D-Search: An Efficient and Exact Search Algorithm for Large Distribution Sets

Yasuko Matsubara[1], Yasushi Sakurai[2] and Masatoshi Yoshikawa[1]

[1]Kyoto University; [2]NTT Communication Science Labs

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 or 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 a wall clock time up to *2,300 times faster* than the naive implementation without sacrificing accuracy.

Keywords: Distribution sets; KL divergence; Singular value decomposition, Likelihood

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.

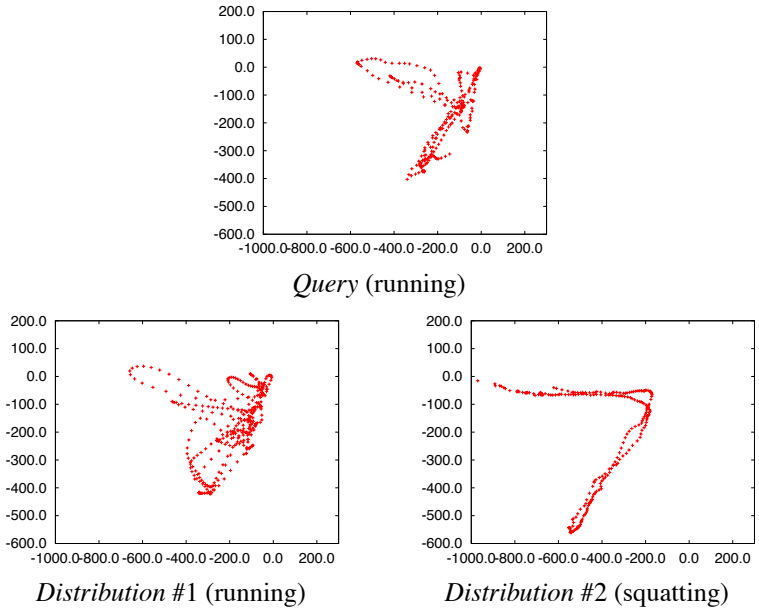


Fig. 1. Three distributions from *MoCap* data: all showing scatter plots of left and right foot kinetic energy values for different motions. The query distribution and *Distribution #1* "look" more similar, while *Distribution #2* "looks" different.

We 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 identifies 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 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 focus mainly 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 a 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 [16, 5, 22, 12, 3, 36, 15]. 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. As another example, consider digital images, which contain a number of rows and columns of pixels. We can find similar images in large digital image datasets by using these numerical attributes.

– *Medical data analysis* : The extraction of meaningful information from large medical datasets is the central theme of many medical research problems [27]. The Brain-Computer Interface (BCI), which is mainly designed to help disabled people control personal computers using biofeedback, is a completely new approach in the field of neurology [25]. Biofeedback is a coaching and training process that helps people learn how to change patterns of behavior, to take greater responsibility for their health and for their mental, physical, emotional and spiritual functions. However, it is undesirable for disabled people to have to adapt to computers. The basic idea behind BCI is that the computer adapts rather than the person. 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 detected by electrodes placed suitably on the scalp. They result from the superposition of three main types of brain potential, namely oscillatory, event-related, and slow potential shifts. Different components of the EEG signal have been widely demonstrated to have measurable correlates with the brain activity involved in specific mental tasks [6, 32]. As another example, consider diabetes diagnosis information, where each diagnosis has a list of numerical attributes (e.g., age in years, body mass index, diastolic blood pressure). For this collection of clouds, we would like to find patterns and groups, e.g., to undertake medical research.
– *Web service* : There are numerous, fascinating applications for web service mining. One example is an SNS system such as a blog hosting service. Consider a large number of blog users each of whom has a list of numerical and categorical attributes (e.g., total number of postings, average posting length, URLs of outgoing links, average number of trackbacks). Now assume that we can record all these attributes, on a daily basis, for all blog users. For this collection of clouds, we would like to find patterns and groups, e.g., to undertake sociological and behavioral research. As another example, let us assume web services such as an ondemand-TV service, which records the viewing of TV programs, 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 perform 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 efficient algorithms called *D-Search*, which can find similar distributions in large distribution datasets. The contributions are as follows: (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 an m-bucketized histogram and a query distribution, our algorithms require only $O(n)$ to compute KL divergence, instead of $O(mn)$ as required by the naive method, and lead to a large reduction in the search cost. (b) Extensive experiments on real multi-dimensional datasets shows that our method is significantly faster than the naive method without sacrificing accuracy. (c) We propose to address the problem of time-series distribution mining and present our solution, which finds similar subsequences in time-series distribution datasets. (d) Finally, we propose the extend version of *D-Search* which finds the top k distributions that maximize the likelihood, given a probability density.

1.3. Outline

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 introduces preliminary concepts, and 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 address the problem of time-series distribution mining. Section 4 describes an extended version of *D-Search*, which finds the top k distributions given a probability density function. Section 5 introduces some of the applications for which our method proves useful, and evaluates our algorithms by conducting extensive experiments. Section 6 concludes the paper.

2. Related Work

Related work falls broadly into three categories: (1) similarity functions between distributions, (2) probabilistic query processing, and (3) model estimation. We provide a survey of the related literature.

2.1. Comparing two distributions

There are several statistical methods [26] for deciding 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* [28] 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 between 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, triplots [34] can find patterns across two large, multi-dimensional sets of points, although they cannot assign a distance score [4]. The most suitable idea for our setting is the Kullback-Leibler (KL) divergence (see Equation (2)), which gives a notion of the distance between two distributions. The KL divergence is commonly used in several fields to measure the distance between two probability density functions (PDFs, as in, e.g., information theory [35], pattern recognition [31, 17]).

