Usefulness the Multi-Valued Function in Machine Learning

Katja B. Rangelov[1]

Abstract – **Learning from data is the central theme of Knowledge Discovery in Databases (KDD) and Machine Learning (ML) community. The increase of data volume caused greater difficulties to extract useful information for decision support or different analysis tasks. The purpose of this paper is to demonstrate the applicability i.o. similarity and differences between machine learning and logic synthesis when is used functional decomposition and multi-valued functions.**

Key words **- functional decomposition, multiple–valued function, machine learning, knowledge, data discovery, inductive learning.**

1. INTRODUCTION

Machine learning is the sub-field of artificial intelligence (AI) concerned with intelligent systems that learn. To understand machine learning, it is helpful to have a clear notion of intelligent systems. This paper is concerned a view of intelligent systems — systems that perceive and act in an environment; the system is intelligent to the degree that its actions are successful.

Machine learning research is relevant to the goals of both artificial intelligence and cognitive psychology. At present, humans are *much better* learners, for the most part, than either machine learning programs or psychological models. Except in certain artificial circumstances, the overwhelming deficiency of current psychological models of learning is their complete incompetence as learners. Since the goal of machine learning is to make better learning mechanisms, and to understand them, results from machine learning will be useful to psychologists at least until machine learning systems approach or surpass humans in their general learning capabilities. All of the issues that come up in machine learning — generalization ability, handling noisy input, using prior knowledge, handling complex environments, forming new concepts, active exploration and so on — are also issues in the psychology of learning and development. Theoretical results on the computational (in)tractability of certain learning tasks apply equally to machines and to humans.

Finally, some artificial intelligence's system designs, such as Newell's SOAR architecture, are also intended as cognitive models. However, that it is often difficult to interpret human learning performance in terms of specific mechanisms.

Learning is often viewed as the most fundamental aspect of intelligence, as it enables the system, also called agent [11], to become independent of its creator. It is an essential component of an agent design whenever the designer has incomplete knowledge of the task environment. It therefore provides *autonomy* in that the agent is not dependent on the designer's knowledge for its success, and can free it from the assumptions built into its initial configuration. Learning may also be the only route by which it can construct very complex intelligent systems. In many application domains, the state-of-the-art systems are constructed by a learning process rather than by traditional programming or knowledge engineering.
The basic problem studied in machine learning has been that of inducing a representation of a function — a systematic relationship between inputs and outputs — from examples.

In general, the way the function is learned is that the feedback is used to indicate the correct (or approximately correct) value of the function for particular inputs, and the agent's representation of the function is altered to try to make it match the information provided by the feedback. Obviously, this process will vary depending on the choice of representation. In each case, however, the generic task—to construct a good representation of the desired function from correct examples—remains the same. This task is commonly called *induction* or *inductive inference* . The term *supervised learning* is also used, to indicate that correct output values are provided for each example.

To specify the task formally, we need to say exactly what we mean by an *example* of a function. Suppose that the function f maps from domain X to range Y (that is, it takes an X as input and outputs a Y). Then an example of f is a pair (x, y) where $x \ 0 \ X$, $y \ 0 \ Y$ and $y = f(x)$, i.e. an example is an input/output pair for the function.

It can define the task of *pure inductive inference*: given a collection of examples of f, return a function h that approximates f as closely as possible. The function returned is called a *hypothesis.* A hypothesis is *consistent* with a set of examples if it returns the correct output for each example, given the input [13,14]. We say that h *agrees* with f on the set of examples. A hypothesis is *correct* if it agrees with f on all possible examples.

To illustrate this definition, suppose it has an automated taxi that learning to drive by watching a teacher. Each example includes not only a description of the current state, represented by the camera input and various measurements from sensors, but also the correct action to do in that state, obtained by "watching over the teacher's shoulder." Given sufficient examples, the induced hypothesis provides a reasonable approximation to a driving function that can be used to control the vehicle [12].
The other important basic category is representation. *Attribute-based representations*: this category includes all *Boolean functions*—functions that provides a yes/no answer

[1]Katja B. Rangelov is with the Technical University of Sofia, the Faculty of Computer Systems and Control, 1756 Sofia, Bulgaria, E-mail: krang@tu-sofia.bg, krang@abv.bg,

based on logical combinations of yes/no input attributes. Attributes can also have multiple values.

