Inferring Approximate Functional Dependencies from Example Data

Tatsuya AKUTSU
Mechanical Engineering Laboratory
1-2 Namiki, Tsukuba, Ibaraki 305, Japan
e-mail:akutsu@mel.go.jp

Atsuhiro TAKASU
National Center for Science Information Systems
3-29-1 Otsuka, Bunkyo, Tokyo 112, Japan
e-mail:takasu@nacsis.ac.jp

Abstract

This paper proposes a kind of PAC (Probably Approximately Correct) learning framework for inferring a set of functional dependencies. A simple algorithm for inferring the set of approximate functional dependencies from a subset of a full tuple set (i.e. a set of all tuples in the relation) is presented. It is shown that the upper bound of the sample complexity, which is the number of example tuples required to obtain a set of functional dependencies whose error is at most ε with a probability of at least $1-\delta$, is $O(\sqrt{\frac{\ln \frac{1}{\delta}}{\varepsilon}}\sqrt{n})$, where n denotes the size of the full tuple set and the uniform distribution of examples is assumed. An experimental result, which confirms the theoretical analysis, is also presented.

1 Introduction

Recently computer systems accumulate various kinds of data such as experimental data, observation data from satellites, performance observation data for computer systems, logs for computer and network managements and so on. Though these raw data are expected to include valuable information, some of them are not fully analyzed. In order to utilize these data, automatic or computer aided data analysis systems are required. The technique of knowledge discovery in databases [5] is prospective method for solving this problem. The volume of raw data is growing larger and larger and we will need to handle tens of millions of records in the near future. In this situation, knowledge discovery from a portion of raw data is a key issue for practical problem solving.

This paper treats data that can be represented in a relation (i.e. a set of tuples) and discusses the problem of inferring a set of functional dependencies that approximately satisfies the data from its subset. For a set of functional dependencies F and a full tuple set T, we define F's error ε as a measure of difficulty in finding the evidence of inconsistency between F and T (see Definition 5 and 7). This means that the smaller the error is, the larger the portion that functional dependencies are consistent with. Then the problem is to show the sample

complexity, that is, the size of the sample required to obtain a set of functional dependencies whose error does not exceed ε with a probability of at least $1 - \delta$.

The functional dependency is a fundamental dependency of relations. It is originally introduced as an integrity constraint and used for relational database design (i.e. building normal form relations). The method of discovering functional dependencies can be applied to relational database design. Kantola et al. developed a database design tool [7]. In this tool, a set of functional dependencies are derived from a relation and the derived functional dependencies are used to decompose the relation. The functional dependency is regarded as cause-effect relationship where left and right sides of functional dependencies represent cause and effect attributes respectively. From this point of view, Ziarko proposed a method to generate a decision table using the discovery of the functional dependency [15]. Thus the discovery of functional dependency has broad applications.

For the discovery of approximate functional dependencies, we use the framework of PAC (Probably Approximately Correct) learning [3, 9] proposed by Valiant [14]. Though the framework seems to be suitable for our problem, it can not be applied directly because of the following problem:

- Whether a functional dependency is consistent or not is determined not for one tuple, but for a set of tuples. That is, positive or negative is not defined for one example.

Therefore, we developed a PAC learning framework for functional dependencies. In this paper, we focus on the number of examples. We consider a very simple algorithm whose output is a set of all the functional dependencies consistent with a sample tuple set. In our previous paper [2], we showed that the sample complexity under an arbitrary and unknown probability distribution is $O(\frac{\sqrt{\ln \frac{1}{\delta}}}{\varepsilon}\sqrt{n})$ where n is the size of a full tuple set (a set of all tuples in the relations) and the number of attributes is assumed to be fixed. However, this value is too large for practical application. Moreover, it is not realistic to consider an arbitrary distribution. In this paper, we consider the uniform distribution and show that the number is improved to $O(\sqrt{\frac{\ln \frac{1}{\delta}}{\varepsilon}}\sqrt{n})$. The lower bound is also improved to $\Omega(\sqrt{\frac{1-\delta}{\varepsilon}}\sqrt{n})$, while it is $\Omega(\frac{\sqrt{1-\delta}}{\varepsilon}\sqrt{n})$ under an arbitrary distribution. Note that the lower bounds shown are not for an arbitrary algorithm, but for the simple algorithm presented in this paper.

As well as performing theoretical analysis, we made experiments for confirming the theoretical results. The experimental results coincide with the theoretical results very well. Moreover, the sample complexity obtained from the experiments is closer to the lower bound than the upper bound. These theoretical and experimental results imply that the method described in this paper can be applied to practical cases.

A lot of studies have been done for learning rules in database systems [7, 10, 11, 12]. Quinlan and Rivest studied a method for inferring decision trees from tuples [11]. Piatetsky-Shapiro studied a method for deriving rules from a part of data [10]. The forms of rules are limited to $Cond(t) \rightarrow (t[A_i] = b)$ and $Cond(t) \rightarrow (b_1 \leq t[A_i] \leq b_2)$. Sonoo et al. studied another method for deriving rules from a part of data [12]. The form of rules is limited to $(t[A_i] = a) \rightarrow (t[A_j] = b_1 \vee b_2 \vee \cdots \vee b_k)$. Functional dependencies can not be described in either form. Kantola et al. studied methods for inferring functional dependencies [7]. They describe an algorithm for enumerating functional dependencies which are consistent with sample data. Although their work is similar to ours, the distinction of our work is in the

approximate functional dependencies and the sample complexity for inferring them, which were not studied in [7].

The contents of this paper are as follows. In Section 2, the usual PAC learning framework is briefly reviewed. In Section 3 and Section 4, the sample complexity for testing a functional dependency is studied. In Section 5, the discussions in Sections 3 and 4 are extended for multiple functional dependencies and the main result of this paper, the sample complexity for inferring approximate functional dependencies under uniform distribution, is described. In Section 6, the results of this paper are compared with the results of our previous paper [2]. In Section 7, an experimental result, which confirms the theoretical analysis, is presented. Finally, in Section 8, we conclude with proposals for future works.

2 PAC Learning

In this section, we briefly review the framework of PAC learning [3, 9, 14] (see [9] for detailed discussion).