2.2. Probabilistic queries

A remotely related problem is the problem of probabilistic queries. Cheng et al. [7] classify and evaluate probabilistic queries over uncertain data based on models of uncertainty. An indexing method for regular probabilistic queries is then proposed in [8]. In [33] Tao et al. present 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 [7] is that we focus on comparing differences between distributions, while Cheng focuses on locating areas in distributions that satisfy a given threshold.

2.3. Model estimation

Recently significant progress has been made on understanding the theoretical issues surrounding learning mixture distributions in theoretical computing [20, 9, 19] and machine learning [37, 30, 21]. Mixture models are used in a broad range of scientific and engineering applications, including computer vision and speech recognition. The effectiveness of modeling hinges on choosing the right parameters for the mixture distribution, and the problem of parameter selection for mixture models has a long history [20]. The most commonly used method for parameter estimation is maximum likelihood estimation (MLE), which suggests choosing the parameters in a way that maximizes the likelihood of the observed data, given a model. In modern practice this is generally accomplished with the iterative optimization technique known as the expectation maximization (EM) algorithm [10]. Recently significant progress on understanding the theoretical issues surrounding learning mixture distributions and EM has been made in theoretical computer science [9, 19, 37, 30, 21, 13].

Distribution search and mining are problems that, to our knowledge, have not been addressed. The distance functions among clouds that we mentioned above either expect a continuous distribution (such as 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 define some fundamental concepts and 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

3.1.1. KL divergence

Given two distributions P and Q, there are several measures of their distance/similarity, as 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 using information theory as a basis, and specifically the Kullback-Leibler (KL) divergence.

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
$\mathcal{Q}$	probability density function
$L(P\|Q)$	likelihood function of Q given P
$l_c(P\|Q)$	upper bounding likelihood of Q given P with c histogram buckets
$l'_c(P\|Q)$	approximate likelihood of Q given P with c SVD coefficients

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. The KL divergence measures the distance from one probability distribution P to another one Q [11]. Intuitively, the (continuous, and asymmetric) KL divergence $D_{CKL}(P, Q)$ corresponds to the extra coding cost we have to suffer, when we try to compress samples from P while using the coding that is optimal for Q. Formally, the KL divergence is defined as follow:

$$D_{CKL}(P, Q) = \int p_x \cdot \log\left(\frac{p_x}{q_x}\right) dx \tag{1}$$

where p_x and q_x are the probability density functions at x.

As defined, the KL divergence has two drawbacks: (a) it needs the probability density functions, which we don't know and (b) it is asymmetric. All the above definitions are for the case of continuous variables. The first issue can be (partially) resolved by bucketization: once we choose a size of grid cells, m (e.g., chosen according to space budget), we can impose a d-dimensional grid on all our clouds, and measure the counts of points in every grid cell. Then, we employ the discrete version of the KL divergence, defined as follows:

$$D_{KL}(P, Q) = \sum_{i=1}^{m} p_i \cdot \log\left(\frac{p_i}{q_i}\right) \tag{2}$$

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 using the symmetric KL

divergence D_{SKL}:

$$D_{SKL}(P,Q) = D_{KL}(P,Q) + D_{KL}(Q,P) = \sum_{i=1}^{m}(p_i - q_i) \cdot \log\left(\frac{p_i}{q_i}\right). \quad (3)$$

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 areas in multi-dimensional space may take a zero value. To avoid this, we introduce the Laplace estimator [23, 18]:

$$p_i = \frac{p'_i + 1}{|P'| + m} \cdot |P'| \quad (4)$$

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.

3.1.2. Singular value decomposition

Here we provide a brief overview of singular value decomposition (SVD). A set $\mathbf{P}$ of n m-bucketized histograms can be transformed using SVD as follows.

Every matrix $\mathbf{P} \in \mathbb{R}^{n \times m}$ can be decomposed into

$$\mathbf{P} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where $\mathbf{U} \in \mathbb{R}^{n \times r}$, $\mathbf{V} \in \mathbb{R}^{m \times r}$ and $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$, with $r \leq min(n, m)$ the rank of $\mathbf{P}$. The columns v_i of $\mathbf{V} \equiv [v_1 \cdots v_r]$ are the right singular vectors of $\mathbf{P}$ and they form an orthonormal basis its row space. Similarly, the columns u_i of $\mathbf{U} \equiv [u_1 \cdots u_r]$ are the left singular vectors and form the basis of the column space of $\mathbf{P}$. Finally $\mathbf{\Sigma} \equiv diag[\sigma_1 \cdots \sigma_r]$ is a diagonal matrix with positive values σ_i, called the singular value of $\mathbf{P}$. The transformed data (i.e., the projection of $\mathbf{P}$) $\mathbf{S_P} \equiv [sp_1 \cdots sp_r]$ is given as $\mathbf{S_P} = \mathbf{U}\mathbf{\Sigma}$.

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 subsequent subsections. (a) How can we represent the distribution of a 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 falls into the i-th bucket.

The naive solution is to maintain an exact 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 argument against it is that we may ignore some sparsely 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:

$$d_c(P,Q) = \sum_{i \in \mathcal{I}_{pq}} (p_i - q_i) \cdot \log\left(\frac{p_i}{q_i}\right). \tag{5}$$

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

$$D(P,Q) \geq d_c(P,Q). \tag{6}$$

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

$$D(P,Q) = d_m(P,Q) \geq d_c(P,Q), \tag{7}$$

which completes the proof. □

3.2.2. Multi-Step Sequential Scan

Instead of operating on lower bounding KL divergence with c buckets of a single computation, we propose using multiple computations, in an attempt to achieve 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 $\mathcal{N}$ indicates 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 the KL divergence. Finally, we compute the exact KL divergences between distributions, which are not pruned in any steps, and the query distribution.