2. MACHINE LEARNING

In machine learning the idea is to find patterns in the data, such that the data can be partitioned into smaller concepts (data blocks), which correspond to the sub-blocks of the decomposition. The principle of using decomposition in machine learning is to reduce a given function specified by a set of care minterms (samples or examples) to a composition of smaller function (concepts).

2.1 MACHINE LEARNING EXAMPLE

An **example** is described by the input attributes and the value of the output of the function (target concept). A complete set of examples is called the **training set**. A training set does not imply that the set fully explores all possible inputs and outputs of a function. For example, in the case of a cancer database, doctors may take data about a person's age, sex, height, weight, x-rays, etc., and whether the person has cancer or not. The number of input data values (variables) that are taken can be enormous while the number of patients that are evaluated in the database can be quite small. For instance, if a doctor asks 20 questions about sex, height, etc., but only evaluates 500 patients, then the number of examples is quite small compared to the number of possible outcomes given the number of questions. In this example, if each question was a yes/no question, then there are $2^{20} \approx$ 1-million combinations given different values of the inputs, while only 500 patients were evaluated. The training set is only a small representation of the possible values that exist, each example is only a small piece or example of the actual function [7].

Mapping from machine learning to logic synthesis is done by mapping the examples to care terms. But, the number of don't knows in the function that will be evaluated with logic synthesis algorithms is quite large. In fact, it is thought that most machine learning problems have more than 99.9% output don't knows. To use a logic synthesis algorithm on machine learning problems, it must be efficient for functions with a significantly large number of don't knows.

The difference between machine learning and logic synthesis is the percentage of known output values for a set of machine learning functions. On some small functions, the functions are completely specified. In general though, most of the functions have a very small percentage of known output values. In fact, a percentage of unknown values that is much larger than that of the 99.9% unknown values discussed above.

2.2. MISSING ATTRIBUTES

The possibility exists that data collected could have *missing attributes* in the example. A missing attribute is represented by an input don't know to denote the missing data. To associate the concepts of a missing attribute to that of logic synthesis, input doesn't know is akin to input don't care. In logic synthesis, an input A is don't care when, given a set of all other inputs, all possible values of A do not result in a change in the output of the function.

Definition 3. *An* **input don't care** *is defined as covering all possible values for a given variable.*

In machine learning, the possibility exists that data collected could have a missing attribute [1, 9, 10] in the data.

Definition 4. *An* **input don't know** *is defined as representing one (unknown) data value, not all possible values for a given variable.*

For example [1], a questionnaire asks for a person's age. It is possible that the person does not want to state his/her age; in general, instead of guessing at an approximate age, a don't know is placed in the category. In terms of logic synthesis, a don't care implies that the person is all the ages between 0 and 120. While a don't know implies that a person has only one age between 0 and 120.

One possible method of determining missing attributes is to find two input cubes C_1 and C_2 that intersect and have the same output value. Thus, $C_1 \cap C_2 \neq \emptyset$ and the minimal solution for the missing attributes for these two cubes is $C_1 \cap C_2$.

Another method of determining missing attributes is to use probabilities. Probabilities are used to determine the value of a missing attribute by determining how probable a value is. A baseline probability is defined as any value occurring more than X-times. If this occurs, then the value is accepted and a don't know can represent all values in the range when the baseline probability is zero or it is possible that no values are in the range. This methodology can be performed by using a pre-processing algorithm that determines (probabilistically) what the don't know values should be changed too.

Another method of determining missing attributes is to use multi-valued relations [2], but the relations are only used to indefiy overlapping cubes. The relations treat don't know as don't cares, but overlapping cubes of different valued .

For logic synthesis methods to be used on machine learning problems, missing attributes/input don't knows need to be represented in the data structure. A possible representation is to create a new cofactor for each variable that contains don't know information.

2.3. REATAINING MVL SYNTHESIS TECHNIQUES

Because Boolean logic synthesis techniques have been practiced since the 1950s, they are well known and fairly well understood compared to multi-valued logic synthesis. That does not imply that they are the future of computing. It also does not imply that machine learning problems should be encoded with Boolean expressions before analysis. The encoding of multi-valued expressions to Boolean expressions may destroy data either by adding or removing components inherent in the multi-valued expressions of the function or data.

Given a four-valued variable a, i.e., a represents the values $\{0,1,2,3\}$. If Boolean encoding is used, two new variables are created, a_1 and a_2, derived from the following encoding:

a	0123
a_0	0011
a_1	0101.

