The VLDB Journal (2009) 18:809–835
DOI 10.1007/s00778-008-0128-8

REGULAR PAPER

Top-*k* typicality queries and efficient query answering methods on large databases

Ming Hua · Jian Pei · Ada W. C. Fu · Xuemin Lin · Ho-Fung Leung

Received: 15 October 2007 / Revised: 15 November 2008 / Accepted: 19 November 2008 / Published online: 15 January 2009

Abstract Finding typical instances is an effective approach to understand and analyze large data sets. In this paper, we apply the idea of typicality analysis from psychology and cognitive science to database query answering, and study the novel problem of answering top-*k* typicality queries. We model typicality in large data sets systematically. Three types of top-*k* typicality queries are formulated. To answer questions like "Who are the top-*k* most typical NBA players?", the measure of simple typicality is developed. To answer questions like "Who are the top-*k* most typical guards distinguishing guards from other players?", the notion of *discriminative typicality* is proposed. Moreover, to answer questions like "Who are the best *k* typical guards in whole representing different types of guards?", the notion of *representative typicality* is used. Computing the exact answer to a top-*k* typicality query requires quadratic time which is often too costly for online query answering on large databases. We develop a series of approximation methods for various situations: (1) the randomized tournament algorithm has linear complexity though it does not provide a theoretical guarantee on the quality of the answers; (2) the direct local typicality approximation using VP-trees provides an approximation quality guarantee; (3) a local typicality tree data structure can be exploited to index a large set of objects. Then, typicality queries can be answered efficiently with quality guarantees by a tournament method based on a Local Typicality Tree. An extensive performance study using two real data sets and a series of synthetic data sets clearly shows that top-*k* typicality queries are meaningful and our methods are practical.

Keywords Typicality analysis · Top-*k* query · Efficient query answering

This paper is a substantial extension of the conference version [32]. The research of Ming Hua and Jian Pei is supported in part by an NSERC Discovery grant and an NSERC Discovery Accelerator Supplements grant. The research of Xuemin Lin is supported in part by the Australian Research Council Discovery Grants DP0987557, DP0881035 and DP0666428 and a Google research award. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies. We are grateful to the anonymous reviewers and Dr. Peter Haas, the associate editor, for their constructive comments which help to improve the quality of the paper substantially. Particularly, we thank them for their advice in improving the mathematical exposition and their suggestions about some important related work.

M. Hua (✉) · J. Pei
Simon Fraser University, Burnaby, Canada
e-mail: mhua@cs.sfu.ca

J. Pei
e-mail: jpei@cs.sfu.ca

A. W. C. Fu · H.-F. Leung
The Chinese University of Hong Kong,
Shatin, NT, Hong Kong, China
e-mail: adafu@cse.cuhk.edu.hk

H.-F. Leung
e-mail: lhf@cse.cuhk.edu.hk

X. Lin
The University of New South Wales, Sydney, Australia
e-mail: lxue@cse.unsw.edu.au

X. Lin
NICTA, Sydney, Australia

1 Introduction

Learning from examples is an effective strategy extensively adopted in practice. For instance, how should one teach the concept of mammal? A helpful approach is to show some *typical* examples of mammals such as leopard and lions. Good

examples are often more effective than feature descriptions as the first step to understand a concept or a large set of objects.

How should good examples be chosen? First of all, those examples must be typical. Using platypuses (which lay eggs instead of giving birth to live young) as examples of mammals may mislead children in learning the concept. In addition, real examples are strongly preferred. Some virtual objects made up by assembling perfect features of the concept may not be easy to understand. More often than not, an ideal case may not be possible. For example, there does not exist a single bird having all the expected features of birds.

Typicality analysis has been studied extensively in psychology and cognitive science (see Sect. 3 for a brief review). Interestingly, the strategy of using typical examples to summarize and analyze a large set of objects can be applied to effective query answering. When a user asks a query which may return a large number of answers, instead of examining those answers one by one, the user may want to obtain a small number of typical answers for digesting.

1.1 Motivating examples

Jeff is a junior basketball player whose dream is to play in the NBA. As the NBA has more than 400 active players, they are quite diverse. Jeff may want to know some representative examples of NBA players. Top-k typicality queries can help.

1.1.1 Top-k simple typicality queries

Jeff asks, "Who are the top-3 most typical NBA players?" We can imagine that, in the space of technical statistics, there exists a likelihood function of being NBA players. The player who has the maximal likelihood of being NBA players is the most typical. This leads to our first typicality measure—the *simple typicality*. A top-k simple typicality query finds the k most typical objects in a set of objects.

1.1.2 Top-k discriminative typicality queries

Jeff is particularly interested in becoming a guard. "Who are the top-3 most typical guards distinguishing guards from other players?" Simple typicality on the set of guards is insufficient to answer the question, since it is possible that a typical guard may also be typical among other players. Instead, players that are typical among all guards but are not typical among all non-guard players should be found.

In order to address this demand, we need the notion of *discriminative typicality*, which measures how an object is typical in one set but not typical in the other set. Given a set of objects and a target subset, a top-k discriminative typicality query finds the k objects with the highest discriminative typicality values.

1.1.3 Top-k representative typicality queries

NBA guards may still have some sub-groups. For example, the fresh guards and the experienced guards, as well as the shooting guards and the point guards. Jeff wants to learn different types of guards, without a clear idea about what types there are. So he asks,"Who are the best 3 typical guards in whole representing different types of guards?"

Simple typicality does not provide the correct answer to this question, since the 3 players with the greatest simple typicality may be quite similar to each other, while some other popular players different from those three may be missed. Discriminative typicality does not help either, because the exact types of guards and their members are unknown.

To solve this problem, we develop the notion of *representative typicality* on a set of objects, which measures how an object is typical among a set of objects different from the already reported typical objects. Given a set of objects, a top-k representative typicality query finds a set of k objects with the highest representative typicality scores.

1.1.4 Efficient query answering

Top-k typicality queries can be used to provide good examples for effective learning and understanding. The above scenarios may happen in many other applications, such as students learning concepts, medical analysts investigating medical cases, and city administration analyzing traffic accidents and crimes. In our empirical study (Sect. 8), we will illustrate the effectiveness of typicality queries using two real data sets.

As more and more data has been accumulated, top-k typicality queries may be applied on large data sets. It is practically desirable that the queries can be answered online. However, we will show that, with a proper notion of top-k typicality, computing the exact answer to a top-k typicality query needs quadratic time which is often too costly for online query answering on large data sets. On the other hand, typical enough examples are often sufficient for understanding and analyzing large data sets. Thus, there are practical demands to have approximation methods to answer top-k typicality queries.

1.2 Our contributions

In this paper, we formulate and tackle the problem of efficiently answering top-k typicality queries, and make the following contributions.

First, we extend the idea of typicality analysis from psychology and cognitive science to database query answering and search, and identify the novel problem of top-k typicality query answering on large databases. We develop three effective measures for typicality.

Second, as computing exact answers to top-k typicality queries on large databases can be costly, we develop a series of approximation query answering algorithms for various situations: (1) the randomized tournament algorithm has linear complexity though it does not provide a theoretical guarantee on the quality of the answers; (2) the direct local typicality approximation using VP-trees provides an approximation quality guarantee; (3) a local typicality tree (LT-tree) can be exploited to index a large set of objects. Then, typicality queries can be answered efficiently with quality guarantees by an LT-tree tournament method.

Last, we conduct an extensive performance study on both real data sets and synthetic data sets to verify the effectiveness of top-k typicality queries and the efficiency of our query evaluation methods.

1.2.1 How is this study different from others?

To the best of our knowledge, we are the first to systematically study how to use typicality in ranking query answers from large databases. We also address the efficient query answering issue.

Top-k queries (also known as ranking queries) have been heavily employed in many applications, such as searching web databases, similarity search, recommendation systems, etc. According to a user-specified preference function, a top-k query returns the best k results. Many previous studies investigated efficient approaches to answering top-k queries using aggregate scoring functions. Such an aggregate function maps an object to a certain preference score. The score depends on only the attribute values of the object, and is often independent from other objects in the database.

Top-k typicality queries are a special type of top-k queries, however, the typicality of an object depends on the distribution of the other objects in question. This poses a major challenge for efficient top-k typicality query answering. To the best of our knowledge, how to answer top-k queries efficiently using such "relative" scoring functions has not been studied well.

The rest of the paper is organized as follows. We propose the typicality measures in Sect. 2 and systematically review the related work in Sect. 3. For top-k simple typicality queries, we develop a randomized tournament algorithm in Sect. 4, and the local typicality approximation methods in Sect. 5. We discuss how to answer top-k discriminative and representative typicality queries in Sects. 6 and 7, respectively. We report a systematic performance study in Sect. 8. The paper is concluded in Sect. 9.

2 Top-K typicality queries

In this section, we define the three types of top-k typicality queries that will be addressed in this paper.

By default, we consider a set of objects S on attributes $A_1, \ldots, A_n$. Let $A_{i_1}, \ldots, A_{i_l}$ be the attributes on which the typicality queries are applied ($1 \leq i_j \leq n$ for $1 \leq j \leq l$) and $d_{A_{i_1},\ldots,A_{i_l}}(x, y)$ be the distance between two objects x and y in S on attributes $A_{i_1}, \ldots, A_{i_l}$. When $A_{i_1}, \ldots, A_{i_l}$ are clear from context, $d_{A_{i_1},\ldots,A_{i_l}}(x, y)$ is abbreviated to $d(x, y)$.

We address the top-k typicality problem in a generic metric space. Therefore, the distance metric d should satisfy the triangle inequality. The definitions and frequently used notations are summarized in Table 1.

2.1 Simple typicality

In a set of objects S, which object is the most typical? To answer this query, we introduce simple typicality.

By intuition and as also suggested by the previous research in psychology and cognitive science (as will be reviewed in Sect. 3), an object o in S is more typical than the others if o is more likely to appear in S. Conceptually, the set of objects S on attributes $A_1, \ldots, A_n$ can be viewed as a set of independent and identically distributed samples of an n-dimensional random vector $\mathcal{Z}$ that takes values in the Cartesian product space $D = D_{A_1} \times \cdots \times D_{A_n}$, where D_{A_i} is the domain of attribute A_i ($1 \leq i \leq n$). The likelihood of $o \in S$, given that o is a sample of $\mathcal{Z}$, can be used to measure the typicality of o.

Definition 1 (*Simple typicality*) Given a set of objects S on attributes $A_1, \ldots, A_n$ and a subset of attributes $A_{i_1}, \ldots, A_{i_l}$ ($1 \leq i_j \leq n$ for $1 \leq j \leq l$) of interest, let $\mathcal{Z}$ be the n-dimensional random vector generating the samples S, the *simple typicality* of an object $o \in S$ with respect to $\mathcal{Z}$ on attributes $A_{i_1}, \ldots, A_{i_l}$ is defined as $T_{A_{i_1},\ldots,A_{i_l}}(o, \mathcal{Z}) = L_{A_{i_1},\ldots,A_{i_l}}(o|\mathcal{Z})$ where $L_{A_{i_1},\ldots,A_{i_l}}(o|\mathcal{Z})$ is the likelihood [23] of o on attributes $A_{i_1}, \ldots, A_{i_l}$, given that o is a sample of $\mathcal{Z}$.

In practice, since the distribution of random vector $\mathcal{Z}$ is often unknown, we use $T_{A_{i_1},\ldots,A_{i_l}}(o, S) = L_{A_{i_1},\ldots,A_{i_l}}(o|S)$ as an estimator of $T_{A_{i_1},\ldots,A_{i_l}}(o, \mathcal{Z})$, where $L_{A_{i_1},\ldots,A_{i_l}}(o|S)$ is the posterior probability of an object o on attributes $A_{i_1}, \ldots, A_{i_l}$ given S [23].

$L_{A_{i_1},\ldots,A_{i_l}}(o|S)$ can be computed using density estimation methods. In this paper, we adopt the commonly used kernel density estimation method, which does not require any distribution assumption on S. The general idea is to use a kernel function to approximate the probability density around each observed sample. More details will be discussed in Sect. 4.1.

Hereafter, unless specified otherwise, the simple typicality measure refers to the estimator $T_{A_{i_1},\ldots,A_{i_l}}(o, S)$. Moreover, for the sake of simplicity, when $A_{i_1}, \ldots, A_{i_l}$ are clear from context, $T_{A_{i_1},\ldots,A_{i_l}}(o, S)$ and $L_{A_{i_1},\ldots,A_{i_l}}(o|S)$ are abbreviated to $T(o, S)$ and $L(o|S)$, respectively.

Given a set of objects S on attributes $A_{i_1}, \ldots, A_{i_l}$ of interest, a predicate P and a positive integer k, a *top-k simple typicality query* returns, from the set of tuples in S satisfying

Table 1 Summary of definitions and frequently used notations

Notation	Description
$o(=(a_1,\ldots,a_n))$	An object on attributes $A_1,\ldots,A_n$ ($o.A_i = a_i$ for $1 \le i \le n$)
$D = D_{A_1} \times \cdots \times D_{A_n}$	The Cartesian product space (D_{A_i} is the domain of attribute A_i for $1 \le i \le n$)
$\mathcal{Z}$ and S	A random vector $\mathcal{Z}$ and a set of objects S that are independent and identically distributed samples of $\mathcal{Z}$
$L(o\|\mathcal{Z})$	The likelihood [23] of an object o given that it is a sample of random vector $\mathcal{Z}$
$L(o\|S)$	The likelihood [23] of an object o given a set of objects S
$T(o,\mathcal{Z}) = L(o\|\mathcal{Z})$	The *simple typicality* of object o (Definition 1)
$T(o,S) = L(o\|S)$	An estimator of simple typicality $T(o,\mathcal{Z})$
$DT(o,\mathcal{U},\mathcal{V}) = T(o,\mathcal{U}) - T(o,\mathcal{V})$	the *discriminative typicality* of o in $C \subset S$, where $\mathcal{U}$ and $\mathcal{V}$ are the random vectors that generate the samples C and $S-C$, respectively (Definition 2)
$DT(o,C,S-C) = T(o,C) - T(o,S-C)$	An estimator of discriminative typicality $DT(o,\mathcal{U},\mathcal{V})$
$D(o,A) = \{x\|x \in D, d(x,o) = \min_{y\in A} d(x,y)\}$	The *representing region* of $o \in A$ (Definition 3)
$N(o,A,S) = \{x\|x \in S \cap D(o,A)\}$	The objects in S that lie in the representing region of $o \in A$
$GT(A,\mathcal{Z}) = \sum_{o\in A} T(o,\mathcal{Z}_{D(o,A)}) Pr(D(o,A))$	The *group typicality* of a set of objects $A \subset S$, where $T(o,\mathcal{Z}_{D(o,A)})$ is the simple typicality of o with respect to $\mathcal{Z}$ in $D(o,A)$ (Definition 4)
$GT(A,S) = \sum_{o\in A} T(o,N(o,A,S)) \frac{\|N(o,A,S)\|}{\|S\|}$	An estimator of group typicality $GT(A,\mathcal{Z})$
$RT(o,A,\mathcal{Z}) = GT(A\cup\{o\},\mathcal{Z}) - GT(A,\mathcal{Z})$	The *representative typicality* of o with respect to the reported answer set $A \subset S$ (Definition 5)
$RT(o,A,S) = GT(A\cup\{o\},S) - GT(A,S)$	An estimator of representative typicality $RT(o,A,\mathcal{Z})$

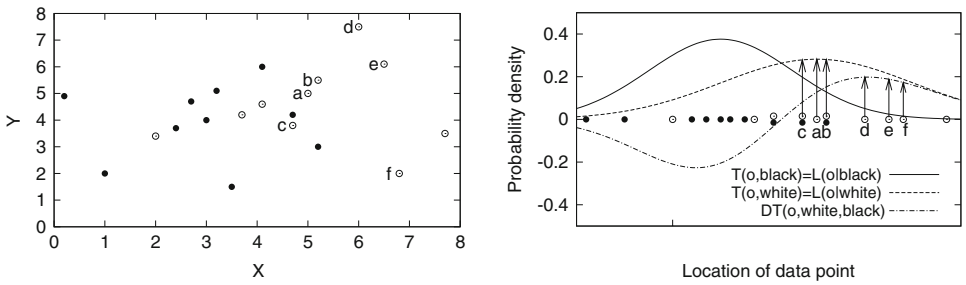


Fig. 1 The simple typicality and discriminative typicality curves of a set of points

(a) A set of points. **(b)** Typicality.

predicate P, the k tuples having the largest simple typicality values that are computed on attributes $A_{i_1},\ldots,A_{i_l}$.