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
```

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

In the previous subsection, we described the basic version of *D-Search*, 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 a more efficient solution in practice requiring high accuracy, not a theoretical guarantee. As the enhanced version of *D-Search*, we propose compressing the histograms using the singular value decomposition (SVD), and *then*

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
```

keeping some appropriate coefficients. As we show later, this decision significantly improves both space and response time, with a negligible effect on the mining results. The only tricky aspect is that if we just keep the top c SVD coefficients, we might not obtain good accuracy for the KL divergence. This led us to the design of our method that we describe next. The main idea 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 coeffi-

cients of P and $\hat{P}$, respectively. Then we have

$$D(P,Q) = \frac{1}{2}\sum_{i=1}^{m} f_{pq}(i) \tag{8}$$
$$f_{pq}(i) = (sp_i - \hat{sq}_i)^2 + (sq_i - \hat{sp}_i)^2 - (sp_i - \hat{sp}_i)^2 - (sq_i - \hat{sq}_i)^2.$$

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

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

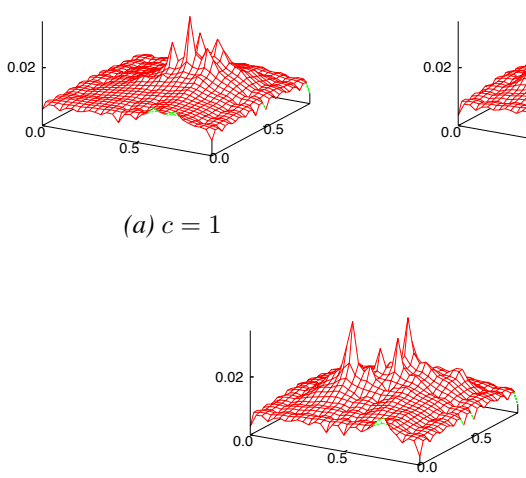
In light of Parseval's theorem, this completes the proof. □

The KL divergence can be obtained from Equation (8) 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 for 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 the SVD coefficients.

For Equation (8), we compute the KL divergence with the top c SVD coefficients, that is, we compute $f_{pq}(i)$ if 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:

$$d'_c(P,Q) = \frac{1}{2}\sum_{i=1}^{c} f_{pq}(i). \tag{9}$$

Figure 2 shows the SVD-based approximation of the probability distribution from *MoCap*. It is represented by a 10×10 bucketized histogram (i.e. full coefficients $c = m = 100$). However, the numerical rank is much lower. In addition to the basic version of *D-Search* described in Section 3.2, the enhanced version also uses the multi-step scan algorithm (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)).



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

(c) Original data

Fig. 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.

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 it reduces the number of buckets (or SVD coefficients) to $O(c)$ and allocates $O(cn)$ memory for computing the KL divergence. 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 the 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 the 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 the number n of input distributions. Again, since c is a small constant value, the time complexity distribution search can be simplified to $O(n)$. □

3.5. Time-series distribution mining

Many data sources consist of 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 a search method for time-series distributions that can find similar subsequences in 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 similar subsequences for multiple windows? In our approach, we choose a geometric progression of windows sizes [29, 24]: 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 behind 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 a

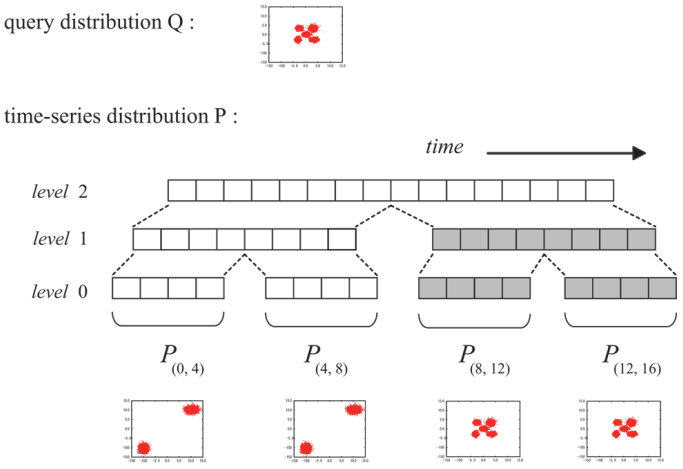


Fig. 3. Time-series distribution search (multiple windows, $w_0 = 4, W = 2$).

general model in time-series processing [38, 14]. By using the sliding window, we can find similar sequences, which are delayed less than the basic window time.

4. Extension to probability density functions

In the previous subsection we presented our distribution search algorithms, which use distributions as queries for search processing. In this section, we extend our concepts and introduce the probability density function (PDF) to find distributions that maximize the likelihood. We first describe how to measure the distance between distributions and probability density. We use the likelihood as the basis throughout this problem. Here we formally define our problem as follows:

Problem 3. Given n distributions and a probability density function $\mathcal{Q}$, find the top k distributions that maximize the likelihood.

Let $X = \{x_1, \ldots x_M\}$ be a distribution with M points, and $\mathcal{Q}$ be the probability density function. The log likelihood function is defined by:

$$L(X|\mathcal{Q}) = \sum_{i=1}^{M} \log \mathcal{Q}(x_i) \tag{10}$$

where x_i is the value of the i-th point of X, and $\mathcal{Q}(x_i)$ shows the probability function of x_i, which is subject to the constraint $0 \leq \mathcal{Q}(x_i) \leq 1$.

