Scalable Algorithms for Distribution Search

Yasuko Matsubara
Kyoto University
y.matsubara@db.soc.i.kyoto-u.ac.jp

Yasushi Sakurai
NTT Communication Science Labs
yasushi.sakurai@acm.org

Masatoshi Yoshikawa
Kyoto University
yoshikawa@i.kyoto-u.ac.jp

Abstract

Distribution data naturally arise in countless domains, such as meteorology, biology, geology, industry and economics. However, relatively little attention has been paid to data mining for large distribution sets. Given n distributions of multiple categories and a query distribution Q, we want to find similar clouds (i.e., distributions), to discover patterns, rules and outlier clouds. For example, consider the numerical case of sales of items, where, for each item sold, we record the unit price and quantity; then, each customer is represented as a distribution of 2-d points (one for each item he/she bought). We want to find similar users, e.g., for market segmentation, anomaly/fraud detection. We propose to address this problem and present D-Search, *which includes fast and effective algorithms for similarity search in large distribution datasets. Our main contributions are (1) approximate KL divergence, which can speed up cloud-similarity computations, (2) multi-step sequential scan, which efficiently prunes a significant number of search candidates and leads to a direct reduction in the search cost. We also introduce an extended version of* D-Search *: (3) time-series distribution mining, which finds similar subsequences in time-series distribution datasets. Extensive experiments on real multi-dimensional datasets show that our solution achieves up to* 2,300 faster *wall-clock time over the naive implementation while it does not sacrifice accuracy.*

1 Introduction

Distribution data naturally arise in countless domains, such as meteorology, biology, geology, industry and economics. Although the datasets generated by the corresponding applications continue to grow in size, a common demand is to discover patterns, rules and outliers. However, relatively little attention has been paid to data mining for large distribution sets. Here we focus on a less-studied problem, namely on "distribution search". Given n distributions of multiple categories and a query distribution Q, we want to find similar clouds (i.e., distributions), to meet the above demand. To solve this problem, we present *D-Search*, which includes fast and effective algorithms for similarity search for large distribution sets.

We will illustrate the main intuition and motivation with a real example. Figure 1 shows three distributions corresponding to three motions. Specifically, they are the scatter plots of left and right foot kinetic energy values. Given a query motion, shown on the left, we would like to discover similar objects in large distribution datasets. Figure 1 shows the output of our approach, which successfully identify similar distributions. For example, *D-Search* detects *Distribution* #1 similar to the query distribution (in fact, they both correspond to "running" motions). In contrast, *Distribution* #2 is not be found as a similar object (in fact, it corresponds to a "squatting" motion).

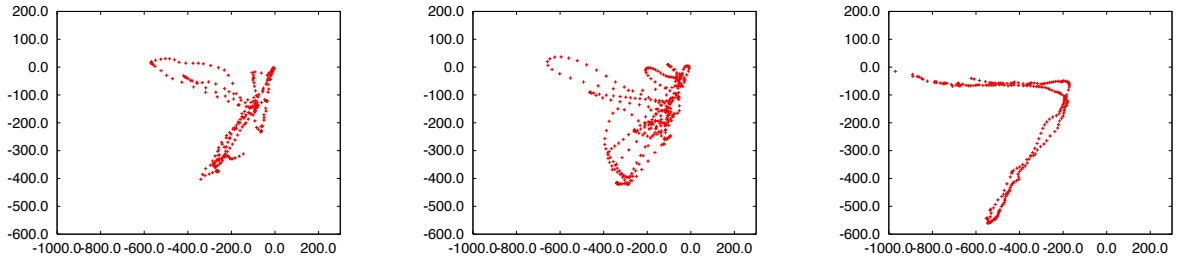
In this paper, we propose efficient algorithms called *D-Search*, which can find similar distributions in large distribution datasets. We mainly focus on similarity search for numerical distribution data to describe our approach. However, our solution, *D-Search* can handle categorical distributions as well as numerical ones. Our upcoming algorithms are completely independent of such choice.

1.1 Example domains and applications

There are many distribution search applications. In this section, we briefly describe application domains and provide some illustrative, intuitive examples of the usefulness of *D-Search*.

- *Multimedia* : Multimodal data mining in a multimedia database is a challenging topic in data mining research [9, 3, 12, 7]. Multimedia data may consist of data in different modalities, such as digital images, audio, video, and text data. For example, consider motion capture datasets, which contain a list of numerical attributes of kinetic energy values. In this case, every motion can be represented as a cloud of hundreds of frames, with each frame being a d-dimensional point. For this collection of clouds, we can find similar motions without using annotations or other meta-data.

- *Medical data analysis* : The extraction of meaningful information from large medical datasets is the central theme in many medical research problems [16]. For example, mental task classification using electroencephalograms (EEG) is an approach to understanding human brain functions. EEG signals are weak voltages resulting from the spatial summation of electrical potentials in the brain cortex, which can easily be de-



Query (running) *Distribution* #1 (running) *Distribution* #2 (squatting)

Figure 1. Three distributions from *MoCap* data: they all show scatter plots of left and right foot kinetic energy values, for different motions. The query distribution and *Distribution #1* "look" more similar, while *Distribution #2* "looks" different.

tected by electrodes suitably placed on the scalp surface [4, 19].

- *Web service* : There are numerous, fascinating applications for *Web service* mining. Let us assume the web services such as an *Ondemand TV* service, which records the viewing of *Ondemand TV* on a daily basis of all users (e.g., the genre of a TV program, the time the user spent on the service). Discovering clusters and outliers in such data (*which groups or communities of users are associated with each other?*) would help in tasks such as service design and content targeting.

