Density-Based Indexing for Approximate Nearest-Neighbor Queries

Kristin P. Bennett
Rensselaer Polytechnic Inst.
and Microsoft Research
bennek@rpi.edu

Usama Fayyad
Microsoft Research
One Microsoft Way
Redmond, WA 98052
fayyad@microsoft.com

Dan Geiger
Computer Science, Technion
and Microsoft Research
dang@cs.technion.ac.il

Abstract

We consider the problem of performing Nearest-neighbor queries efficiently over large high-dimensional databases. To avoid a full database scan, we target constructing a multi-dimensional index structure. It is well-accepted that traditional database indexing algorithms fail for high-dimensional data (say $d > 10$ or 20 depending on the scheme). Some arguments have advocated that nearest-neighbor queries do not even make sense for high-dimensional data. We show that these arguments are based on over-restrictive assumptions, and that in the general case it is meaningful and possible to build an index for such queries. Our approach, called DBIN, scales to high-dimensional databases by exploiting statistical properties of the data. The approach is based on statistically modeling the density of the content of the data table. DBIN uses the density model to derive a single index over the data table and requires physically re-writing data in a new table sorted by the newly created index (i.e. create a clustered-index). The indexing scheme produces a mapping between a query point (a data record) and an ordering on the clustered index values. Data is then scanned according to the index. We present theoretical and empirical justification for DBIN. The scheme supports a family of distance functions which includes the traditional Euclidean distance measure.

1 Introduction

Nearest-neighbor queries (NN), also known as *similarity queries* [19], are an important class of queries in data mining applications. The basic problem is: given a data table of records, find the nearest (most similar) records to a given query record. Specifically, given a data record represented as a d-dimensional vector of values q, and a data set D, determine a record $x \in D$ such that $\text{Dist}(x, q) = (x - q)^T S(x - q)$ is minimized. S is a positive semi-definite matrix. If the distance measure is the Euclidean distance or 2-norm, then S is the identity matrix. This definition can be generalized to retrieving the k nearest records.

The many applications of NN queries include: predictive modeling, product catalog navigation (find similar products based on a long list of specifications), fraud detection, customer support, problem diagnosis, and management of knowledge bases. NN queries also provide a more flexible means for querying databases by supporting a "find similar" capability useful in text or image databases, e.g., finding similar documents based on long vectors of keyword counts, or retrieving an image from a large database of images based on a query image. Even if comparisons are not done on a per-pixel basis, typically feature extraction over an image or document will result in many features and a dimensionality in the tens or hundreds. With the growth of Data Warehousing, NN queries are also useful for flexible querying as well as data cleaning applications (e.g. matching records for merge-purge, or retrieving partial matches).

Defining efficient methods for indexing very large databases to support NN queries remains an unsolved problem despite a large literature on the topic. The obvious approach to finding the NN is to scan the data and compute the distance between every record in the database and the query record – an unacceptable solution in massive databases. Indexing schemes developed in the database literature focused primarily on efficient data structures for retrieving matches from disk. Many studies have found that the traditional indexing methods frequently fail to be useful if the dimensionality of the data is high [16, 1, 5, 4, 3, 20, 14, 18]. There is increasing interests in avoiding this "curse of dimensionality" by performing *approximate* NN queries [13]. In data mining applications, the NN distance metric is a mathematical approximation of a user-defined similarity criterion which is often vague. Thus, improving performance by providing good

KDD-99 San Diego CA USA

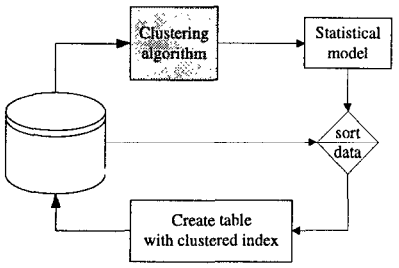

Figure 1: Overview of derivation of index structure

approximate NN estimates is an appropriate tradeoff.

In this work we introduce a probability density based indexing (DBIN) method for calculating **probabilistic** or **p-approximate NN queries** (p-NN). A point s is a p-NN of q if with probability greater than $1-p$, s is a NN of q. Note that p-NN differs from the prior $\epsilon-$approximate NN (ϵ-NN) [13]. A point s is the ϵ-NN of q if for all $x \in D$, $dist(s,q) \leq (1+\epsilon)dist(x,q)$. We assume the data table is modeled as a probability density function (pdf) consisting of a mixture of components (e.g. Gaussians). The offline component of DBIN estimates this pdf and uses it to construct a clustered index of the data. At query time this pdf is used to prioritize and terminate the search for the NN.

In the offline phase described in Section 3 and illustrated in Figure 1, the estimated pdf is calculated using scalable clustering algorithms [8, 7]. Each cluster corresponds to a Gaussian. Using the pdf, we produce a global ordering on the data records that maximizes proximity of records that are "near" each other according to a family of distance measures. The resulting data clusters are used to decide the ordering of the records in the table. Essentially the records need to be sorted by their cluster ID and the table materialized. This table will be scanned via a clustered index for the purposes of finding NN.

Section 4 describes the query process. The query point is mapped to relevant values of the index column. The estimated pdf is used to **prioritize** search by producing an ordering by which the data pages should be scanned. As data is scanned, the statistical model is used to calculate the probability that the current NN or k-NN estimate is correct. When this estimated probability attains some threshold, the scan is stopped. It is important to note that at query time we require one additional query lookup stage to determine the cluster to be scanned next. The lookup consists of using the model to determine which cluster to scan next for the given query item. The clustered index is then used to scan the members of that cluster. In a file system implementation we simply write all members of a cluster in a separate file. This provides a simple means of scanning the file containing all the data items that belong to the cluster under consideration.