4.1. Upper bounding likelihood

Given a set of distributions and a PDF $\mathcal{Q}$, the naive solution is to compute the likelihood using all the data points of the distributions with $\mathcal{Q}$, as in Equation (10). However, this idea is too expensive to compute, especially for a large number of data points (i.e., M is too large). We thus propose bucketizing X and $\mathcal{Q}$. The concept of our method is illustrated in Figure 4. Instead of using all the points x_i $(i = 1, \ldots, M)$ of X for

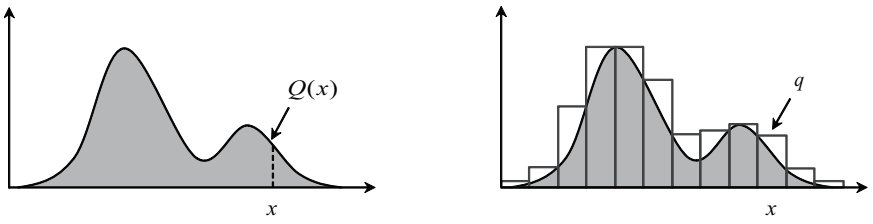


Fig. 4. Upper bounding likelihood: instead of using all the data points of the distributions with PDF $\mathcal{Q}$, we compute the discrete version of the likelihood.

computing the likelihood, we bucketize both X and $\mathcal{Q}$ with m grid cells. Here, we use the upper bound of probability for $\mathcal{Q}$.

Let $P = \{p_1, \dots p_m\}$ be a bucketized distribution X with m grid cells, and $Q = \{q_1, \dots q_m\}$ be the upper bounds of $\mathcal{Q}$ with m buckets. The upper bounding likelihood is defined by:

$$L(P|Q) = \sum_{i=1}^{m} p_i \log q_i \tag{11}$$

where p_i is the value of the i-th bucket of P, and q_i shows the i-th value of Q.

Lemma 7. For any distributions, the following inequality holds.

$$L(X|\mathcal{Q}) \leq L(P|Q). \tag{12}$$

Proof. Let $x_1, x_2, \dots, x_s$ be points of X, which belong to the i-th bucket p_i of P. For any x, we have $\sum_{j=1}^{s} \log \mathcal{Q}(x_j) \leq p_i \cdot \log q_i$. This completes the proof. □

Here, a subtle point should be noted. As well as the KL divergence, we can compute the likelihood with c histogram values, that is, we compute $p_i \cdot \log q_i$ 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 upper bounding likelihood of P given Q:

$$l_c(P|Q) = \sum_{i \in \mathcal{I}_{pq}} p_i \cdot \log q_i. \tag{13}$$

Lemma 8. For any distributions, the following inequality holds.

$$L(P|Q) \leq l_c(P|Q). \tag{14}$$

Proof. From the definition, since $\forall i, p_i \cdot \log q_i \leq 0$, for any c value $(1 \leq c \leq m)$, we have

$$L(P|Q) = l_m(P|Q) \leq l_c(P|Q), \tag{15}$$

which completes the proof. □

4.2. Likelihood approximation

Next, we describe how to approximate the likelihood of distributions by using the SVD approach. Let $\hat{Q} = (\hat{q}_1, \dots, \hat{q}_m)$ be the histogram of the logarithms of $Q =$

$(q_1, \ldots, q_m)$, i.e., $\hat{q}_i = \log q_i$. Let S_p and $\hat{S}_q$ be the SVD coefficients of P and $\hat{Q}$, respectively. We present our solution using S_p and $\hat{S}_q$.

Theorem 2. Let $S_p = (sp_1, \ldots, sp_m)$ and $\hat{S}_q = (\hat{sq}_1, \ldots, \hat{sq}_m)$ be the SVD coefficients of P and $\hat{Q}$, respectively. Then we have

$$L(P|Q) = \frac{1}{2} \sum_{i=1}^{m} g_{pq}(i) \tag{16}$$

$$g_{pq}(i) = sp_i^2 + \hat{sq}_i^2 - (sp_i - \hat{sq}_i)^2$$

Proof. From the definition,

$$L(P|Q) = \sum_{i=1}^{m} p_i \log q_i.$$

Then we have

$$L(P|Q) = \frac{1}{2} \sum_{i=1}^{m} g_{pq}(i).$$

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

For Equation (16), we compute the likelihood with the top c SVD coefficients. Thus, we obtain the approximate likelihood of P and Q:

$$l'_c(P|Q) = \frac{1}{2} \sum_{i=1}^{c} g_{pq}(i). \tag{17}$$

We also use the multi-step scan algorithm (Algorithms 1 and 2), which provides an efficient solution for probability density functions.

5. 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 1.86 GHz with 4 GB 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 well does *D-Search* work for probability density functions?
3. How does it scale with the sequence lengths n in terms of computational time?
4. How well does it approximate the KL divergence?

5.1. Pattern discovery in time-series distributions

In this section we describe some of the applications where *D-Search* proves useful. Figures 5-8 show 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.

