Monotonic and Non-monotonic Inductive Inference

Klaus P. JANTKE

Department of Mathematics and Informatics
Leipzig University of Technology
P.O. Box 66, Leipzig 7030, Germany

Received 21 November 1990

Abstract The main goal is to exhibit the relationship between research work on non-monotonic reasoning and recursion-theoretically based approaches to inductive learning. There are introduced the concepts of monotonic and weakly monotonic inductive inference. It is proved that these concepts are considerably distinguished from other classical concepts of inductive inference, i.e. non-monotonic reasoning is inherently required in several approaches to inductive inference.

Keywords: Machine Learning, Inductive Inference, Non-monotonic Reasoning

1 Introduction

The research work presented here originates from the author's visit to Karlsruhe University in summer 1989, where Peter H. Schmitt raised the question about the relationship between non-monotonic reasoning and inductive inference. The results presented below are intended to exhibit the importance of non-monotonic reasoning for inductive inference.

There is some evidence of a deeper relationship between both areas. For example, Shapiro's technique of contradiction backtracing for inductively synthesizing Prolog programs (cf. [4]) turns out to be a particular reason maintenance technique.

This paper is based on a couple of well-studied problem classes (cf. [1, 3]). The author introduces two new problem classes $\mathcal{MON}$ and $\mathcal{WMON}$ denoting inductive inference problems which are uniformly solvable by monotonic resp. weakly
monotonic inductive inference algorithms. Their relation to the other problem classes illuminates the necessity of non-monotonic reasoning.

Inductive inference usually means the area of algorithmic learning from possibly incomplete information. The paper [1] by D. Angluin and C. H. Smith is an excellent survey in this regard. In [2] the author has recently given an easy introduction. Inductive inference of total recursive functions is one of the best developed inductive inference areas. A large number of results (cf. [1, 3]) illustrate particular inductive inference techniques as well as the effects of special approaches and restrictions. Basic results apply to several areas of learning from incomplete information.

In Chapter 2 below there are introduced basic notions and notations. Chapter 3 is dedicated to the introduction of monotonic inductive inference. Chapter 4 contains the results relating monotonic inductive inference to classical inductive inference. It follows a summary and the list of references.

2 Notions and Notations

By \( N \) we denote the set of natural numbers. \( P^n \) and \( R^n \) denote the classes of \( n \)-ary partial recursive resp. total recursive functions. If \( n \) equals 1, it will usually be omitted. Total recursive functions are considered to be objects to be identified (learned, synthesized) from incomplete information.

Finite sets of input/output examples are the weakest imaginable form of information about some target function. A lot of quite illustrative work (cf. [1]) illuminates the possibility of learning recursive functions from input/output examples only. It is quite important whether or not there can be assumed a certain fixed ordering underlying the information presented. (The reader should consult [3] in this regard.) If any ordering is acceptable, it is represented by any complete sequence \( X \) of natural numbers. For the problems investigated here, one may assume that \( X \) is any permutation of \( N \), i.e. \( X \) is repetition-free. The standard sequence of natural numbers \( X_0 = \{0, 1, 2, \ldots\} \) is usually dropped in every notation. If a finite list of input/output examples about some total recursive function \( f \) is given as \( f(x_0), f(x_1), \ldots, f(x_n) \) with respect to some ordering \( X = \{x_0, x_1, x_2, \ldots\} \), this sequence is abbreviated (effectively encoded) by \( f_X[n] \). Similarly, \( f[n] \) refers to \( X_0 \). At some places, it is necessary to refer to the corresponding list of arguments \( x_0, \ldots, x_n \) abbreviated by \( X[n] \).

If one is mainly interested in fundamental questions of solvability or unsolvability of learning problems, it may be sufficient to consider learning methods as any recursively computable devices. Therefore, we allow any partial recursive function \( S \) to compete for solving a given learning problem. If some information \( f_X[n] \) is presented to \( S \), the corresponding hypothesis generated by \( S \) is denoted by \( S(X[n], f_X[n]) \). If the underlying ordering is known to be \( X_0 \), the notation may be shortened to \( S(f[n]) \).

Every hypothesis is intended to describe some recursive function. Therefore, we have to assume some semantics on the space of hypotheses. A mathematically