- *E-commerce* : Consider an e-commerce setting, where we wish to find customers according to their purchasing habits. Suppose that for every sale we can obtain the time the customer spent browsing, the number of items bought, their genres and sales price. Thus, each customer is a cloud of 4-d points (one for each purchase). The e-store would like to classify these clouds, to do market segmentation, rule discovery (*is it true that the highest volume customers spend more time on our web site?*) and spot anomalies (e.g., identity theft).

1.2 Contributions

We introduce an efficient algorithms called *D-Search*, which can find similar distributions in large distribution datasets. The contributions are the following: (a) we examine the time and space complexity of our solutions and compare them with the complexity of the naive solution. Given n distributions of a m-bucketized histogram and a query distribution, our algorithms require only $O(n)$ to compute KL divergence, instead of $O(mn)$ the naive method required, and lead to a dramatically reduction in the search cost. (b) Extensive experiments on real multi-dimensional datasets shows that our method is significantly faster than the naive method, while it does not sacrifice accuracy. We also introduce an extended version of *D-Search* : (c) time-series distribution mining, which finds similar subsequences in time-series distribution datasets.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 introduces preliminary concepts, describes the proposed method, and identifies the main tasks of *distribution* search. We then explain the techniques and constraints we use to realize efficient KL divergence calculations. We also describe an extended version of *D-Search*, namely, time-series distribution mining. Section 4 introduces some of the applications for which our method proves useful, and evaluates our algorithms by conducting extensive experiments. Finally, Section 5 concludes the paper.

2 Related Work

Related work falls broadly into two categories. The first category includes work on similarity functions between distributions. The other category includes probabilistic queries.

2.1 Comparing two distributions

There are several statistical methods [15] to decide whether two distributions are the same (Chi-square, Kolmogorov-Smirnoff). However, they do not give a score; only a yes/no answer; and some of them cannot be easily generalized for higher dimensions.

Functionals that return a score are motivated from image processing and image comparisons: the so-called *earth-moving distance* [17] between two images is the minimum energy (mass times distance) to transform one image into the other, where mass is the gray-scale value of each pixel. For two clouds of points P and Q (= black-and-white images), there are several measures of their distance/similarity: one alternative is the distance of the closest pair (min-min distance); another is the Hausdorff distance (max-min - the maximum distance of a set P, to the nearest point in the set Q); another would be the average of all pairwise distances among P-Q pairs. Finally, tri-plots [21] can find patterns across two large, multi-dimensional sets of points, although they cannot assign a distance score. The

Table 1. Symbols and definitions.

Symbol	Definition
n	number of distributions
m	number of buckets
P, Q	two histograms of numerical and/or categorical distributions
$\hat{P}$	histogram of the logarithm of P
p_i, $\hat{p}_i$	i-th bucket of P, $\hat{P}$
S_p	SVD coefficients of P
$\hat{S}_p$	SVD coefficients of $\hat{P}$
sp_i	i-th coefficient of S_p
$D(P, Q)$	symmetric KL divergence of P and Q
$d_c(P, Q)$	lower bounding KL divergence of P and Q with c histogram buckets
$d'_c(P, Q)$	approximate KL divergence of P and Q with c SVD coefficients

most suitable idea for our setting is the Kullback-Leibler (KL) divergence (see Equation (1)), which gives a notion of distance between two distributions. The KL divergence is commonly used in several fields to measure the distance between two PDFs (probability density functions, as in, e.g., information theory [22], pattern recognition [18, 10]).

2.2 Probabilistic queries

A remotely related problem is the problem of probabilistic queries. Cheng et al. [5] classifies and evaluates probabilistic queries over uncertain data based on models of uncertainty. An indexing method for regular probabilistic queries is then proposed in [6]. In [20] Tao et al. presents an algorithm for indexing uncertain data for any type of PDFs. Distributions for our work can be expressed by PDFs as well as histograms. However, the main difference between our study and [5] is that our work focuses on comparing differences between distributions, while Cheng et al.s' work focuses on locating areas in distributions that satisfy a given threshold.

Distribution search and mining are problems that, to our knowledge, have not been addressed. The distance functions among clouds that we mentioned earlier either expect a continuous distribution (like a probability density function), and/or are too expensive to compute.

3 Proposed Method

We now present our distribution search algorithms. In this section, we introduce some background concepts, define the problem of distribution search, and then propose algorithms for solving it. We also introduce time-series distribution search, as an extended version of *D-Search*.

3.1 Preliminaries

Given two distributions P and Q, there are several measures of their distance/similarity, as we described in the literature survey section. However, the above distances suffer from one or more of the following drawbacks: they are either too fragile (like the min-min distance); and/or they do not take all the available data points into account; and/or they are too expensive to compute. Thus we propose to use information theory as the basis, and specifically the Kullback-Leibler (KL) divergence.

Let us assume for a moment that the two clouds of points P and Q consist of samples from two (continuous) probability density functions $\mathcal{P}$ and $\mathcal{Q}$, respectively. If we knew $\mathcal{P}$ and $\mathcal{Q}$ we could apply the continuous version of the KL divergence; however, we do not. Thus we propose to bucketize all our distributions using a grid with m grid-cells, and employing the discrete version of the KL divergence, defined as follows:

$$D_{KL}(P, Q) = \sum_{i=1}^{m} p_i \cdot \log\left(\frac{p_i}{q_i}\right) \quad (1)$$

where p_i, q_i are the i-th buckets of distributions P and Q, respectively. That is, $\sum_{i=1}^{m} p_i = \sum_{i=1}^{m} q_i = 1$.

The above definition is asymmetric, and thus we propose to use the symmetric KL divergence D_{SKL}:

$$\begin{aligned} D_{SKL}(P, Q) &= D_{KL}(P, Q) + D_{KL}(Q, P) \\ &= \sum_{i=1}^{m} (p_i - q_i) \cdot \log\left(\frac{p_i}{q_i}\right). \end{aligned} \quad (2)$$

In the rest of the paper, we shall simply denote $D_{SKL}(P, Q)$ as $D(P, Q)$.