The basic intuition behind the method is that clusters exist in real data and that there is a strong correspondence between cluster membership and proximity according to the distance measure. In this paper we develop a theory and empirical evidence to justify this intuition. Much of the existing database literature has analyzed the NN problem under assumptions that data is identical and independently distributed (iid) in each dimension. Under these assumptions Beyer et al. [6] showed that NN queries in high dimensions are asymptotically meaningless–the ratio of the nearest-neighbor and the farthest-neighbor distance approaches one as the dimensionality increases. So, for example, if data is drawn from a single Gaussian in high dimensions, then all the points are roughly the same distance apart. Herein we extend their results to show that for clustered data generated from a mixture model of well-separated Gaussians, NN is meaningful. In addition, efficient p-NN indexing methods exist based on clustering the data and modeling the contents statistically. We derive *stability conditions* under which DBIN (our indexing method) will perform optimally. Our computational results in Section 6, show that when these stability conditions are violated the performance of the method degrades slowly and is still more efficient than a full scan. The statistical model we utilize also allows us to detect when our indexing structure is unlikely to be useful. Hence an optimizer may use this information to decide the tradeoff between a sequential scan and using the index structure. The result is an efficient technique for indexing databases for multi-dimensional p-NN queries for a general family of distance functions that includes the most commonly used ones.

2 Related Indexing Methods

Much of the existing work on k-NN queries use tree-based indexing. Some strategy is used to group the data by proximity. Each group is characterized by some bounding object and the bounding objects are organized in a tree structure to support efficient querying. For example Vornoi-based indexing and X-trees use bounding boxes [5, 4], SS-Trees use bounding hyperspheres [20]. Consequently, points or regions of the data space can be eliminated as possible candidate NN. Most methods work well in low dimensions but exhibit poor behavior in high-dimensions [13]. A common strategy is to map the data into a lower-dimensional space and perform the NN search in that space [10]. Some hope has been expressed for avoiding the "curse of dimensionality" by using approximate NN

queries [13]. Locality-Sensitive Hashing [12] provides a general scheme for mapping data into a lower-dimensional space and finding NN in the reduced space that are approximate ϵ-NN in the original space. DBIN also finds approximate NN but in a probabilistic sense.

Why are approximations needed? Bounding objects do not effectively constrain search in high-dimensional spaces. For exact NN queries, if the estimated query ball intersects the bounded regions, that region must be investigated. Unfortunately, as the dimensionality grows, the intersected area grows dramatically. Consider this simple example. Suppose a cluster of data is in a d-dimensional sphere of radius r inscribed in a minimum bounding box. The ratio of the volume of the inscribed sphere to the volume of the minimum bounding box converges rapidly to 0 as d goes to infinity[1]. When d is large, most of the box volume is in the corners (i.e. outside the ball) which contain no data. A NN query will intersect most data pages, causing them to be scanned needlessly. Increase in overlap with dimensionality is a manifestation of this phenomena [5]. In contrast our probabilistic approximate NN identifies data partitions most likely to contain relevant data and avoids regions unlikely to contain relevant points.

3 Preparing the Index: Offline Phase

Step 1 (Estimating Density): The first step in the offline preprocessing phase is estimating the density of the data. By density we mean a joint probability density function f governing the distribution of data values. We chose to model the data using a mixture of Gaussians. Hence the pdf has the form: $f(x) = \sum_{i=1}^{K} p_i G(x|\mu_i, \Sigma_i)$ where $G(x|\mu_i, \Sigma_i)$ is a Gaussian pdf parameterized with a mean vector μ_i and a covariance matrix Σ [2]. There are several desirable properties of the Gaussian mixture model for this problem including:

1. Expressive power: any distribution can be represented as a mixture of Gaussians [17].
2. Computational efficiency: using our existing scalable approaches to estimate the density in this form [8].
3. Suitability to distance measure. We can exploit this form to help construct index structures for a large family of distance functions.
4. Ability to analyze the model, study its properties, and convenience of its use to compute probabilities of regions and events of interest.

[1] Specifically, $\left[\frac{r^d\sqrt{\Pi d}}{\Gamma(d/2+1)}\right]\left[\frac{1}{(2r)^d}\right] \to 0$.

[2] Recall that a given Gaussian G with dimension d, mean vector μ and covariance matrix Σ is given by the standard form: $G(x|\mu,\Sigma) = \frac{1}{\sqrt{2\pi^d|\Sigma|}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$.

There are many ways to find a mixture-of-Gaussians pdf that fits data. We use data clustering (a.k.a. segmentation) algorithms as an efficient means of obtaining the statistical model [8]. In clustering, the input is a number of clusters, an initial guess at where the clusters are centered, and what their initial parameters (means and covariances) are. The initial guess is typically some random setting unless one has prior knowledge of the data. We use a scalable EM algorithm [8] that was developed to cluster large databases using a mixture-of-Gaussians pdf model. The actual algorithm is discussed in detail in [8] and can build models over large databases in a single scan. In this paper we assume that the clustering model is given as an input. We focus on its use to derive a multidimensional index.

Step 2 (Determine Clusters): The model consisting of a mixture of K Gaussians determined by scalable EM [8] is used to precompute the index structure based on cluster membership. Membership of a data point $x \in D$ in a cluster $C_j \in \{C_1, \ldots, C_K\}$ is determined to be the cluster with highest probability:

$$\Pr(x|C_j) = \frac{G(x|\mu_j, \Sigma_j)}{\sum_{i=1}^{K} G(x|\mu_i, \Sigma_i)}$$

where $G(x|\mu_i, \Sigma_i)$ is the Gaussian representing cluster C_i. The optimal Bayes decision rule for a mixture-of-Gaussians pdf [9, Chapter 2] is as follows: a data point x is assigned to cluster C_l if $i = l$ maximizes the quantity

$$g_i = -\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i) - \frac{1}{2}\log|\Sigma_i| + \log p_i. \quad (1)$$

This decision rule maximizes the probability that x is generated by the i^{th} Gaussian distribution (cluster C_i) and in that sense it is an optimal decision. This decision rule defines a set of K regions $R_1, \ldots, R_K$ in high-dimensional Euclidean space where R_i contains the set of points assigned to cluster C_i. We define $R(x)$ to be the *region-identification function*; that is, each data point x belongs to region $R(x)$ where $R(x)$ is one of the K regions $R_1, \ldots, R_K$. An in-memory directory structure consisting of the means, variances and weights of the K Gaussians is required to determine $R(x)$.

