance function learning. Specifically, a standard classification model is constructed on this new set of features in order to relate them to the user-defined similarity values. We note that the classification model often works much more effectively on this new set of features, because they often incorporate the inherent characteristics of natural similarity (or distance) functions. For example, the difference functional transformation automatically stores the relationship between two of the columns which is often likely to be very relevant to the final distance value. While a purely learning strategy may also try to learn this relationship, it is likely to be significantly more inaccurate when insufficient amounts of data are available. In fact,as our empirical results will show, even the feature transformational models do not work as well as the more restricted parametric forms when small amounts of training data are available. In general, a direct learning strategy often finds it more difficult to model the natural characteristics of similarity calculations.

4. COMBINING STRATEGIES BY HOLDOUTS

One of the interesting points to be noted is that no particular strategy may provide an optimum response. This is because a given feature transformation or canonical form may incorporate certain advantages for a particular data domain, while it may not be quite as effective for other domains. Unfortunately, without additional domain knowledge, it is not possible to know the most effective transformation a-priori. In order to handle this problem, it is possible to create a *holdout strategy* in picking the most effective canonical form or transformation.

In order to compare different strategies, we divided the data set $\mathcal{D}$ of size N into two subsets $\mathcal{D}_1$ and $\mathcal{D}_2$ of size N_1 and N_2 respectively. The set $\mathcal{D}_2$ is defined as the *holdout* set. Once these two data sets have been constructed, we build each model separately on the data set $\mathcal{D}_1$. Next, we determine the estimate of the distance function value for each record in the data set $\mathcal{D}_2$, using the model constructed on the data set $\mathcal{D}_1$. The average error in the distance function using each canonical form or feature transformation is calculated and used for the determining which of the various methods is most appropriate for that particular data domain.

We note that a normalization issue needs to be taken into account while combining different strategies. Since some of the methods discussed above correspond to maximization objective functions, whereas others correspond to minimization objective functions, we need to utilize the user feedback in a consistent way. In order to achieve this, we assume that $\alpha_1 \ldots \alpha_N$ are generated using the minimization convention, whereas the corresponding maximization counterparts are generated as $M-\alpha_1 \ldots M-\alpha_N$. Here M is an upper bound on any value of α_i and may be chosen as $M = \max_i\{\alpha_i\}$. We note that while using this convention the absolute inaccuracy in modelling the maximization objective function value is directly comparable to that of modelling the minimization value. This makes it possible to compare different distance function models in order to test the appropriateness of that approach for the given data set.

5. EMPIRICAL RESULTS

The aim of this section is to show that the models discussed in this paper can effectively capture the distance functions quite accurately for a number of real data sets. All results were implemented on an AIX4.1.4 system at 233MHz and 200MB of main memory.

5.1 Data Generation

In order to test the performance of our methodology, we used a number of data sets from the UCI machine learning[1] repository. While all data sets were based on real domains, the user responses were synthetically generated from within the data. We used one of the feature variables in order to synthetically generate the distance function responses $\alpha_1 \ldots \alpha_N$. This feature was stripped from the other records. Let the records in the original databases $\mathcal{D}$ be denoted by $W_1 \ldots W_r$. We will denote the values of the special column from which the objective function was generated by $z_1 \ldots z_r$. In order to actually generate the objective function values, we randomly picked N pairs of objects from the records $W_1 \ldots W_r$. Let us assume that the ith pair of objects picked is denoted by $(X_i, Y_i) = (W_{n_i}, W_{m_i})$. Thus, n_i and m_i are indices of the object pairs which are sampled. The value of the objective function α_i was given by $||z_{m_i} - z_{n_i}||$ while testing for minimization objective functions[2] such as the parametric Minkowski distance function or the difference functional transformation. On the other hand, while dealing with maximization objective functions such as the parametric cosine model or the product functional transformation, this value was set at $M - ||z_{m_i} - z_{n_i}||$, where M is an upper bound on the value of $||z_{m_i} - z_{n_i}||$. The value of M is chosen to be $\max_i\{z_i\} - \min_i\{z_i\}$. We note that this method of creating the similarity value is synthetic as opposed to manual. Since a difference function was used on a particular column to consistently generate the similarity values, this generates a similarity function with a particular bias, as is the case with most user defined similarity perceptions. For example, when the column for "Age" is used in order to define the similarity function, the difference function on this column creates a function which will make two objects similar only when they are similar in age. Some of the features are likely to be more important than others while making such a determination. However, the aim of this section is to show the *effectiveness* of these techniques in modelling any kind of similarity, whether they be manually defined or synthetic. The advantage of using synthetic data sets is to be able to provide objective measures of accuracy of the method.