For an alphabet Σ, a subset c of Σ^* is called a *concept*. A concept can be regarded as a function $c : \Sigma^* \to \{0,1\}$ such that $c(x) = 1$ if $x \in c$, $c(x) = 0$ otherwise. A *concept class* F is a (nonempty) set of concepts. For a concept c, an *example* of c is a pair $(x, c(x))$ for $x \in \Sigma^*$. An example $(x, c(x))$ is said to be *positive* (resp. *negative*) if $c(x) = 1$ (resp. $c(x) = 0$). Each example $(x, c(x))$ is given randomly according to $P(x)$ where $P(x)$ is an arbitrary and unknown probability distribution of $\Sigma^{[n]}$ ($\Sigma^{[n]}$ is the set of all strings of length at most n). For concepts c_1 and c_2, $c_1 \triangle c_2$ denotes the symmetric difference $(c_1 - c_2) \cup (c_2 - c_1)$ and $P(c_1 \triangle c_2)$ denotes $\sum_{x \in c_1 \triangle c_2} P(x)$.

[Definition 1] ([9]) An algorithm A_{PAC} is a *probably approximately correct* (PAC) learning algorithm for a class of concepts F if

- A_{PAC} takes as input $\varepsilon \in (0,1]$, $\delta \in (0,1]$ and n where ε and δ are the *error* and the *confidence* parameters respectively, and n is the length parameter.
- A_{PAC} takes examples where each example is given randomly according to an arbitrary and unknown probability distribution P on $\Sigma^{[n]}$.
- For all concepts $f \in F$ and all probability distributions P, A_{PAC} outputs a concept $g \in F$ such that $P(f \triangle g) \leq \varepsilon$ with a probability of at least $1 - \delta$.

The number of examples and the computational complexity required to output a concept whose error is at most ε with a probability of at least $1 - \delta$ have been studied for various concept classes [3, 9, 14].

In our framework, a set of functional dependencies corresponds to a concept and a tuple corresponds to an example. However, the framework of Definition 1 cannot be applied directly since the consistency of a set of functional dependencies is not determined for a tuple, but determined for a set of tuples. While a concept is defined as a subset of elements (a subset of Σ^*) in Definition 1, a set of functional dependencies does not correspond to a subset of tuples. The same set of functional dependencies holds not only for one subset of tuples, but also for multiple subsets of tuples. Thus, we developed a modified PAC learning framework for inferring functional dependencies. Of course, extensions of the PAC learning framework have been studied extensively [1, 4, 6, 8], but they are mainly concerned with probabilistic concepts and their results do not seem to be applicable in our case.

3 An Algorithm to Test a Functional Dependency

To learn a set of functional dependencies, we use a simple algorithm as most PAC learning algorithms do [3, 9, 14]. That is, we derive all functional dependencies consistent with example tuples. In Sections 3 and 4, we consider the case of testing one functional dependency. In Section 5, we consider the case of deriving a set of functional dependencies.

3.1 Functional Dependency

In this subsection, we briefly review the concept of the functional dependency [13]. In this paper, we fix the domain of the relation to $D_1 \times D_2 \times \cdots \times D_N$ where every D_i is a (sufficiently large) finite set. An element of $D_1 \times D_2 \times \cdots \times D_N$ is called as a tuple. A_i denotes the ith attribute and $t[A_i]$ denotes the ith attribute value of a tuple t. $t[A_{i_1} A_{i_2} \cdots A_{i_k}]$ denotes a subtuple $(t[A_{i_1}], t[A_{i_2}], \cdots, t[A_{i_k}])$. A relation T is defined as a subset of $D_1 \times D_2 \times \cdots \times D_N$.

[Definition 2] A functional dependency $A_{i_1} A_{i_2} \cdots A_{i_p} \rightarrow A_q$ is consistent with a set of tuples S if, for every $t \in S$ and $s \in S$, $(1 \leq \forall h \leq p)(t[A_{i_h}] = s[A_{i_h}]) \Rightarrow t[A_q] = s[A_q]$ holds.

Of course, a functional dependency can be defined as $A_{i_1} A_{i_2} \cdots A_{i_p} \rightarrow A_{j_1} A_{j_2} \cdots A_{j_q}$. However, we consider only the form of Definition 2 since

$$(\forall j_k)(A_{i_1} A_{i_2} \cdots A_{i_p} \rightarrow A_{j_k}) \Longleftrightarrow A_{i_1} A_{i_2} \cdots A_{i_p} \rightarrow A_{j_1} A_{j_2} \cdots A_{j_q}$$

holds. Moreover, we do not consider the case where $A_q = A_{i_h}$ holds for some A_{i_h} since such a functional dependency trivially holds for any set S. In Sections 3 and 4, we fix the functional dependency to $A_{i_1} A_{i_2} \cdots A_{i_p} \rightarrow A_q$ without loss of generality. F denotes $A_{i_1} A_{i_2} \cdots A_{i_p} \rightarrow A_q$. Moreover, for a tuple t, $L(t)$ and $R(t)$ denote $t[A_{i_1} A_{i_2} \cdots A_{i_p}]$ and $t[A_q]$, respectively.

Note that the consistency of functional dependencies is defined not for a tuple, but for a set of tuples. For example, consider a set of tuples $\{(a,1,p),(a,2,q),(b,1,p)\} \subset D_1 \times D_2 \times D_3$ and a functional dependency: $A_1 \rightarrow A_2$. In this case, it is meaningless to discuss about whether a tuple $(a,1,p)$ (or $(a,2,q)$ or $(b,1,p)$) is consistent with $A_1 \rightarrow A_2$ or not. However, it is meaningful to discuss about whether a subset of tuples is consistent with $A_1 \rightarrow A_2$ or not. For example, $\{(a,1,p),(b,1,p)\}$ is consistent and $\{(a,1,p),(a,2,q)\}$ is inconsistent. Since the conventional PAC learning framework is based on the property that the consistency of a concept is determined for each example [9], it can not be applied in our case. Moreover, the error cannot be defined in the usual way since positivity or negativity are not defined for each tuple. Thus, the conventional PAC learning framework can not be applied directly.

3.2 An Algorithm

The following algorithm (Algorithm A1) tests the consistency of F with a set of tuples S.