Step 3 (Materialize Sorted Data Table): The next step is to materialize the table on disk sorted by cluster membership. If this new table is to be used within a database, then we assume a clustered index will be created on the new column. Alternatively, data from each cluster can be written in a separate file if the scheme is to be used with a file system (no DBMS). The primary distinguishing characteristic of DBIN is its use of a model of the density to reorganize the data. Most other indexing schemes, due to efficiency considerations,

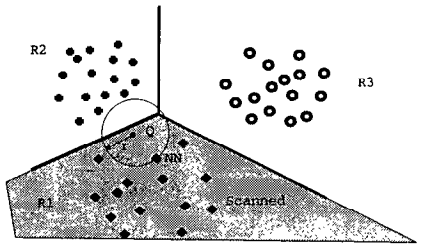

Figure 2: Stable Query Example

define regions by examining one dimension at a time. Projections to a single dimension can cause confusion when the data is high-dimensional. In contrast, clustering works on all dimensions simultaneously.

4 The Query Algorithm and Its Probabilistic Analysis

The second component of DBIN is activated at query time. It finds with high probability a NN of a query point q, denoted by $n(q)$. The probability model is used to prioritize the order of clusters to be scanned and to terminate search once the NN has been estimated with high probability. For its description we use the notation $B(q,r)$ to denote the query ball, a sphere centered on q and having radius r. We also denote by E the set of regions scanned so far, and by e the knowledge learned by scanning the regions in E. A NN of q is then found as follows.

NearestNeighbor (q ; $R_1, \ldots, R_K, f$)

1. Let C_j be the cluster assigned to q using the optimal Bayes decision rule (Equation 1);
2. Scan data in R_j and determine nearest neighbor $n(q)$ in R_j and its distance r from q;
3. Set $E = \{R_j\}$;
4. While $\mathrm{P}(B(q,r)$ *is Empty* $\mid e) <$ *tolerance*

   a. Find a cluster C_j not in E which minimizes $\mathrm{P}(B(q,r) \cap R_j$ *is Empty* $\mid e)$;
   b. Scan the data in R_j; Set $E = E \cup \{R_j\}$;
   c. If a data point closer to q is found in R_j, let $n(q)$ be that point,
   and set r to be the new minimum distance.

The quantity $\mathrm{P}(B(q,r)$ *is Empty* $\mid e)$ is the probability that $B(q,r)$ is empty, given the evidence e collected so far. The evidence consists simply of the list of points included in the regions scanned so far. Before we show how to compute this quantity and how to compute $\Pr(B(q,r) \cap R_i$ *is Empty* $\mid e)$, we explain the algorithm using the simple example depicted in Figure 2.

In this example, the optimal Bayes decision rule generated three regions R_1, R_2 and R_3 whose boundaries are shown. A given query point q is found to reside in region R_1. The algorithm scans R_1, a current NN estimate is found, a current minimum distance r is determined, and a query ball $B(q,r)$ is formed. The query ball is shown in the figure. If our algorithm was deterministic it would be forced to scan R_2 and R_3 since they intersect the query ball. Instead the algorithm determines the probability that the ball is empty given the fact that region R_1 has been scanned. Suppose this probability does not exceed the tolerance. The algorithm must now choose between scanning R_2 and scanning R_3. A choice should be made according to the region that maximizes the probability to find a NN once that region is scanned, namely, the algorithm should scan the region that minimizes $\mathrm{P}(B(q,r) \cap R_i$ *is Empty* $\mid e)$. This quantity is hard to compute and so the algorithm approximates this quantity (using Equation 4 which we develop below). In this example, region R_2 is selected to be scanned. The algorithm now halts because $\mathrm{P}(B(q,r)$ *is Empty* $\mid e)$ is sufficiently large once R_1 and R_2 have been scanned.

We compute $\mathrm{P}(B(q,r)$ *is Empty* $\mid e)$ based on the assumption that the points are independent and identically distributed (iid) samples from the estimated mixture-of-Gaussians pdf f. In other words, we take f to be the true model that generated the database. The sensitivity of the algorithm to this assumption must be tested using real data sets. By the iid assumption, the probability the ball is empty is the product of the probabilities that each point does not fall in the ball. Consequently, we have

$$P(B(q,r) \textit{is empty} \mid e) = \prod_{i=1}^{n} [1 - P(x_i \in B(q,r) \mid x_i \in R(x_i))] \quad (2)$$

where $R(x_i)$ is the region of x_i and n is the number of of data points. If $R(x_i)$ has been scanned then $P(x_i \in B(q,r) \mid x_i \in R(x_i)) = 0$. In general, $P(x_i \in B(q,r) \mid x_i \in R(x_i))$ is not computable. Fortunately, Theorem 4.1 below shows that we can use the approximation

$$P(x_i \in B(q,r) \mid x_i \in R(x_i)) \approx P(x_i \in B(q,r) \mid x_i \text{ generated by } C_j) \quad (3)$$

where C_j is the cluster assigned to x_i using the optimal Bayes decision rule. The same reasoning as above and the fact $P(x_i \in B(q,r) \cap R_j | x_i \in R_j) = P(x_i \in B(q,r) | x_i \in R_j)$,

$$\begin{aligned} &\mathrm{P}(B(q,r) \cap R_j \textit{ is Empty} \mid e) \\ &= [1 - P(x_i \in B(q,r) \mid x_i \in R_j)]^{n(R_j)} \\ &\approx [1 - P(x_i \in B(q,r) \mid x_i \text{ generated by } C_j)]^{n(R_j)} \end{aligned} \quad (4)$$

where $n(R_j)$ is the number of points falling in region R_j.

The remaining task of computing

$$P(x \in B(q,r) \mid x \text{ generated by } C_l)$$

has been dealt with in the statistical literature in a more general setting. This probability can be calculated numerically using a variety of approaches [15]. In particular, numerical approaches have been devised to calculate probabilities of the form $P[(x-q)^T D(x-q) \leq r^2]$ where x is a data point assumed to be generated by a multivariate Gaussian distribution $G(x|\mu, \Sigma)$ and D is a positive semi-definite matrix. When Euclidean distance is used to measure distances between data points, as we do herein, D is simply the identity matrix. Since numerical methods apply to distance functions of the form $d(x,q) = (x-q)^T D(x-q)$ where D is a positive semi-definite matrix, DBIN can be readily applied to any distance metric of this form.