There is a subtle point we need to address. The KL divergence expects non-zero values, however, histogram buckets corresponding to sparse area in multi-dimensional space may take a zero value. To avoid this, we introduce the Laplace estimator [13, 11]:

$$p_i = \frac{p'_i + 1}{|P'| + m} \cdot |P'|. \quad (3)$$

where p'_i is the original histogram value ($i = 1, \ldots, m$) and p_i is the estimate of p'_i. $|P'|$ is the total number of points (i.e., $|P'| = \sum_{i=1}^{m} p'_i$).

Another solution is that we could simply treat empty cells as if they had "minimum occupancy" of ϵ. The value for "minimum occupancy" should be $\epsilon << 1/|P'|$. We chose the former since it provides better search accuracy, although our algorithms are completely independent of such choices.

3.2 D-Search

The first problem we want to solve is as follows:

Problem 1 (Distribution search) *Given n distributions of an m-bucketized histogram and a query distribution Q, find the top k distributions that minimize the KL divergence.*

This involves the following sub-problems, which we address in each of the upcoming subsections. (a) How can we represent a distribution of histogram compactly, and accelerate distance calculations? (b) How can we prune a significant number of search candidates and achieve a direct reduction in the search cost? (c) What is the space and time complexity of our method?

3.2.1 Lower bounding KL divergence

We described how to measure the distance between distributions. The first question is how to store the distribution information, in order to minimize space consumption and response time. Recall that for each m-bucketized distribution P, we could keep the fraction of p_i that fall into the i-th bucket.

The naive solution is exactly to maintain an m-bucketized histogram for each distribution, and to use such histograms to compute the necessary KL divergences, and eventually to run the required mining algorithm (e.g., the nearest neighbor search).

However, this may require too much space, especially for higher dimensions. One solution would be to use the top c most populated buckets, in the spirit of 'high end histograms'. The reason against it is that we may ignore some sparse-populated bins, whose logarithm would be important for the KL divergence.

For the definition, we compute the KL divergence with c histogram values, that is, we compute $(p_i - q_i) \cdot \log(p_i/q_i)$ if we select either p_i or q_i, otherwise, we can simply ignore these values since they are very close to zero. Consider that the sequence describing the positions of the top c values of P and Q is denoted as $\mathcal{I}_{pq}$. We then obtain the lower bounding KL divergence of P and Q:

$$d_c(P,Q) = \sum_{i \in \mathcal{I}_{pq}} (p_i - q_i) \cdot \log\left(\frac{p_i}{q_i}\right). \quad (4)$$

Lemma 1 *For any distributions, the following inequality holds.*

$$D(P,Q) \geq d_c(P,Q). \quad (5)$$

Proof: From the definition,

$$D(P,Q) = \sum_{i=1}^{m} (p_i - q_i)(\log p_i - \log q_i).$$

Since $\forall i, (p_i - q_i)(\log p_i - \log q_i) \geq 0$, for any c value $(1 \leq c \leq m)$, we have

$$D(P,Q) = d_m(P,Q) \geq d_c(P,Q), \quad (6)$$

which completes the proof. □

Algorithm 1 D-Search(Q, k)

```
/* N is the sorted nearest neighbor list */
initialize N
for i := 1 to h do
   N = MultiStepScan(N, Q, k, c_i)
end for
for all P ∈ database do
   compute D(P, Q)
   if D(P, Q) ≤ D_cb then
      add P to N and update D_cb
   end if
end for
return N
```

Algorithm 2 MultiStepScan($\mathcal{N}, Q, k, c$)

```
/* N_app is the sorted nearest neighbor list */
initialize N_app
/* compute approximate KL divergence */
for all P ∈ database do
   compute d_c(P, Q)
   if d_c(P, Q) ≤ d_cb then
      add P to N_app and update d_cb
   end if
end for
/* compute exact KL divergence */
for all P ∈ N_app do
   compute D(P, Q)
   if D(P, Q) ≤ D_cb then
      add P to N and update D_cb
   end if
end for
/* prune the search candidates */
for all P ∈ database do
   if d_c(P, Q) > D_cb then
      remove P from database
   end if
end for
return N
```

3.2.2 Multi-Step Sequential Scan

Instead of operating on lower bounding KL divergence with c buckets of a single computation, we propose to use multiple computations, trying to balance to a trade-off between accuracy and comparison speed. As the number of buckets c increases, the lower bounding KL divergence becomes tighter, but the computation cost also grows. Accordingly, we gradually increase the number of buckets, and thus improve the accuracy of the approximate distance, during the course of query processing.

Algorithm 1 shows our proposed method, which uses the lower bounding KL divergence. In this algorithm $\mathcal{N}$ shows the k-nearest neighbor list, and D_{cb} shows the exact KL divergence of the current k-th nearest neighbor (i.e., D_{cb} is the current best). As the multi-step scan, the algorithm uses breadth-first traversal, and it prunes unlikely distributions at each step, as follows:

1. We first obtain the set of k-nearest neighbor candidates ($\mathcal{N}_{app}$) based on the approximate KL divergence (i.e., the lower bounding KL divergence) with the top c histogram buckets.

2. We then compute the exact KL divergence between candidate distributions ($\mathcal{N}_{app}$) and the query distribution. When we find a distribution whose exact KL divergence is smaller than D_{cb} we update the candidate ($\mathcal{N}$).

3. For all distributions, if the lower bounding KL divergence is larger than D_{cb}, we exclude the distribution since it cannot be one of the k-nearest neighbors.

