Comprehensibility of Data Mining Algorithms

Zhi-Hua Zhou
Nanjing University, China

INTRODUCTION

Data mining attempts to identify valid, novel, potentially useful, and ultimately understandable patterns from huge volume of data. The mined patterns must be ultimately understandable because the purpose of data mining is to aid decision-making. If the decision-makers cannot understand what does a mined pattern mean, then the pattern cannot be used well. Since most decision-makers are not data mining experts, ideally, the patterns should be in a style comprehensible to common people. So, comprehensibility of data mining algorithms, that is, the ability of a data mining algorithm to produce patterns understandable to human beings, is an important factor.

BACKGROUND

A data mining algorithm is usually inherently associated with some representations for the patterns it mines. Therefore, an important aspect of a data mining algorithm is the comprehensibility of the representations it forms. That is, whether or not the algorithm encodes the patterns it mines in such a way that they can be inspected and understood by human beings. Actually, such an importance has been argued by a machine learning pioneer many years ago (Michalski, 1983):

The results of computer induction should be symbolic descriptions of given entities, semantically and structurally similar to those a human expert might produce observing the same entities. Components of these descriptions should be comprehensible as single 'chunks' of information, directly interpretable in natural language, and should relate quantitative and qualitative concepts in an integrated fashion.

Craven and Shavlik (1995) have indicated a number of concrete reasons why the comprehensibility of machine learning algorithms is very important. With slight modification, these reasons are also applicable to data mining algorithms.

- **Validation:** If the designers and end-users of a data mining algorithm are to be confident in the performance of the algorithm, they must understand how it arrives at its decisions.
- **Discovery:** Data mining algorithms may play an important role in the process of scientific discovery. An algorithm may discover salient features and relationships in the input data whose importance was not previously recognized. If the patterns mined by the algorithm are comprehensible, then these discoveries can be made accessible to human review.
- **Explanation:** In some domains, it is desirable to be able to explain actions a data mining algorithm suggest take for individual input patterns. If the mined patterns are understandable in such a domain, then explanations of the suggested actions on a particular case can be garnered.
- **Improving performance:** The feature representation used for a data mining task can have a significant impact on how well an algorithm is able to mine. Mined patterns that can be understood and analyzed may provide insight into devising better feature representations.
- **Refinement:** Data mining algorithms can be used to refine approximately-correct domain theories. In order to complete the theory-refinement process, it is important to be able to express, in a comprehensible manner, the changes that have been imparted to the theory during mining.

MAIN THRUST

It is evident that data mining algorithms with good comprehensibility are very desirable. Unfortunately, most data mining algorithms are not very comprehensible and therefore their comprehensibility has to be enhanced by extra mechanisms. Since there are many different data mining tasks and corresponding data mining algorithms, it is difficult for such a short article to cover all of them. So, the following discussions are restricted to the comprehensibility of classification algorithms, but some essence is also applicable to other kinds of data mining algorithms.

Some classification algorithms are deemed as comprehensible because the patterns they mine are expressed in an explicit way. Representatives are decision tree algorithms that encode the mined patterns in the form of a

decision tree which can be easily inspected. Some other classification algorithms are deemed as incomprehensible because the patterns they mine are expressed in an implicit way. Representatives are artificial neural networks that encode the mined patterns in real-valued connection weights. Actually, many methods have been developed to improve the comprehensibility of incomprehensible classification algorithms, especially for artificial neural networks.

The main scheme for improving the comprehensibility of artificial neural networks is *rule extraction*, that is, extracting symbolic rules from trained artificial neural networks. It originates from Gallant's work on connectionist expert system (Gallant, 1983). Good reviews can be found in (Andrews, Diederich, & Tickle, 1995; Tickle, Andrews, Golea, & Diederich, 1998). Roughly speaking, current rule extraction algorithms can be categorized into four categories, namely the *decompositional*, *pedagogical*, *eclectic*, or *compositional* algorithms. Each category is illustrated with an example below.

The decompositional algorithms extract rules from each unit in an artificial neural network and then aggregate. A representative is the RX algorithm (Setiono, 1997), which prunes the network and discretizes outputs of hidden units for reducing computational complexity in examining the network. If a hidden unit has many connections then it is split into several output units and some new hidden units are introduced to construct a subnetwork, so that the rule extraction process is iteratively executed. The RX algorithm is summarized in Table 1.

The pedagogical algorithms regard the trained artificial neural network as an *opaque* and aim to extract rules that map inputs directly into outputs. A representative is the TREPAN algorithm (Craven & Shavlik, 1996), which regards the rule extraction process as an inductive learning problem and uses oracle queries to induce an ID2-of-3 decision tree that approximates the concept represented by a given network. The pseudo-code of this algorithm is shown in Table 2.

The eclectic algorithms incorporate elements of both the decompositional and pedagogical ones. A representative is the DEDEC algorithm (Tickle, Orlowski, & Diederich, 1996), which extracts a set of rules to reflect the functional dependencies between the inputs and the outputs of the artificial neural networks. Fig. 1 shows its working routine.

The compositional algorithms are not strictly decompositional because they do not extract rules from individual units with subsequent aggregation to form a global relationship, nor do them fit into the eclectic category because there is no aspect that fits the pedagogical profile. Algorithms belonging to this category are mainly designed for extracting deterministic finite-state automata (DFA) from recurrent artificial neural networks. A representative is the algorithm proposed by Omlin and Giles (1996), which exploits the phenomenon that the outputs of the recurrent state units tend to cluster, and if each cluster is regarded as a state of a DFA then the relationship between different outputs can be used to set up the transitions between different states. For example, assuming there are two recurrent state units s_0 and s_1, and their outputs appear as nine clusters, then the working style of the algorithm is shown in Fig. 2.

During the past years, powerful classification algorithms have been developed in the ensemble learning area. An ensemble of classifiers works through training multiple classifiers and then combining their predictions, which is usually much more accurate than a single classifier (Dietterich, 2002). However, since the classification is made by a collection of classifiers, the comprehensibility of an ensemble is poor even when its component classifiers are comprehensible.

A pedagogical algorithm has been proposed by Zhou, Jiang, and Chen (2003) to improve the comprehensibility of ensembles of artificial neural networks, which utilizes the trained ensemble to generate instances and then extracts symbolic rules from them. The success of this

Table 1. The RX algorithm

1. Train and prune the artificial neural network.
2. Discretize the activation values of the hidden units by clustering.
3. Generate rules that describe the network outputs using the discretized activation values.
4. For each hidden unit:
    1) If the number of input connections is less than an upper bound, then extract rules to describe the activation values in terms of the inputs.
    2) Else form a subnetwork:
        (a) Set the number of output units equal to the number of discrete activation values. Treat each discrete activation value as a target output.
        (b) Set the number of input units equal to the inputs connected to the hidden units.
        (c) Introduce a new hidden layer.
        (d) Apply RX to this subnetwork.
5. Generate rules that relate the inputs and the outputs by merging rules generated in Steps 3 and 4.