Example 1 (*Top-k simple typicality queries*) Consider the set of points in Fig. 1a. A top-3 simple typicality query on attribute X with predicate $COLOR = white$ returns the 3 white points having the largest simple typicality values computed on attribute X.

Figure 1b projects the points in T to attribute X. The likelihood function of the white points and that of the black points on attribute X are labeled as $L(o|white)$ and $L(o|black)$ in the figure, respectively, while we will discuss how to compute the likelihood values in Sect. 4.1. Points a, b and c have the highest likelihood values among all white points, and thus should be returned as the answer to the query.

2.2 Discriminative typicality

Simple typicality works when a set of objects are considered independently from other objects not in the set. In some scenarios, a user may have multiple classes of objects, such as the guards and all other NBA players as illustrated in Sect. 1. To understand the discriminative features of one class against the others, one may want to find the objects that are typical in the target class, but not in the others.

Given a set of objects S and a target subset C, which object is the most typical in C but not in $(S-C)$? We use the discriminative typicality to answer such a question. By intuition, an object $o \in C$ is typical and discriminative in C if the difference between its typicality in C and that in $(S-C)$ is large.

Definition 2 (*Discriminative typicality*) Given a set of objects S on attributes $A_1,\ldots,A_n$ and a *target subset* $C \subset S$, let $\mathcal{U}$ and $\mathcal{V}$ be the n-dimensional random vectors generating the samples C and $S-C$, respectively, the *discriminative typicality* of an object $o \in C$ on attributes $A_{i_1},\ldots,A_{i_l}$ ($1 \le i_j \le n$ for $1 \le j \le l$) is $DT(o,\mathcal{U},\mathcal{V}) = T(o,\mathcal{U}) - T(o,\mathcal{V})$, where $T(o,\mathcal{U})$ and $T(o,\mathcal{V})$ are the simple typicality values of object o with respect to $\mathcal{U}$ and $\mathcal{V}$, respectively.

In the definition, the discriminative typicality of an object is defined as the difference of its simple typicality in the target set and that in the rest of the data set. One may wonder whether using the ratio $\frac{T(o,\mathcal{U})}{T(o,\mathcal{V})}$ may also be meaningful. Unfortunately, such a ratio-based definition may not choose a typical object that has a large simple typicality value with respect to $\mathcal{U}$. Consider an extreme example. Let o be an object that is very atypical with respect to $\mathcal{U}$ and has a typicality value of nearly 0 with respect to $\mathcal{V}$. Then, o still has an infinite ratio $\frac{T(o,\mathcal{U})}{T(o,\mathcal{V})}$. Although o is discriminative between $\mathcal{U}$ and $\mathcal{V}$, it is not typical with respect to $\mathcal{U}$ at all.

Due to the unknown distribution of random vectors $\mathcal{U}$ and $\mathcal{V}$, we use $DT(o, C, S-C) = T(o, C) - T(o, S-C)$ to estimate $DT(o, \mathcal{U}, \mathcal{V})$, where $T(o, C)$ and $T(o, S-C)$ are the estimators of $T(o, \mathcal{U})$ and $T(o, \mathcal{V})$, respectively.

Given a set of objects S on attributes $A_{i_1}, \ldots, A_{i_l}$ of interest, a predicate P and a positive integer k, a *top-k discriminative typicality query* treats the set of tuples in S satisfying P as the target subset, and returns the k tuples in the target subset having the largest discriminative typicality values computed on attributes $A_{i_1}, \ldots, A_{i_l}$.

Example 2 (*Top-k discriminative typicality queries*) Consider the set of points in Fig. 1a again and a top-3 discriminative typicality query on attribute X with predicate $COLOR = white$.

The discriminative typicality $DT(o, white, black)$ for each object $o \in white$ is plotted in the figure, where $white$ and $black$ denote the set of white points and black points, respectively. To see the difference between discriminative typicality and simple typicality, consider objects a, b and c, which have large simple typicality values among all white points. However, they also have relatively high simple typicality values as a member in the subset of black points comparing to other white points. Therefore, they are not discriminative. Points $\{d, e, f\}$ are the answer to the query, since they are discriminative.

2.3 Representative typicality

The answer to a top-k simple typicality query may contain some similar objects, since the objects with similar attribute values may have similar simple typicality scores. However, in some situations, it is redundant to report many similar objects. Instead, a user may want to explore the data set by viewing typical objects that are different from each other but jointly represent the whole data set well. For example, as illustrated in Sect. 1, a user may want to get a small subset of guards representing the whole set of guards nicely. Reporting similar typical objects as simple typicality queries does not meet the requirement.

Suppose a subset $A \subset S$ is chosen to represent S. Each object in $(S - A)$ is best represented by the closest object in A. For each $o \in A$, we define the representing region of o.

Definition 3 (*Representing region*) Given a set of objects S on attributes $A_1, \ldots, A_n$ and a subset $A \subset S$, let $D = D_{A_1} \times \cdots \times D_{A_n}$ where D_{A_i} is the domain of attribute A_i $(1 \le i \le n)$, the *representing region* of an object $o \in A$ is $D(o, A) = \{x | x \in D, d(x, o) = \min_{y \in A} d(x, y)\}$, where $d(x, y)$ is the distance between objects x and y.

To make A representative as a whole, the representing region of each object o in A should be fairly large and o should be typical in its own representing region.

Definition 4 (*Group typicality*) Given a set of objects S on attributes $A_1, \ldots, A_n$ and a subset $A \subset S$, let $\mathcal{Z}$ be the n-dimensional random vector generating the samples S, the *group typicality* of A on attributes $A_{i_1}, \ldots, A_{i_l}$ $(1 \le i_j \le n, 1 \le j \le l)$ is $GT(A, \mathcal{Z}) = \sum_{o \in A} T(o, \mathcal{Z}_{D(o,A)}) \cdot Pr(D(o, A))$, where $T(o, \mathcal{Z}_{D(o,A)})$ is the simple typicality of o with respect to $\mathcal{Z}$ in o's representing region $D(o, A)$ and $Pr(D(o, A))$ is the probability of $D(o, A)$.

Since the distribution of $\mathcal{Z}$ is unknown, we can estimate the group typicality $GT(A, \mathcal{Z})$ as follows. For any object $o \in A$, let $N(o, A, S) = \{x | x \in S \cap D(o, A)\}$ be the set of objects in S that lie in $D(o, A)$, $Pr(D(o, A))$ can be estimated using $\frac{|N(o,A,S)|}{|S|}$. The group typicality $GT(A, \mathcal{Z})$ is estimated by $GT(A, S) = \sum_{o \in A} T(o, N(o, A, S)) \cdot \frac{|N(o,A,S)|}{|S|}$, where $T(o, N(o, A, S))$ is the estimator of simple typicality $T(o, \mathcal{Z}_{D(o,A)})$, since $N(o, A, S)$ can be viewed as a set of independent and identically distributed samples of $\mathcal{Z}$ that lie in $D(o, A)$.

The group typicality score measures how representative a group of objects is. The *size-k most typical group problem* is to find k objects as a group such that the group has the maximum group typicality. Unfortunately, the problem is NP-hard, since it has the discrete k-median problem as a special case, which was shown to be NP-hard [15].

Moreover, top-k queries are generally expected to have the monotonicity in answer sets. That is, the result of a top-k query is contained in the result of a top-k' query where $k < k'$. However, an object in the most typical group of size k may not be in the most typical group of size k' $(k < k')$. For example, in the data set illustrated in Fig. 2, the size-1 most typical group is $\{A\}$ and the size-2 most typical group is $\{B, C\}$, which does not contain the size-1 most typical group. Therefore, the size-k most typical group is not suitable to define the top-k representative typicality.

To make representative typicality practical on large data sets, we adopt a greedy approach.

Definition 5 (*Representative typicality*) Given a set of objects S and a *reported answer set* $A \subset S$, let $\mathcal{Z}$ be the

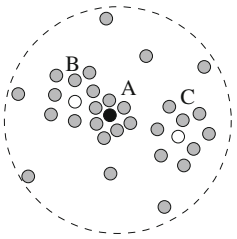


Fig. 2 Non-monotonicity of size-k most typical group

random vector with respect to samples S, the *representative typicality* of an object $o \in (S - A)$ is $RT(o, A, \mathcal{Z}) = GT(A \cup \{o\}, \mathcal{Z}) - GT(A, \mathcal{Z})$, where $GT(A \cup \{o\}, \mathcal{Z})$ and $GT(A, \mathcal{Z})$ are the group typicality values of subsets $A \cup \{o\}$ and A, respectively.

In practice, we use $RT(o, A, S) = GT(A \cup \{o\}, S) - GT(A, S)$ to estimate $RT(o, A, \mathcal{Z})$, where $GT(A, S)$ and $GT(A \cup \{o\}, S)$ are the estimators of $GT(A, \mathcal{Z})$ and $GT(A \cup \{o\}, \mathcal{Z})$, respectively.

Given a set of objects S on attributes $A_{i_1}, \ldots, A_{i_l}$ of interest, a predicate P and a positive integer k, a *top-k representative typicality query* returns k objects $o_1, \ldots, o_k$ from the set of tuples in S satisfying predicate P, such that o_1 is the object having the largest simple typicality, and, for $i > 1$,

$$o_i = \arg \max_{o \in S - \{o_1, \ldots, o_{i-1}\}} RT(o, \{o_1, \ldots, o_{i-1}\}, S).$$

The representative typicality values are computed on attributes $A_{i_1}, \ldots, A_{i_l}$.

Example 3 (*Top-k representative typicality queries*) Consider the set of points in Fig. 3a and a top-2 representative typicality query on attribute X with predicate $COLOR = white$.

We project the white points to attribute X and plot the simple typicality scores of the white points, as shown in Fig. 3b. Points a and c have the highest simple typicality scores. However, if we only report a and c, then the dense region around a is reported twice, but the dense region around b is missed. A top-2 representative typicality query will return a and b as the answer.

2.4 Comparison with other analysis

Typicality is different from the following existing notions of representative objects.

2.4.1 Medians, means and typical objects

In many previous studies, medians and means are often used to represent the aggregate of a set of objects. The mean of a set of points is the geometric center of the set, while the median is the point in the set that is closest to the mean. Can medians and means approximate typical objects well?

Consider a set of points in Fig. 4. Points A and C are the median and the mean of the set, respectively, and point B (the white point) is the most typical point. Clearly, the most typical point B is quite different from the median and the mean in this example.

As the geometric centers, medians and means are not necessarily related to probability distribution which is captured by the typical objects. In Sect. 8.1, we will use real data sets to further elaborate the differences between medians, means, and typical objects.

2.4.2 Clustering and typicality analysis

Clustering analysis partitions a set of objects into subsets so that the objects in each subset share similar features. It is different from typicality analysis in the following two aspects. First, clustering analysis focuses on grouping objects into clusters. The relationship among objects in a cluster is often not explored. Typicality analysis finds the most representative objects, but does not deal with grouping of objects.

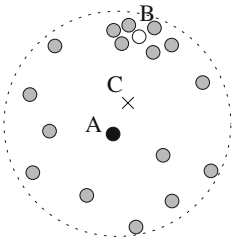
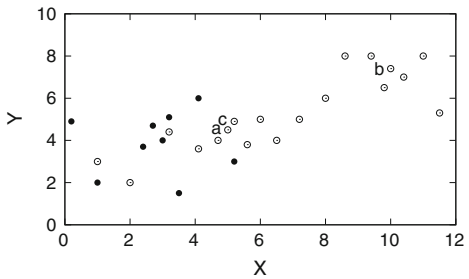
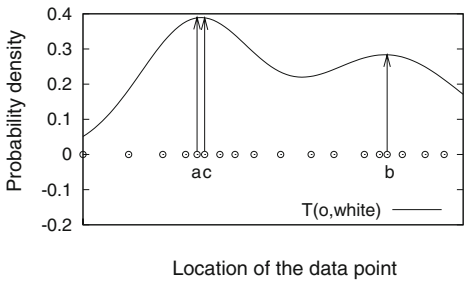


Fig. 4 Medians, means and typical objects



(a) A set of points.



(b) Representative typicality.

Fig. 3 The answer to a top-2 representative typicality query on a set of points

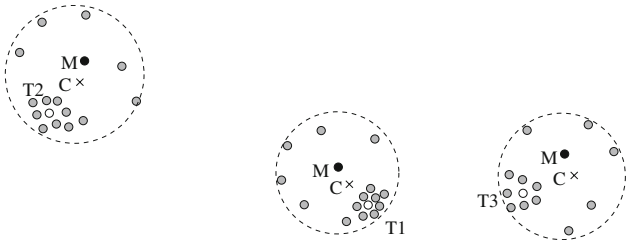


Fig. 5 Cluster centroids and typical objects

Second, although the centroid in a cluster sometimes may be used to represent the cluster, it may not be representative in general. More detailed comparisons between clustering and the typicality analysis can be found in Sect. 3.4.

Figure 5 shows a set of points with three clusters. The clusters are computed based on the distance between points: two or more points belong to the same cluster if their pairwise distances are smaller than their distances to the points in other clusters. The mean and the median of the points in each cluster are labeled C and M, respectively. Medians are often considered as cluster centroids. The answer of the top-3 most representative typicality query contains white points T_1, T_2 and T_3, which are quite different from the means and medians. Moreover, the typical objects T_1, T_2 and T_3 are more similar (with shorter distance) to the objects around them than medians.

3 Related Work

3.1 Typicality in psychology and cognitive science

Typicality of objects has been widely discussed in psychology and cognitive science [22,48]. People judge some objects to be "better examples" of a concept than others. This is known as the *graded structure* [42] of a category. Generally, the graded structure is a continuum of category representativeness, beginning with the most typical members of a category and continuing through less typical members to its atypical members.

There are several determinants of graded structure. One determinant is the *central tendency* [6] of a category. Central tendency is either one or several very representative exemplar(s), either existing in the category or not. An exemplar's similarities to the central tendency determine its typicality in this category. Another determinant of typicality is the *stimulus similarity* [41]. Generally, the more similar an instance is to the other members of its category, and the less similar it is to members of the contrast categories, the higher the typicality rating it has.

The prototype view [43] suggests that a concept be represented by a prototype, such that objects "closer to" or "more similar to" the prototype are considered to be better examples of the associated concept. The exemplar view [12] is an alternative to the prototype view that proposes using real objects as exemplars instead of abstract prototypes that might not exist in real life. Finally, the schema view [16] improves the prototype view by modeling concepts in schema theory and artificial intelligence knowledge representation.

Although typicality has not been used before in query answering on large databases, the idea of typicality was recently introduced into ontology design and conceptual modeling [5], which are generally related to database design.
How is our study related? Our typicality measures are in the general spirit of typicality measures used in psychology and cognitive science. As suggested by the previous studies in psychology and cognitive science, typicality measures may vary in different applications. In our study, we propose simple typicality, discriminative typicality, and representative typicality for different application requirements.

Studies on typicality in psychology and cognitive science often do not address the concerns about efficient query answering from large databases. Complementary to those studies, we focus on efficient query answering.

3.2 Top-k (ranking) queries

There are numerous query answering algorithms proposed for top-k queries in the literature. The threshold algorithm (TA) [39] is one of the best algorithms. TA first sorts the values in each attribute and then scans the sorted lists in parallel. A "stopping value" is maintained, which acts as a threshold to prune the tuples in the rest of the lists if they cannot have better scores than the threshold. Several variants of TA have also been proposed, such as [25]. The recent development and extension of top-k query answering include using views to answer top-k queries efficiently [19], removing redundancy in top-k patterns [52], applying multidimensional analysis in top-k queries [53], continuous monitoring of top-k queries over a sliding window [38], and so forth.
How is our study related? Our top-k typicality queries provide a list of objects with the highest typicality scores according to the proposed typicality measures. This is similar to the existing ranking queries in general.