Procedure $TestFD(S, F)$
  begin
    for all tuples $t \in S$ **do**
      if there is a tuple $s \in S$ such that $(L(t) = L(s) \wedge R(t) \neq R(s))$
      then return FALSE;
    return TRUE
  end

There is an efficient implementation of Algorithm A1. If tuples are sorted according to $A_{i_1}A_{i_2}\cdots A_{i_p}A_q$ and N can be considered as a constant, the test is done in $O(m \log m)$ time including the time for sorting where $|S| = m$. Of course, any algorithm to test the consistency of the functional dependency can be used in place of Algorithm A1.

4 The Number of Examples for Testing a Functional Dependency

In this section, we consider the number of example tuples for testing F whose error is at least ε with the probability at least $1 - \delta$. For that purpose, we consider the probability that F is consistent with example tuples although F is inconsistent with a full tuple set (i.e. a set of all tuples in the relation). That is, we consider the probability that Algorithm A1 returns TRUE although F is inconsistent with the full tuple set. Note that error is one-sided since Algorithm A1 never returns FALSE when F is consistent with the full tuple set.

4.1 Preliminaries

[Definition 3] For a set of tuples U, $vs(U, F)$ denotes a set $\{U_1, U_2, \cdots, U_r\}$ which satisfies the following conditions:

- $(\forall U_i)(U_i \subset U)$,
- $(\forall i)(\forall j)(i \neq j \Longrightarrow U_i \cap U_j = \emptyset)$,
- $(\forall U_i)(\forall t \in U_i)(\forall s \in U_i)(L(t) = L(s))$,
- $(\forall U_i)(\exists t \in U_i)(\exists s \in U_i)(R(t) \neq R(s))$,
- Every U_i is a maximal set.

[Definition 4] For a set of tuples U, a set of violation pairs $pairs(U, F)$ is defined as

$$pairs(U, F) \equiv \{\ U_i \mid |U_i| = 2 \wedge U_i \in vs(U, F)\ \}\ .$$

Hereafter, T ($|T| = n$) denotes a full tuple set (a set of all the tuples in the relation) and $S \subset T$ ($|S| = m$) denotes a sample tuple set (a set of example tuples) in T. Since uniform distribution is assumed, each set S such that $|S| = m$ appears with the same probability, that is, with the probability $1/\binom{n}{m}$. Note that an identical tuple appears in S at most once since S is a set.

[Definition 5] For a set of tuples T and a functional dependency F, the error $\varepsilon(T, F)$ is defined as the minimum $\frac{|V|}{|T|}$ such that F is consistent with $T - V$.

The definition of the error seems reasonable since F becomes consistent if V is removed from T. In the case of an arbitrary distribution (see Section 6), the error is defined as the minimum $\sum_{t \in V} P(t)$, where $P(t)$ denotes the probability that t appears as an example. Note that an identical tuple may appear more than once in the case of an arbitrary distribution.

[Definition 6] $P(m, T, F)$ denotes the probability that F is inconsistent with S where each $S \subset T$ such that $|S| = m$ is given with the same probability $1/\binom{n}{m}$. Moreover, $Q(m, T, F)$ denotes $1 - P(m, T, F)$.

[Example 1] In this example, we consider a domain of the relation $D_1 \times D_2 \times D_3$ and a functional dependency: $A_1 \rightarrow A_2$. Let T be $\{(a,1,p),(a,1,q),(a,1,r),(a,2,p),(a,2,r),(a,3,p),(b,1,p),(b,2,q),(c,1,p),(c,1,q)\}$. Then, $vs(T, F) =$ $\{\ \{(a,1,p),(a,1,q),(a,1,r),(a,2,p),$

$(a,2,r),(a,3,p)\},\{(b,1,p),(b,2,q)\}$ } and $pairs(T,F)$ = { $\{(b,1,p),(b,2,q)\}$ }. Moreover, $\varepsilon(T,F) = 0.4$ for $V = \{(a,2,p),(a,2,r),(a,3,p),(b,1,p)\}$ or $V = \{(a,2,p),(a,2,r),(a,3,p),(b,2,q)\}$.

4.2 Upper Bound of the Number of Examples

In this subsection, we show the upper bound of the number of example tuples for testing F such that $\varepsilon(T,F) \geq \varepsilon$ ($\varepsilon > 0$) with the probability at least $1-\delta$. For that purpose, we derive the upper bound of $Q(m,T,F)$.

The following lemma shows that we need only consider the simple case (i.e., the case where $vs(T,F) = pairs(T,F)$).

[Lemma 1] For any T and F, there exists T' which satisfies the following conditions:

- $P(m,T,F) \geq P(m,T',F)$,
- $\frac{1}{2}\varepsilon(T,F) \leq \varepsilon(T',F)$,
- $|T'| = |T|$,
- $vs(T',F) = pairs(T',F)$.

(Proof) Let $\{t_1^1,\cdots,t_{n_1}^1,t_1^2,\cdots,t_{n_2}^2,\cdots,t_1^k,\cdots,t_{n_k}^k\}$ be an arbitrary element of $vs(T,F)$. We assume w.l.o.g. (without loss of generality) that $(\forall p,q)(R(t_p^i) = R(t_q^i))$ and $(\forall i,j,p,q)(i \neq j \rightarrow R(t_p^i) \neq R(t_q^j))$ hold.

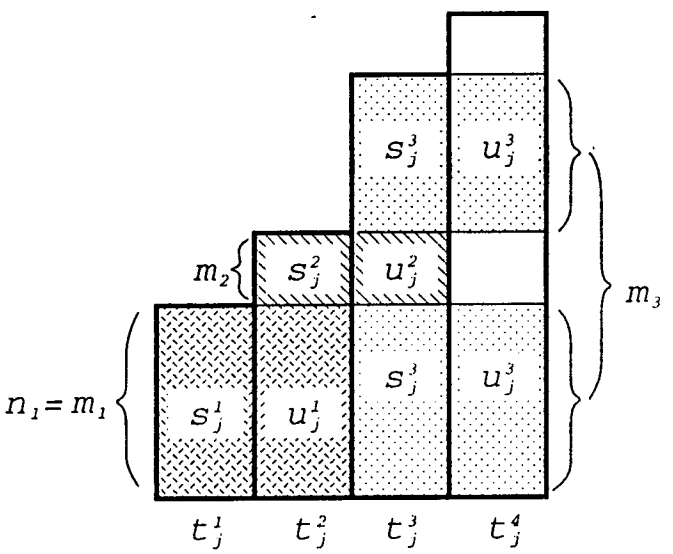

Figure 1: Construction of s_j^i, u_j^i in Lemma 1.