5.2 Accuracy Measurements

In addition to the N data pairs which are used for distance function modelling, we also generate a separate set of N' data pairs for testing the effectiveness of the distance function. Let us assume that these N' pairs are denoted by $(X'_1, Y'_1) \ldots (X'_{N'}, Y'_{N'})$ along with corresponding test similarity values of $\beta_1 \ldots \beta_{N'}$. Once the appropriate model has

[1] http://www.cs.uci.edu/~mlearn.

[2] Smaller values are better for the case of minimization objective functions.

been constructed, we use it to predict the similarity values $\beta'_1 \ldots \beta'_{N'}$ for each of the N' test cases. The absolute difference between each value of β_i and β'_i (denoted by $||\beta_i - \beta'_i||$) is the inaccuracy on that particular test example. The overall inaccuracy $\mathcal{I}$ is then determined by averaging the same calculation over all the different test examples. Therefore, we have:

$$\mathcal{I} = \sum_{i=1}^{N'} ||\beta_i - \beta'_i||)/N' \quad (12)$$

We calculated this error value for each individual model as well as the composite model by obtaining the best canonical form in the holdout model.

While the above measures give insight into the absolute errors of the different strategies, the real effectiveness of a distance function is determined by how well the distance function orders the different objects in terms of their relative distance values. Thus, for example, if objects A and B are closer to one another than the user perceives objects C and D, then we would like that the model is able to accurately predict this fact. To this effect, we sampled two data pairs at a time and calculated the fraction of time that the correct ordering was maintained. We will refer to this value as the ordering accuracy. Therefore, if n_1 and n_2 be the respective number of times that the correct and incorrect order was maintained between randomly chosen pairs of objects, then the ordering accuracy was given by $n_1/(n_1+n_2)$. Such a measure is independent of the absolute values of the similarity and measures quality in terms of percentage of ordering correctness. This makes it possible to compare distance functions with widely varying absolute values.

5.3 Results

In Table 1, we have illustrated the statistics of the different data sets. For each data set, the column which was used to generate the distance function value is denoted by *columnid*. This column was always chosen to be numeric in order to define the distance function effectively. This column was also removed from the data set, so that the distance function was modelled using only the remaining features. The resulting dimensionality (after removal of this column) is indicated in Table 1. In the table we have also illustrated the maximum and minimum values of the target objective function once they were synthetically generated. These values provide an idea of the range of the absolute error in the estimation of the final objective function.

The absolute error for each data set is illustrated in Table 2. Each of the different methods proposed in this paper were tested together with a pure learning methodology as a baseline. The learning method used was a decision (regression) tree classifier [17]. It is interesting to see that the pure learning method was never as effective as either the parametric functional methods or the transformational learning methods in any case. The reason for this is that the parametric distance functions and the transformational features encode critical information about the underlying data. The other salient observation is that the effectiveness of the different distance functions varies from one data set to another. For example, the parametric distance functions are more effective than the transform learning functions for the Ecoli and Machine data sets, whereas the transformational features are more effective for the other cases. Even among the different kinds of parametric forms, there were slight variations

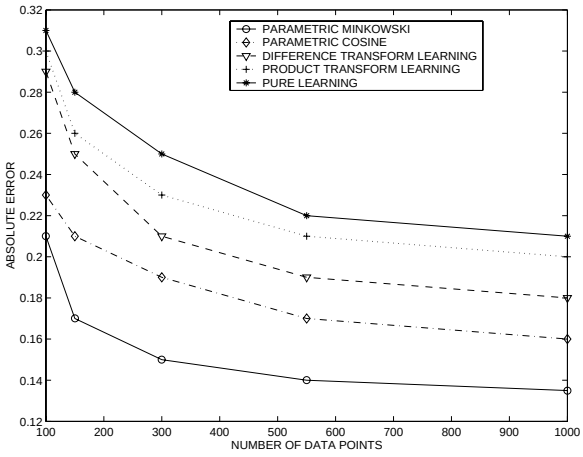

Figure 1: Accuracy versus Data Set Size (Ecoli Data Set)