However, our scoring functions (typicality measures) are different from most of the scoring functions studied before in top-k queries. The typicality score for each object is based on its relationship with the other objects in question. This is much more complicated than scoring an individual object based on its own attributes independently, like most other scoring functions do. Therefore, some classic and efficient query answering algorithms for top-k queries cannot be directly applied to answer top-k typicality queries. We have to develop new efficient query answering algorithms.

3.3 The (discrete) k-median problem

Finding typical objects is broadly related to the k-median problem in computational geometry. Given a set S of n points, the *k-median problem* is to find a set M of k points minimizing the sum of distances from all points in S to M. Points in M are called the medians of S. Under the constraint that points in M belong to S, it is known as the *discrete k-median problem*. When $k = 1$, we can find the exact median in $O(n^2)$ time. When k is an arbitrary input parameter, the discrete k median problem on any distance metric is NP-hard [15].

Several approximation algorithms have been proposed to compute the approximate 1-median efficiently. [8] proposes a quad-tree based data structure to support finding the approximate median with a constant approximation ratio in $O(n \log n)$ time. A randomized algorithm is proposed in [14], which computes the approximate median in linear time. Although the approximation ratio cannot be bounded, it performs well in practice. [33] provides a $(1 + \delta)$-approximation algorithm with runtime $O(n/\delta^5)$ based on sufficiently large sampling. [7] proposes an algorithm to solve the median problem in L_1 metric in $O(n \log n)$ time.

How is our study related? The top-k simple typicality query and the discrete k-median problem both want to find the objects in a data set optimizing the scores with respect to their relationship to other objects. However, as will be clear in Sect. 4.1, the functions to optimize are different. The methods of the discrete k-median problem cannot be applied directly to answer top-k typicality queries.

Moreover, in discrete k-median problem, there is no ranking among the k median objects. The top-k representative typicality queries as defined will return k objects in an order.

3.4 Clustering analysis

Clustering analysis partitions a set of data objects into smaller sets of similar objects. [54] is a nice survey of various clustering methods.

The clustering methods can be divided into the following categories. The *partitioning methods* partition the objects into k clusters and optimize some selected partitioning criterion, where k is a user specified parameter. K-means [29], K-medoids [36] and CLARANS [40] are examples of this category. The *hierarchical methods* perform a series of partitions and group data objects into a tree of clusters. BIRCH [56], CURE [27] and Chameleon [35] are examples of hierarchical methods. The *density-based methods* use a local cluster criterion and find the regions in the data space that are dense and separated from other data objects by regions with lower density as clusters. The examples of density-based methods include DBSCAN [24], OPTICS [3] and DENCLUE [30]. The *grid-based methods* use multi-resolution grid data structures and form clusters by finding dense grid cells. STING [51] and CLIQUE [2] are examples of grid-based methods.

How is our study related? Typicality analysis and clustering analysis both consider similarity among objects. However, the two problems have different objectives. Clustering analysis focuses on partitioning data objects, while typicality analysis aims to find representative objects.

In some studies, cluster centroids are used to represent the whole clusters. However, in general the centroid of a cluster may not be a representative point. For example, medians are often considered as cluster centroids in partitioning clustering methods, but they are not the most typical objects as shown in Sect. 2.4.

In the density-based clustering method DBSCAN [24], the concept of "core point" is used to represent the point with high density. For a core point o, there are at least $MinPts$ points lying within a radius Eps from o, where $MinPts$ and Eps are user input parameters. However, "core points" are significantly different from "typical points" in the following two aspects.

First, a "core point" may not be typical. Consider an extreme case where there are two groups of points: the first group of points lie close to each other with a size much larger than $MinPts$, while the second group only contain $MinPts$ points lying within a radius Eps from a point o that are far away from the points in the first group. then, o is a core point but it is not typical at all. Second, a typical point may not be a "core point", either. It is possible that a typical point does not have $MinPts$ points lying within a distance Eps from it, but it still has a high typicality score. A comparison between clustering and typicality analysis on real data sets is given in Sect. 8.1.

It is possible to extend the existing clustering methods to answer typicality queries, by defining the most typical object in a cluster as the centroid and using the maximal group typicality of clusters as the clustering criteria, which is in the same spirit as our typicality query evaluation algorithms.

3.5 Other related models

Typicality probability [13,26] in statistical discriminant analysis is defined as the Mahalanobis distance between an object and the centroid of a specified group, which provides an absolute measure of the degree of membership to the specified group.

Spatially-decaying aggregation [17,18] is defined as the aggregation values influenced by the distance between data items. Generally, the contribution of a data item to the aggregation value at certain location decays as its distance to that location increases. Nearly linear time algorithms are proposed to compute the ϵ-approximate aggregation values when the metric space is defined on a graph or on the Euclidean plane.

How is our study related? Discriminant analysis mainly focuses on how to correctly classify the objects. It does not consider the typicality of group members. Our definition of discriminative typicality combines both the discriminability and the typicality of the group members, which is more powerful in capturing the "important" instances in multi-class data sets. Moreover, [13,26] do not discuss how to answer those queries efficiently on large data sets.

Spatially decaying sum with *exponential decay function* [17,18] is similar to our definition of simple typicality. However, in [17,18], the spatially decaying aggregation problem is defined on graphs or Euclidean planes, while we assume only a generic metric space. The efficiency in [17,18] may not be carried forward to the more general metric space. The techniques developed in this paper may be useful to compute spatially-decaying aggregation on a general metric space. When typicality queries are computed on graphs or Euclidean planes, some ideas in [17,18] may be borrowed.

4 Answering simple typicality queries

In this section, we first discuss how to compute likelihood values, then, we show that answering top-k typicality queries is quadratic in nature. Last, we present a randomized tournament approximation algorithm (RT). The approximation algorithm developed for simple typicality computation in this section can be extended to answer top-k discriminative typicality queries and top-k representative typicality queries, as will be discussed later in Sects. 6 and 7, respectively.

4.1 Likelihood computation

For an object o in S, likelihood $L(o|S)$ is the posterior probability of o given data S, which can be computed using probability density estimation methods. There are several model estimation techniques in the literature [21], including parametric and non-parametric density estimation. Parametric density estimation requires a certain distribution assumption, while non-parametric estimation does not. Among the various techniques proposed for non-parametric density estimation [20], histogram estimation [34], kernel estimation [1,11] and nearest neighbor estimation [37] are the most popular. In this paper, we use kernel estimation, because it can estimate unknown data distributions effectively [28].

Kernel estimation is a generalization of sampling. In random sampling, each sample point carries a unit weight. However, an observation of the sample point increases the chance of observing other points nearby. Therefore, kernel estimator distributes the weight of each point in the nearby space around according to a *kernel function* K. The commonly used kernel functions are listed in Table 2, where $I(|u| \leq 1)$ in the kernel functions denotes the value 1 when $|u| \leq 1$ holds, and 0 when $|u| \leq 1$ does not hold.

Table 2 The commonly used kernel functions

Name	Kernel function
Uniform	$K(u) = \frac{1}{2} I(\|u\| \leq 1)$
Triangle	$K(u) = (1 - \|u\|) I(\|u\| \leq 1)$
Epanechnikov	$K(u) = \frac{3}{4}(1 - u^2) I(\|u\| \leq 1)$
Quartic	$K(u) = \frac{15}{16}(1 - u^2)^2 I(\|u\| \leq 1)$
Triweight	$K(u) = \frac{35}{32}(1 - u^2)^3 I(\|u\| \leq 1)$
Gaussian	$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$
Cosine	$K(u) = \frac{\pi}{4} \cos(\frac{\pi}{2} u) I(\|u\| \leq 1)$

A bandwidth parameter (also known as the smoothing parameter) h is introduced to control the distribution among the neighborhood of the sample. As shown in [47], the quality of the kernel estimation depends mostly on the bandwidth h and lightly on the choice of the kernel K. Too small bandwidth values cause very spiky curves, and too large bandwidth values smooth out details. A class of effective methods are data-driven least-squares cross-validation algorithms [9,44–46], which select the bandwidth value that minimizes integrated square error.

In this paper, we choose the commonly used Gaussian kernels. Our approach can also be adapted using other kernel functions. We set the bandwidth of the Gaussian kernel estimator $h = \frac{1.06s}{\sqrt[5]{n}}$ as suggested in [46], where n is the size of the data set and s is the standard deviation of the data set which can be estimated by sampling. In Sect. 8.3, we evaluate the sensitivity of the answers to top-k typicality queries with respect to the choice of kernel functions and bandwidth values. The results show that the answers computed using different kernel functions listed in Table 2 are mostly consistent. Moreover, using different bandwidth values around $h = \frac{1.06s}{\sqrt[5]{n}}$ also provide consistent answers.

Outliers in data sets may increase the standard deviation of the data sets, and thus lead to larger bandwidth values, which may impair the quality of the answers to typicality queries. Therefore, for better performance, we can remove outliers in data sets as preprocessing. There are extensive studies on effective and efficient outlier detection [31,49,50], which can be used as a screening step in our methods. Moreover, it is shown from the experimental results in Sect. 8.3 that, even on data sets containing a non-trivial amount of noise, the results returned by top-k typicality queries are often consistent with the results found when outliers are removed.

Since we address the top-k typicality problem in a generic metric space, the only parameter we use in density estimation is the distance (or similarity) between two objects. Formally, given a set of objects $S = (o_1, o_2, \ldots, o_n)$ in a generic metric space, the underlying likelihood function is approximated as

$$L(x|S) = \frac{1}{n} \sum_{i=1}^{n} G_h(x, o_i) = \frac{1}{n\sqrt{2\pi}} \sum_{i=1}^{n} \mathrm{e}^{-\frac{d(x,o_i)^2}{2h^2}} \quad (1)$$

Algorithm 1 ExactTyp(S,k)

Input: a set of objects $S = \{o_1, \ldots, o_n\}$ and positive integer k
Output: the k objects with the highest simple typicality values
Method:
1: **for all** object $o \in S$ **do**
2: set $T(o, S) = 0$
3: **end for**
4: **for** $i = 1$ to n **do**
5: **for** $j = i + 1$ to n **do**
6: $w = \frac{1}{n\sqrt{2\pi}} e^{-\frac{d(o_i,o_j)^2}{2h^2}}$
7: $T(o_i, S) = T(o_i, S) + w$
8: $T(o_j, S) = T(o_j, S) + w$
9: **end for**
10: **end for**
11: return the top-k objects according to $T(o, S)$

where $d(x, o_i)$ is the distance between x and o_i in the metric space, and $G_h(x, o_i) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{d(x,o_i)^2}{2h^2}}$ is a *Gaussian kernel*.

4.2 An exact algorithm and complexity

Hereafter, by default, we assume that outliers are removed using the techniques discussed in Sect. 4.1.

Theoretically, given a set of objects S, if the likelihood of an object $o \in S$ satisfies $L(o|S) \propto \frac{1}{\sum_{o' \in S} d(o,o')}$, then the discrete 1-median problem can be reduced to a special case of the top-1 simple typicality query problem. As so far no better than quadratic algorithm has been found for exact solutions to the general discrete 1-median problem (except in L_1 metric space), it is challenging to find a better than quadratic algorithm for computing exact answers to general top-k typicality queries.

Algorithm 1 is a straightforward quadratic method that computes the exact answer to a top-k simple typicality query. It computes the exact simple typicality for each object using two nested loops, and then selects the top-k objects. Quadratic algorithms are often too costly for online queries on large databases, while good approximations of exact answers are good enough for typicality analysis. This motivates our development of approximation algorithms.

4.3 A randomized tournament algorithm

Inspired by the randomized tournament method [14] for the discrete 1-median problem, we propose a randomized tournament algorithm for answering top-k simple typicality queries as follows.

Let t be a small integer, called the *tournament group size*. To find the most typical object in a set S of n objects, we partition the objects into $\lceil \frac{n}{t} \rceil$ groups randomly such that each group has t objects. For each group, we use the exact algorithm to find the object that has the largest simple typicality value in the group. Only the winner objects in the groups are sent to the next round.

Algorithm 2 RandomTyp(S,k,t,v)

Input: a data set S, positive integer k, tournament size t and number of validations v
Output: approximation to the answer to a top-k simple typicality query $\widetilde{A}$
Method:
1: $\widetilde{A} = \emptyset$
2: **for** $i = 1$ to k **do**
3: $S' = S - \widetilde{A}$
4: $candidate = \emptyset$
5: **for** $j = 1$ to v **do**
6: **repeat**
7: $G = \{g_i\}\ (1 \le i \le \lceil \frac{n}{t} \rceil, |g_i| = t, \bigcup_{g_i \in G} g_i = S')$
8: **for all** group $g \in G$ **do**
9: $winner_g = ExactTyp(g, 1)$
10: $S' = S' - g \cup winner_g$
11: **end for**
12: **until** $|S'| = 1$
13: $candidate = candidate \cup S'$
14: **end for**
15: $\widetilde{A} = \widetilde{A} \cup \{\arg\max_{o \in candidate}\{T(o, S)\}\}$
16: **end for**

The winners of the previous round are again partitioned randomly into groups such that each group contains t objects. The most typical object in each group is selected and sent to the next round. The tournament continues until only one object is selected as the winner. The final winner is an approximation of the most typical object.

To approximate the second most typical object, we run the randomized tournament again with the following constraint: the most typical object already chosen in the previous tournament cannot be selected as the winner in this tournament. The final winner in the second tournament is the approximation of the second most typical object. Continuing in this manner, we can find an approximation to the set of top-k typical objects by running a total of k tournaments.

In order to achieve a higher accuracy, we can run this randomized tournament several times for selecting the approximation of the i-th most typical object ($1 \le i \le k$), and pick the object with the largest simple typicality among all the final winners. The procedure is given in Algorithm 2.

The typicality computation within one group has the time complexity of $O(t^2)$. There are $\lceil \log_t n \rceil$ tournament rounds in total. Without loss of generality, let us assume $n = t^m$. Then, the first round has $\frac{n}{t}$ groups, the second round has $\frac{\frac{n}{t}}{t} = \frac{n}{t^2}$ groups, and so forth. The total number of groups is $\sum_{1 \le i \le \log_t n} \frac{n}{t^i} = \frac{n}{t-1}(1 - \frac{1}{t^m}) = O(\frac{n}{t})$. The complexity of selecting the final winner is $O(t^2 \cdot \frac{n}{t}) = O(tn)$. If we run each tournament v times for better accuracy, and run tournaments to choose top-k typical objects, the overall complexity is $O(kvtn)$.

The randomized algorithm runs in linear time with respect to the number of objects. However, the accuracy of the approximation to the answer is not guaranteed in theory, though in practice it often has reasonable performance.

5 Local typicality approximation

While the randomized tournament method is efficient, it cannot guarantee the approximation quality. Can we provide some quality guarantee and at the same time largely retain the efficiency? In this section, we develop several heuristic local typicality approximation methods. Our discussion in this section is for simple typicality. The methods will be extended to other typicality measures later in the paper.

5.1 Locality of typicality approximation

In Gaussian kernel estimation, given two points a and p, the contribution from p to $T(a, S)$, the simple typicality score of a, is $\frac{1}{n\sqrt{2\pi}}e^{-\frac{d(a,p)^2}{2h^2}}$, where n is the size of the data set. The contribution of p decays exponentially as the distance between a and p increases. Therefore, if p is remote from a, p contributes very little to the simple typicality score of a.

Moreover, in a metric space, given three points a, b and p, the triangle inequality $|d(a, p)-d(b, p)| < d(a, b)$ holds. If $d(a, p) \gg d(a, b)$, then $d(a, p) \approx d(b, p)$. Therefore, the objects far away from a and b will have similar contributions to the probability density values $T(a, S)$ and $T(b, S)$.

Based on the above observations, given a set of objects S and a subset $C \subseteq S$, can we use the locality to approximate the object having the largest simple typicality value in C?

Definition 6 (*Neighborhood region*) Given a set of objects S, a neighborhood threshold σ, and a subset $C \subseteq S$, let $D = D_{A_1} \times \cdots \times D_{A_n}$ where D_{A_i} is the domain of attribute A_i $(1 \leq i \leq n)$, the *σ-neighborhood region* of C is defined as $D(C, \sigma) = \{o | o \in D, \min_{o' \in C}\{d(o, o')\} \leq \sigma\}$.