We assume w.l.o.g. $n_1 \leq n_2 \leq \cdots \leq n_k$. We construct T' such that a set of tuples

$$\{s_1^1,u_1^1,\cdots,s_{m_1}^1,u_{m_1}^1,\ s_1^2,u_1^2,\cdots,s_{m_2}^2,u_{m_2}^2,\ \cdots\cdots,\ s_1^{k-1},u_1^{k-1},\cdots,s_{m_{k-1}}^{k-1},u_{m_{k-1}}^{k-1}\}$$

is included and the following conditions are satisfied (see also Fig.1):

- $(\forall i,j)(L(s_j^i) = L(u_j^i) \wedge R(s_j^i) \neq R(u_j^i))$,
- $(\forall s_j^i)(\forall t \in T')((t \neq s_j^i \vee t \neq u_j^i) \rightarrow L(t) \neq L(s_j^i))$,
- $(m_1 = n_1) \wedge (\forall i \geq 1)(m_{i+1} = n_{i+1} - m_i)$.

It is easy to see that such T' satisfies the conditions of the lemma. □

Lemma 2 is used to obtain the upper bound of $Q(m,T,F)$.

[Lemma 2] Assume that $vs(T,F) = \{ \{t_1,t'_1\},\{t_2,t'_2\},\cdots,\{t_k,t'_k\} \}$, $\frac{n}{2} - 1 > k > 0$, $n > 3$ and $m > 0$ hold. When $(\forall i)(t_i \notin S \vee t'_i \notin S)$ holds, the expected number of $|S \cap \{t_1,t'_1,\cdots,t_k,t'_k\}|$ is less than $\frac{3mk}{n}$.
(Proof) We prove the lemma by induction on k. Let $E_k(m,n)$ denote the expected number of $|S \cap \{t_1,t'_1,\cdots,t_k,t'_k\}|$. First, it is easy to see that $E_k(1,n) = \frac{2k}{n} < \frac{3k}{n}$ holds for any k. Next, we assume $m > 1$.

If $k = 1$, the lemma holds since

$$E_1(m,n) = \frac{2m(n-m)}{2m(n-m)+(n-m)(n-m-1)} = \frac{2m}{n+m-1} < \frac{3m}{n} .$$

Next, we assume that the lemma holds for k. Then, the lemma holds for $k+1$ since

$$\begin{aligned} E_{k+1}(m,n) &= (\frac{2m}{n+m-1})(1+E_k(m-1,n-2)) + (1-\frac{2m}{n+m-1})E_k(m,n-2) \\ &\leq (\frac{2m}{n+m-1})(1+\frac{3(m-1)k}{n-2}) + (\frac{n-m-1}{n+m-1})(\frac{3mk}{n-2}) < \frac{3m(k+1)}{n} . \end{aligned}$$

The last inequality is derived from

$$\begin{aligned} & \frac{3m(k+1)}{n} - (\frac{2m}{n+m-1})(1+\frac{3(m-1)k}{n-2}) - (\frac{n-m-1}{n+m-1})(\frac{3mk}{n-2}) \\ = & \frac{m(-6(m-1)k+n^2+3mn-6m-5n+6)}{n(n^2+mn-2m-3n+2)} \\ = & (\frac{6m(m-1)}{n((n-3)n+(n-2)m+2)})(-k+\frac{n^2-2n}{6(m-1)}+\frac{n}{2}-1) > 0 . \end{aligned}$$

□

Lemma 3 shows the upper bound of $Q(m,T,F)$.

[Lemma 3] Assume that $n > m > 8$ and $k < \frac{n}{8}$ hold. Let $vs(T,F) = \{ \{t_1,t'_1\},\{t_2,t'_2\},\cdots, \{t_k,t'_k\} \}$. Then, $Q(m,T,F) < (1-\frac{1}{2}(\frac{m-2}{n}-\frac{6km}{n^2})^2)^k$.
(Proof) We consider the case where the elements of $\{t_i,t'_i\}$ are picked from T without being replaced, and this operation is done for k times from $i = 1$ to k. Consider the case where i pairs are already picked and $(\forall j \leq i)(t_j \notin S \vee t'_j \notin S)$ holds. Let U be a set of remained elements in T. By Lemma 2, $|U \cap S| > m - \lceil\frac{6im}{n}\rceil$ holds with a probability of more than $\frac{1}{2}$. Therefore, the probability that $(t_{i+1} \notin S \vee t'_{i+1} \notin S)$ holds is less than

$$1-\frac{1}{2}(\frac{(m-\lceil\frac{6im}{n}\rceil)(m-1-\lceil\frac{6im}{n}\rceil)}{(n-2i)(n-2i-1)}) < 1-\frac{1}{2}(\frac{m-2-\frac{6km}{n}}{n})^2$$

for any $i < k$. Note that $m-2-\frac{6km}{n} > 0$ holds since $m > 8$ and $k < \frac{n}{8}$ are assumed. By multiplying it k times, the lemma is proved. □

[Theorem 1] Assume that $\varepsilon < \frac{1}{4}$. For any T and F such that $\varepsilon(T,F) \geq \varepsilon$, the number of examples required to test F correctly with a probability of at least $1-\delta$ is $O(\sqrt{\frac{\ln \frac{1}{\delta}}{\varepsilon}}n)$.