We compute h steps that form an arithmetic progression: $c = \{c_1, 2c_1, 3c_1, ..., h \cdot c_1\}$, or more generally, for steps of $c_i := i \cdot c_1$ for $i = 1, 2, ..., h$.

The search algorithm gradually enhances the accuracy of the lower bounding KL divergence and prunes dissimilar distributions to reduce the computation cost of KL divergence. Finally, we compute the exact KL divergences between distributions, which are not pruned in any steps, and the query distribution.

Lemma 2 *For any distributions,* D-Search *guarantees exactness when finding distributions that minimize the KL divergence for the given query distribution.*

Proof:
From Lemma 1, we obtain $D(P,Q) \geq d_c(P,Q)$ for any granularity, for any distribution. For $\mathcal{N}$, $D_{cb} \geq d_c(P,Q)$ holds. In the search processing, since $D_{cb} \leq D(P,Q)$, the lower bounding KL divergence of $\mathcal{N}$ is less than D_{cb}. The algorithm discards P if (and only if) $d_c(P,Q) > D_{cb}$. Therefore, the final k-nearest neighbors in $\mathcal{N}$ cannot be pruned during the search processing. □

Although we described only a search algorithm for k-nearest neighbor queries, *D-Search* can be applied to range queries. It utilizes the current k-th nearest neighbor distance D_{cb} for k-nearest neighbor queries, and the search range is used to handle range queries.

3.3 Enhanced D-Search

We described the basic version of *D-Search* in the previous subsection, which guarantees the exactness for distribution search while the algorithm efficiently finds distributions that minimize the KL divergence. The question is what can we do in the highly likely case that the users need more efficient solution while they practically require high accuracy, not a theoretical guarantee. As the enhanced version of *D-Search*, we propose to compress the histograms using SVD (Singular Value Decomposition), and *then* keeping some appropriate coefficients. As we show later, this decision significantly improves both space as well as response time, with negligible effects on the mining results. The only tricky aspect is that if we just keep the top c SVD coefficients, we might not get good accuracy for the KL divergence. This led us to the design of our method that we describe next. The main idea behind *SVD* is to keep the top c SVD coefficients for the histogram $P(m) = (p_1, \ldots p_m)$, *as well as* the top c coefficients for the histogram of the *logarithms* $(\log p_1, \ldots \log p_m)$. We elaborate next.

Let $\hat{P} = (\hat{p}_1, \ldots, \hat{p}_m)$ be the histogram of the logarithms of $P = (p_1, \ldots, p_m)$, i.e., $\hat{p}_i = \log p_i$. Let S_p and $\hat{S}_p$ be the SVD coefficients of P and $\hat{P}$, respectively. We present our solution using S_p and $\hat{S}_p$.

Proposed Solution: We represent each distribution as a single vector; we compute S_p and $\hat{S}_p$ from P and $\hat{P}$ for each distribution, and then we compute the necessary KL divergences from the SVD coefficients. Finally, we apply a search algorithm (e.g., the nearest neighbor search) to the SVD coefficients.

The cornerstone of our method is Theorem 1, which effectively states that we can compute the symmetric KL divergence using the appropriate SVD coefficients.

Theorem 1 *Let* $S_p = (sp_1, \ldots, sp_m)$ *and* $\hat{S}_p = (\hat{sp}_1, \ldots, \hat{sp}_m)$ *be the SVD coefficients of* P *and* $\hat{P}$*, respectively. Then we have*

$$D(P,Q) = \frac{1}{2}\sum_{i=1}^{m} f_{pq}(i) \tag{7}$$

$$f_{pq}(i) = (sp_i - \hat{sq}_i)^2 + (sq_i - \hat{sp}_i)^2 - (sp_i - \hat{sp}_i)^2 - (sq_i - \hat{sq}_i)^2.$$

Proof: From the definition,

$$D(P,Q) = \sum_{i=1}^{m} (p_i - q_i) \cdot \log\left(\frac{p_i}{q_i}\right).$$

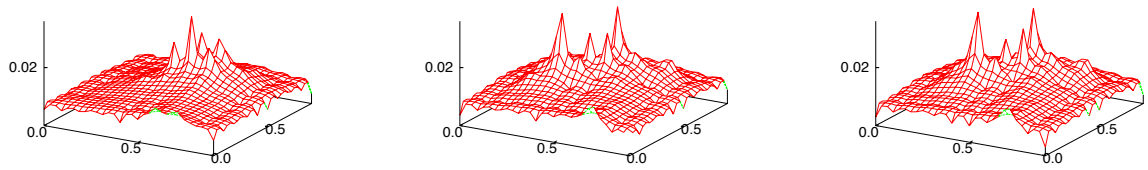
Then we have

$$\begin{aligned} D(P,Q) &= \sum_{i=1}^{m} (p_i - q_i) \cdot (\log p_i - \log q_i) \\ &= \frac{1}{2}\sum_{i=1}^{m} f_{pq}(i). \end{aligned}$$

In light of Parseval's theorem, this completes the proof. □

The KL divergence can be obtained from Equation (7) using the SVD calculated from histogram data. The number of buckets of a histogram (i.e., m) could be large, especially for high-dimensional spaces, while the most of buckets may be empty. The justification of using SVD is that very few of the SVD coefficients of real datasets are often significant and the majority are small, thus, the error is limited to a very small value. When calculating the SVD from the original histogram, we select a small number of SVD coefficients (say c coefficients) that have the largest energy from the original SVD array. This indicates that these coefficients will yield the lowest error among all SVD.