Definition 7 (*Local simple typicality*) Given a set of objects S, a neighborhood threshold σ, and a subset $C \subseteq S$, let $\mathcal{Z}$ be the random vector that generates samples S, the *local simple typicality* of an object $o \in C$ is defined as $LT(o, C, \mathcal{Z}, \sigma) = L(o|\mathcal{Z}_{D(C,\sigma)})$ where $L(o|\mathcal{Z}_{D(C,\sigma)})$ is the likelihood of o given that it is a sample of $\mathcal{Z}$ in region $D(C, \sigma)$.

In practice, for each object $o \in S$, we use the set of objects in S that lie in o's σ-neighborhood region to estimate the simple typicality of o.

Definition 8 (*Local neighborhood*) Given a set of objects S, a neighborhood threshold σ, and a subset $C \subseteq S$, The *σ-neighborhood* of C is defined as $LN(C, S, \sigma) = \{o | o \in S \cap D(C, \sigma)\}$, where $D(C, \sigma)$ is the σ-neighborhood region of C.

$LN(C, S, \sigma)$ is the set of objects in S whose distance to at least one object in C is at most σ. Then, $LT(o, C, \mathcal{Z}, \sigma)$ can be estimated using $LT(o, C, S, \sigma) = L(o|LN(o, C, \sigma))$, where $L(o|LN(o, C, \sigma)$ is the likelihood of o given objects $LN(o, C, \sigma)$.

The following result uses local simple typicality to approximate the simple typicality with a quality guarantee. The proof can be found in Appendix A.

Theorem 1 (Local typicality approximation) *Given a set of objects S, neighborhood threshold σ, and a subset $C \subseteq S$, let $\widetilde{o} = \arg\max_{o_1 \in C}\{LT(o_1, C, S, \sigma)\}$ be the object in C having the largest local simple typicality value, and $o = \arg\max_{o_2 \in C}\{T(o_2, S)\}$ be the object in C having the largest simple typicality value. Then,*

$$T(o, S) - T(\widetilde{o}, S) \leq \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}} \tag{2}$$

Moreover, for any object $x \in C$,

$$T(x, S) - LT(x, C, S, \sigma) < \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}} \leq \frac{1}{\sqrt{2\pi}} \tag{3}$$

From Theorem 1, we can also derive the minimum neighborhood threshold value σ to satisfy certain approximation quality.

Corollary 1 (Choosing neighborhood threshold) *Given a set of objects S, an object $x \in S$, and a quality requirement θ $(\theta < \frac{1}{\sqrt{2\pi}})$, if $\sigma \geq \sqrt{-2\ln\sqrt{2\pi}\theta} \cdot h$, then $T(x, S) - LT(x, C, S, \sigma) < \theta$ for any subset C $(x \in C)$.*

The proof of Corollary 1 is given in Appendix B.

5.2 DLTA: direct local typicality approximation using VP-trees

Inequality 3 in Theorem 1 can be used immediately to approximate simple typicality computation with quality guarantee. Given a neighborhood threshold σ, for each object $x \in S$, the direct local typicality approximation (DLTA) algorithm computes the σ-neighborhood of $\{x\}$, i.e., $LN(\{x\}, S, \sigma)$ and the local simple typicality $LT(x, \{x\}, S, \sigma)$. Then, it returns the k objects with the highest local simple typicality values as the approximation to the answer of the top-k simple typicality query.

The quality of the approximation answer is guaranteed by the following theorem, whose proof is given in Appendix C.

Theorem 2 (Approximation quality) *Given a set of objects S, neighborhood threshold σ and integer k, let A be the k objects with the highest simple typicality values, and $\widetilde{A}$ be*

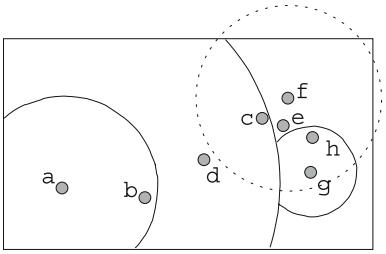


Fig. 6 Decomposing a set of objects in a VP-tree

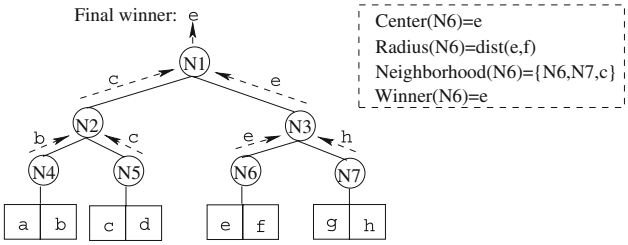


Fig. 7 Computing the approximate most typical point

the k objects with the highest local simple typicality values. Then,

$$\frac{\sum_{o\in A} T(o,S) - \sum_{\widetilde{o}\in\widetilde{A}} T(\widetilde{o},S)}{k} < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \tag{4}$$

Searching the σ-neighborhood for each object can be very costly. To implement the direct local typicality approximation efficiently, we can use the VP-tree index [55] to support σ-neighborhood searches effectively.

A VP-tree [55] is a binary space partitioning (BSP) tree. Given a set of objects S, a VP-tree T indexes the objects in S. Each node in a VP-tree represents a subset of S. Roughly speaking, for each non-leaf node N and the set of nodes S_N at N, a *vantage point* is selected to divide the set S_N into two exclusive subsets S_{N_1} and S_{N_2} ($S_N = S_{N_1} \cup S_{N_2}$) such that, to search the objects within distance σ to an object $p \in N$, likely we only need to search either S_{N_1} or S_{N_2} but not both. S_{N_1} and S_{N_2} are used to construct the two children of N. For example, the VP-tree in Fig. 7 indexes the objects in Fig. 6.

A VP-tree can be constructed top-down starting from the root which represents the whole set of objects. A sampling method is given in [55] to select vantage points for internal nodes. Then, the first half subset of objects that are close to the vantage point form the left child of the root, and the other objects form the right child. The left and the right children are further divided recursively until a node contains only one object (a leaf node). A VP-tree can be constructed in cost $O(|S| \log |S|)$.

Searching a VP-tree for the σ-neighborhood of a query point is straightforward using the recursive tree search. Once an internal node in the tree can be determined in the σ-neighborhood of the query point, all descendant objects of the internal node are in the neighborhood and no subtrees need to be searched. For example, the σ-neighborhood of node $N_6 = \{e, f\}$ in Fig. 6 is represented by the dashed circle. To find all points in the σ-neighborhood of N_6, we search the VP-tree in Fig. 7 from the root node N_1, and recursively examine each internal node. During the search, node N_4 can be pruned since all points in N_4 lie out of the σ-neighborhood of N_6.

The cost of computing the local simple typicality of an object x is $O(|LN(\{x\}, S, \sigma)|)$. Then, computing the local simple typicality of all objects in S takes $O(\sum_{x\in S} |LN(\{x\}, S, \sigma)|)$ time. Although the search can be sped up using a VP-tree, the complexity of the DLTA algorithm is still $O(|S|^2)$. The reason is, in the worst case where σ is larger than the diameter (i.e., the largest pairwise distance) of the data set, the σ-neighborhood of each object contains all other objects in S.

5.3 LT3: local typicality approximation using tournaments

To reduce the cost of computing local simple typicality further, we incorporate the tournament mechanism, and propose a local typicality approximation algorithm using tournaments (the LT3 algorithm). The basic idea is to group the objects locally, and conduct a tournament in each local group of objects. The object with the largest local simple typicality is selected as the winner. The winners are sent to the next round of tournament. The tournaments terminate when only one object is left as the winner. A sampling method is employed to reduce the computational cost.

5.3.1 Local typicality trees (LT-trees)

A local typicality tree (LT-tree) is an MVP-tree [10] with auxiliary information that supports local typicality calculation and tournaments. Given a set of objects S, an LT-tree can be constructed as follows.

First, we construct an MVP-tree [10] on S, which is a t-nary VP-tree that uses more than one vantage point to partition the space. Without loss of generality, let us assume $t = 2^l$ and the data set contains t^m objects. We assign a layer number to each node in the MVP-tree. The root node has layer number 0, and a node is assigned layer number $(i + 1)$ if its parent has layer number i. We remove all those nodes in the MVP-tree whose layer number is not a multiple of t. For a node N of layer number jt ($j \geq 1$), we connect N to its ancestor in the MVP-tree of layer $(j - 1)t$.

Second, we compute three pieces of information, the approximate center, the radius, and the σ-neighborhood, for each node in the LT-tree.

Approximate center. For a node N in the LT-tree, let S_N be the set of objects at N. To compute the approximate center at a node N in the LT-tree, we draw a sample R of $\sqrt{|S_N|}$

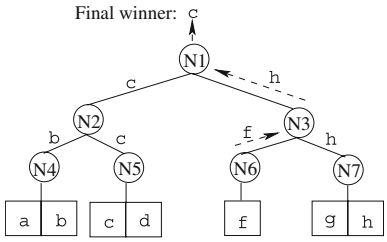


Fig. 8 Computing the approximate second most typical point

objects from S_N, and compute the pairwise distance between every two objects in R. Then, for each object $x \in R$, the center-score of x is the maximum distance from x to another point in R. The object in R of the minimum center-score is chosen as the center. This center approximation procedure is popularly used in computational geometry. It takes $O(|S_N|)$ time for each node N, and $O(|S| \log_t |S|)$ time for all nodes in the LT-tree.

Radius. Once the center c of a node N is chosen, the radius is given by the maximum distance between c and the other objects at N. This can be computed in time $O(|S_N|)$ for each node N, and $O(|S| \log_t |S|)$ for all nodes in the LT-tree.

σ-neighborhood. We use a range query in the LT-tree to compute a superset of the σ-neighborhood of S_N for every node N in the LT-tree, which always achieves a typicality approximation no worse than using the σ-neighborhood. To compute the superset, we start from the root and iteratively search for the nodes that completely lie in the σ-neighborhood of N, using the approximate center and radius of N. Once all objects at a node N' are in the σ-neighborhood of N, we use N' to represent them and do not search any subtrees of N'.

5.3.2 Query answering

To answer a top-k simple typicality query, we run tournaments on the LT-tree bottom-up. First, a tournament is run for each leaf node in the LT-tree. The winner enters the tournament at the parent node. The winner o_1 at the root node is the approximation of the most typical object. Figure 7 illustrates the procedure of computing the approximate most typical point in the set of points in Fig. 6 using an LT-tree. During the tournaments in the leaf nodes, $\{b, c, e, h\}$ are selected as local winners and sent to the parent nodes. Then, $\{c, e\}$ are selected in the tournaments in nodes N_2 and N_3. Finally, e is selected as the winner, which approximates the most typical point in the data set.

To find the approximation of the second most typical object, we do not need to completely run the tournaments again. Instead, we can reuse most of the results in the tournaments of finding the most typical object w_1. The only tournaments we need to rerun are on the nodes containing w_1.

Algorithm 3 LT3Typ(S,k,T)

Input: a set of n objects $S = \{o_1, \ldots, o_n\}$ and positive integer k and an LT-tree T (with m levels) built on S whose root node is N_R
Output: approximation to the answer to a top-k simple typicality query $\widetilde{A}$
Method:
1: $\widetilde{A} = \emptyset$
2: **for** $j = m$ to 0 **do**
3: **for all** node N at level L_j **do**
4: $winner_N = \arg\max_{o \in N}\{LT(o, N, S, \sigma)\}$
5: $N_p = N_p \cup \{winner_N\}$ {$*N_p$ is the parent node of N}
6: **end for**
7: **end for**
8: $\widetilde{A} = \widetilde{A} \cup \{winner_{N_R}\}$
9: $w_1 = winner_{N_R}$
10: **for** $i = 2$ to k **do**
11: find N such that $w_{i-1} \in N$ {$*w_{i-1}$ is the last output winner}
12: **while** $N \neq N_R$ **do**
13: $winner_N = \arg\max_{o \in N, o \neq w_{i-1}}\{LT(o, N, S, \sigma)\}$
14: $N_p = N_p \cup \{winner_N\}$
15: $N \leftarrow N_p$
16: **end while**
17: $\widetilde{A} = \widetilde{A} \cup \{winner_{N_R}\}$
18: $w_i = winner_{N_R}$
19: **end for**

First, we run a new tournament at the leaf node N containing w_1, but do not include w_1 in the new tournament. Then, the winner w_1' is sent to N_p, the parent of N, and a new tournament is run there by replacing w_1 by w_1'. A series of m tournaments are needed to find a new winner w_2 in the root node, which is the approximation of the second most typical object. At each level of the LT-tree, only one node needs to run a tournament. For example, in the LT-tree in Fig. 8, after e is selected as the first final winner, it is removed from node N_6. Then, only the tournaments in nodes N_6, N_3 and N_1 need to be re-conducted. Finally, c is selected as the final winner to approximate the second most typical point. The complete procedure is shown in Algorithm 3.

The quality of the answers returned by the LT3 algorithm can be guaranteed by the following theorem, whose proof is given in Appendix D.

Theorem 3 *In a set S, let o be the object with the largest simple typicality and $\widetilde{o}$ be an object computed by the LT3 method using local typicality approximation and the tournament group size t. Then,*

$$T(o, S) - T(\widetilde{o}, S) < \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}\right) \cdot \lceil \log_t |S| \rceil$$

The LT3 algorithm combines the merits of both local typicality approximation and the tournament mechanism. It achieves better accuracy than the randomized tournament algorithm, thanks to the local grouping. It is more efficient than the DLTA algorithm because of the tournament mechanism. As shown in our experiments, the approximations of the most typical objects computed by the LT3 algorithm are very close to the exact ones. LT3 is very efficient and scalable.

5.3.3 A sampling method for bounding runtime

To make the analysis complete, here we provide a sampling method which provides an upper bound on the cost of local typicality computation with quality guarantee.

Suppose we want to compute the local simple typicality $LT(p, C, S, \sigma)$. We consider the contribution of an object $o \in LN(C, S, \sigma)$ to $LT(p, C, S, \sigma)$, denoted by

$$\eta(o) = \frac{1}{|LN(C, S, \sigma)|} G_h(p, o) = \frac{e^{-\frac{d(p,o)^2}{2h^2}}}{|LN(C, S, \sigma)|\sqrt{2\pi}}$$

We can draw a sample of $LN(C, S, \sigma)$ to estimate the expectation of $\eta(o)$. Please note that

$$LT(p, C, S, \sigma) = |LN(C, S, \sigma)| \cdot E[\eta(o)]$$

where $E[\eta(o)]$ is the expectation of $\eta(o)$ for $o \in LN(C, S, \sigma)$.

Using a special form of the Chernoff–Hoeffding bound [4], we have the following result.

Theorem 4 *For any* δ $(0 < \delta < 1)$ *and* ϵ $(\epsilon > 0)$ *and a sample* R *of* $LN(C, S, \sigma)$, *if* $|R| > \frac{3\sqrt{2\pi} \cdot e^{\frac{\sigma^2}{2h^2}} \cdot \ln \frac{2}{\delta}}{\epsilon^2}$, *then*

$$Pr\left\{\left|\frac{|LN(C, S, \sigma)|}{|R|} \sum_{o \in R} \eta(o) - LT(p, C, S, \sigma)\right| > \epsilon \cdot LT(p, C, S, \sigma)\right\} < \delta$$

Theorem 4 provides an upper bound of the sample size, which is independent of the size of data sets. The larger ϵ and δ, the smaller the sample size. The larger σ, the larger the sample size.

Using the sampling method suggested by Theorem 4, we can have a tournament algorithm using an LT-tree of cost $O(n \log n)$. The algorithm provides a theoretical bound on the runtime.

However, the sampling method cannot be practically gainful unless on extremely large data sets. The LT-tree already exploits the locality of objects nicely. When the data set is not extremely large, the number of objects in the σ-neighborhood of a node is usually (substantially) smaller than the number of samples required for high approximation quality. In our experiments, the above case is always true. Thus, we do not include the experimental results on this sampling method.

6 Answering discriminative typicality queries

Discriminative typicality can be calculated as follows. For each object o in the target subset $C \subset S$, Algorithm 1 can be used to compute the simple typicality scores of o in C and $S - C$, respectively. The difference between the two is the discriminative typicality of o.

Suppose there are m objects in the target subset C. To compute the discriminative typicality score of an object $o \in C$, we have to compute the simple typicality scores of o in both C and $S - C$, which takes $O(n)$ time, where n is the size of the whole data set S. Therefore, answering top-k discriminative typicality queries using the exact algorithm takes time $O(mn)$.