(Proof) We assume w.l.o.g. (without loss of generality) that $n > m > 8$ holds and $\frac{\varepsilon n}{2}$ is an integer. By Lemma 1, $Q(m,T,F) \leq Q(m,T',F)$ holds for T' where $vs(T',F) = pairs(T',F)$ and $\varepsilon(T',F) \geq \frac{\varepsilon}{2}$. Thus, by Lemma 3, $Q(m,T,F) < (1 - \frac{1}{2}(\frac{m-2}{n} - \frac{6km}{n^2})^2)^k$ holds where $k = \frac{\varepsilon n}{2}$. By solving the inequality $(1 - \frac{1}{2}(\frac{m-2}{n} - \frac{6km}{n^2})^2)^k < \delta$,

we get $\quad m > \frac{1}{1-3\varepsilon}(n\sqrt{2(1-\delta^{\frac{2}{\varepsilon n}})} + 2)$. $\quad -(A1)$

Since $a^x > 1 + \ln(a)x$ holds for sufficiently small $x > 0$ where a is a constant such that $a < 1$,

$$\frac{1}{1-3\varepsilon}(n\sqrt{2(1-\delta^{\frac{2}{\varepsilon n}})} + 2) < \frac{1}{1-3\varepsilon}(2\sqrt{\frac{-\ln(\delta)}{\varepsilon}}n + 2) \qquad -(A2)$$

holds if $n \to \infty$. Therefore, the number of examples is $O(\sqrt{\frac{\ln \frac{1}{\delta}}{\varepsilon}}n)$. □

4.3 Lower Bound of the Number of Examples

In this subsection, we derive a lower bound of the number of example tuples for Algorithm A1.

[Theorem 2] For any T and F such that $\varepsilon(T,F) \geq \varepsilon$, the number of examples required to test F correctly with a probability of at least $1-\delta$ by Algorithm A1 is $\Omega(\sqrt{\frac{1-\delta}{\varepsilon}}n)$.

(Proof) We consider w.l.o.g. the case where $vs(T,F) = \{\ \{t_1, t_1'\}, \cdots, \{t_k, t_k'\}\ \}$ and $\varepsilon = \frac{k}{n}$ hold. Since the probability that $(t_i \in S \wedge t_i' \in S)$ holds is less than $(\frac{m}{n-1})^2$ for each i, $P(m,T,F) \leq k(\frac{m}{n-1})^2$ holds. Note that $P(m,T,F)$ is the probability that $(\exists i)(t_i \in S \wedge t_i' \in S)$ holds.

By solving $k(\frac{m}{n-1})^2 \leq 1-\delta$, we get $\quad m \leq \sqrt{\frac{1-\delta}{\varepsilon}}\frac{n-1}{\sqrt{n}}$. $\quad -(B)$ □

Note that a bad case ($vs(T,F) = pairs(T,F)$) is considered in this theorem and a smaller number of examples may be enough in other cases.

Here, we compare the upper bound with the lower bound. Note that the gap lies between $O(\sqrt{1-\delta})$ and $O(\sqrt{\ln \frac{1}{\delta}})$. Table 1 shows several values of the expressions (A2) and (B) where each value denotes a coefficient for $\sqrt{n}$. It is seen that the gap is not so large.

5 Inferring the Approximate Functional Dependencies

In this section, we consider the original problem, that is, to infer the approximate functional dependencies for the full tuple set from example tuples probably approximately correctly. For the problem, we consider a simple algorithm. It enumerates all functional dependencies and then tests the consistency of each dependency by means of Algorithm A1. We call it as Algorithm A2. Note that the error of Algorithm A2 is one-sided as in the case of Algorithm

Table 1: Comparison of Upper Bound and Lower Bound

ε	0.1	0.01	0.01	0.00001
δ	0.1	0.01	0.00001	0.00001
upper bound	13.7	44.2	70.0	2146.0
lower bound	3.0	9.9	10.0	316.2

(coefficients for $\sqrt{n}$)

A1. Although Algorithm A2 is not a polynomial time algorithm, it works in $O(m\log(m))$ time if the domain of the relation is fixed. Of course, any algorithm which generates a set of all functional dependencies consistent with example tuples can be used in place of Algorithm A2. For example, an algorithm developed by Kantola et al. [7] can be applied. However, it seems impossible to develop an algorithm whose time complexity is a polynomial of N since the number of functional dependencies is exponential.

[Definition 7] For a set of tuples T and a set of functional dependencies FS, the error $\varepsilon(T, FS)$ is defined as the minimum $\frac{|V|}{|T|}$ such that each functional dependency in FS is consistent with $T - V$.

Note that, for any set FS which does not contain inconsistent functional dependencies, $\varepsilon(T, FS) = 0$ holds. If we allow any set, an algorithm which always outputs an empty set can be used as a learning algorithm with error 0. Therefore, we consider only the sets in which all the consistent functional dependencies are included.

[Example 2] In this example, we consider a domain of the relation $D_1 \times D_2 \times D_3$ and a set of tuples $T = \{(a,1,p),(a,1,q),(b,2,p),(b,2,r),(c,3,q),(c,3,r),(d,1,r)\}$. The set of all consistent functional dependencies for T is $FS_0 = \{A_1 \rightarrow A_2, A_1A_3 \rightarrow A_2, A_2A_3 \rightarrow A_1\}$, and then $\varepsilon(T, FS_0) = 0$. Thus, each set must include FS_0 as its subset. $FS_1 = \{A_1 \rightarrow A_2, A_1A_3 \rightarrow A_2, A_2A_3 \rightarrow A_1, A_2 \rightarrow A_1\}$ satisfies this condition. Since FS_1 is consistent with $T - \{(d,1,r)\}$, $\varepsilon(T, FS_1) = \frac{1}{7}$.

Next, we count the number of all (consistent or inconsistent) functional dependencies. It is easy to see that the number is at most $\sum_{i=1}^{N} N\binom{N}{i} < N2^N$. Then, we get the following result.

[Theorem 3] For any T, the number of examples required to infer a set of functional dependencies FS such that $\varepsilon(T, FS) \leq \varepsilon$ holds and all functional dependencies consistent with T are included by FS with a probability of at least $1 - \delta$ is $O(\sqrt{\frac{N2^N \ln \frac{N2^N}{\delta}}{\varepsilon}} n)$.

(Proof) We assume w.l.o.g. that $N > 1$ holds and $\frac{\varepsilon n}{N2^{N+1}}$ is an integer.