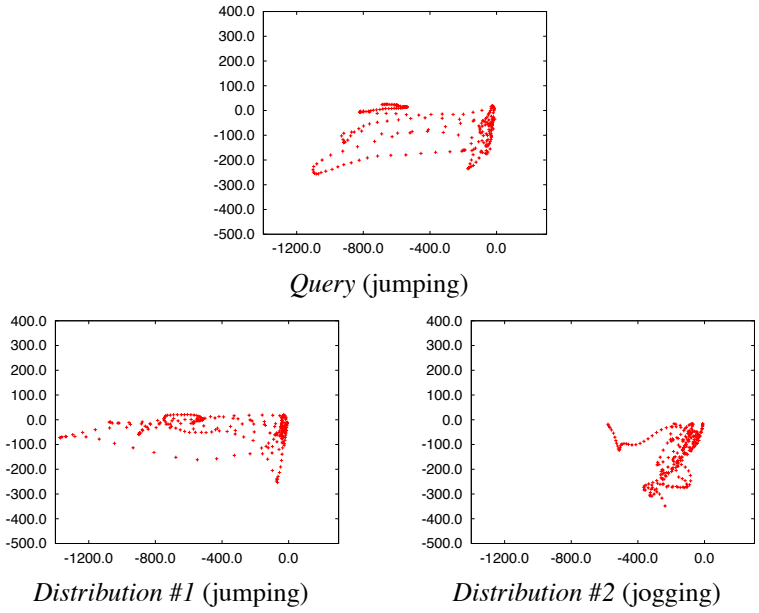


Fig. 5. Discovery of subsequences in *MoCap*. We choose a window size (i.e., w_0) of 1 sec. on the lowest level for this dataset.

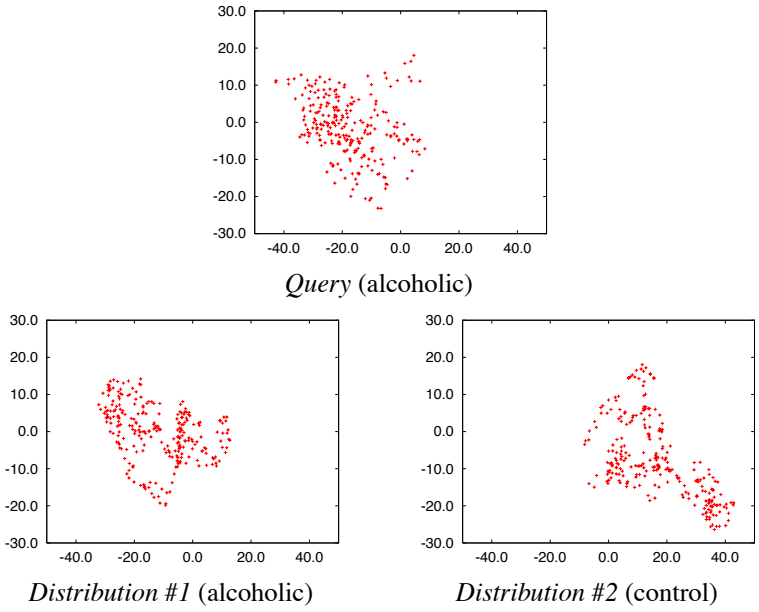
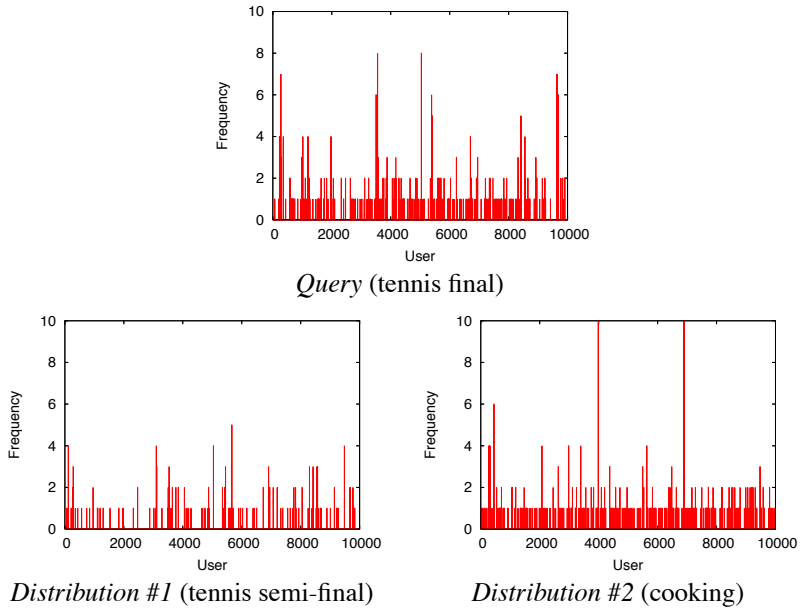


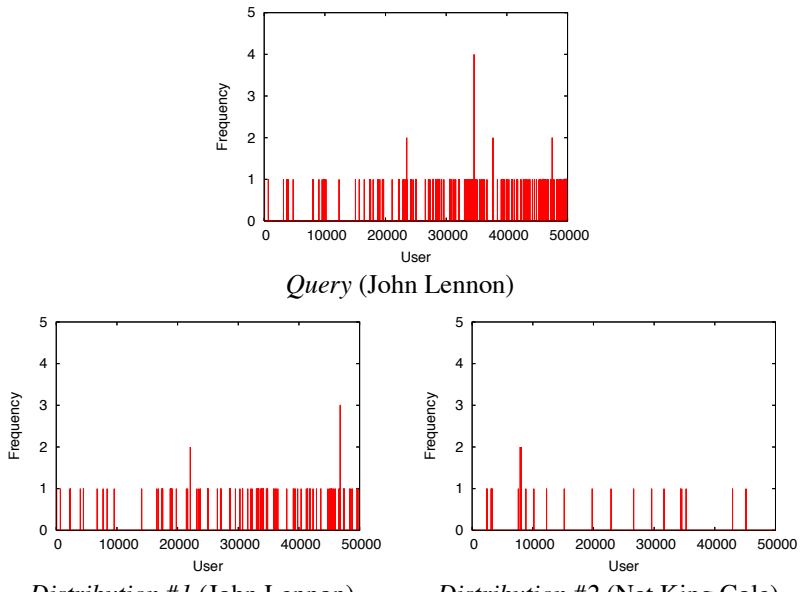
Fig. 6. Discovery of subsequences in *EEG*. We choose window a size (i.e., w_0) of 1 sec. on the lowest level for this dataset.