The approximation methods developed for top-k simple typicality queries can also be adopted to answer top-k discriminative typicality queries.

6.1 A randomized tournament algorithm

Generally, the randomized tournament algorithm can be used to answer top-k discriminative typicality queries if the discriminative typicality measure is applied. The difference is that only the objects in the target subset C are involved in the tournament. The other objects are only used to compute the approximate discriminative typicality scores of the objects in C.

The cost of discriminative typicality computation within one group is $O(\frac{m}{n}t^2)$. Since there are $O(\frac{n}{t})$ groups in total, the complexity of selecting the final winner is $O(mt)$. If we run each tournament v times for better accuracy, then the overall complexity of answering a top-k discriminative typicality query is $O(kvtm)$.

6.2 Local typicality approximation

Similar to the idea in Sect. 5.1, we can define the local discriminative typicality as follows.

Definition 9 (*Local discriminative typicality*) Given a set of objects S on attributes $A_1, \ldots, A_n$, a neighborhood threshold σ, and a target subset $C \subset S$, let $\mathcal{U}$ and $\mathcal{V}$ be the random vectors generating C and $S - C$, respectively, the *local discriminative typicality* of an object $o \in C$ on attributes $A_{i_1}, \ldots, A_{i_l}$ is defined as $LDT(o, \mathcal{U}, \mathcal{V}, \sigma) = LT(o, \{o\}, \mathcal{U}, \sigma) - LT(o, \{o\}, \mathcal{V}, \sigma)$, where $LT(o, \{o\}, \mathcal{U}, \sigma)$ and $LT(o, \{o\}, \mathcal{V}, \sigma)$ are the local simple typicality values of o in $\mathcal{U}$ and $\mathcal{V}$, respectively.

$LDT(o, \mathcal{U}, \mathcal{V}, \sigma)$ can be estimated using $LDT(o, C, S - C, \sigma) = LT(o, \{o\}, C, \sigma) - LT(o, \{o\}, S - C, \sigma)$, where $LT(o, \{o\}, C, \sigma)$ and $LT(o, \{o\}, S - C, \sigma)$ are the estimators of local simple typicality values of o in C and $S - C$, respectively.

Similar to Theorem 1, we have the following quality guarantee of local discriminative typicality approximation.

Theorem 5 (Local discriminative typicality approximation) *Given a set of objects* S, *a neighborhood threshold* σ, *and a*

target subset $C \subseteq S$, let $\widetilde{o} = \arg\max_{o_1 \in C}\{LDT(o_1, C, S, \sigma)\}$ be the object in C having the largest local discriminative typicality value, and $o = \arg\max_{o_2 \in C}\{DT(o, C, S)\}$ be the object in C having the largest discriminative typicality value. Then,

$$DT(o, C, S - C) - DT(\widetilde{o}, C, S - C) < \frac{2}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \quad (5)$$

Moreover, for any $x \in C$,

$$|DT(x, C, S - C) - LDT(x, C, S - C, \sigma)| < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \quad (6)$$

6.2.1 DLTA: direct local typicality approximation

Theorem 5 can be directly used to approximate discriminative typicality. Given a neighborhood threshold σ, for each object $x \in C$, we compute the σ-neighborhood of $\{x\}$ in C and $S - C$, respectively, and thus its local discriminative typicality. Searching the σ-neighborhood can also be done using a VP-tree, as described in Sect. 5.2.

The cost of computing the local discriminative typicality of an object $x \in C$ is $O(|LN(\{x\}, S, \sigma)|)$. The overall cost of computing the local discriminative typicality of all objects in the target subset C is $O(\sum_{x \in C} |LN(\{x\}, S, \sigma)|)$. As analyzed in Sect. 5.2, the σ-neighborhood of an object may contain the whole data set in the worst case. Thus, the time complexity of the direct local typicality approximation method for discriminative typicality is $O(mn)$, where m is the size of the target subset C, and n is the size of the whole data set. However, data is often distributed as clusters in practice, and the number of objects contained in the σ-neighborhood of each object is often small.

6.2.2 LT3: local typicality approximation using tournaments

The local typicality approximation method using tournaments (LT3) method can be extended to answer top-k discriminative typicality queries, which follows the same framework as answering top-k simple typicality queries. The only difference is that, in the tournament in each node, local discriminative typicality is computed, instead of local simple typicality.

Similar to Theorem 3, we have the following guarantee of the quality of answering top-k discriminative typicality queries using the LT3 method.

Theorem 6 *In a set S, let o be the object of the largest discriminative typicality and $\widetilde{o}$ be an object computed by the LT3 method using local discriminative typicality approximation with the tournament group size t. Then,*

$$DT(o, C, S - C) - DT(\widetilde{o}, C, S - C) < \frac{2}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \cdot \lceil \log_t |S| \rceil$$

Sampling method introduced in Sect. 5.3.3 can also be used to bound the runtime of the LT3 algorithm for discriminative typicality approximation.

7 Answering representative typicality queries

In this section, we first propose a straightforward approach to find the exact answer of a top-k representative typicality query. Then, we will discuss how to extend the approximation techniques proposed for simple typicality queries to efficiently answer top-k representative typicality queries.

7.1 An exact algorithm and complexity

When the answer set A is empty, the most representatively typical object is simply the most typical object o_1, which can be computed using Algorithm 1. After o_1 is added to A, the group typicality $GT(A, S)$ is the simple typicality score $T(o_1, S)$, since all members in S are represented by o_1.

Then, in order to compute the next object with the maximal representative typicality score, according to Definition 5, we have to compute the group typicality score $G(A \cup \{o\}, S)$ for each object $o \in (S - A)$ and select the object with the maximal score.

To compute $GT(A \cup \{o\}, S)$, we first construct $N(o, A, S)$ for object o as follows. We scan all objects in $(S - A)$. For each object $x \in (S - A)$, suppose $x \in N(o', A, S)$ for an object $o' \in A$, which means that o' is the object closest to x in A. If $d(o, x) < d(o', x)$, then x is removed from $N(o', A, S)$ and is added to $N(o, A, S)$. To make the computation efficient, we maintain the minimum distance from an object $x \in (S - A)$ to the objects in A by a 1-dimensional array. The minimum distances are updated every time after a new object is added into A.

Then, $T(o, N(o, A, S))$, the simple typicality of o in $N(o, A, S)$, is computed using Algorithm 1. Probability $Pr(N(o, A, S))$ is $\frac{|N(o,A,S)|}{|S|}$. For other objects $o' \in A$, since $N(o', A, S)$ may be changed, the simple typicality scores $T(o', N(o', A, S))$ and $Pr(N(o', A, S))$ are updated accordingly. Last, $GT(A \cup \{o\}, S)$ can be calculated according to Definition 4.

The above procedure is repeated to find the next most representatively typical object, until k objects are found.

The complexity of the exact algorithm is $O(kn^2)$ because each time after an object is added to A, the representative typicality scores of all objects in $(S - A)$ need to be recomputed

to find the next object with the largest representative typicality score.

7.2 A randomized tournament method

A top-k representative typicality query can be answered using the randomized tournament method.

At the beginning, the answer set A is empty, so the randomized tournament method works exactly the same as finding the most typical object using the randomized tournament method as described in Sect. 4.3. The winner object of the tournament is added to A.

To compute the ith ($i > 1$) object with the highest approximate representative typicality score, a randomized tournament is conducted from bottom up, similar to finding the first answer in A. The only difference is that the representative typicality score of each object in each group is computed, instead of the simple typicality score. The object with the maximal representative typicality score in each group is the winner and is sent to the next round of tournament. The final winner is an approximation of the ith most representatively typical object, and is added to A.

A top-k representative typicality query can be answered by k randomized tournaments. To ensure a higher accuracy, we can run each tournament several times, and pick the winner object with the highest representative typicality score on the whole data set.

The complexity of the randomized tournament to find the ith object ($i \leq k$) with the highest representative typicality score is $O(vtn)$, where v is the number of times the tournament is run, t is the group size, and n is the size of the data set. This is because, finding the object with the highest representative typicality score in each group takes $O(t^2)$ time, and there are $O(\frac{n}{t})$ groups in total. To answer a top-k representative typicality query, k randomized tournaments need to be conducted. Therefore, the overall complexity is $O(kvtn)$.

7.3 Local typicality approximation methods

The locality property in simple typicality approximation can be extended to address the representative typicality approximation.

Let A be the current reported answer set. The local group typicality of A is computed by only considering the object in the σ-neighborhood of $o \in A$. The intuition is, if an object is not in the σ-neighborhood of o, then the contribution from o to this object is small and can be ignored.

Definition 10 (*Local group typicality*) Given a set of objects S, a neighborhood threshold σ and a subset $A \subset S$, let $\mathcal{Z}$ be the random vector that generates the samples S, the *local group typicality* of A is

$$LGT(A, \mathcal{Z}, \sigma) = \sum_{o \in A} LT(o, \{o\}, \mathcal{Z}_{D(o,A)}, \sigma) \cdot Pr(N)$$

where $LT(o, \{o\}, \mathcal{Z}_{D(o,A)}, \sigma)$ is the local simple typicality of o in its representing region $D(o, A)$ and $N = D(o, A) \cap D(\{o\}, \sigma)$ is the σ-neighborhood region of o in its representing region $D(o, A)$.

Definition 11 (*Local representative typicality*) Given a set of objects S, a neighborhood threshold σ and a *reported answer set* $A \subset S$, let $\mathcal{Z}$ be the random vector generating the samples S, the *local representative typicality* of an object $o \in (S - A)$ is $LRT(o, A, \mathcal{Z}, \sigma) = LGT(A \cup \{o\}, \mathcal{Z}, \sigma) - LGT(A, \mathcal{Z}, \sigma)$.

For any object $o \in A$, let $N(o, A, S) = \{x | x \in S \cap D(o, A)\}$ be the set of objects in S that lie in $D(o, A)$, then $LN(\{o\}, N(o, A, S), \sigma)$ is the σ-neighborhood of o in $N(o, A, S)$. The local group typicality $LGT(A, \mathcal{Z}, \sigma)$ can be estimated using $LGT(A, S, \sigma) = \sum_{o \in A} LT(o, \{o\}, N(o, A, S), \sigma) \cdot \frac{|LN(\{o\}, N(o,A,S),\sigma)|}{|S|}$. Hence, the local representative typicality is estimated by $LRT(o, A, S, \sigma) = LGT(A \cup \{o\}, S, \sigma) - LGT(A, S, \sigma)$.

Local representative typicality can approximate representative typicality with good quality, as shown below. The proof can be found in Appendix E.

Theorem 7 (Local representative typicality approximation) *Given a set of objects S, a neighborhood threshold σ, and an answer set $A \subset S$, let $\widetilde{o} = \arg\max_{o_1 \in (S-A)}\{LRT(o_1, A, S, \sigma)\}$ be the object in $(S - A)$ having the largest local representative typicality value, and $o = \arg\max_{o_2 \in (S-A)}\{RT(o, A, S)\}$ be the object in $(S - A)$ having the largest representative typicality value. Then,*

$$RT(o, A, S) - RT(\widetilde{o}, A, S) < \frac{2}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \tag{7}$$

Moreover, for any $x \in (S - A)$,

$$|RT(x, A, S) - LRT(x, A, S, \sigma)| < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \tag{8}$$

7.3.1 DLTA: direct local typicality approximation

Direct local representative typicality approximation (DLTA) follows the similar framework of the exact algorithm described in Sect. 7.1. The only difference is that the local representative typicality score instead of the representative typicality score is computed.

To compute the local representative typicality score of an object o given answer set A, one critical step is to compute the local group typicality score $LGT(A \cup \{o\}, S, \sigma)$. The computation is similar to the exact algorithm elaborated in Sect. 7.1, except that the local simple typicality instead of the simple typicality of o in $N(o, A, S)$ is used to compute $LGT(A \cup \{o\}, S, \sigma)$.

Suppose the current reported answer A_i ($0 \leq i < k$, $A_0 = \emptyset$) contains the first i answers to a top-k representative typicality query, computing the local simple typicality of an object $o \in (S - A_i)$ takes $O(|LN(\{o\}, N(o, A, S), \sigma)|)$, where $LN(\{o\}, N(o, A, S), \sigma)$ is the σ-neighborhood of o in the set of its represented members $N(o, A, S)$. Thus, the complexity of computing the $(i + 1)$-th answer is $O(\sum_{o \in (S-A_i)} |LN(\{o\}, N(o, A, S), \sigma)|)$.

The overall complexity of answering a top-k representative typicality query is $O(\sum_{i=0}^{k-1} \sum_{o \in (S-A_i)} |LN(\{o\}, N(o, A, S), \sigma)|)$. In the worst case, the local neighborhood of any object o may contain the whole data set. Moreover, A_i contains i objects, so $|S - A_i| = n - i$. Therefore, the overall complexity of the DLTA algorithm is $O(\sum_{i=0}^{k-1}((n-i) \cdot n)) = O(n \cdot \frac{(2n-k-1)k}{2}) = O(kn^2)$.

7.3.2 LT3: local typicality approximation using tournaments

The LT3 algorithm for simple typicality approximation can be used to answer top-k representative typicality queries. To find the object with the largest representative typicality score. After the first answer object is added into A, to find the approximation of the next most representatively typical object, a tournament is conducted from bottom up. In each node of the LT-tree, we compute the local representative typicality of each object, and select the object with the greatest local representative typicality score as the winner, and let it go to the tournament in the parent node. The computation of local representative typicality is similar to the local representative typicality computation in the DLTA algorithm.

There is one critical difference between the LT3 algorithm for simple typicality computation and the LT3 algorithm for representative typicality computation. In simple typicality computation, once the first winner object is computed, we only need to re-conduct part of the tournament to find the next winner object. However, the representative typicality score of each object changes once the reported answer set is updated. Thus, no results can be reused. A new tournament among the rest of the objects should be conducted from bottom up completely. Therefore, the LT3 method for representative typicality computation is not as efficient as the LT3 method for simple typicality or discriminative typicality computation.

Similar to Theorem 3, the quality of answering top-k representative typicality queries using the LT3 method is guaranteed.

Theorem 8 *In a set S, let o be the object of the largest representative typicality and $\widetilde{o}$ be an object computed by the LT3 method using local representative typicality approximation and the tournament group size t. Then,*

$$RT(o, A, S) - RT(\widetilde{o}, A, S) < \frac{2}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \cdot \lceil \log_t |S| \rceil$$

8 Empirical evaluation

In this section, we report a systematic empirical study using real data sets and synthetic data sets. All the experiments were conducted on a PC computer with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk, running the Microsoft Windows XP Professional Edition operating system. Our algorithms were implemented in Microsoft Visual C++ V6.0.

8.1 Typicality queries on real data sets

In this section, we use two real data sets to illustrate the effectiveness of typicality queries on real applications.

8.1.1 Typicality queries on the Zoo data set

We use the Zoo Database from the UCI Machine Learning Database Repository,[1] which contains 100 tuples on 15 Boolean attributes and 2 numerical attributes, such as *hair* (Boolean), *feathers* (Boolean) and *number of legs* (numeric). The Euclidean distances are computed between objects by treating Boolean values as binary values. All tuples are classified into 7 categories (*mammals*, *birds*, *reptiles*, *fish*, *amphibians*, *insects* and *invertebrates*). We apply the simple typicality, discriminative typicality and representative typicality queries on the Zoo Database. The results of the three queries all match the common sense of typicality.

We compute the simple typicality for each animal in the data set. Table 3 shows the most typical and the most atypical animals of each category. Since some tuples, such as those 10 most typical animals in category *mammals*, have the same values on all attributes, they have the same typicality value. The most typical animals returned in each category can serve as good exemplars of the category. For example, in category *mammals*, the most typical animals are more likely to be referred to as a mammal than the most atypical one, *platypuses*, which are one of the very few mammal species that lay eggs instead of giving birth to live young.

We apply *discriminative typicality analysis* on the Zoo Database to find the discriminative typical animals for each category. The results are listed in Table 3 as well. In some categories, the tuples having the largest simple typicality value also have the highest discriminative typicality value, such as categories *mammals*, *birds*, *fish*, *invertebrates*, and *amphibians*. In some categories such as *insects* and *reptiles*, the most

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html.