Note that Algorithm A2 outputs at most $N2^N$ functional dependencies. If the error of each functional dependency does not exceed $\frac{\varepsilon}{N2^N}$, the total error does not exceed ε. If the probability that each functional dependency whose error exceeds $\frac{\varepsilon}{N2^N}$ is consistent with example

tuples is at most $\frac{\delta}{N2^N}$, the probability that at least one functional dependency whose error exceeds $\frac{\varepsilon}{N2^N}$ is derived is at most δ. Thus, by replacing ε and δ with $\frac{\varepsilon}{N2^N}$ and $\frac{\delta}{N2^N}$ respectively in the expression (A1) of Theorem 1, we get $m > (\frac{N2^N}{N2^N - 3\varepsilon})(n\sqrt{2(1 - (\frac{\delta}{N2^N})^{\frac{N2^N+1}{\varepsilon n}}) + 2})$ (C). Using a similar discussion as in Theorem 1, the theorem follows. □

Note that the upper bound shown in Theorem 3 is not optimal. A better bound might be obtained if the relation between functional dependencies is analyzed and utilized for deriving the upper bound.

6 Comparison with the Case of an Arbitrary Distribution

Here, we compare the results of this paper with the results of our previous paper [2]. First, we briefly review the results of the previous paper.

In the previous paper, an arbitrary probability distribution for tuples was considered. As in Section 2, let $P(t)$ denote the probability that a tuple $t \in T$ appears when one tuple is selected as an example tuple from T. Note that $\sum_{t \in T} P(t) = 1$ must hold. For simplicity, we allowed that the same tuple t may appear in a sample tuple set more than once. Thus, if a sample tuple set $S \subset T$ such that $|S| = m$ is selected, the probability that t appears in S is $1 - (1 - P(t))^m$. The error $\varepsilon(F, T, P)$ for a functional dependency F was defined as the minimum $\sum_{t \in V} P(t)$ such that F is consistent with $T - V$. Similarly, the error $\varepsilon(FS, T, P)$ for a set of functional dependencies FS was defined as the minimum $\sum_{t \in V} P(t)$ such that FS is consistent with $T - V$. Algorithms A_1 and A_2 were used for an arbitrary distribution, too. Under an arbitrary distribution, the sample complexities in Theorem 1 and 2 were replaced by $O(\frac{\sqrt{\ln \delta}}{\varepsilon}\sqrt{n})$ and $\Omega(\frac{\sqrt{1-\delta}}{\varepsilon}\sqrt{n})$, respectively. Thus, the both bounds for tests are improved with a factor by $O(\frac{1}{\sqrt{\varepsilon}})$ by considering the uniform distribution. The sample complexity in Theorem 3 was replaced by $O((\frac{N2^N}{\varepsilon})\sqrt{\ln \frac{N2^N}{\delta}}\sqrt{n})$ under an arbitrary distribution. Thus, the upper bound for inferring the approximate functional dependencies is improved by a factor of $O(\sqrt{\frac{N2^N}{\varepsilon}})$ by considering the uniform distribution.

Next, we compare more precise numbers of upper bounds. Under an arbitrary distribution, the number corresponding to (C) was shown to be $\frac{N2^N\sqrt{32 \ln \frac{N2^N}{\delta}}}{\varepsilon}\sqrt{n-1} + 2$. If n is sufficiently large, this number is approximately $\frac{N2^N\sqrt{32 \ln \frac{N2^N}{\delta}}}{\varepsilon}\sqrt{n}$ and the number of (C) is approximately $\frac{N2^{N+1}}{N2^N - 3\varepsilon}\sqrt{\frac{N2^N \ln \frac{N2^N}{\delta}}{\varepsilon}}\sqrt{n}$. Table 2 shows coefficients for $\sqrt{n}$ for several cases. It is seen that the number under an arbitrary distribution is far from practical, but, the number under the uniform distribution is not so large. Thus, when a uniform (or near uniform) distribution can be assumed, our method will be practical.

Table 2: Comparison of the Upper Bounds

ε	0.1	0.01	0.01	0.00001	0.01
δ	0.1	0.01	0.00001	0.01	0.01
N	4	4	4	4	10
uniform distribution	129.2	473.9	633.7	1.50×10^4	7529.0
arbitrary distribution	9202.8	1.07×10^5	1.43×10^5	1.07×10^8	2.15×10^7

(coefficients for $\sqrt{n}$)

7 Experiments

In this section, we describe the results of experiments with the sample complexity. We made experiments on the sample complexity required to test a fixed functional dependency for various ε, δ and n.

Experiments were done on a UNIX workstation using C-language. Since large real data in which approximate functional dependencies are held were not available to us, we used the following data. The domains are fixed to $D_1 \times D_2$ where each of D_1 and D_2 is the set of integers $\{1, \cdots, n+1\}$, and only the dependency $F = A_1 \rightarrow A_2$ is considered. For each ε, a relation (i.e., a set of tuples)

$$T = \{(i,i) \mid 1 \leq i \leq (1-\varepsilon)n\} \bigcup \{(i,i+1) \mid 1 \leq i \leq \varepsilon n\}$$

is created. Next, these n tuples are arranged in the appropriate order. Note that, in this case, $vs(T,F) = pairs(T,F)$ holds and the error $\varepsilon(T,F)$ is equal to ε except for the effect of truncation error. Next, for various values of m, m example tuples are chosen randomly by repeating the following procedure. A random number i from 1 to n is generated using the random() function in UNIX system, and the i'th tuple is selected as an example tuple if the same tuple has not been selected previously.

Since the number of example tuples for δ can not be obtained directly, it is obtained in the following way. For each m, a set of m tuples is chosen and tested to see whether or not $F = A_1 \rightarrow A_2$ holds for the set. This test is repeated 500 times. Let f be the number of tests such that no violation pair is found in the example tuples. Then, $\delta(m) = \frac{f}{500}$ is calculated. It corresponds to the probability δ. Then, for a given ε, δ and n, various kinds of m where $\delta(m)$ is around δ are tried and the one at which $\delta(m)$ is closest to δ is selected as the sample complexity.

The experimental relation between the sample complexity and $\sqrt{n}$ is plotted in Fig.2 and Fig.3, while the experimental relation between the sample complexity and $\frac{1}{\sqrt{\varepsilon}}$ is plotted in Fig.4 and Fig.5. They show that the sample complexity is approximately proportional to both $\sqrt{n}$ and $\frac{1}{\sqrt{\varepsilon}}$. They also coincide with both the upper bound and the lower bound except the multiplicative constant factor.