The pdf of the random variable $(x-q)^T D(x-q)$ is a χ^2 distribution when $D = I$ and $q = \mu$. It is a noncentralized χ^2 when $D = I$ and $q \neq \mu$. It is a sum of non-centralized χ^2 pdfs in the general case of a positive semi-definite quadratic form. The general method uses a linear transformation to calculate the cumulative distribution of a linear combination of chi-squares. We use the method of Sheil and O'Murcheartaigh [15].

Note that the needed probability is computable for the general case, but we illustrate the idea of the computation assuming that Σ is diagonal and that the Euclidean distance metric is used (D is the identity matrix). We start with

$$(x-q)^T D(x-q) = \sum_{j=1}^{n} (x_j - q_j)^2, \qquad (5)$$

where q_j is the j-th component of q, and transform x_j to a standard normal random variable z_j.

$$(x_j - q_j)^2 = \sigma_j^2 \frac{((x_j - \mu_j) + (\mu_j - q_j))^2}{\sigma_j^2} = \sigma_j^2 (z_j + \delta_j)^2$$

where $\delta_j = \frac{(q_j - \mu_j)}{\sigma_j}$ and μ_j is the j-th component of μ. Now $(z - \delta_j)^2$ has a noncentral chi-square distribution with 1 degree of freedom and noncentrality parameter δ_j^2. We denote this as $\chi^2(1, \delta_j^2)$. The final result is

$$P[(x-q)^T D(x-q) \leq r^2] = P\left[\sum_{j=1}^{n} \sigma_j^2 \chi^2(1, \delta_j^2) \leq r^2\right]$$

This cumulative distribution function (cdf) can be expanded into an infinite series. In our implementation we used, AS 204 [11], the terms of the series are calculated until an acceptable bound on the truncation error is achieved. Other techniques for calculating or estimating quadratic normal forms have been proposed [15] and may also be suitable.

The next theorem shows that what we compute with these known techniques is not very far from what is desired. Let B be the event "$B(q,r)$ *is Empty*," C_1 be the event "*x is generated by* C_1," and R_i be the event "$x \in R_i$".

Theorem 4.1 *Let the events B, R_1, and C_1 be as defined above. Then, $|P(B|R_1) - P(B|C_1)|$ is bounded by*

$$P(B \cap R_1 \cap C_1) \cdot \left[\left|\frac{1}{P(R_1)} - \frac{1}{P(C_1)}\right| + P(B)\left|\frac{1}{P(C_1 \cap B)} - \frac{1}{P(R_1 \cap B)}\right|\right]$$

Proof. Using the definition of conditional probability and the triangle inequality, we have

$$\begin{aligned}
&|P(B|R_1) - P(B|C_1)| \\
&= |\tfrac{P(B\cap R_1 \cap C_1)}{P(R_1)} + \tfrac{P(B\cap R_1 \cap \bar{C}_1)}{P(R_1)} \\
&\quad - \tfrac{P(B\cap R_1 \cap C_1)}{P(C_1)} - \tfrac{P(B\cap \bar{R}_1 \cap C_1)}{P(R_1)} \\
&= |P(B \cap R_1 \cap C_1)(\tfrac{1}{P(R_1)} - \tfrac{1}{P(C_1)}) \\
&\quad + P(B)(P(\bar{C}_1|R_1 \cap B) - P(\bar{R}_1|C_1 \cap B))| \\
&= |P(B \cap R_1 \cap C_1)(\tfrac{1}{P(R_1)} - \tfrac{1}{P(C_1)}) + \\
&\quad P(B)(P(R_1|C_1 \cap B) - P(C_1|R_1 \cap B))| \\
&= |P(B \cap R_1 \cap C_1)(\tfrac{1}{P(R_1)} - \tfrac{1}{P(C_1)}) + \\
&\quad P(B)P(B \cap R_1 \cap C_1)(\tfrac{1}{P(C_1 \cap B)} - \tfrac{1}{P(R_1 \cap B)})| \\
&\leq P(B \cap R_1 \cap C_1)\left[|\tfrac{1}{P(R_1)} - \tfrac{1}{P(C_1)}| + \right. \\
&\quad \left. P(B)|\tfrac{1}{P(C_1 \cap B)} - \tfrac{1}{P(R_1 \cap B)}|\right].
\end{aligned}$$

□

The bound gets arbitrarily tight as the sample size increases because the radius of NN ball converges to zero as the sample size increases. Also the smaller the probability the ball intersects the region the better the bound. The optimal Bayes decision rule minimizes the differences between $P(C_i)$ and $P(R_i)$ which means that this bound for a typical pair R_i, C_i is on average as tight as possible when the optimal Bayes decision rule is used to cluster

5 Nearest-Neighbor Behavior within Gaussian Mixtures

Under what conditions should DBIN perform well? Assuming a mixture of Gaussians, we define stability conditions that ensure that NN queries are meaningful and that the NN of a point is in the same cluster as the point itself. If the NN is always in the same cluster, then DBIN need only scan one cluster. Note that these results are asymptotic in the sense that they hold as the attribute dimensionality increases to infinity.

Our results extend those of Beyer et al [6]. They proved the disturbing result that NN neighbor queries are meaningless in high dimensions under commonly used assumptions pertaining to data distributions. The concern raised in their work is that the ratio of nearest and farthest neighbor distance converges in probability to 1 as the dimensionality increases. We show that under reasonable assumptions this negative result is not applicable. We first present the notation and results in [6].

Definition 5.1 (Notation)
d is dimensionality of a Euclidean space.
n is number of samples taken.
$F_{1,i}, F_{2,i}, \ldots, \; i = 1, \ldots, n$ are sequences of data distributions.
$Q_1, Q_2, \ldots$ is a sequence of query distributions.
For any d, $x_{d,1}, x_{d,2}, \ldots, x_{d,n}$ are n independent data points per d such that $x_{d,i}$ is generated from $F_{d,i}$.
q_d is a query point generated from Q_d.
We use the squared Euclidean distance $||x_{d,i} - q_d||^2$, however, our results generalize to other distance measures as well.
$DMIN_d = \min_i\{||x_{d,i} - q_d||^2 \mid 1 \leq i \leq n\}$.
$DMAX_d = \max_i\{||x_{d,i} - q_d||^2 \mid 1 \leq i \leq n\}$.