Table 3 The most typical, the most discriminatively typical, and the most atypical animals

Category	# Tuples	Most typical	Most discriminative typical	Most atypical
Mammal	40	Boar, Cheetah, Leopard, Lion, Lynx, Mongoose, Polecat, Puma, Raccoon, Wolf ($T = 0.16$)	Boar, Cheetah, Leopard, Lion, Lynx, Mongoose, Polecat, Puma, Raccoon, Wolf ($DT = 0.08$)	Platypus ($T = 0.01$)
Bird	20	Lark, Pheasant, Sparrow, Wren ($T = 0.15$)	Lark, Pheasant, Sparrow, Wren ($DT = 0.04$)	Penguin ($T = 0.04$)
Fish	14	Bass, Catfish, Chub, Herring, Piranha ($T = 0.15$)	Bass, Catfish, Chub, Herring, Piranha ($DT = 0.03$)	Carp ($T = 0.03$)
Invertebrate	10	Crayfish, Lobster ($T = 0.16$)	Crayfish, Lobster ($DT = 0.01$)	Scorpion ($T = 0.08$)
Insect	8	Moth, Housefly ($T = 0.13$)	Gnat ($DT = 0.02$)	Honeybee ($T = 0.06$)
Reptile	5	Slowworm ($T = 0.17$)	Pitviper ($DT = 0.007$)	Seasnake ($T = 0.08$)
Amphibian	3	Frog ($T = 0.2$)	Frog ($DT = 0.008$)	Newt, Toad ($T = 0.16$)

T simple typicality value, DT discriminative typicality value

Table 4 The most representatively typical and the most typical animals

Top-10 most representatively typical animals				Top-10 most typical animals			
Rank	Animal	Representative typicality score	Category	Rank	Animal	Simple typicality score	Category
1	Boar	0.0874661	Mammal	1	Boar	0.0855363	Mammal
2	Lark	0.135712	Bird	1	Cheetah	0.0855363	Mammal
3	Bass	0.176546	Fish	1	Leopard	0.0855363	Mammal
4	Gnat	0.198142	Insect	1	Lion	0.0855363	Mammal
5	Aardvark	0.213342	Mammal	1	Lynx	0.0855363	Mammal
6	Wallaby	0.225642	Mammal	1	Mongoose	0.0855363	Mammal
7	Starfish	0.236235	Invertebrate	1	Polecat	0.0855363	Mammal
8	Slug	0.246638	Invertebrate	1	Puma	0.0855363	Mammal
9	Dolphin	0.236347	Mammal	1	Raccoon	0.0855363	Mammal
10	Frog	0.265012	Amphibian	1	Wolf	0.0855363	Mammal

typical animals are not the most discriminatively typical. For example, in category *reptiles*, the most discriminatively typical animal is *pitvipers* in stead of *slowworm*, because *slowworm* is also similar to some animals in other categories besides *reptiles*, such as *newts* in category *amphibians*. On the other hand, *pitvipers* are venomous. Very few animals in the other categories are venomous. The result matches the above analysis.

In some situations, the results from the simple typicality queries may have a bias on the most popular categories. Table 4 lists the top-10 most typical animals in the Zoo Database. The 10 animals are all mammals, since *mammals* are the largest category in the Zoo Database. As a result, the top-10 most typical animals as a whole is not representative. The animals in other categories cannot be represented well.

Representative typicality queries avoid this problem. The top-10 most representatively typical animals are also listed in Table 4, which cover 6 out of the 7 categories in the Zoo Database. The only missed category is *Reptile*, which only contains 5 animals. *Boar* is the first animal in the answer set of both queries, since it is the most typical animal in the whole data set. Note that the representative typicality score and the simple typicality score of *boar* are slightly different, because the bandwidth parameter h is computed via sampling, and thus may have a small difference in each computation. The second most representatively typical animal is *Lark*, which is the most typical animal in the second most popular category *Bird*. *Dolphin* is in the answer to the top-10 representative typicality query, since it represents a set of aquatic mammal in the Zoo database, such as porpoise and mink. They are not typical mammals, but they are an important category if we would like to explore different kinds of mammals.

To show the difference between typicality analysis and clustering analysis, we apply the k-medoids clustering algorithm [36] to the Zoo data set. We first compute the 10 clusters of the Zoo data set. The median animals of the clusters are {*Starfish, Boar, Lark, Tuatara, Dolphin, Flea, Bass, Mink, Scorpion, Hare*}. The group typicality score of the

Table 5 The most typical NBA players in 2005-2006 Season

Name	T	Position	Minutes	Points per game	3-Point throw	Rebounds	Assists	Blocks
Danny Granger	0.0383	Forwards	22.6	7.5	1.6	4.9	1.2	0.8
Devean George	0.0382	Forwards	21.7	6.3	3	3.9	1.0	0.5
Michael Finley	0.0378	Guards	26.5	10.1	5	3.2	1.5	0.1

T simple typicality values

Table 6 The answers to top-10 simple typicality/discriminative typicality/representative typicality queries on the NBA guards

Top-10 simple typicality query				Top-10 discriminative typicality query				Top-10 representative typicality query			
Name	T	3PT	AST	Name	DT	3PT	AST	Name	RT	3PT	AST
Ronald Murray	0.076	2.4	2.6	Delonte West	0.0095	4.3	4.6	Ronald Murray	0.076	2.4	2.6
Marko Jaric	0.075	2.3	3.9	David Wesley	0.0092	5.2	2.9	Andre Owens	0.06	0.8	0.4
Keith Bogans	0.074	3.7	1.8	Speedy Claxton	0.0092	1.1	4.8	Charlie Bell	0.037	4.1	2.2
Kevin Martin	0.074	3.4	1.4	Eddie Jones	0.0085	6.8	2.4	Mike James	0.023	6.9	5.8
Anthony Johnson	0.072	2.8	4.3	Chris Duhon	0.0083	5.2	5	Brent Barry	0.017	3.8	1.7
Jalen Rose	0.072	3.2	2.5	T. J. Ford	0.0082	1.9	6.6	Calbert Cheaney	0.017	0.2	0.5
Michael Finley	0.071	5	1.5	Jalen Rose	0.0082	3.2	2.5	Alex Acker	0.014	1.2	0.8
Chucky Atkins	0.071	5.3	2.8	Kirk Hinrich	0.0079	5.8	6.4	Leandro Barbosa	0.009	4.9	2.8
Chris Duhon	0.071	5.2	5	Jason Terry	0.0078	7.3	3.8	Zoran Planinic	0.007	1.2	1
Eddie Jones	0.07	6.8	2.4	Mike Miller	0.0077	6.5	2.7	Keyon Dooling	0.007	1.2	2.2

T simple typicality values, DT discriminative typicality values, RT representative typicality values, and 3PT for 3-point throw

set of median animals is 0.182. At the same time, the group typicality score of the answer set of the top-10 most representatively typical animals shown in Table 4 is 0.216. Therefore, the set of animals found by the clustering analysis is only 84% as representative as the set of animals found by the top-k representative typicality queries.

8.1.2 Typicality queries on the NBA data set

We apply typicality queries on the NBA 2005–2006 Season Statistics.[2] The data set contains the technical statistics of 458 NBA players, including 221 guards, 182 forwards and 55 centers, on 16 numerical attributes.

Table 5 shows the top-3 most typical players, and some of the attribute values. The results answer Jeff's question in Sect. 1.1.1.

To answer Jeff's question in Sect. 1.1.2, we conduct a top-10 discriminative typicality query on guards. The results are shown in Table 6. For comparison, in the same table we also list the answer to the top-10 simple typicality query on guards. To explain the results, we list some selected attributes as well. The most discriminatively typical guards have better performance than those of the highest simple typicality in *3-point throws* or *assists*, which are the skills popular in guards, but may not be common in other players.

In Table 6, we list the answers to the top-10 representative typicality query on guards. Comparing to the answers to a top-10 simple typicality query listed in Table 6, the top-10 representatively typical guards are quite different from each other in 3 point throws and assists. For example, *Ronald Murray*, the most typical guard, represents the NBA guards who are experienced and perform well, while *Andre Owens*, the second most representatively typical guard, represents a group of NBA guards whose performances are relatively poorer.

We use the NBA data set to examine the differences among medians, means, and typical objects. The results are shown in Table 7. The simple typicality scores of the medians and the means are often substantially lower than the most typical players, which justifies that the geometric centers may not reflect the probability density distribution. A typical player can be very different from the median player and the mean. For example, Ronald Murray is identified as the most typical guard, but Charlie Bell is the median guard. The technical statistics show that Murray makes fewer rebounds than Bell, but contributes more assists. To this extent, Murray is more typical than Bell as a guard. Moreover, Ronald Murray played 76 games in the season, while Charlie Bell only played 59 games. If we take the range $76 \pm 6 = [70, 82]$, then there are 92 guards whose numbers of games played are in the range; while there are only 31 guards whose numbers of games played are in the range $59 \pm 6 = [53, 65]$. That is, much more guards played a similar number of games as Murray.

[2] http://sports.yahoo.com/nba/stats/.

Table 7 Comparison among medians, means, and typical players in the NBA data set

Category (position)	Median/mean/most typical	Name	Simple typicality	# Games	Avg. min. per game
All players	Median	Ryan Gomes	0.0271	61	22.6
	Mean	N/A	0.0307	54.4	20.51
	Most typical	Danny Granger	0.3830	78	22.6
Centers	Median	Jake Voskuhl	0.0903	51	16
	Mean	N/A	0.0679	52.42	17.36
	Most typical	Francisco Elson	0.1041	72	21.9
Forwards	Median	Al Jefferson	0.0747	59	18
	Mean	N/A	0.0509	54.83	19.97
	Most typical	Maurice Taylor	0.0910	67	18.1
Guards	Median	Charlie Bell	0.0488	59	21.7
	Mean	N/A	0.0230	54.54	21.73
	Most typical	Ronald Murray	0.0756	76	27.8

To compare the difference between typicality analysis and clustering analysis, we compute 2 clusters of all guards using the k-medoids clustering algorithm [36]. The median players of clusters are {*Ronald Murray, Stephon Marbury*}, whose group typicality score is 0.105. The group typicality score of the top-2 most representatively typical guards in Table 6 (i.e.,{Ronald Murray, Andre Owens}) is 0.161. The set of players found by the clustering analysis is only 65% as representative as the set of players found by the top-k representative typicality queries.

8.2 Approximation quality

To evaluate the query answering quality on large data sets, we use the Quadraped Animal Data Generator also from the UCI Machine Learning Database Repository to generate synthetic data sets with up to 25 numeric attributes. We test the approximation quality of the RT (randomized tournament) method, the DLTA (direct local typicality approximation) method, and the LT3 (local typicality approximation using tournaments) method on top-k simple typicality queries, top-k discriminative typicality queries, and top-k representative typicality queries, respectively. The results are reported in the rest of this section.

First of all, we test RT, DLTA, and LT3 for top-k simple typicality queries. To measure the error made by an approximation algorithm, we use the following *error rate* measure. For a top-k typicality query Q, let A be the set of k objects returned by the exact algorithm, and $\widetilde{A}$ be the set of k objects returned by an approximation algorithm. Then, the error rate e is

$$e = \frac{\sum_{o\in A} T(o,S) - \sum_{o\in \widetilde{A}} T(o,S)}{\sum_{o\in A} T(o,S)} \times 100\% \quad (9)$$

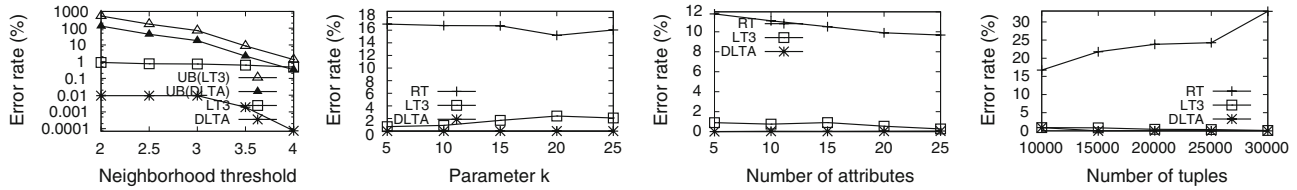
The error rate of the exact algorithm is always 0.

We compare three approximation algorithms: the randomized tournament method (RT, Algorithm 2), the direct local typicality approximation method (DLTA, Sect. 5.2), and the LT-tree tournament method (LT3, Sect. 5.3). By default, we set the number of tuples to 10, 000, the dimensionality to 5 attributes, and conduct top-10 simple typicality queries. When local typicality is computed, by default we set the neighborhood threshold to $2h$, where h is the bandwidth of the Gaussian kernel. In such a case, according to Theorem 1, the difference between the simple typicality value and the local simple typicality value of any object is always less than 0.05. In the randomized tournament method, by default the tournament group size is 10 and 4 times validation are conducted. We observe that although with more rounds of validations, the quality of randomized tournament may increase, but after 3 rounds, the quality improvement is very small.

Figure 9a shows the change of approximation quality with respect to the neighborhood threshold. In the figure, the error bounds given by Theorems 1 and 3 are also plotted, which are labeled as $UB(DLTA)$ and $UB(LT3)$, respectively. To make the curves legible, the error rates are in the logarithmic scale. Clearly, the larger the neighborhood threshold, the more accurate the local typicality approximation. Our methods perform much better than the error bounds, which shows that they are effective in practice.

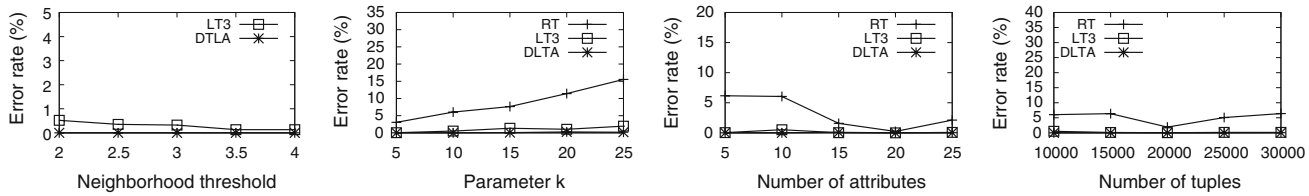
In Fig. 9b, we vary the value of k in the top-k queries. The approximation quality of RT is not sensitive to k, since it runs k times to select the top-k answers. Both DLTA and LT3 see a larger error rate with a larger value of k, this is because the distant neighbors may get a better chance to play a role in typicality when more objects are returned.

Figure 9c shows the impact of dimensionality on the error rate. DLTA achieves the best approximation quality, the error rate is up to 0.066%. LT3 has an accuracy close to DLTA,



(a) Error rate vs. neighborhood. **(b)** Error rate vs. k. **(c)** Error rate vs. dimensionality. **(d)** Error rate vs. cardinality.

Fig. 9 Approximation quality of answering top-k simple typicality queries



(a) Error rate vs. neighborhood. **(b)** Error rate vs. k. **(c)** Error rate vs. dimensionality. **(d)** Error rate vs. cardinality.

Fig. 10 Approximation quality of answering top-k discriminative typicality queries

and is much better than RT. The error rate decreases as the dimensionality increases, since the average pairwise distance in the data set also increases and the local typicality approximation becomes more effective.

Figure 9d tests the approximation quality versus the number of tuples in the data set. When the cardinality increases, the data set becomes denser, and the local typicality approximation is more accurate. That is why LT3 and DLTA perform better with larger data sets. However, the approximation quality of RT decreases in large data sets, since with a fixed tournament group size, the larger the data set, the more likely the most typical object in a random group is biased.

In summary, DLTA and LT3 both achieve better accuracy than RT, which strongly justifies the effectiveness of our local typicality approximation technique. The accuracy of LT3 is slightly lower than DLTA, but as we will show in Sect. 8.4, LT3 is much more efficient than DLTA.

We also report the approximation quality of top-k discriminative typicality query answering algorithms. By default, the data set contains 10, 000 tuples with 5 attributes. We randomly assign 20% of the tuples into the target subset C, and conduct top-10 discriminative typicality queries. The neighborhood threshold σ of DLTA and LT3 is set to $2h$, where h is the bandwidth of Gaussian kernels. The group size of randomized tournament is set to 50, and 4 validations are conducted. Here we increase the tournament size to 50, since only 20% objects are actually involved in the tournament, as we explained in Sect. 6.1.