By these experiments, the theoretical result is confirmed. Furthermore, it is shown that the sample complexity obtained by the experiments is much closer to the theoretical lower bound, that is, a much smaller number of examples are sufficient than for the theoretical upper bound. For example, for $n = 100,000$, $\varepsilon = 0.01$ and $\delta = 0.1$, the sample complexity calculated from the theoretical upper bound is about 9900, while the sample complexity estimated from the experiments is about 4700.

8 Conclusion

In this paper, the number of example tuples for inferring a set of functional dependencies probably approximately correctly was studied. It has been shown that the sample complexity is improved considerably if we assume a uniform distribution in place of an arbitrary distribution. The theoretical results are confirmed by the experiments, too. Moreover, the experimental results suggest that the real sample complexity is closer to the lower bound than to the upper bound. These theoretical and experimental results imply that the sample complexity is not too large for practical applications when a uniform distribution can be assumed. Of course, we cannot always assume the uniform distribution. However, an arbitrary distribution seems to be too general since the worst case of probability distribution is included in the case of an arbitrary distribution. We think that there are many practical cases where distributions can be regarded as a uniform distribution or near uniform distribution. Moreover, our method can be used in such a way as described in [10] where the uniform distribution can be assumed. Anyway, our method should be tested with real-world data since we do not know which distribution should be used. We plan to make experiments for multiple functional dependencies with real-world data.

A simple method for inferring approximate functional dependencies, that is, a method for enumerating only the dependencies consistent with examples, is considered in this paper. However, other methods may work better depending on applications. For example, a method which outputs a set of functional dependencies whose error for example tuples is very small can be considered. Such a method seems to be more robust than the method described in this paper. Therefore, other methods for finding a set of functional dependencies probably approximately correctly should be developed and examined.

Although only functional dependencies are considered in this paper, our framework might be applied to other types of constraints such as multi-valued dependencies and more general logical formulae. Such extensions would be very useful and should be studied.

References

[1] N. Abe, J. Takeuchi, and M. K. Warmuth. "Polynomial learnability of probabilistic concepts with respect to the Kullback-Leibler divergence". In *Proceedings of the Forth Annual Workshop on Computational Learning Theory*, pp. 277–289, 1991.

[2] T. Akutsu and A. Takasu. "On PAC learnability of functional dependencies". In *Proceedings of the Third Workshop on Algorithmic Learning Theory (ALT'92)*, pp. 229–239, 1992.

[3] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. "Learnability and the Vapnik-Chervonenkis dimension". *Journal of the ACM*, Vol. 36, pp. 929–965, 1990.

[4] P. Fischer, S. Pölt, and H. U. Simon. "Probably almost bayes decisions". In *Proceedings of the Forth Annual Workshop on Computational Learning Theory*, pp. 88–94, 1991.

[5] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. "Knowledge Discovery in Databases: An Overview". In *Knowledge Discovery in Databases*, pp. 1–27. AAAI Press, 1991.

[6] D Haussler. "Generalizing the PAC model: sample size bounds from metric dimension-based uniform convergence results". In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pp. 40–45, 1989.

[7] M. Kantola, H. Mannila, K. Rälhä, and H. Siirtola. "Discovering functional and inclusion dependencies in relational databases". *International Journal of Intelligent Systems*, Vol. 7, pp. 591–607, 1992.

[8] J. K. Kearns, R. E. Schapire, and M. S. Linda. "Towards efficient agnostic learning". In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 341–353, 1992.

[9] B. K. Natarajan. *"Machine Learning - A Theoretical Approach"*. Morgan Kaufmann, CA, 1991.

[10] G. Piatetsky-Shapiro. "Discovery and analysis of strong rules in databases". In *Proceedings of Advanced Database System Symposium '89*, pp. 135–142, 1989.

[11] J. R. Quinlan and R. L. Rivest. "Inferring decision trees using the minimum description length principle". *Information and Computation*, Vol. 80, pp. 227–248, 1989.

[12] K. Sonoo, H. Kawano, S. Nishio, and T. Hasegawa. "Accuracy evaluation of rules derived from sample data in VLKD". In *Proceedings of the 5th Conference of Japanese Society of Artificial Intelligence*, pp. 181–184, (in Japanese) 1991.

[13] J. D. Ullman. *"Principles of Database and Knowledge-Base Systems - Volume 1"*. Computer Science Press, 1988.

[14] L. G. Valiant. "A theory of the learnable". *Communications of the ACM*, Vol. 27, pp. 1134–1142, 1984.

[15] W. Ziarko. "The discovery, analysis, and representation of data dependencies in databases". In *Knowledge Discovery in Databases*, pp. 195–209. AAAI Press, 1991.

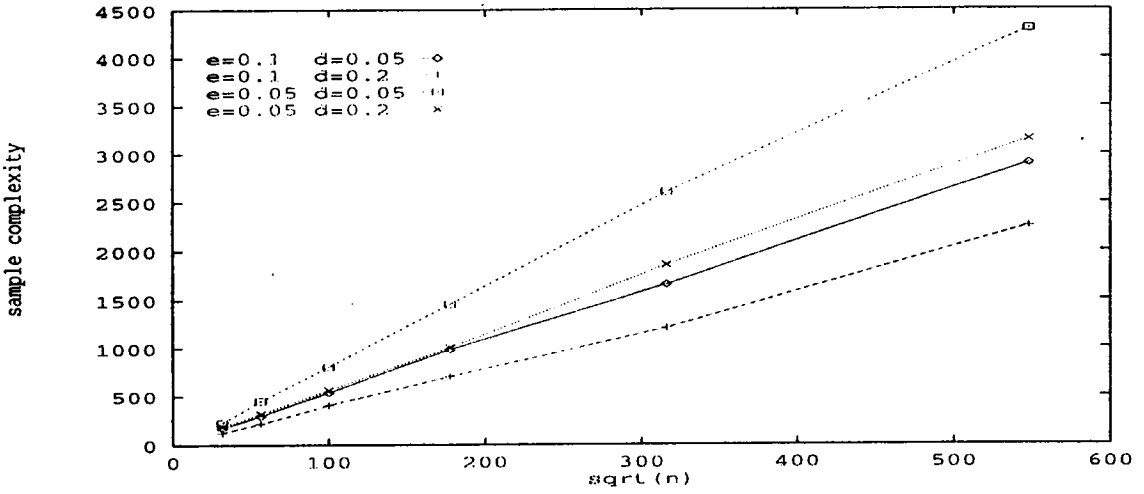

Figure 2: Relation Between Size and Sample Complexity (d denotes δ and e denotes ε).