For Equation (7), we compute the KL divergence with the top c SVD coefficients, that is, we compute $f_{pq}(i)$ if



(a) $c = 1$ *(b)* $c = 16$ *(c) Original data*

Figure 2. Approximation of probability distribution from running motion: three probability distributions are shown here, from running motion of *MoCap*, which is shown in Figure 1 (*Query* distribution). They are approximations of $c = 1$, $c = 16$ and the original data, respectively.

we select either sp_i or sq_i ($\hat{sp}_i$ or $\hat{sq}_i$), otherwise, we can simply ignore these coefficients since they are very close to zero. Thus, we obtain the approximate KL divergence of P and Q:

$$d'_c(P,Q) = \frac{1}{2}\sum_{i=1}^{c} f_{pq}(i). \quad (8)$$

Figure 2 shows the SVD-based approximation of probability distribution from *MoCap*. It is represented by a 10×10 bucketized histogram (i.e. full coefficients $c = m = 100$). The numerical rank, however, is much lower. As well as the basic version of *D-Search* described in Section 3.2, the enhanced version also uses the algorithm of multi-step scan (see Algorithms 1 and 2), which efficiently finds similar distributions using their SVD-based approximate KL divergences. Figure 2 shows the gradual 'refinement' of the approximation. In Figure 2 (a), we compute the approximate distance from the coarsest version of a distribution P as the first step of the refinement. If the distance is greater than the current k-th nearest neighbor distance (i.e., D_{cb}), we can prune P. Otherwise, we compute the distance from the more accurate version as the second refinement step (see Figure 2 (b)). We compute the exact distance from the original representation of P only if the approximate distance does not exceed D_{cb} (see Figure 2 (c)).

3.4 Theoretical Analysis

In this section we examine the time and space complexity of our approach and compare it with the complexity of the naive solution. Recall that n is the number of input distributions, m is the number of buckets that we impose on the address space, and c is the number of buckets or SVD coefficients that our methods keeps.

3.4.1 Space Complexity

Naive method

Lemma 3 *The naive method requires $O(mn)$ space.*

Proof: The naive method requires the storage of m-bucketized histograms of n distributions, hence the complexity is $O(mn)$. □

Proposed Solution (*D-Search*)

Lemma 4 *The proposed algorithms require $O(m + n)$ space.*

Proof: *D-Search* initially allocates memory to store histogram of m buckets. The basic version of *D-Search* selects the top c most populated buckets, and keeps them. The enhanced version calculates the SVD coefficients and keeps only the top c coefficients. Then they reduce the number of buckets (or SVD coefficients) to $O(c)$ and allocate $O(cn)$ memory for computing the criterion. However, c is normally a very small constant, which is negligible. We sum up all the allocated memory and we obtain a space complexity of $O(m + n)$. □

3.4.2 Time Complexity

Naive method

Lemma 5 *The naive method requires $O(mn)$ time to compute KL divergence for the k-nearest neighbor search.*

Proof: Computing the KL divergence requires $O(m)$ time for every distribution pair. For n distributions, it would take $O(mn)$ time. □

Proposed Solution (*D-Search*)

Lemma 6 *The proposed algorithms require $O(n)$ time to compute approximate KL divergence for the k-nearest neighbor search.*

Proof: The calculation of the nearest neighbor search requires $O(cn)$ time. We handle c histogram values (or SVD coefficients) for each distribution. This is repeated for n number of input distributions. Again, since c is a small constant value, the time complexity distribution search can be simplified to $O(n)$. □

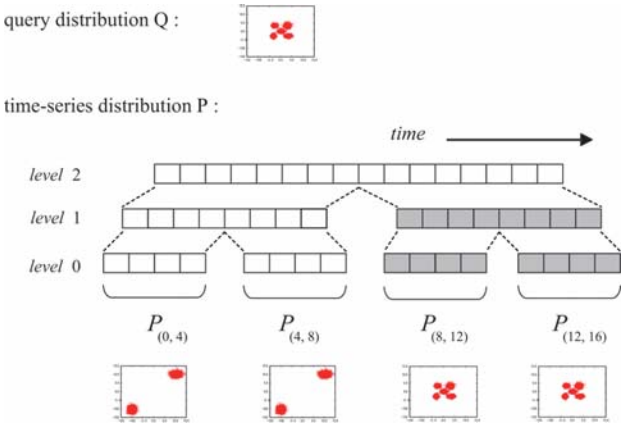


Figure 3. Time-series distribution search (multiple windows, $w_0 = 4$, $W = 2$).

3.5 Time-series distribution mining

Many data sources are observations that evolve over time leading to time-series distribution data. For example, financial datasets depict the prices of every stock over time, which is a common example of time-series distribution data. Reporting meteorological parameters such as temperature readings from multiple sensors gives rise to a numerical distribution sequence. Business warehouses represent time-series categorical distribution sequences such as the sale of every commodity over time. Time-series distribution data depict the trends in the observed pattern over time, and hence capture valuable information that users may wish to analyze and understand.

In this section we introduce an extended version of *D-Search*, which can find similar subsequences in time-series distribution datasets. The problem we propose and solve is as follows:

Problem 2 (Time-series distribution mining) *Given time-series distribution datasets and query distribution Q, find subsequences whose distribution minimizes the KL divergence.*

Consider the time ordered series distribution of d-dimensional points. Distributions performed at different times or by different subjects have different durations, and data sampling rates can also be different at various times. We should solve the following question: How do we efficiently find the similar subsequences for multiple windows? In our approach, we choose a geometric progression of windows sizes [14]: rather than estimating the patterns for windows of lengths $w_0, w_0+1, w_0+2, w_0+3, ...$, we estimate them for windows of w_0, $2w_0$, $4w_0$, ..., or, more generally, for windows of length $w_l := w_0 \cdot W^l$ for $l = 0, 1, 2, ...$ Thus, the size of the window set $\mathcal{W}$ we need to examine is dramatically reduced.