for different data sets. For example, for the Ecoli data set the parametric Minkowski function had a lower error than the parametric cosine function, whereas for the glass data set, the reverse was true. The same observation was true about the transformational features. In terms of the absolute values of the error rates, the error values for the pure learning strategy are significantly higher, whereas the differences between the other four methods are relatively small. This tends to indicate that any of the latter methods would be significantly robust for most data sets. We also note that by using the hold out strategy discussed earlier, it is possible to isolate and use the most effective distance function for a given data set.

In Table 3, we have illustrated the ordering accuracy of the different methods for each of the data sets. In this case, we have omitted the pure learning strategy, because our results in Table 2 show that it is conclusively worse than any of the other techniques in each case. Instead, we have used the ordering error on the simple Euclidean function as a baseline in order to show the advantages of this method over the traditional Euclidean metric. The results in Table 2 indicate that the relative ordering accuracies of the different methods closely mirror the absolute inaccuracies of the different methods. Furthermore, in each case, the techniques turn out to be more effective than the traditional Euclidean distance function. The reason for this is that the Euclidean function is not geared to provide the particular notion of nearness which is application dependent and data driven. Thus, the combined results of Table 2 and 3 show that the careful method of designing the distance function is able to achieve results that either traditional distance functions or pure learning strategies cannot achieve.

5.4 Data Size Requirements

The previous section showed that while the quality of results obtained are approximately comparable, they provide little guidance about the comparative data size requirements in order to achieve effective results. We note that since the focus of this paper is to learn distance functions from domain specific data, the amount of data available may often be limited. Therefore, it is useful to have an idea of how well the different methods compare in terms of the amount

Table 1: Statistics of Different Data Sets

Data Set	Dimensionality	Column Id	Minimum Target Similarity	Maximum Target Similarity
Ecoli	6	2	0	0.89
Glass	8	6	0	5.29
Machine	7	10	0	1213
Auto-Mpg	7	7	0	12
Housing	12	14	0	45

Table 2: Absolute Estimation Error with Different Methods

Data Set	Parametric Minkowski Function	Parametric Cosine Function	Difference Transform Learning	Product Transform Learning	Pure Learning
Ecoli	0.101	0.110	0.121	0.127	0.153
Glass	0.443	0.412	0.4000	0.353	0.561
Machine	41.3	40.8	43.3	41.5	92.6
Auto-Mpg	2.7	2.9	2.5	2.6	4.6
Housing	7.6	7.1	5.4	5.3	9.7

Table 3: Ordering Accuracy Comparisons

Data Set	Parametric Minkowski Function	Parametric Cosine Function	Difference Transform Learning	Product Transform Learning	Euclidean Function
Ecoli	0.81	0.75	0.72	0.71	0.69
Glass	0.80	0.81	0.82	0.78	0.71
Machine	0.89	0.90	0.86	0.88	0.74
Auto-Mpg	0.78	0.76	0.79	0.78	0.65
Housing	0.71	0.73	0.77	0.78	0.62

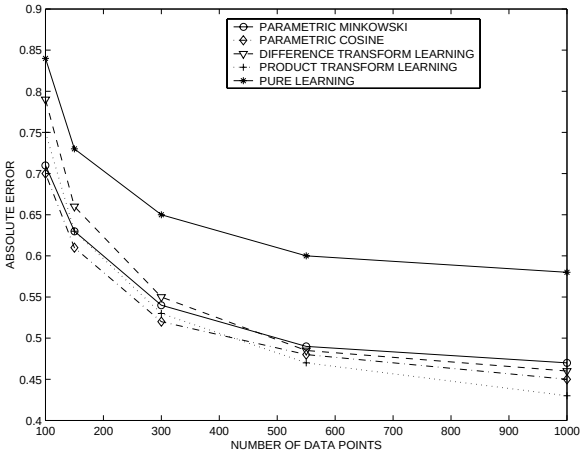

Figure 2: Accuracy versus Data Set Size (Glass Data Set)