Theorem 5.1 (Meaningless NN Queries [6]) *Let $x_{d,i}$ and q_d be random variables with pdf $F_{d,i}$ and Q_d, respectively. If*

$$\lim_{d\to\infty} \frac{var(||x_{d,i} - q_d||^2)}{E[||x_{d,i} - q_d||^2]^2} = 0, \qquad (6)$$

then for every $\epsilon > 0$,
$\lim_{d\to\infty} P[DMAX_d \leq (1+\epsilon)DMIN_d] = 1$.

In a mixture of Gaussian distributions, each data point or query is generated by a single Gaussian distribution. We can think of a random set of points generated by such a mixture model as being clustered by the Gaussian distribution that generated them. Theorem 5.1 applies in particular to data generated by a single Gaussian distribution and so it shows that the distance between arbitrary two points in the same cluster approach the mean *within cluster* distance as the dimensionality d increases. We say that the *within cluster* distance is **unstable** because roughly every point in a cluster is the same distance apart. Specifically,

Corollary 5.1 (*Within Cluster* Dist. Converge) *If* $\lim_{d\to\infty} \frac{var(||x_{d,i} - q_d||^2)}{E(||x_{d,i} - q_d||^2)} = 0$, *then* $\frac{||x_{d,i} - q_d||^2}{E(||x_{d,i} - q_d||^2)} \to_p 1$.

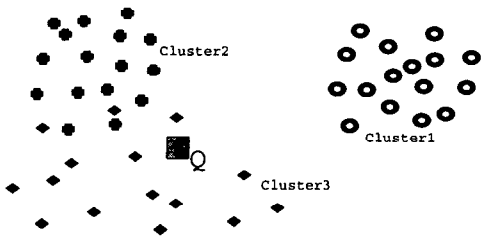

Figure 3: Unstable Query Example

Similarly, the distance between any two points from two distinct clusters approaches the mean *between cluster* distance.

Corollary 5.2 (*Between Cluster* Dist. Converge) *Let $x_{d,i}$, $x_{d,j}$, and q_d be random variables with pdf $F_{d,i}$, $F_{d,j}$, and Q_d, respectively. If*

$$\lim_{d\to\infty} \frac{var(||x_{d,j} - q_d||^2)}{E(||x_{d,i} - q_d||^2)^2} = 0$$

and

$$\lim_{d\to\infty} \frac{E(||x_{d,j} - q_d||^2)}{E(||x_{d,i} - q_d||^2)} = \Delta,$$

then $\frac{||x_{d,j} - q_d||^2}{E(||x_{d,i} - q_d||^2)} \longrightarrow_p \Delta$.

If for two clusters, the *between cluster* distance dominates the *within cluster* distance, we say the clusters are **stable** with respect to each other. In Figure 3, Clusters 1 and 2 are stable and Clusters 2 and 3 are not stable.

Definition 5.2 (Pairwise Stability) *Let $x_{d,i}$, $x_{d,j}$, and q_d be random variables with pdf $F_{d,i}$, $F_{d,j}$, and Q_d, respectively. If* $\frac{||x_{d,i} - q_d||^2}{E(||x_{d,i} - q_d||^2)} \to_p 1$ *and* $\frac{||x_{d,j} - q_d||^2}{E(||x_{d,i} - q_d||^2)} \to_p \Delta > 1$, *then Clusters i and j are pairwise stable with parameter Δ.*

Theorem 5.2 (Stable Cluster Distances) *If Clusters i and j are pairwise stable with parameter Δ, then for any $\epsilon > 0$,*

$$\lim_{d\to\infty} P(||x_{d,j} - q_d||^2 \geq (\Delta - \epsilon)||x_{d,i} - q_d||^2) = 1$$

Proof. Let $\mu_d = E(||x_{d,i} - q_d||^2)$.
Let $V_d = \frac{||x_{d,i} - q_d||^2}{\mu_d}$ and let $W_d = \frac{||x_{d,j} - q_d||^2}{\mu_d}$. We know $V_d \to_p 1$ and $W_d \to_p \Delta$. Thus $\frac{||x_{d,j} - q_d||^2}{||x_{d,i} - q_d||^2} = \frac{\mu_d||x_{d,j} - q_d||^2}{\mu_d||x_{d,i} - q_d||^2} = \frac{W_d}{V_d} \to_p \Delta$. By definition of convergence in probability, for any $\epsilon > 0$
$\lim_{d\to\infty} P\left[\left|\frac{||x_{d,j} - q_d||^2}{||x_{d,i} - q_d||^2} - \Delta\right| \leq \epsilon\right] = 1$. So
$\lim_{d\to\infty} P\left[\Delta - \epsilon \leq \frac{||x_{d,j} - q_d||^2}{||x_{d,i} - q_d||^2} \leq \Delta + \epsilon\right] = 1$, which

implies $\lim_{d\to\infty} P(\|x_{d,j}-q_d\|^2 \geq (\Delta-\epsilon)\|x_{d,i}-q_d\|^2) = 1$. □

If every cluster is stable with respect to at least one other cluster then NN is well defined in the sense that the nearest and farthest neighbor distances are bounded apart. With probability 1 as d grows, the ratio of the farthest and NN is bigger than some constant greater than 1. For example in Figure 3, Cluster 1 is stable with respect to both Clusters 2 and 3 so NN is well defined.