Query (tennis final)

Distribution #1 (tennis semi-final) *Distribution #2* (cooking)

Fig. 7. Discovery of subsequences in *Ondemand TV*. We choose a window size (i.e., w_0) of 3 hours on the lowest level for this dataset.



Query (John Lennon)

Distribution #1 (John Lennon) *Distribution #2* (Nat King Cole)

Fig. 8. Discovery of subsequences in *MusicStore*. We choose a window size (i.e., w_0) of 3 hours on the lowest level for this dataset.

MoCap

This dataset consists of 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.

One of the results on this dataset was already presented in Section 1. As shown in this figure, *D-Search* can successfully identify similar distributions. Figure 5 also shows that *D-Search* can successfully find similar distributions. For example, *D-Search* detects *Distribution #1* (the subsequence from data no.14-14 2-4 sec.) similar to the query distribution (the subsequence from data no.14-20 2-4 sec.). In fact, they both correspond to a jumping motion. In contrast, *Distribution #2* (from the same data no.14-14, 4-6 sec.) is not found as a similar object. In fact, it corresponds to a jogging motion.

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 that were sampled at 256 Hz (3.9-msec epoch) for 1 sec., 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 6 shows that *D-Search* can classify the query distribution and *Distribution #1*, (a subsequence from co2a0000364 of 36-37 sec., and a subsequence from co3a0000451 of 56-57 sec., 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., goes to another group (in fact, it belongs to "control").

Ondemand TV

This dataset is from an ondemand TV service and consists of 13,231 programs that users viewed in a 6-month timeframe (from 14th May to 15th Nov. 2007). We randomly selected 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 7, our method can find a similar ondemand TV program. 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, the 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 to 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), and date of purchase/sale.

Figure 8 shows that *D-Search* can identify similar user groups. For example, *D-Search* found that the 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 a similar distribution. In fact, *Distribution #2* was a purchaser histogram of Nat King Cole's "L-O-V-E".

5.2. Similarity search with probability density functions

To evaluate the effectiveness of *D-Search* for probability density functions, we present case studies on two datasets.

Gaussians

This is a synthetic example of numerical datasets, each of whose distributions is a mixture of Gaussians. The dataset contains n=400,000 distributions, each with 10,000 points. Figures 9 and 10 show the results on the *Gaussians* dataset. Given PDFs of the Gaussian mixture, our approach can correctly find the appropriate distributions from this dataset.

MoCap

Figures 11 and 12 show that *D-Search* can successfully identify appropriate distributions on the *MoCap* dataset. For example, the top row in Figure 11 shows the PDF, which corresponds to a moving hands motion. *D-Search* recognizes that *Distribution #1* and *Distribution #2* are similar and dissimilar distributions to the PDF. Similarly, *D-Search* identifies that the PDF of balancing on one leg is similar to *Distribution #1* in Figure 12. In fact, it corresponds to the same motion. In contrast, *Distribution #2* corresponds to boxing motion, and it was not found to be a similar object.

5.3. Performance

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

Figure 13 compares our algorithms with the naive method in terms of computation cost. The database size varied from 100,000 to 400,000. Note that the y-axis uses a logarithmic scale. We conducted this experiment with a histogram of $m = 10,000$, and $k = 1$. Each result reported here is the average of 100 trials. To evaluate the efficiency of *D-Search*, We used the three datasets, *MoCap*, *EEG*, and *Ondemand TV*, and we also used the *Gaussians* dataset for probability density functions. *D-Search* provided a dramatic reduction in computation time. Specifically, the enhanced (basic) version achieved 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. While we evaluated performance for the nearest neighbor search (i.e., $k = 1$), our method also achieves a dramatic improvement for the k-nearest neighbor search ($k > 1$).

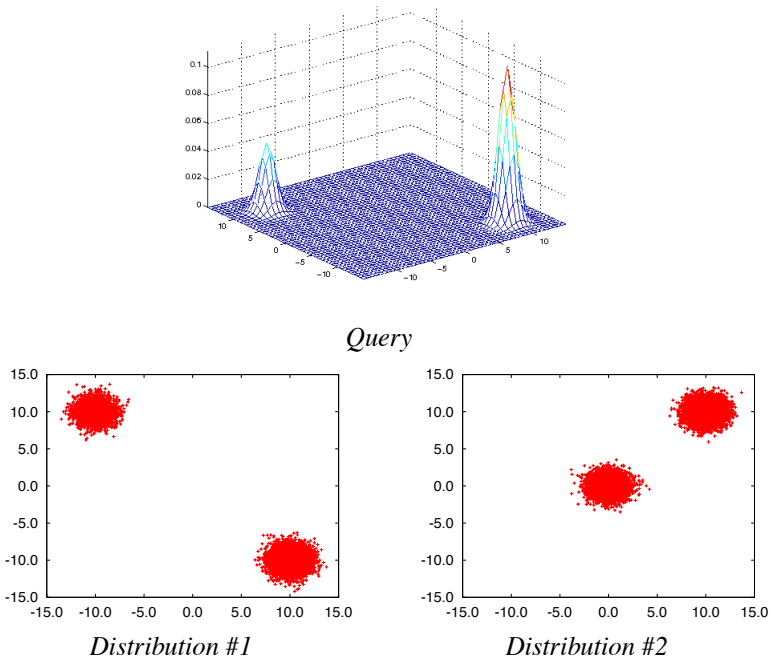


Fig. 9. Discovery of distributions in *Gaussians*. The top row shows the probability density of the Gaussian mixture model, and the middle and bottom rows show similar and dissimilar distributions.