The error rate measure is defined as follows. We normalize $DT(o, C, S-C)$ as $DT'(o, C, S-C) = DT(o, C, S-C) - \min_{x \in C} DT(x, C, S-C)$, in order to make the $DT(o, C, S-C)$ value always non-negative. For a top-k discriminative typicality query Q, let A be the set of k objects returned by the exact algorithm, and $\widetilde{A}$ be the set of k objects returned by an approximation algorithm. Then, the error rate e is

$$e = \frac{\sum_{o \in A} DT'(o, C, S-C) - \sum_{o \in \widetilde{A}} DT'(o, C, S-C)}{\sum_{o \in A} DT'(o, C, S-C)} \times 100\% \quad (10)$$

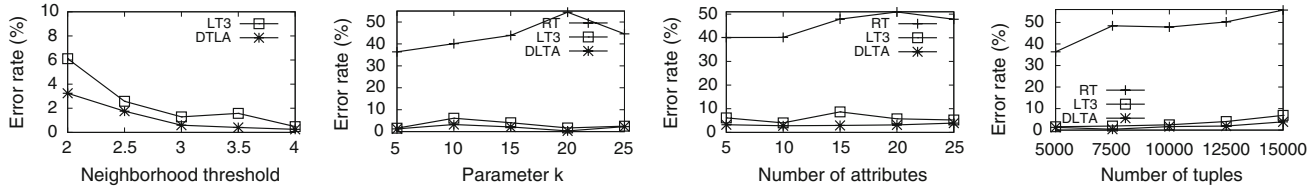
The approximation quality of the three methods are shown in Fig. 10. In general, the comparison among RT, DLTA and LT3 is similar to the situation of the top-k simple typicality query evaluation listed in Fig. 9.

To test the approximation quality for representative typicality, we conducted various experiments. By default, the data set contains 5, 000 tuples with 5 attributes, and conduct top-10 representative typicality queries. The neighborhood threshold σ of DLTA and LT3 is set to $2h$, where h is the bandwidth of Gaussian kernels. The group size of randomized tournament is set to 10, and 4 validations are conducted.

We adopt the following error rate measure. For a top-k representative typicality query Q, let A be the set of k objects returned by the exact algorithm, and $\widetilde{A}$ be the set of k objects returned by an approximation algorithm. $GT(A, S)$ and $GT(\widetilde{A}, S)$ are the group typicality scores of A and $\widetilde{A}$, respectively. Then, the error rate e is

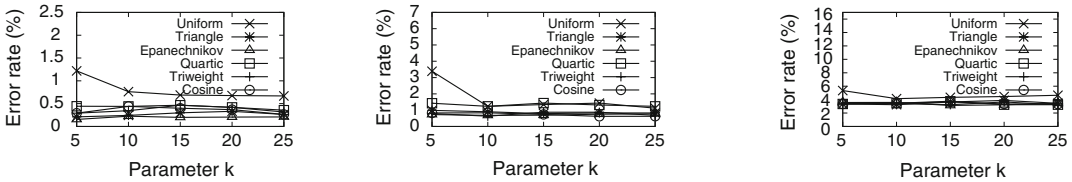
$$e = \frac{|GT(A, S) - GT(\widetilde{A}, S)|}{GT(A, S)} \times 100\% \quad (11)$$

The error rate measure computes the difference between the group typicality of the exact answer and the group typicality of the approximate answer. If the error rate is small, even the objects in the two answer sets are different, the approximation to the answer still represents the whole data set well. The approximation quality of representative typicality approximation is shown in Fig. 11. The explanations are similar to the situations of simple typicality queries.



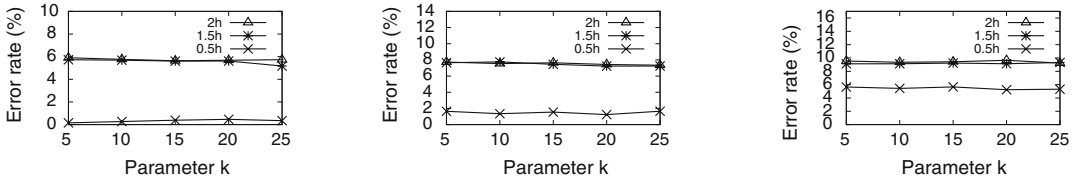
(a) Error rate vs. neighborhood. **(b)** Error rate vs. k. **(c)** Error rate vs. dimensionality. **(d)** Error rate vs. cardinality.

Fig. 11 Approximation quality of answering top-k representative typicality queries



(a) Top-k simple typicality queries. **(b)** Top-k discriminative typicality queries. **(c)** Top-k representative typicality queries.

Fig. 12 The error rates of using different kernel functions with respect to k



(a) Top-k simple typicality queries. **(b)** Top-k discriminative typicality queries. **(c)** Top-k representative typicality queries.

Fig. 13 The error rates of using different bandwidth values with respect to k

In summary, for all three types of typicality queries, DLTA has the best approximation quality, while RT gives the largest error rates. LT3 has comparable approximation quality to DLTA.

8.3 Sensitivity to parameters and noise

To test the sensitivity of the answers of top-k typicality queries with respect to the kernel function and the bandwidth value, we use the Quadraped Animal Data Generator to generate synthetic data sets with 10, 000 tuples and 5 attributes.

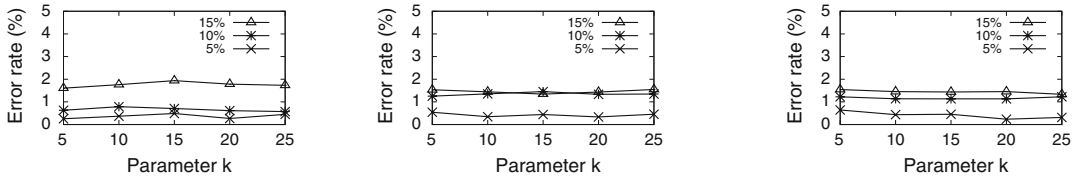
We first fix the bandwidth value $h = \frac{1.06s}{\sqrt[5]{n}}$ as discussed in Sect. 4.1, and use the kernel functions listed in Table 2 to answer top-k simple typicality/ discriminative typicality/ representative typicality queries. We compare the results computed using the Gaussian kernel function and the results computed using some other kernel functions as follows.

Let the results returned by using the Gaussian kernel be A and the results returned by using other kernel functions be $\widetilde{A}$, the error rates of the answers to the three typicality queries are computed using Eqs. 9, 10 and 11, respectively. The curves are shown in Fig. 12. The results match the discussion in Sect. 4.1: the answers computed using some other kernel functions are similar to the answers computed using the Gaussian kernel, since the error rates are very low.

We then use the Gaussian kernel function and vary the bandwidth value from $0.5h$ to $2h$, where $h = \frac{1.06s}{\sqrt[5]{n}}$ is the default bandwidth value used in other experiments. Let A be the answer set computed using the default bandwidth value h and $\widetilde{A}$ be the answer set computed using other bandwidth values, the error rates are computed using Eqs. 9, 10 and 11, respectively. From the results shown in Fig. 13, we can see that the answers computed using different bandwidth values are similar in their typicality/discriminative typicality/group typicality score. Moreover, using smaller bandwidth values causes less difference than using larger bandwidth values. Larger bandwidth values smooth out the peaks of the density curves, which are the most typical points.

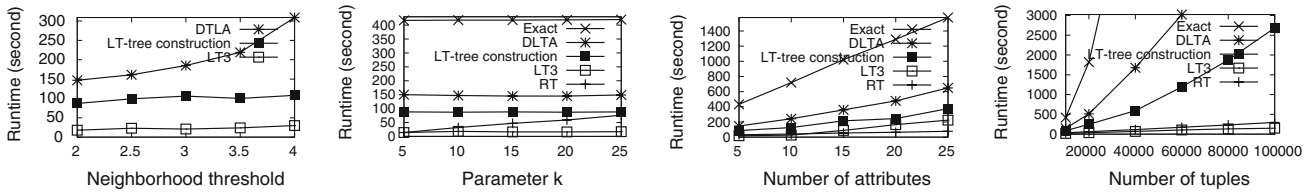
In summary, the answers to top-k simple typicality queries, top-k discriminative typicality queries and top-k representative typicality queries are insensitive to the choice of kernel functions and the bandwidth values.

Moreover, we evaluate the sensitivity of top-k typicality queries with respect to noise in data sets. We use the Quadraped Animal Data Generator to generate synthetic data sets with 10, 000 tuples and 5 attributes. In addition, we add 5 to 15% noise tuples whose attribute values are uniformly distributed in the domain of each attribute. Gaussian kernel function and bandwidth $h = \frac{1.06s}{\sqrt[5]{n}}$ are used. The answers returned are denoted by $\widetilde{A}$, and the answers computed when



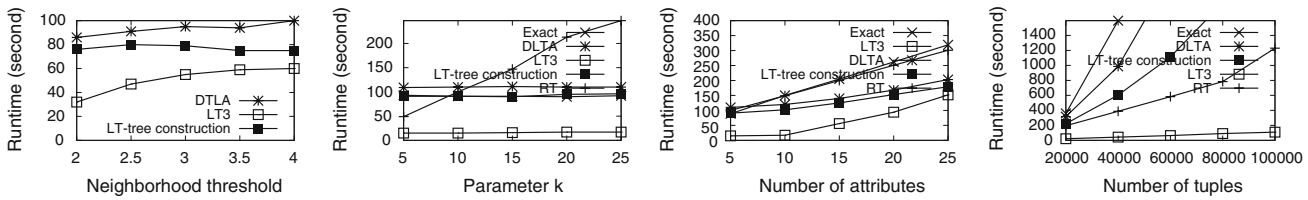
(a) Top-k simple typicality queries. **(b)** Top-k discriminative typicality queries. **(c)** Top-k representative typicality queries.

Fig. 14 The error rates of having different amount of noises with respect to k



(a) Runtime vs. neighborhood. **(b)** Runtime vs. k. **(c)** Runtime vs. dimensionality. **(d)** Runtime vs. cardinality.

Fig. 15 Efficiency and scalability of answering top-k simple typicality queries



(a) Runtime vs. neighborhood. **(b)** Runtime vs. k. **(c)** Runtime vs. dimensionality. **(d)** Runtime vs. cardinality.

Fig. 16 Efficiency and scalability of answering top-k discriminative typicality queries

removing the noises are denoted by A. The error rates in Fig. 14a, b, and c are computed using Eqs. 9, 10 and 11, respectively. Clearly, the results of top-k typicality queries are not sensitive to noise and outliers in data sets.

8.4 Efficiency and scalability

To test the efficiency and the scalability of our methods, we report in Figs. 15, 16, and 17 the runtime in the experiments conducted in Figs. 9, 10, and 11, respectively.

As shown in Fig. 15a, the runtime of DLTA increases substantially when the neighborhood threshold increases, but the increase of runtime for the LT3 method is mild, thanks to the tournament mechanism.

Figure 15b shows that the runtime of DLTA and LT3 is insensitive to the increase of k. LT3 incrementally computes other top-k answers after the top-1 answer is computed. Thus, computing more answers only takes minor cost. The RT method has to run the tournaments k rounds, and thus the cost is linear to k.

As shown in Fig. 15c, among the four methods, RT is the fastest and the exact algorithm is the slowest. LT3 and DLTA are in between, and LT3 is faster than DLTA. All methods are linearly scalable with respect to dimensionality.

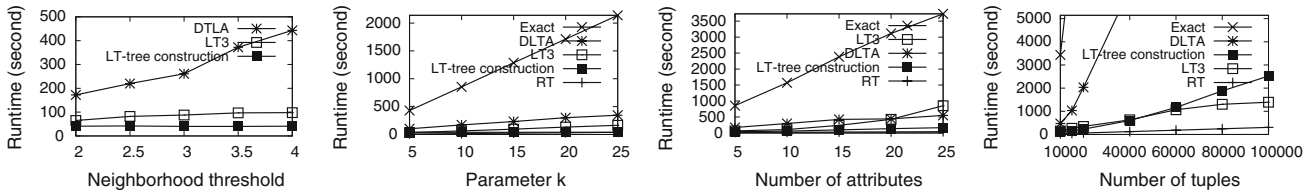
Figure 15d shows the scalability of the four algorithms with respect to database size. RT has a linear scalability. LT3 clearly has the better performance and scalability than DLTA on large data sets.

The trends for discriminative typicality queries and representative typicality queries are similar, as shown in Figs. 16 and 17, respectively.

In summary, as RT has linear complexity, when runtime is the only concern, RT should be used. While DLTA and LT3 are much more scalable than the exact algorithm and are more accurate than RT, they are good when both accuracy and efficiency matter. LT3 has the better efficiency and scalability than DLTA, while achieving comparable accuracy to DLTA.

9 Conclusions

In this paper, we apply the idea of typicality analysis from psychology and cognitive science to query answering, and study the novel problem of answering top-k typicality queries. The simple typicality, the discriminative typicality and the representative typicality measures are proposed for different applications. As computing the exact answers to top-k typicality queries on large data sets can be too costly for



(a) Runtime vs. neighborhood. **(b)** Runtime vs. k. **(c)** Runtime vs. dimensionality. **(d)** Runtime vs. cardinality.

Fig. 17 Efficiency and scalability of answering top-k representative typicality queries

online queries, we develop a series of approximation methods. By a systematic empirical evaluation using both real data sets and synthetic data sets, we illustrate the effectiveness of top-k typicality queries, and verify the accuracy and the efficiency of our methods.

Typicality can find other applications in databases. As future work, we would like to explore the potential of typicality analysis in data summarization, data warehousing and data mining.

Appendix

A: Proof of Theorem 1

For any object $x \in C$,

$$T(x,S) = \frac{1}{|S|}\left(\sum_{y\in LN(C,S,\sigma)} G_h(x,y) + \sum_{z\in(S-LN(C,S,\sigma))} G_h(x,z)\right)$$

Since $LT(x,C,S,\sigma)=\frac{1}{|LN(C,S,\sigma)|}\sum_{y\in LN(C,S,\sigma)} G_h(x,y)$,

$$T(x,S) = \frac{1}{|S|}\left(|LN(C,S,\sigma)|\cdot LT(x,C,S,\sigma) + \sum_{z\in(S-LN(C,S,\sigma))} G_h(x,z)\right) \quad (12)$$

Because $LN(C,S,\sigma)\subseteq S$, $\frac{|LN(C,S,\sigma)|}{|S|}\leq 1$. Thus,

$$T(x,S) \leq LT(x,C,S,\sigma) + \frac{1}{|S|}\sum_{z\in(S-LN(C,S,\sigma))} G_h(x,z) \quad (13)$$

According to the definition of local neighborhood, $d(x,z)>\sigma$ for any $z\in(S-LN(C,S,\sigma))$. Thus,

$$\frac{1}{|S|}\sum_{y\in(S-LN(C,S,\sigma))} G_h(x,y) < \frac{1}{\sqrt{2\pi}}e^{-\frac{\sigma^2}{2h^2}} \quad (14)$$

Inequality 3 follows from inequalities 13 and 14 immediately.

Applying Eq. 12 to o and $\widetilde{o}$, respectively, we have

$$T(o,S)-T(\widetilde{o},S) = \frac{|LN(C,S,\sigma)|}{|S|}(LT(o,C,S,\sigma)-LT(\widetilde{o},C,S,\sigma)) + \frac{1}{|S|}\sum_{z\in(S-LN(C,S,\sigma))}(G_h(o,z)-G_h(\widetilde{o},z))$$

Using inequality 14, we have

$$\frac{1}{|S|}\sum_{z\in(S-LN(C,S,\sigma))}(G_h(o,z)-G_h(\widetilde{o},z)) \leq \frac{1}{\sqrt{2\pi}}e^{-\frac{\sigma^2}{2h^2}}$$

Since $LT(\widetilde{o},C,S,\sigma)\geq LT(o,C,S,\sigma)$, $LT(o,C,S,\sigma)-LT(\widetilde{o},C,S,\sigma)\leq 0$. Thus,

$$T(o,S)-T(\widetilde{o},S) \leq \frac{1}{\sqrt{2\pi}}e^{-\frac{\sigma^2}{2h^2}}$$

Inequality 2 is shown.