Theorem 5.3 (NN Well-Defined) *If Cluster i and Cluster j are pairwise stable with parameter Δ, then for any $\epsilon > 0$,*
$\lim_{d\to\infty} P[DMAX_d \geq (\Delta-\epsilon)DMIN_d] = 1$

Proof. By conditions of the theorem $\frac{\|x_{d,i}-q_d\|^2}{E(\|x_{d,i}-q_d\|^2)} \to_p 1$ *and* $\frac{\|x_{d,j}-q_d\|^2}{E(\|x_{d,i}-q_d\|^2)} \to_p \Delta$. By definition of minimum and maximum $\frac{DMIN_d}{E(\|x_{d,i}-q_d\|^2)} \leq \frac{\|x_{d,i}-q_d\|^2}{E(\|x_{d,i}-q_d\|^2)}$ and $\frac{DMAX_d}{E(\|x_{d,i}-q_d\|^2} \geq \frac{\|x_{d,j}-q_d\|^2}{E(\|x_{d,i}-q_d\|^2)}$. This implies $\frac{DMAX_d}{DMIN_d} \geq \frac{\|x_{d,j}-q_d\|^2}{\|x_{d,i}-q_d\|^2}$. We know by the Theorem 5.2 that for any $\epsilon > 0$, $\lim_{p\to\infty} P[\frac{\|x_{d,j}-q_d\|^2}{\|x_{d,i}-q_d\|^2} \geq \Delta-\epsilon] = 1$. From the last two statements, $\lim_{d\to\infty} P[\frac{DMAX_d}{DMIN_d} \geq \frac{\|x_{d,j}-q_d\|^2}{\|x_{d,i}-q_d\|^2} \geq \Delta-\epsilon] = 1$. □

If every cluster is stable with respect to every other cluster then if a point belongs to one cluster, its NN also belongs to that cluster. Therefore if we partition our data by cluster membership then with probability 1, as d grows, our index will only need to visit one cluster to find the NN. With probability one, other clusters can be skipped with no false drops of points.

Theorem 5.4 (Nearest Neighbor in Cluster) *If Cluster i is pairwise stable with every Cluster j, $j = 1,\ldots,n$, $j \neq i$, with parameter $\Delta_{ij} > \Delta > 1$ respectively, then for any point $x_{d,j}$ from cluster $j \neq i$, and any $\epsilon > 0$,*
$\lim_{d\to\infty} P[\|x_{d,j}-q_d\|^2 \geq (\Delta-\epsilon)\|x_{d,i}-q_d\|^2] = 1$.

Proof. By Theorem 5.2 for every $x_{d,j}$ $j \neq i$ and every $\epsilon > 0$,
$\lim_{d\to\infty} P[\|x_{d,j}-q_d\|^2 \geq (\Delta_{i,j}-\epsilon)\|x_{d,i}-q_d\|^2] = 1$.
Since $\Delta_{i,j} > \Delta$, the result follows. □

These results show that if we have a stable mixture of Gaussians where the between cluster distance dominates the within cluster distance, then if we partition by a cluster membership function that assigns all data generated by the same Gaussian to the same partition, the index would work perfectly for NN queries generated by the same distribution. The higher the dimensionality, the better it would work. There is no "curse of dimensionality" in this case. For example in [2], it was shown that clusters generated by a mixture of spherical Gaussian with identical covariance matrices Σ such that the Mahalanobis distance between the cluster centers grows at least as fast as $\sqrt{d}$ then the clusters are stable.

Note that we actually do not know which Gaussian generates a data point. But, as discussed in Section 4, we can use the optimal Bayes decision rule [9] to estimate the Gaussian that generated a point with minimum error. Figure 2 illustrates stable clusters and the corresponding partitioning of the data space into regions.

6 Computational Results

Our initial experimental goals were to confirm the validity of the theoretical results, to determine the accuracy of the probability estimates, and to establish that DBIN would be practical on real-world data with unknown distribution. We assumed that if a cluster is visited that the entire cluster is scanned. We did not address paging of data within a cluster. We experimented with both synthetic and real-world databases. The purpose of synthetic databases is to study the behavior of DBIN in well-understood situations. The experiments on the real data sets verify that our assumptions are not too restrictive and apply in natural situations.

6.1 Synthetic Databases

First we verified the stability theory introduced in Section 5 by applying DBIN to both stable and unstable mixtures of Gaussian data. We used synthetic data sets drawn from a mixture of ten Gaussians. First, we used *stable* clusters with a known generating model, then *unstable* clusters with a known generating model, and finally, *stable* clusters with an unknown generating model, i.e. a situation in which we had to estimate the density (and do the clustering).

Stable Case, Known Probability Density: We generated a mixture of ten Gaussians in d dimensions with distance τ_d between the means of the distribution. Each Gaussian had covariance matrix $\sigma^2 I$ where I is the identity matrix and $\sigma^2 = .01$. As discussed in Section 5 and [2], if $\tau_d > \sigma\sqrt{d}$, the clusters are stable. In order to fix the distance between the means to be τ_d apart, we set the i^{th} mean $\mu_i = \sqrt{\tau_d/2}e_i$ where e_i is a vector of zeros with a one in the i^{th} component. The size of the database was fixed and we generated $500000/d$ points for $d < 100$ dimensions and $1000000/d$ for $d \geq 100$. DBIN was used to find the 2 NN for 250 query points randomly selected from the database. To remove any variation due to inaccuracies in clustering, we first used the *true generating density* as input to DBIN. Hence, this represents an extreme for a best-case scenario.

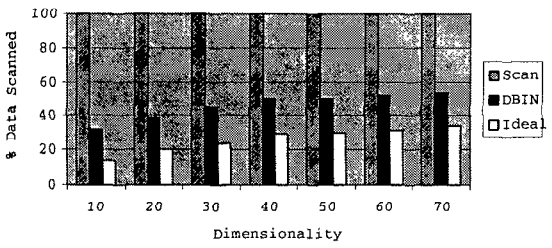

Figure 4: Fraction scanned: unstable case $\tau_d = 2\sigma$.

The stable case, with $\tau_d = \sigma d$ was evaluated on 10, 20, 30, 40, 50, 60, 70, 100, 200, and 500 dimensions. In every case the 2 NN were found by examining only one cluster. In addition DBIN correctly determined that no additional clusters needed to be searched. Since each cluster contained ten percent of the data, each query required a scan of ten percent of the data. Similar experiments for 10 NN produced the same conclusion. In the best of worlds when the clusters are stable and the model is known, DBIN works perfectly. As the theory in Section 5 predicted, the curse of dimensionality vanishes when the distributions are stable.

Unstable Case, Known Probability Density For unstable data, we fixed $\tau_d = 2\sigma$. With this distribution, any problem with dimensionality over 4 is unstable. The amount of overlap of the Gaussians is growing exponentially with d. This is a worst-case scenario as there is no separation between the clusters and the asymptotic stability theory does not apply.