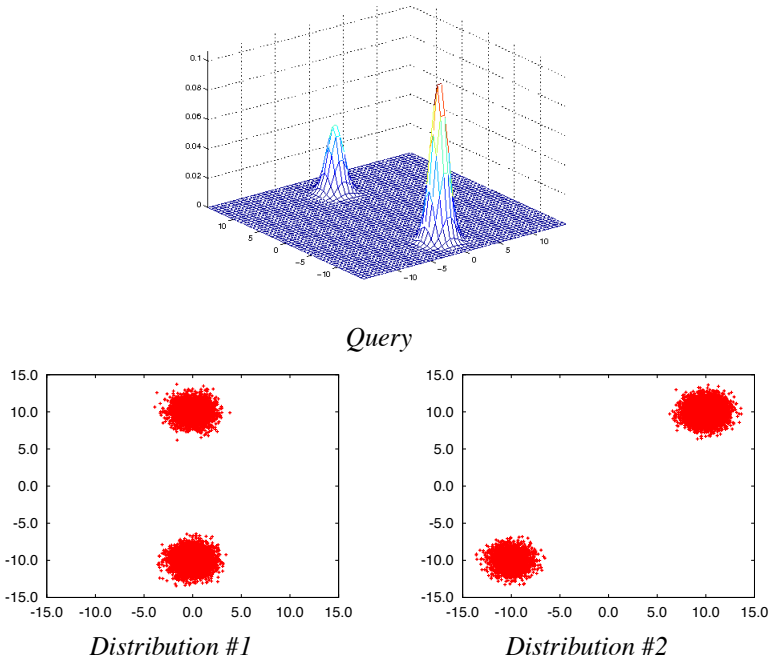
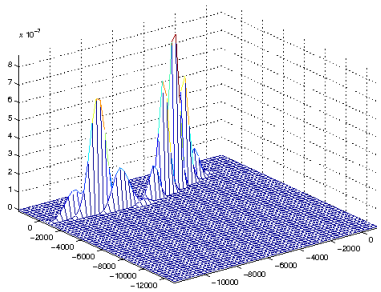
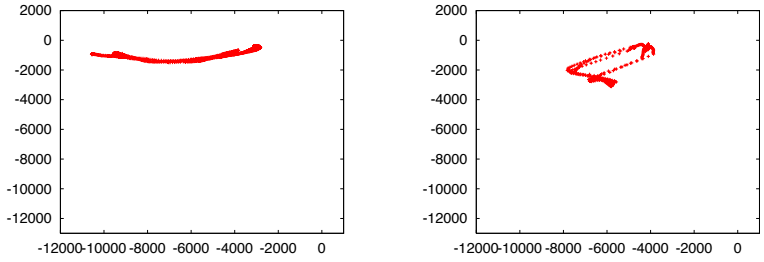


Fig. 10. Discovery of distributions in *Gaussians*.

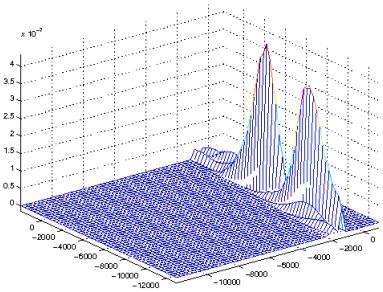


Query (moving hands)

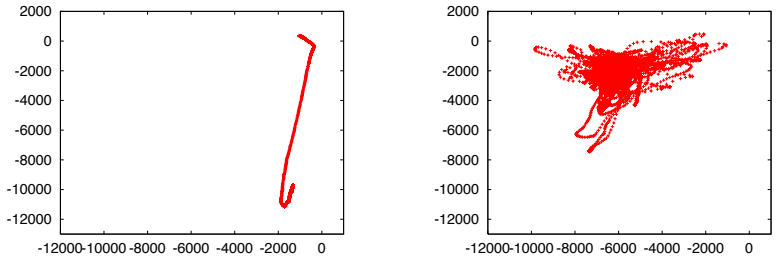


Distribution #1 (moving hands) *Distribution #2* (walking)

Fig. 11. Discovery of distributions in *MoCap*.



Query (balancing on one leg)



Distribution #1 (balancing on one leg) *Distribution #2* (boxing)

Fig. 12. Discovery of distributions in *MoCap*.

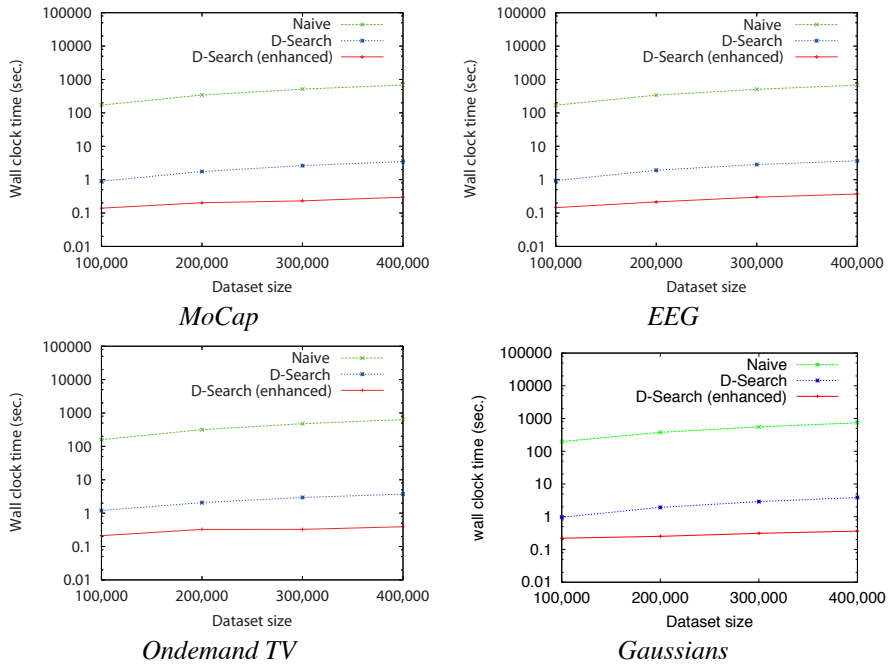


Fig. 13. Scalability: wall clock time vs. dataset size n (= number of distributions). *D-Search* can be up to 2,300 times faster than the naive implementation.