B: Proof of Corollary 1

From Theorem 1, For any object $o\in C$, subset $C\subseteq S$ and neighborhood threshold σ, we have

$$T(x,S)-LT(x,C,S,\sigma) < \frac{1}{\sqrt{2\pi}}e^{-\frac{\sigma^2}{2h^2}}$$

In order to meet the quality requirement θ ($\theta<\frac{1}{2\pi}$), it should hold that $\frac{1}{\sqrt{2\pi}}e^{-\frac{\sigma^2}{2h^2}}\leq\theta$. Therefore, $\sigma\geq\sqrt{-2\ln\sqrt{2\pi}\theta}\cdot h$.

C: Proof of Theorem 2

If $A\cap\widetilde{A}\neq\emptyset$, then let $A=A-A\cap\widetilde{A}$ and $\widetilde{A}=\widetilde{A}-A\cap\widetilde{A}$. So in the rest of the proof, we assume $A\cap\widetilde{A}=\emptyset$.

We sort the objects in A in the descending order of their typicality values, and sort the objects in $\widetilde{A}$ in the descending order of their local typicality values. Let o be the ith object in A, and $\widetilde{o}$ be the ith object in $\widetilde{A}$ ($1\leq i\leq k$). From

Inequality 3, we have

$$0 \le T(o, S) - LT(o, \{o\}, S, \sigma) < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}$$

and

$$0 \le T(\widetilde{o}, S) - LT(\widetilde{o}, \{\widetilde{o}\}, S, \sigma) < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}$$

Moreover, since $o \notin \widetilde{A}$, it holds that $LT(o, \{o\}, S, \sigma) < LT(\widetilde{o}, \{\widetilde{o}\},$
$S, \sigma)$. Thus,

$$\begin{aligned} &T(o, S) - T(\widetilde{o}, S) \\ &= (T(o, S) - LT(o, \{o\}, S, \sigma)) \\ &\quad -(T(\widetilde{o}, S) - LT(\widetilde{o}, \{\widetilde{o}\}, S, \sigma)) + (LT(o, \{o\}, S, \sigma) \\ &\quad -LT(\widetilde{o}, \{\widetilde{o}\}, S, \sigma)) < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \end{aligned}$$

Inequality 4 follows by summing up the above difference at each rank i $(1 \le i \le k)$.

D: Proof of Theorem 3

In the worst case, object o is not selected as the winner in the first level of tournaments.

Let o_1 be the winner of the group containing o in the first level of tournaments, then we have

$$T(o, S) - T(o_1, S) < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}$$

as indicated by inequality 2 in Theorem 1.

If o_i fails the $(i+1)$th level of tournaments, let o_{i+1} be the winner of the group containing o_i in this tournament, then again we have

$$T(o_i, S) - T(o_{i+1}, S) < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}$$

For a set of objects S, there are $\lceil \log_t |S| \rceil$ levels of tournaments. The final winner $\widetilde{o}$ is the winner in the $\lceil \log_t |S| \rceil$th level of tournaments. That is, $\widetilde{o} = o_{\lceil \log_t |S| \rceil}$. Then,

$$\begin{aligned} T(o, S) - T(\widetilde{o}, S) = &T(o, S) - T(o_1, S) \\ &+T(o_1, S) - T(o_2, S) + \cdots \\ &+T(o_{\lceil \log_t |S| \rceil - 1}, S) - T(o_{\lceil \log_t |S| \rceil}, S) \\ < &\frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \cdot \lceil \log_t |S| \rceil \end{aligned}$$

The inequality in the theorem holds.

E: Proof of Theorem 7

Proof of Theorem 7 To prove Theorem 7, we need the following lemma.

Lemma 1 (Local group typicality score approximation) *Given a set of objects S, a neighborhood threshold σ and a reported answer set $A \subset S$.*

$$GT(A, S) - LGT(A, S, \sigma) < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \tag{15}$$

Proof For each object $o \in A$, let $N(o, A, S)$ be the set of objects in S that lie in $D(o, A)$, according to Eq. 1, we have

$$\begin{aligned} &T(o, N(o, S, A)) \cdot Pr(N(o, S, A)) \\ &= \frac{1}{|N(o, S, A)|} \sum_{x \in N(o,S,A)} G_h(x, o) \times \frac{|N(o, S, A)|}{|S|} \\ &= \frac{1}{|S|} \sum_{x \in N(o,S,A)} G_h(x, o) \end{aligned}$$

Let $N = LN(\{o\}, N(o, A, S), \sigma)$ be the σ-neighborhood of o in $N(o, A, S)$, then

$$\begin{aligned} &LT(o, \{o\}, N(o, S, A), \sigma) \cdot Pr(N) \\ &= \frac{1}{|N|} \sum_{x \in N} G_h(x, o) \times \frac{|N|}{|S|} \\ &= \frac{1}{|S|} \sum_{x \in N} G_h(x, o) \end{aligned}$$

Thus,

$$\begin{aligned} &T(o, N(o, S, A)) Pr(N(o, S, A)) \\ &\quad -LT(o, \{o\}, N(o, S, A), \sigma) Pr(N) \\ &= \frac{1}{|S|} \left(\sum_{x \in N(o,S,A)} G_h(x, o) - \sum_{x \in N} G_h(x, o) \right) \\ &= \frac{1}{|S|} \sum_{x \in N(o,S,A) - N} G_h(x, o) \end{aligned}$$

For object $x \in N(o, S, A) - N$, x is not in the σ-neighborhood of o, so $d(x, o) > \sigma$. Therefore

$$\frac{1}{|S|} \sum_{x \in N(o,S,A) - N} G_h(x, o) < \frac{1}{|S|} \sum_{x \in N(o,S,A) - N} \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}$$

Thus, we have

$$\begin{aligned} &GT(A,S) - LGT(A,S,\sigma) \\ &= \sum_{o\in A}(T(o,N(o,S,A))Pr(N(o,S,A)) \\ &\quad -LT(o,\{o\},N(o,S,A),\sigma)Pr(N)) \\ &= \sum_{o\in A}\frac{1}{|S|}\sum_{x\in N(o,S,A)-N} G_h(x,o) \\ &< \frac{1}{|S|}\sum_{o\in A}\sum_{x\in N(o,S,A)-N}\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}} \\ &< \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}} \end{aligned}$$

Equation 15 holds. □

Proof of Theorem 7 For any object $x \in S$,

$$RT(o,A,S) = GT(A\cup\{o\},S) - GT(A,S)$$
$$LRT(o,A,S,\sigma) = LGT(A\cup\{o\},S,\sigma) - LGT(A,S,\sigma)$$

Therefore,

$$\begin{aligned} &RT(o,A,S) - LRT(o,A,S,\sigma) \\ &= (GT(A\cup\{o\},S) - LGT(A\cup\{o\},S,\sigma)) \\ &\quad -(GT(A,S) - LGT(A,S,\sigma)) \end{aligned}$$

Using Lemma 1, we have

$$0 \le GT(A\cup\{o\},S) - LGT(A\cup\{o\},S,\sigma) < \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}}$$

and

$$0 \le GT(A,S) - LGT(A,S,\sigma) < \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}}$$

Thus,

$$\begin{aligned} -\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}} &< RT(o,A,S) - LRT(o,A,S,\sigma) \\ &< \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}} \end{aligned}$$

Inequality 8 follows from the above inequality immediately.

Applying inequality 8 to o and $\widetilde{o}$, we have

$$|RT(o,A,S) - LRT(o,A,S,\sigma)| < \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}}$$

and

$$|RT(\widetilde{o},A,S) - LRT(\widetilde{o},A,S,\sigma)| < \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\sigma^2}{2h^2}}$$

Since $LRT(\widetilde{o},A,S,\sigma) \ge LRT(o,A,S,\sigma)$

$$\begin{aligned} &RT(o,A,S) - RT(\widetilde{o},A,S) \\ &= (RT(o,A,S) - LRT(o,A,S,\sigma)) \\ &\quad -(RT(\widetilde{o},A,S) - LRT(\widetilde{o},A,S,\sigma)) \\ &\quad +(LRT(o,A,S,\sigma) - LRT(\widetilde{o},A,S,\sigma)) \\ &\le \frac{2}{\sqrt{2\pi}}e^{-\frac{\sigma^2}{2h^2}} \end{aligned}$$

Inequality 7 is shown. □

References

1. Abramson, I.S.: On bandwidth variation in kernel estimates—a square root law. Ann. Stat. **10**(4), 1217–1223 (1982)
2. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. of ACM SIGMOD (1998)
3. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: ordering points to identify the clustering structure. In: Proc. of ACM SIGMOD (1999)
4. Angluin, D., Valiant, L.G.: Fast probabilistic algorithms for hamiltonian circuits and matchings In: Proc. of STOC (1977)
5. Au Yeung, C.M., Leung, H.F.: A formal model of ontology for handling fuzzy membership and typicality of instances. Comput. J. (to appear)
6. Barsalou, L.W.: The instability of graded structure: implications for the nature of concepts. Concepts and conceptual development, pp. 101–140 (1987)
7. Bespamyatnikh, S., Kedem, K., Segal, M.: Optimal facility location under various distance functions. In: Proc. of WADS (1999)
8. Bose, P., Maheshwari, A., Morin, P.: Fast approximations for sums of distances, clustering and the fermat-weber problem. Compu. Geom. Theory Appl. **24**(3), 135–146 (2003)
9. Bowman, A.W.: An alternative method of cross-validation for the smoothing of density estimates. Biometrika **71**(2), 353–360 (1984)
10. Bozkaya, T., Ozsoyoglu, M.: Indexing large metric spaces for similarity search queries. ACM Trans. Database Syst. **24**(3), 361–404 (1999)
11. Breiman, L., Meisel, W., Purcell, E.: Variable kernel estimates of multivariate densities. Technometrics **19**(2), 135–144 (1977)
12. Brooks, L.R.: Nonanalytic concept formation and memory for instances. In: Rosch, E.H., Lloyd B.B. (eds.) Cognition and Categorization, pp. 169–211. Hillsdale, New York, NY (1973)
13. Campbell, N.A.: Some aspects of allocation and discrimination. In: Multivariate Statistical Methods in Physical Anthropology, pp. 177–192 (1984)
14. Cantone, D., Cincotti, G., Ferro, A., Pulvirenti, A.: An efficient approximate algorithm for the 1-median problem in metric spaces. SIAM J. Optim. **16**(2), 434–451 (2005)
15. Charikar, M., Guha, S., Tardos, E., Shmoys, D.B.: A constant-factor approximation algorithm for the k-median problem. In: Proceedings of the Symposium on Theory of Computing, pp. 1–10. ACM Press, New York (1999)
16. Cohen, B., Murphy, G.L.: Models of concepts. Cogn. Sci. **8**, 27–58 (1984)
17. Cohen, E., Kaplan, H.: Spatially-decaying aggregation over a network: model and algorithms. In: Proc. of ACM SIGMOD (2004)
18. Cohen, E., Kaplan, H.: Spatially-decaying aggregation over a network. J. Comput. Syst. Sci. **73**(3), 265–288 (2007)
19. Das, G., Gunopulos, D., Koudas, N., Tsirogiannis, D.: Answering top-*k* queries using views. In: Proc. of VLDB (2006)
20. Devroye, L.: A Course in Density Estimation. Birkhauser, Basel (1987)
21. Devroye, L., Lugosi, G.: Combinatorial Methods in Density Estimation, 1st edn. Springer, Berlin (2001)
22. Dubois, D., Prade, H., Rossazza, J.: Vagueness, typicality, and uncertainty in class hierarchies. Int. J. Intell. Syst. **6**, 167–183 (1991)
23. Edwards, A.W.F.: Likelihood, 1st edn. Cambridge University Press, London (1985)
24. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of ACM SIGKDD (1996)

25. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: Proc. of PODS (2001)
26. Foody, G.M., Campbell, N.A., Trodd, N.M., Wood, T.F.: Derivation and applications of probabilistic measures of class membership from the maximum likelihood classification. Photogramm. Eng. Remote Sens. **58**, 1335–1341 (1992)
27. Guha, S., Rastogi, R., Shim, K.: Cure: an efficient clustering algorithm for large databases. SIGMOD Rec. **27**(2), 73–84 (1998). doi:10.1145/276305.276312
28. Gunopoulos, D., Kollios, G., Tsotras, V., Domeniconi, C.: Selectivity estimators for multi-dimensional range queries over real attributes. VLDB J. **14**(2), 137–154 (2005)
29. Hartigan, J.A., Wong, M.A.: A K-means clustering algorithm. Appl. Stat. **28**, 100–108 (1979)
30. Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. In: Proc. of ACM SIGKDD (1998)
31. Hodge, V., Austin, J.: A survey of outlier detection methodologies. Artif. Intell. Rev. **2**(2), 85–126 (2004)
32. Hua, M., Pei, J., Fu, A.W.C., Lin, X., Leung, H.F.: Efficiently answering top-*k* typicality queries on large databases. In: Proc. of VLDB (2007)
33. Indyk, P.: Sublinear time algorithms for metric space problems. In: Proc. of STOC (1999)
34. Kanazawa, Y.: An optimal variable cell histogram based on the sample spacings. Ann. Stat. **20**(1), 291–304 (1992)
35. Karypis, G., Han, E.H.S., Kumar, V.: Chameleon: hierarchical clustering using dynamic modeling. Computer **32**(8), 68–75 (1999)
36. Kaufmann, L., Rousseeuw, P.J.: Clustering by means of medoids. In: Dodge, Y. (ed.) Statistical Data Analysis based on the L1 Norm, pp. 405–416. Elsevier/North Holland, Amsterdam (1987)
37. Mack, Y., Rosenblatt, M.: Multivariate k-nearest neighbor density estimates. J. Multivar. Anal. **9**, 1–15 (1979)
38. Mouratidis, K., Bakiras, S., Papadias, D.: Continuous monitoring of top-*k* queries over sliding windows. In: Proc. of ACM SIGMOD (2006)
39. Nepal, S., Ramakrishna, M.: Query processing issues in image(multimedia) databases. In: Proc. of ICDE (1999)
40. Ng, R.T., Han, J.: Clarans: a method for clustering objects for spatial data mining. IEEE Trans. Knowl. Data Eng. **14**(5), 1003–1016 (2002)
41. Nosofsky, R.M.: Similarity, frequency, and category representations. J. Exp. Psychol. Learn. Memory Cogn. **14**(1), 54–65 (1988)
42. Rosch, E.: On the internal structure of perceptual and semantic categories. In: Cognitive Development and Acquisition of Language, pp. 111–144 (1973)
43. Rosch, E.: Cognitive representations of semantic categories. J. Exp. Psychol. Gen. **104**, 192–233 (1975)
44. Rudemo, M.: Empirical choice of histograms and kernel density estimators. Scand. J. Stat. **9**, 65–78 (1982)
45. Sain, S.R., Baggerly, K.A., Scott, D.W.: Cross-validation of multivariate densities. J. Am. Stat. Assoc. **89**(427), 807–817 (1994)
46. Scott, D., Sain, S.: Multi-dimensional density estimation. Handbook of Statistics: Data Mining and Computational Statistics, vol. 23 (2004)
47. Silverman, B.W.: Density Estimation for Statistics and Data Analysis (Hardcover). Chapman and Hall, London (1986)
48. Tamma, V., Bench-Capon, T.: An ontology model to facilitate knowledge-sharing in multi-agent systems. Knowl. Eng. Rev. **17**(1), 41–60 (2002)
49. Tarter, M.: Density estimation applications for outlier detection. Comput. Programs Biomed. **10**(1), 55–60 (1979)
50. Walfish, S.: A review of statistical outlier methods. Pharm. Technol. **30**(11), 82–88 (2006)
51. Wang, W., Yang, J., Muntz, R.R.: Sting: a statistical information grid approach to spatial data mining. In: Proc. of VLDB (1997)
52. Xin, D., Cheng, H., Yan, X., Han, J.: Extracting redundancy-aware top-*k* patterns. In: Proc. of ACM SIGKDD (2006)
53. Xin, D., Han, J., Cheng, H., Li, X.: Answering top-*k* queries with multi-dimensional selections: the ranking cube approach. In: Proc. of VLDB (2006)
54. Xu, R., Wunch II, D.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)
55. Yianilos, P.N.: Data structures and algorithms for nearest neighbor search in general metric spaces. In: Proc. of SODA (1993)
56. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. In: Proc. of ACM SIGMOD (1996)