A Simple Greedy Algorithm for Finding Functional Relations: Efficient Implementation and Average Case Analysis

Tatsuya Akutsu\textsuperscript{1}, Satoru Miyano\textsuperscript{1}, and Satoru Kuhara\textsuperscript{2}

\textsuperscript{1} Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan
\{akutsu,miyano\}@ims.u-tokyo.ac.jp

\textsuperscript{2} Graduate School of Genetic Resources Technology, Kyushu University, Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
kuhara@grt.kyushu-u.ac.jp

Abstract. Inferring functional relations from relational databases is important for discovery of scientific knowledge because many experimental data in science are represented in the form of tables and many rules are represented in the form of functions. A simple greedy algorithm has been known as an approximation algorithm for this problem. In this algorithm, the original problem is reduced to the set cover problem and a well-known greedy algorithm for the set cover is applied. This paper shows an efficient implementation of this algorithm that is specialized for inference of functional relations. If one functional relation for one output variable is required, each iteration step of the greedy algorithm can be executed in linear time. If functional relations for multiple output variables are required, it uses fast matrix multiplication in order to obtain non-trivial time complexity bound. In the former case, the algorithm is very simple and thus practical. This paper also shows that the algorithm can find an exact solution for simple functions if input data for each function are generated uniformly at random and the size of the domain is bounded by a constant. Results of preliminary computational experiments on the algorithm are described too.

1 Introduction

Many scientific rules are represented in the form of functions. For example, an output value $y_j$ may be a function of several input variables $x_{i_1}, \ldots, x_{i_d}$ (i.e., $y_j = f_j(x_{i_1}, \ldots, x_{i_d})$). For another example, a simple differential equation of the form $\frac{dy_j}{dx} = f_j(x_{i_1}, \ldots, x_{i_d})$ can also be considered as a function if we can know the values of $\frac{dy_j}{dx}$ (e.g., using $\frac{\Delta y_j}{\Delta x}$ in place of $\frac{dy_j}{dx}$). Moreover, many experimental data in sciences are represented in the form of tables. Therefore, inferring functional relations from tables is important for scientific discovery. Since a relational database consists of tables, this problem is almost equivalent to inference of functional relations from relational databases.

© Springer-Verlag Berlin Heidelberg 2000
A Simple Greedy Algorithm for Finding Functional Relations

Inference of functional relations (or almost equivalently, inference of functional dependencies) from relational databases is rather a classical problem in the field of KDD (knowledge discovery in databases) [2,9–11]. Since \( y_j = f^i_j(x_{i_1}, \ldots, x_{i_d}) \) holds for all \( x_{i_{d+1}}, \ldots, x_{i_d} \), if \( y_j = f^i_j(x_{i_1}, \ldots, x_{i_d}) \) holds, it is usually required to find the minimum set or the minimal sets of input variables. Mannila and Räihä proposed a heuristic algorithm for finding functional dependencies [10]. Inference of functional dependencies with small noises was also studied and PAC-type analysis was made [2,9]. Unfortunately, Mannila and Räihä proved that finding a functional dependency with the minimum number of input attributes (i.e., \( d \) is minimum) is NP-hard [11]. Therefore, development of heuristic algorithms and/or approximation algorithms is important. As mentioned before, Mannila and Räihä proposed a heuristic algorithm [10]. Akutsu and Bao proposed a simple greedy algorithm in which the original problem was reduced to the set cover problem [3]. Although an upper bound on the approximation ratio (on \( d \)) is given, the time complexity is not low if it is implemented as it is. Even for finding a functional relation for one output variable (i.e., one \( y_j \)), it takes \( O(m^2ng) \) time (\( O(m^2n) \) time using an efficient implementation for the set cover problem [7]), where \( n \) denotes the number of attributes, \( m \) denotes the number of tuples and \( g \) denotes the number of main iterations in the greedy algorithm. This time complexity is too high for applying the greedy algorithm to large databases.

This paper gives a simple implementation of the greedy algorithm, which runs in \( O(mng) \) time in the case of finding a functional relation for one output variable. Each iteration can be done in linear time since the size of input data (i.e., input table) is \( O(mn) \). This complexity is reasonable because \( g \) is usually small (e.g., \( < 10 \)). This algorithm has some similarity with decision tree construction algorithms, where the similarity and difference are to be discussed in the final section. By the way, in some applications, it is required to infer functional relations for multiple output variables simultaneously. For example, in inference of genetic networks [4], functional relations should be inferred for all genes (i.e., for \( n \) genes). In such a case, \( O(m^2ng) \) time is still required using the efficient implementation mentioned above. Therefore, we developed an improved algorithm for a special case in which \( g \) can be regarded as a constant and the size of the domain is bounded by a constant. This algorithm is based on a fast matrix multiplication algorithm [6] as in [4], and the time complexity is \( O(m^{\omega-2}n^2 + mn^{2+\omega-3}) \), where \( \omega \) is the exponent of matrix multiplication (currently, \( \omega < 2.376 [6] \)). Although this algorithm is not practical, it is faster than the \( O(m^{2g}) \) time algorithm when \( m \) is large.

This paper also gives an average case analysis of the greedy algorithm for simple functions (such as AND of literals, OR of literals), under the condition that input data are generated uniformly at random and the size of the domain is bounded by a constant. In this case, the greedy algorithm finds an exact solution with high probability, where the probability is taken over all possible input data. This gives another theoretical guarantee to the algorithm. Recall that it is already known that the greedy algorithm outputs a solution with a guaranteed