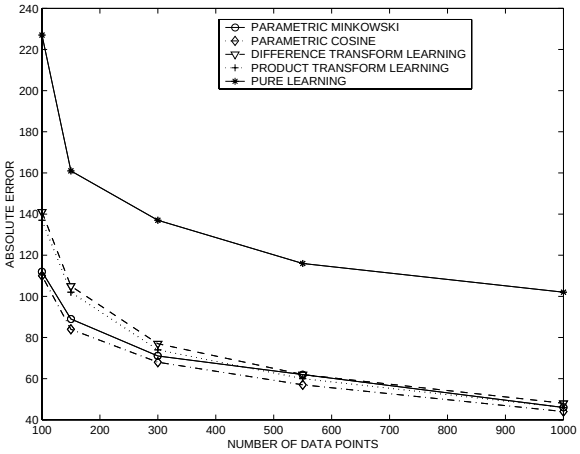

Figure 3: Accuracy versus Data Set Size (Machine Data Set)

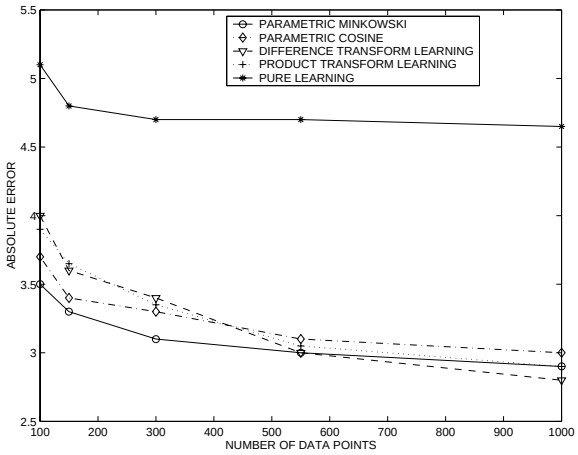

Figure 4: Accuracy versus Data Set Size (Auto-Mpg Data Set)

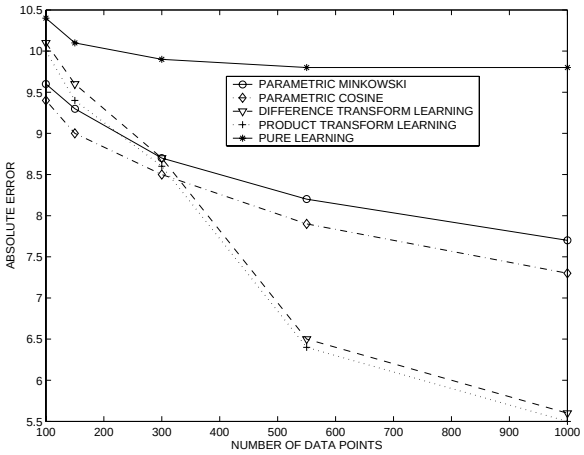

Figure 5: Accuracy versus Data Set Size (Housing Data Set)

of data required in order to achieve accurate modelling of the distance function. The results for the Ecoli, Glass, Machine, Glass, Auto-Mpg and Housing data sets are illustrated in Figures 1, 2, 3, 4 and 5 respectively. On the X-axis, we have illustrated the number of data pairs used for training, whereas on the Y-axis, we have illustrated the absolute error resulting from the training procedure. In each case, the error reduces with increasing number of training pairs, but stabilizes at some point. We note that the results in Table 2 are each based on 10,000 data pairs which is this stable region. An interesting common characteristic of all these results is that in each case, the parametric forms require much fewer number of records to reach the region of stability than the functional feature transformations. In most cases only about 500 to 1000 data pairs were sufficient to reach the region of stability. In fact, in some of the cases such as the Glass, Auto-Mpg and Housing data sets (Figures 2, 4 and 5), the error rate of the parametric forms is lower for smaller number of records, even though the steady state value for larger number of records is higher. This behavior is particularly evident in the housing data set of Figure 5. The reason

for this is that the parametric models are somewhat more restrictive in performing distance function modeling. Thus, fewer number of data pairs are required in order to reach the optimal error value. At the same time, Table 2 shows that the qualitative results are not very different between the transformational feature techniques and the parametric forms. This indicates an overall advantage for the parametric method since the same overall qualitative results can be obtained with much less data. Furthermore, the parametric forms have the advantages of being compact, since they can be easily computed and expressed in closed form. When combined with the advantages of low data size requirements, it is clear that parametric modeling is a promising technique for systematic design of distance functions.