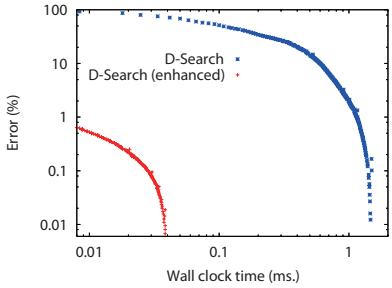


Fig. 14. Approximation quality: relative approximation error vs. wall clock time.

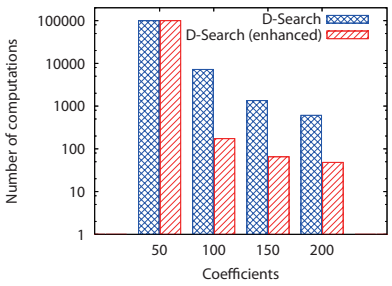


Fig. 15. Frequency of approximation use: number of computations vs. coefficients. Our method significantly reduces the search cost for each step of multi-step sequential scan.

5.4. Analysis of proposed algorithms

D-Search exploits multiple computations for the approximation of KL divergence. In this section we analyze the approximation quality of each granularity. We conducted this experiment with the *Ondemand TV* dataset.

Figure 14 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 re-

sults 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 15 shows how often each approximation was used in the basic version and the enhanced version for a dataset size of 100,000. We chose a starting coefficient $c_1 = 50$, and a step $h = 4$. As shown in the figure, most of the data sequences are excluded with the approximations of four steps ($c = \{50, 100, 150, 200\}$). Compared with $O(m)$, which the naive implementation requires, the approximation technique provides a dramatic reduction in computation time. The coarser approximation provides reasonable approximation quality, and its calculation speed is very high. On the other hand, although the approximation with higher granularity is not very fast compared to the coarser version, it offers good approximation quality. Accordingly, using approximations with various granularities offers significant advantages in terms of approximation quality and calculation speed. Our algorithms, and especially the enhanced version, can efficiently prune a large number of search candidates, which leads to a significant reduction in the search cost. Similar trends were observed in other experiments (*MoCap*, *EEG*, *Ondemand TV*, *MusicStore*, *Gaussians*).

6. 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 required by the naive solution, our solution needs only $O(n)$ time to compute the 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 capturing all similar distributions perfectly. Furthermore, our algorithms can be extended to time-series distribution mining and probability density functions. 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] C. C. Aggarwal. On classification and segmentation of massive audio data streams. *Knowl. Inf. Syst.*, 20(2):137–156, 2009.

[4] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of VLDB*, pages 81–92, Berlin, Germany, Sept. 2003.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] S. Dasgupta. Learning mixtures of gaussians. In *FOCS*, 1999.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[11] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

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

[13] V. Gandhi, J. M. Kang, S. Shekhar, J. Ju, E. D. Kolaczyk, and S. Gopal. Context inclusive function evaluation: a case study with em-based multi-scale multi-granular image classification. *Knowl. Inf. Syst.*, 21(2):231–247, 2009.

[14] 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.

[15] Z. Gong and Q. Liu. Improving keyword based web image search with visual feature distribution and term expansion. *Knowl. Inf. Syst.*, 21(1):113–132, 2009.

[16] 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.

[17] 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.

[18] 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.

[19] R. Kannan, H. Salmasian, and S. Vempala. The spectral method for general mixture models. In *18th Annual Conference on Learning Theory (COLT)*, pages 444–457, 2005.

[20] K.Pearson. Contributions to the Mathematical Theory of Evolution. *l. Trans. Royal Soc., 185A*, pages 71–110, 1894.

[21] M. H. C. Law, M. A. T. Figueiredo, and A. K. Jain. Simultaneous feature selection and clustering using mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1154–1166, 2004.

[22] 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.

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

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

[25] G. Pfurtscheller, D. Flotzinger, and C. Neuper. Differentiation between finger, toe and tongue movement in man based on 40 hz eeg. In *Electroencephalography and Clinical Neurophysiology*, pages 456–460, 1994.

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

[27] 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.

[28] 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.

[29] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *Proceedings of ACM SIGMOD*, pages 599–610, Baltimore, Maryland, June 2005.

[30] T. Shi, M. Belkin, and B. Yu. Data spectroscopy: learning mixture models using eigenspaces of convolution operators. In *ICML*, pages 936–943, 2008.

[31]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.

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

[33]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.

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

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

[36]W. L. Woon and K.-S. Wong. String alignment for automated document versioning. *Knowl. Inf. Syst.*, 18(3):293–309, 2009.

[37]Z. Zhang, B. T. Dai, and A. K. H. Tung. Estimating local optimums in em algorithm over gaussian mixture model. In *ICML*, pages 1240–1247, 2008.

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

Correspondence and offprint requests to: Yasuko Matsubara, Kyoto University, Email: y.matsubara@db.soc.i.kyoto-u.ac.jp