The main idea of our approach is shown in Figure 3. We compute the KL divergence of data points falling within a window, and organize all the windows hierarchically. In this case, query distribution in Figure 3 is similar to $P_{(8,12)}$, $P_{(12,16)}$ at the level 0 ($l = 0$), and, $P_{(8,16)}$ at the level 1 ($l = 1$), which are shaded in Figure 3.

With our method, we can also optimally use the sliding window, which is used as general model in time-series processing [23, 8]. By using the sliding window, we can find similar sequences, which are delayed less than the basic window time.

4 Experimental Evaluation

To evaluate the effectiveness of *D-Search*, we carried out experiments on real datasets. Our experiments were conducted on an Intel Core 2 Duo 1.86GHz with 4GB of memory, running Linux.

The experiments were designed to answer the following questions:

1. How successful is *D-Search* in capturing time-series distribution patterns?
2. How does it scale with the sequence lengths n in terms of the computational time?
3. How well does it approximate the KL divergence?

4.1 Pattern discovery in time-series distributions

In this section we describe some of applications where *D-Search* proves useful. Figure 4 shows how *D-Search* finds similar distributions. Note that, for all experimental results, the enhanced version perfectly captures all similar distributions, that is, the output of the enhanced version is exactly the same as that of the naive method and the basic version.

MoCap

This dataset is the subject numbers 7, 13, 14, 16 and 86, taken from the CMU motion capture database [1]. In our framework, a motion is represented as a distribution of hundreds of frames, with each frame being a d-dimensional point. It contains 26 sequences, each consisting of approximately 8000 frames. Each sequence is a series of simple motions. Typical human activities are represented, such as walking, running, exercising, twisting, jogging and jumping.

The results on this dataset were already presented in Section 1. As shown in Figure 1, *D-Search* can successfully identify similar distributions.

EEG

This dataset was taken from a large study that examined the EEG correlates of the genetic predisposition to alcoholism downloaded from the UCI website [2]. It contains measurements from 64 electrodes placed on subjects' scalps

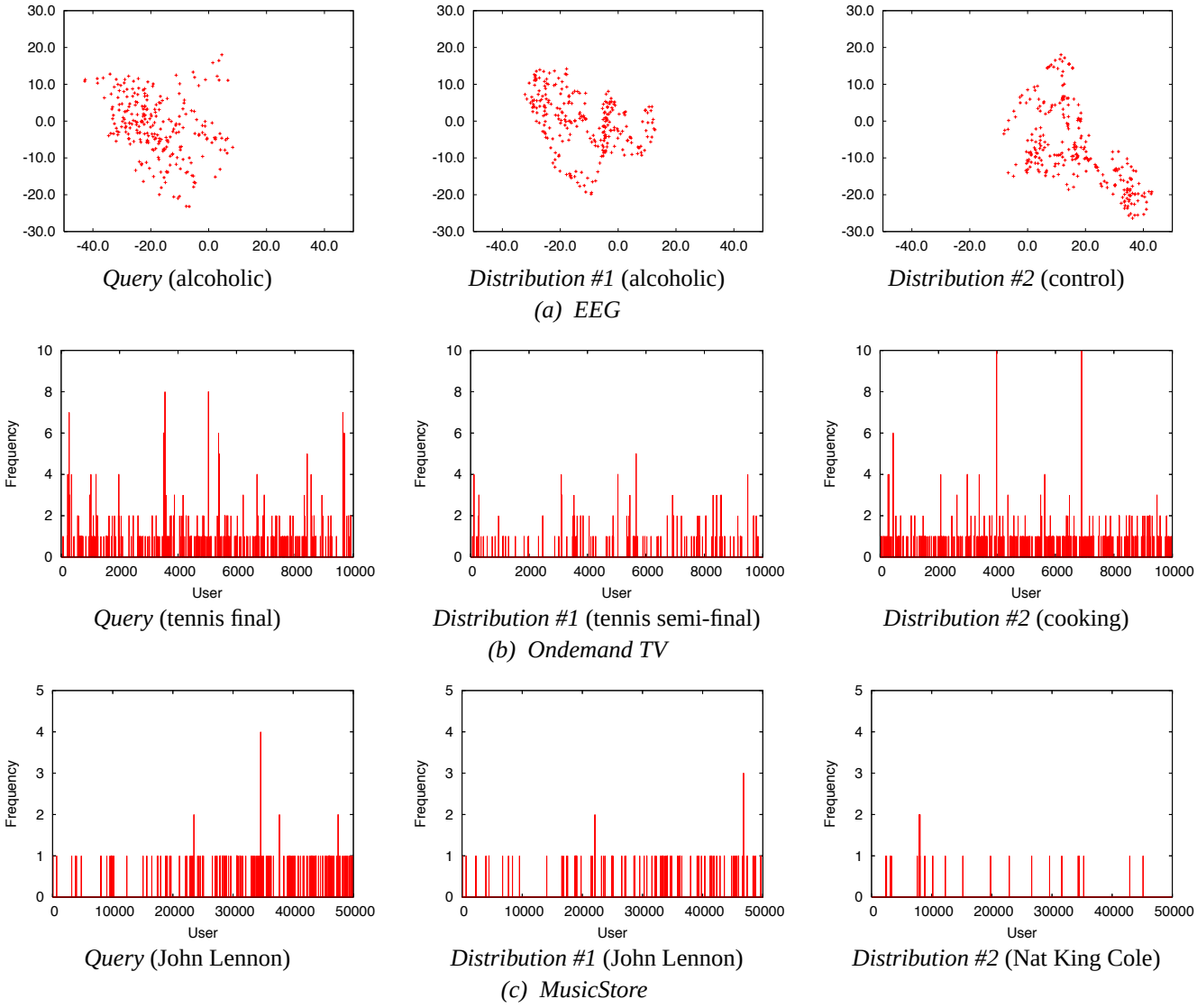


Figure 4. Discovery of subsequences in *EEG*, *Ondemand TV*, and *MusicStore*. We choose window sizes (i.e., w_0) 1 sec., 3 hours, 3 hours on the lowest level for these datasets.