6. CONCLUSIONS AND SUMMARY

While distance functions continue to be one of the most important and widely used element of many data mining problems, they are often implemented using naive methods for many data domains. In this paper, we discussed a user-centered and systematic method for modeling distance functions effectively. We illustrated the advantages of a parametric approach over both a fixed distance function and a purely learning methodology. We also discussed the usefulness of transformational features in distance function modeling. Given the importance and application specificity of distance functions, the results in this paper can be valuable for a large number of data mining problems.

7. REFERENCES

[1] C. C. Aggarwal. Re-designing distance functions and distance based applications for high dimensional data. *ACM SIGMOD Record*, March 2001.

[2] C. C. Aggarwal. Towards Meaningful High Dimensional Nearest Neighbor Search by Human-Computer Interaction. *ICDE Conference*, 2001.

[3] C. C. Aggarwal, P. S. Yu. The IGrid Index: Reversing the Dimensionality Curse for Similarity Indexing in High Dimensional Space. *KDD Conference*, 2001.

[4] C. C. Aggarwal, A. Hinneburg, D. A. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. *ICDT Conference*, 2001.

[5] R. Agrawal, K.-I. Lin, H. S. Sawhney, K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. *VLDB Conference*, pages 490-501, 1995.

[6] D. Bertsekas. Nonlinear Programming. *Athena Scientific*, 2nd Edition, 1999.

[7] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft. When is Nearest Neighbors Meaningful? *ICDT Conference*, 1999.

[8] G. Das, H. Mannila. Context-Based Similarity Measures for Categorical Databases. *PDKK Conference*, 2000.

[9] J. Foote. A Similarity Measure for Automatic Audio Classification. *AAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*, 1997.

[10] D. Gunopulos, G. Das. Time Series Similarity Measures and Time Series Indexing. *ACM SIGMOD Conference*, 2001.

[11] A. Hinneburg, C. C. Aggarwal, D. Keim. What is the nearest neighbor in high dimensional spaces? *VLDB Conference*, 2000.

[12] S. Deerwester et al. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6): 391-407, 1990.

[13] N. Katayama, S. Satoh. Distinctiveness Sensitive Nearest Neighbor Search for Efficient Similarity Retrieval of Multimedia Information. *ICDE Conference*, 2001.

[14] E. Keogh, M. Pazzini. Scaling up Dynamic Time Warping to Massive Data Sets. *Principles of Data Mining and Knowledge Discovery*, pages 1-11, 1999.

[15] P. Moen. Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining. ining. PhD Thesis, Report A-2000-1, Department of Computer Science, University of Helsinki, February 2000. http://www.cs.helsinki.fi/TR/A-2000/1/.

[16] A. Hinneburg, D. A. Keim, M. Wawryniuk. HD-Eye: Visual Mining of High Dimensional Data. *IEEE Comp. Graphics and Applications*, 19(5), pp. 22-31, 1999.

[17] M. James. Classification Algorithms, *Wiley*, 1985.

[18] M. M. Richter. On the notion of similarity in case-based reasoning. Mathematical and Statistical Methods in Artificial Intelligence (ed. G. della Riccia et al). *Springer Verlag*, 1995, p. 171-184.

[19] G. Salton, M. J. McGill. Introduction to Modern Information Retrieval. *Mc Graw Hill*, New York, 1983.

[20] T. Seidl, H.-P. Kriegel: Efficient User-Adaptable Similarity Search in Large Multimedia Databases. *VLDB Conference*, 1997.

[21] A. Singhal, G. Salton, M. Mitra, C. Buckley. Pivoted Document Length Normalization. *ACM SIGIR Conference*, 1996.

[22] L. Wu, C. Faloutsos, K. Sycara, T. Payne. FALCON: Feedback Adaptive Loop for Content-Based Retrieval. *VLDB Conference*, 2000.