To evaluate how well DBIN estimated when to stop scanning, we compared it with the Ideal approach that scans the clusters in the prioritized order predicted by DBIN and stops scanning additional clusters once the NN is found. DBIN stops scanning when the NN is found with high confidence. Since this case is very unstable, we would expect any algorithm to require a scan of much of the database. The percentages of data scanned by a full scan, DBIN, and the Ideal approach are given in Figure 4. Table 1 provides the percentage accuracy of DBIN on this unstable data. An estimate is considered correct if the estimated k-nearest neighbor distance does not exceed the actual k-NN distance. The percentage of the data scanned increased gradually with dimensionality. The Ideal algorithm scanned less data. This difference between the Ideal algorithm and DBIN indicates that the DBIN probability estimate is conservative. In many cases DBIN slightly overestimates the probability of a NN residing in a cluster so there may be an opportunity for tightening the probability estimate used.

Dim.	10	20	30	40	50	60	70
Acc.	98.8	96.4	93.6	93.2	94.8	96.4	92.4

Table 1: Percentage accuracy of DBIN on unstable data

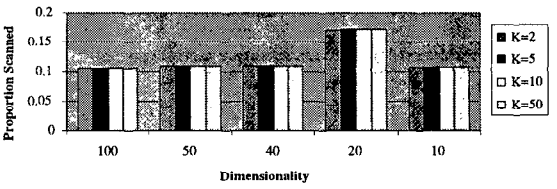

Figure 5: Scan fraction: unknown stable model.

Random Stable Clusters, Unknown Probability Density, Varying K: In the next set of experiments we applied the full DBIN approach to generated data. We utilized some data and corresponding clusters generated originally for [8]. The data were generated from 10 Gaussians with means independently and identically distributed in each dimension from a uniform distribution on $[-5, 5]$. Each diagonal element of covariance matrices was generated from a uniform distribution on $[0.7, 1.5]$. Hence this data is well-separated and should be stable, but is not at an extreme of stability. We used clusters generated by the EM algorithm without using knowledge of the known distribution except the number of clusters. Results on problems with 10 to 100 dimensions, with K (number of nearest neighbors to find) varying from 2 to 50, are given in Figure 5. Note that DBIN *made no errors at all in finding the nearest neighbors for all values of K.* Hence there is no point in plotting those results.

It turns out this distribution is stable in practice. This is no surprise since in high dimensions the distance between the uniformly generated means approaches the expected value which is proportional to $\sqrt{d}$. The 20-dimensional result is an artifact of the fact that our EM clustering algorithm happened to find a non-optimal solution. hence more than 1 cluster is scanned on average. Note that because we limited ourselves to 10 clusters, 0.1 is the minimum possible scan fraction here. hence we are near the optimal. Larger number of clusters will result in less data scanning.

6.2 Real Database Results

We experimented on real-world datasets. Overall the results are promising, but we found some additional

K	2	5	10	50
Accuracy	99.0%	100%	100%	97.1%
Fraction Scanned	12.5%	13.9%	14.2%	16.6%

Table 2: Accuracy and Fraction Scanned Results for 11-dimensional Census Data

K	2	5	10	50
Accuracy	94.7%	90.0%	85.0%	78.6%
Disc. Accuracy	97.3%	96.4%	95.6%	95.2%
Fraction Scanned	16.8%	17.2%	17.4%	17.8%

Table 3: Accuracy and Fraction Scanned Results for 29-dimensional Astronomy Data

issues to resolve. In the current implementation of DBIN, we did not address categorical data fields. Some numerical stability problems occur with sparse datasets that result in zero variance clusters. The algorithm still performed as we expected. We need to extend the theory to the case where data is not truly continuous. Since our primary goal in this paper is to introduce the basic idea, and demonstrate that it is possible to index higher dimensional data, the results presented here are not optimized (i.e. we did not build the best possible clustering model). Nor did we optimize the algorithm to gracefully deal with small variances or discrete data masquerading as continuous values. The goal is to demonstrate that in principle we can derive useful indexing information from statistical structure. All results are averaged over 1000 query points drawn randomly from the data.

U.S. Census Data: publicly available U.S. Census Bureau data set consisting of 300K records, and we selected the 11 dimensions that appeared to be numeric. These were fields such as ages, incomes, taxes, etc. Table 2 shows the results for varying K. The model we constructed had 100 clusters, so in principle the minimum possible scan fraction is around 0.01.

Astronomy Data: This data set consists of 650K records of measurements on sky objects. For each object there are 29 measurements and we clustered the data into 10 clusters. Again, this was by no means an optimized clustering, and a larger number of clusters would imply a lower possible minimum fraction of data to scan. Results are shown in Table 3.

Here we introduce the notion of *Discounted Accuracy*. Our accuracy measure counts a full error if for K=50, we found 49 out of 50 true nearest neighbors. Discounted accuracy gives an error of $\frac{1}{50}$ in this case and gives more insight into what is happening.

Investor Data Sets: The last data set we evaluated is derived from a financial database consisting on historical stock prices, market valuations and so forth. In all, we used 34 dimensions. The data consisted of 834K records. We tried two experiments, one with 20 clusters and one with 47 clusters. Again these clusters were not optimized. The results for K=2 gave an error of 0.01 with 45% of the data scanned in case of 20 clusters. With 47 clusters the accuracy was 100% with 42% of the data scanned. As we increased K, performance deteriorated quickly. This, we believe is due to problems in fitting a model to data. We need to optimize the model. Also, numerical instability and the problem of zero variances showed up in spades in this data set. However, the results at least show than in principle the structure is useful (i.e. we did not have to scan 100% of the data).

7 Conclusions and Future Work

In the discussion so far, we have not addressed the issue of determining the number of clusters required. From the perspective of indexing, the more clusters we have, the less data we are likely to have to scan. However, recall that determining which cluster to scan next requires a lookup into a table of clusters. If there are too many clusters, this lookup becomes too expensive. Consider the extreme case where each point in the database is its own cluster. In this case, each cluster identifies the result directly but the lookup into the cluster table is as expensive as scanning the entire database.