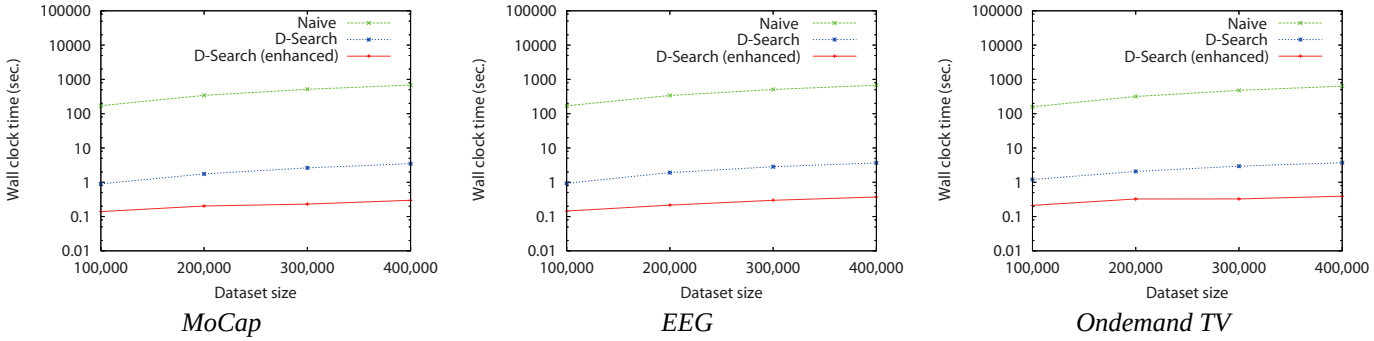


Figure 5. Scalability: wall clock time vs. dataset size n (= number of distributions). *D-Search* can be up to 2,300 times faster than the naive implementation.

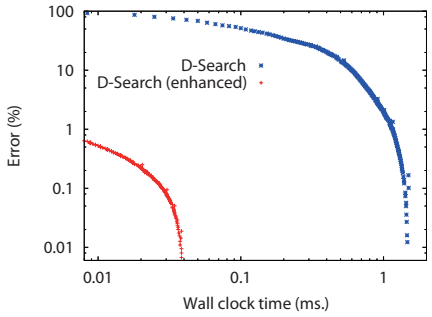


Figure 6. Approximation quality: relative approximation error vs. wall clock time.

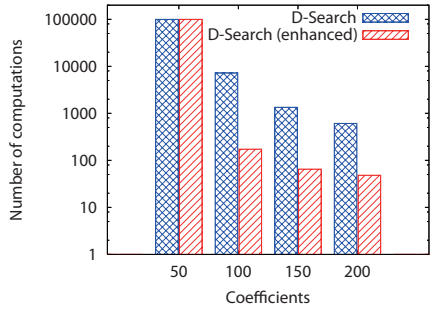


Figure 7. Frequency of approximation use: number of computations vs. coefficients.

that were sampled at 256 Hz (3.9-msec epoch) for 1 second, that is, the length of each sequence is 256. There were two groups of subjects: alcoholic and control.

Our approach is also useful for classification. Figure 4 (a) shows that *D-Search* can classify the query distribution and *Distribution #1*, (a subsequence from co2a0000364 of 36-37sec., and a subsequence from co3a0000451 of 56-57sec., respectively), into the same group (in fact, they both corresponded to "alcoholic"). In contrast, *Distribution #2*, which is a subsequence from co2c0000364 of 75-76sec., go to another group (in fact, it belongs to "control").

Ondemand TV

This dataset is from the *Ondemand TV* service of 13,231 programs that users viewed in a 6-month timeframe (from 14th May. 2007 to 15th Nov. 2007). We randomly select 10,000 anonymous users from the dataset. Each distribution sequence contains a list of attributes (e.g., content ID, the date the user watched the content, the ID of the user who watched the content).

As shown in Figure 4 (b), our method can find similar *Ondemand TV* content. For example, *D-Search* found that *Distribution #1* was a similar distribution, and *Distribution #2* was a dissimilar distribution to the query distribution. In fact, query distribution, *Distribution #1* and *Distribution #2* are "Sports: Australian Open Tennis Championships 2007 Women's Final (from 1st Feb. 2007 to 1st Apr. 2008)", "Sports: Australian Open 2007 Tennis Championships Women's Semifinal (from 1st Feb. 2007 to 1st Apr. 2008)", "Cooking: Oliver's Twist No.1 (from 23rd Oct. 2006 1st Aug. 2008")

MusicStore

This dataset consists of the purchasing records from *MusicStore* obtained over 16 months, (from 4th Apr. 2005 to 1st Jul. 2006). Each record has 3 attributes: user ID (50,000 anonymous, randomly selected users), music ID (43,896 items of music), date of purchase/sale.

Figure 4 (c) shows that *D-Search* can identify similar user groups. For example, *D-Search* found that query distribution was similar to *Distribution #1*. In fact, the query distribution and *Distribution #1* are histograms of purchasers of John Lennon's "Woman", and John Lennon's "Love", respectively. In contrast, *Distribution #2* was not found to be as a similar distribution. In fact, *Distribution #2* was a purchaser histogram of Nat King Cole's "L-O-V-E".

4.2 Performance

To evaluate the search performance, we compared the basic version and the enhanced version with the naive approach. We present experimental results on search performance for when the data set size varies.