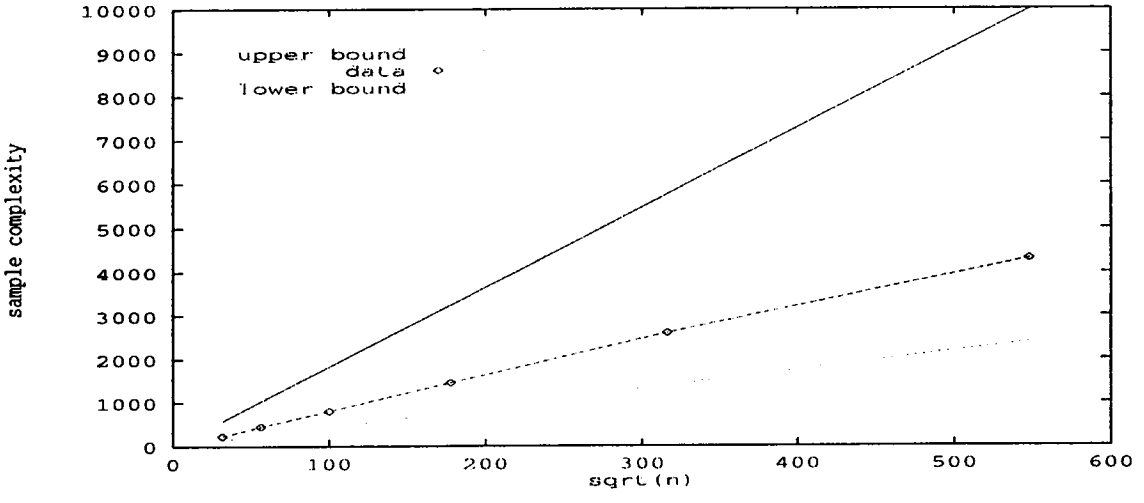

Figure 3: Comparison of Experimental Data with Theoretical Bounds ($\varepsilon = 0.05, \delta = 0.05$).

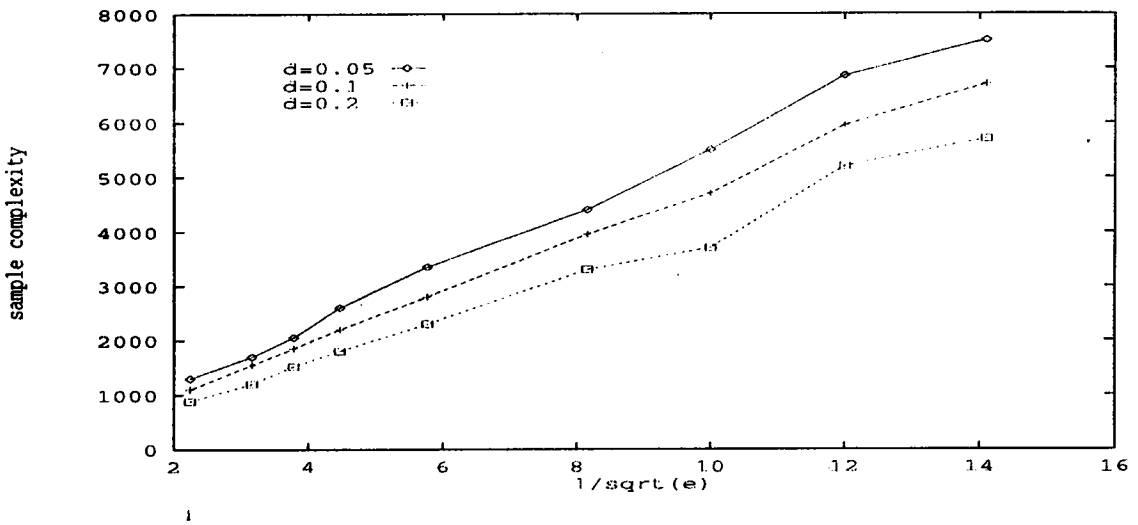

Figure 4: Relation Between Error and Sample Complexity ($n = 100,000$).

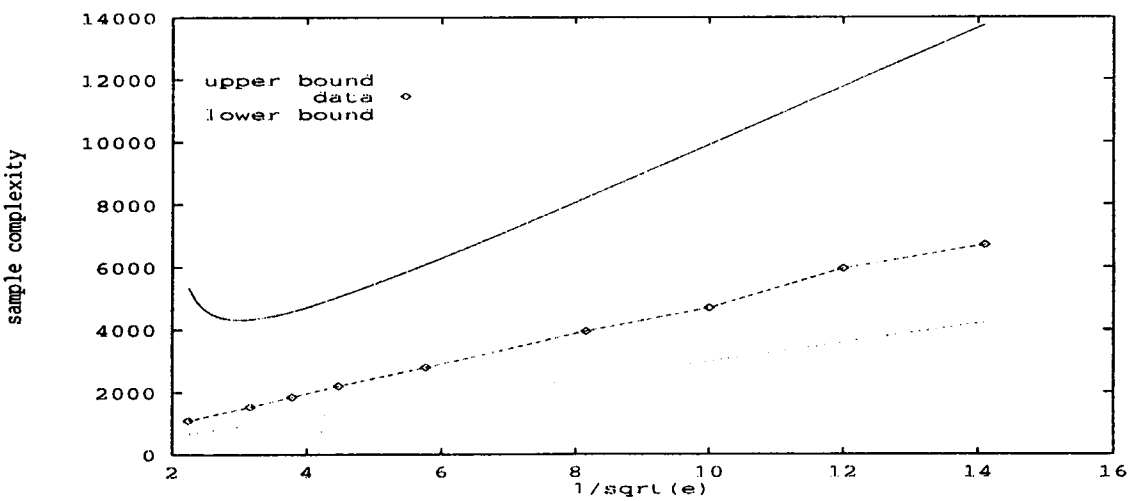

Figure 5: Comparison of Experimental Data with Theoretical Bounds ($n = 100,000, \delta = 0.1$).