Generally, we use a small number of clusters (5 to 100). The cost of computing probabilities from the model in this case is fairly negligible. Note that with 5 clusters, assuming well-separated clusters, we can expect an 80% savings in scan cost. So not many clusters are needed. There are also clustering algorithms that choose K as part of the clustering session. In this case, we let the algorithm choose the most appropriate K to fit the data, so long as K does not get too large. The trade-off between cluster lookup cost and data scan cost can be optimized on a per application basis.

We presented DBIN, a scheme that exploits structure in data to derive a multi-dimensional index for NN queries. We also developed a probabilistic theory to support the method. We analyzed the NN query problem under the assumption that data is modeled by a mixture of Gaussians. We defined the notion of cluster stability which gives us the means to assess if a data set is amenable to our method. We derived two key results. The first is that nearest neighbor queries can be meaningful in high dimensions. The second is that

it is possible to exploit the statistical structure of the data to construct a multi-dimensional indexing scheme. We derived a criterion for estimating the likelihood of payoff of scanning further, enabling a stopping criterion. The result is a new indexing algorithm based on density estimation. It provides a confidence level in the answer found so far. Because it is based on modeling the data content, DBIN provides sufficient information regarding the suitability of its indexing scheme to a given database. This can enable an optimizer to decide whether to invoke DBIN's indexing structure or opt for the sequential scan when appropriate.

We showed empirically that the proposed algorithm works when the data is stable. We used synthetic data to verify this and to study behavior when data is dramatically unstable. We also demonstrated for several real-world databases that real data is not uniformly distributed or concentrated in a single cluster. Our method showed significant improvement over a sequential scan.

DBIN has many properties that are desirable from a data mining perspective. It is an *anytime algorithm*: it can provide the "best answer so far" whenever queried. The user can then stop the scan if the answer is satisfactory. It also provides a confidence level on the answer found so far. It can provide accurate estimates of how much work is left, allowing the user application to perform a cost-benefit analysis. It can utilize much of the existing SQL backend assuming clustered indices are supported. It leverages new work on scalable clustering techniques. Finally, we have found empirically that a full data scan is almost always avoided.

The results indicate that visiting data in order of nearest clusters first is sound. In fact, we have found that in many of the instances when our stopping criterion does not cause us to stop after the first cluster, the actual NN turned out to be in the first scanned cluster after all. This suggests that it is possible to derive improved bounds for stopping earlier.

Additional steps are needed to make DBIN into a practical system. A full comparison with existing indexing methods is needed to assess the overall advantages of DBIN and whether performance increases are due to improved clustering or approximate NN processing. One challenge associated with DBIN is that it must sometimes calculate probabilities that are very close to zero without underflowing. In these cases, we run into numerical stability issues. Numbers get small due to high dimensionality and to large numbers of records. While high-dimensionality is handled by the theory we develop, in practice we are left with computing vanishingly small probabilities. A more robust scheme for computing these is desirable. This paper validates the basic framework and theory rather than provide complete details for implementation. We would like to generalize the theory to models other than Gaussians. Also, an evaluation of an implementation tied to a SQL backend that utilizes native indexing structures would be useful to demonstrate practical feasibility.

Acknowledgments

This work was done while Kristin Bennett and Dan Geiger were visiting researchers at Microsoft Research.

References

[1] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R* tree: an efficient and robust access method for points and rectangles. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 323–331, 1990.

[2] K. Bennett, U. Fayyad, and D. Geiger. Density based indexing for nearest-neighbor queries. Technical Report MSR-TR-98-58, Microsoft Research, Redmond, WA, 1998.

[3] S. Berchtold, D. Keim B. Ertl, and H.-P. Kriegel. Fast nearest neighbor search in high-dimensional space. In *Proc. 14th Int. Conf. on Data Engineering, Orlando, Fl*, 1998.

[4] S. Berchtold, C. Bohm, and H.-P. Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. In *Proc. of ACM SIGMOD, Seattle, WA*, pages 142–153, 1998.

[5] S. Berchtold, D. Keim, and H-P. Kriegel. The X-tree: An index structure for high-dimensional data. In *Proc. of the 22nd Conf. on Very Large Databases, Bombay, India*, pages 28–39, 1996.

[6] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? In *Proc. of the 7th Int'l Conf. on Database Theory (ICDT), Jerusalem, Israel,1999*, 1998. To appear.

[7] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining*, pages 9 – 15, 1998.

[8] P. Bradley, U. Fayyad, and C. Reina. Scaling EM (expectation maximization) clustering to large databases. Technical Report MSR-TR-98-35, Microsoft Research, Redmond, WA, USA, 1998.

[9] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973.

[10] C. Faloutsos and K.-I Lin. Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceeding of ACM SIGMOD Int'l Conf. on Management of Data, San Jose*, pages 231–262, 1995.

[11] R. Farebrother. Algorithm as 204: The distribution of a positive linear combination of chi-square random variables. *Applied Statistics*, 32(3):332–337, 1983.

[12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions. Submitted for publication, 1998.

[13] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC'98. Proceedings of the Thirteenth annual ACM symposium on Theory of computing*, pages 604 – 613, 1998.

[14] N. Katayama and S. Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data, Tucson, Arizona*, 1997.

[15] A. Mathai and S. Provost. *Quadratic Forms in Random Variables.* Marcel Dekker, Inc, New York, 1992.

[16] J. T. Robinson. The K-D-B tree: A search structure for large multidimensional indexes. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data, Ann Arbor, MI*, pages 10–18, 1981.

[17] D. W. Scott. *Density Estimation.* Wiley, New York, 1992.

[18] T. Seidl and H.-P. Kriegel. Optimal multi-step k-nearest neighbor search. In *Proc. of ACM SIGMOD, Seattle, WA*, pages 154–165, 1998.

[19] K. Shim, R. Srikant, and R. Agrawal. High-dimensional similarity joins. In *13th Int'l Conf. on Data Engineering*, 1997.

[20] D. White and R. Jain. Similarity indexing with the SS-tree. In *Proc. of the 12th Int'l Conf. on Data Engineering, New Orleans*, pages 516–523, 1996.