Figure 5 compares our algorithms with the naive method in terms of computation cost. Database size varied from 100,000 to 400,000. Note that the y-axis uses logarithmic scale. We conducted this experiment with a histogram of $m = 10,000$, starting coefficient $c_1 = 50$, and step $h = 4$. Each result reported here is the average of 100 trials.

There, we show the wall-clock time versus the database size n for three datasets. *D-Search* provides a dramatic reduction in computation time. Specifically, the enhanced (basic) version achieves up to 2,300 times (230 times) faster than the naive implementation in this experiment.

In addition to high-speed processing, our method achieves high accuracy; the output of the enhanced version is exactly the same as those of the naive algorithm and the basic version.

4.3 Analysis of proposed algorithms

D-Search exploits multiple computations for the approximation of KL divergence. In this section we discuss the approximation quality of each granularity.

Figure 6 shows scatter plots of the computation cost versus the approximation quality. The x-axis shows the computation cost for KL divergences, and the y-axis shows their relative approximation error rate. We compare the basic version and the enhanced version in the figure. The figure implies a trade-off between quality and cost, but the results of the enhanced version are close to the lower left for both datasets, which means that the enhanced version provides benefit in terms of quality and cost.

Figure 7 shows how often each approximation was used in the basic version and the enhanced version for a dataset

size of 100,000. As shown in the figure, most of the data sequences are excluded with the approximations of $c = \{50, 100, 150, 200\}$. The coarser approximation provides reasonable approximation quality, and its calculation speed is high. On the other hand, although the approximation with higher granularity is not very fast, it offers good approximation quality. Accordingly, using approximations with various granularities offers significant advantages in terms of approximation quality and calculation speed. Our algorithms, especially the enhanced version efficiently prunes a large number of search candidates, which leads to a significant reduction in the search cost.

5 Conclusion

We introduced the problem of distribution search, and proposed *D-Search*, which includes fast and effective algorithms, as its solution. *D-Search* has all the desired characteristics:

- High-speed search: Instead of $O(mn)$ time the naive solution requires, our solution needs $O(n)$ time to compute distance for distribution search.
- Exactness: It guarantees no false dismissals.
- It can be extended to time-series distribution mining, which can find similar subsequences in time-series distribution datasets.

Our experimental results reveal that *D-Search* is significantly faster than the naive method, and occasionally up to 2,300 times faster while it perfectly captures all similar distributions. Furthermore, our algorithms can be extended to time-series distribution mining. In fact, we present case studies on real datasets and demonstrate the effectiveness of our approach in discovering patterns among time-series distribution datasets. We believe that the addressed problem and our solution will be of fundamental interest in data mining.

References

[1] *CMU Graphics Lab Motion Capture Database*. http://mocap.cs.cmu.edu/.

[2] *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml/.

[3] J. Barbic, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard. Segmenting motion capture data into distinct behaviors. In *Graphics Interface*, pages 185–194, 2004.

[4] S. D. Bay, D. F. Kibler, M. J. Pazzani, and P. Smyth. The uci kdd archive of large data sets for data mining research and experimentation. In *SIGKDD Explorations*, pages 81–85, 2000.

[5] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of ACM SIGMOD*, pages 551–562, San Diego, California, June 2003.

[6] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *Proceedings of VLDB*, pages 876–887, Toronto, Canada, August/September 2004.

[7] S. Fischer, R. Lienhart, and W. Effelsberg. Automatic recognition of film genres. In *ACM Multimedia*, pages 295–304, 1995.

[8] L. Gao and X. S. Wang. Continuous similarity-based queries on streaming time series. In *IEEE Trans. Knowl. Data Eng. (TKDE)*, pages 1320–1332, 2005.

[9] Z. Guo, Z. Zhang, E. P. Xing, and C. Faloutsos. Enhanced max margin learning on multimodal data mining in a multimedia database. In *KDD*, pages 340–349, 2007.

[10] X. Huang, S. Z. Li, and Y. Wang. Jensen-shannon boosting learning for object recognition. In *Proceedings of IEEE Computer Society International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 144–149, 2005.

[11] Y. Ishikawa, Y. Machida, and H. Kitagawa. A dynamic mobility histogram construction method based on markov chains. In *Proceedings of Int. Conf. on Statistical and Scientific Database Management (SSDBM)*, pages 359–368, 2006.

[12] C. Li, P. Zhai, S.-Q. Zheng, and B. Prabhakaran. Segmentation and recognition of multi-attribute motion sequences. In *ACM Multimedia*, pages 836–843, 2004.

[13] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[14] S. Papadimitriou and P. S. Yu. Optimal multi-scale patterns in time series streams. In *SIGMOD*, pages 647–658, 2006.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

[16] M. L. Raymer, T. E. Doom, L. A. Kuhn, and W. F. Punch. Knowledge discovery in medical and biological datasets using a hybrid bayes classifier/evolutionary algorithm. In *IEEE Transactions on Systems*, pages 802–813, 2003.

[17] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, 2000.

[18] Z. Sun. Adaptation for multiple cue integration. In *Proceedings of IEEE Computer Society International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 440–445, 2003.

[19] P. Sykacek and S. J. Roberts. Adaptive classification by variational kalman filtering. In *NIPS*, pages 737–744, 2002.

[20] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *Proceedings of VLDB*, pages 922–933, Trondheim, Norway, August/September 2005.

[21] A. Traina, C. Traina, S. Papadimitriou, and C. Faloutsos. Tri-plots: Scalable tools for multidimensional data mining. *KDD*, Aug. 2001.

[22] J.-P. Vert. Adaptive context trees and text clustering. *IEEE Transactions on Information Theory*, 47(5):1884–1901, 2001.

[23] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369, 2002.