Given two four-valued variables, b and c, that are encoded as shown above into four new variables: b_0, b_1, c_0 and c_1. From the encoded values, say that there is a correlation of the $\{0,1\}$ attributes of b with the $\{2,3\}$ attributes of c. Because variables b and c have been encoded, the only way that the correlation between the two-variable attributes will be found is if all four encoded variables are used. Also, if b_0 and c_0 are found to have a correlation, what is actually learned from this? Because b_0 and c_0 are encoded variables they are subsets of the original function but do not contain enough information about b and c to be useful alone.

Boolean encoding can also add values that are not in the original functional representation. Given that a is a three-valued variable then a can be encoded as follows:

a	012
a_0	001
a_1	010.

There is no instance when $a_0 = 1$ and $a_1 = 1$. Where encoding is used in decomposition, the case that $a_0 = 1$ and $a_1 = 1$ is a don't know/don't care case. This adds don't cares to the representation of the function which makes the function more complex.

One important aspect of machine learning that is not highly publicized is that an expert system will usually perform best by the way the data is given to it by a user. This implies that if the user has some understanding of the data being inputted into the expert system, the user may control the way the machine learning algorithms are applied to the data. When the data is encoded to Boolean expressions, the user may loose all intuition of what the data imply and how variables interact with each other.

In machine learning it is sometimes important that the algorithms find hypothesis functions are not a *black-box* concept, but knowledge-oriented, concept-learning systems[10]. It is important that the system communicates the form of the hypothesis function in an operationally effective symbolic form. This implies that the hypothesis function is expressible in a form that a human may understand and use to extract information about the function. The class of functions that include artificial neural networks and statistical methods (including AIM), create *black-box* representations of the hypothesis function and are not understandable. But, the result of functional decomposition methods is in the class of hypothesis functions that are highly understandable. This is another reason to use functional decomposition on machine learning problems. But, if the data has been encoded to Boolean values, it is possible that data in a form that is obvious to a human, has been changed drastically (during the encoding) and the hypothesis function may mean nothing to a human trying to evaluate it.

Each of the above discussions illustrate why it is important to leave machine learning problems in their original form, and why it is important to use multi-valued logic synthesis techniques, instead of Boolean techniques.

2.4. NOISE IN THE DATA

Another issue in machine learning that is different from logic synthesis is the concept of noise.

Definition: Noise *is defined as faulty values in either the attributes or the output in an example of the function.*

Noise in machine learning may be caused by many different things. For example, noise could occur in writing down some data, writing the data down in the wrong place or writing down the wrong information, etc. This can cause problems in both the machine learning algorithms and the logic synthesis algorithms. How do you detect that the example is actually incorrect and not the correct value?

There are two types of noise, noise that is in the input, and noise that is generated on the output. The distinction between input and output noise is that input noise causes the output to be associated with the wrong set of attributes, while output noise occurs when the output value is incorrect for a given attribute. Both types of noise cause different problems in determining that noise is actually in the data set. But the removal of noise is the same for both types of noise. The following is the one of solution [1,8] for the removal of noise:

It is assumed that the amount of error in a data set is a percentage of the total number of data points in the set. By understanding the data, a user may be able to give the percentage of error that may exist. This percentage may then be used to determine compatible elements. Two sets are compatible if the percentage of incompatible elements (compared to compatible elements) in the sets is less than the error percentage supplied by the user. For instance, given two sets with 1000 elements in each and both sets are equal except for one pairing, then should this pairing be thought of as noise? If the user states that the amount of error in the data is 0%, then the sets are not compatible. If the percentage of allowed error is greater than $\frac{1}{1000} \times 100\%$, then the sets are compatible. The use of an allowed error percentage should be variable because there may not be any noise on some data sets, while other data sets may have a large amount of error. Of course, low thresholds may allow sets to be combined, showing that the noise exists, when in fact there are no incorrect values.

Another problem that may be encountered is what happens when the algorithm finds noisy data and what should the noisy values become? If given two values (assuming that one of the

values is noise, but it is not known which value is noisy), should the resulting value become one of the values or an average of the two values, a don't know, etc. Noise is notably a very deep theoretical issue in machine learning, and needs to be explored further.

2.5. CONTINUOUS DATA

Machine learning problems also contain continuous data or data in the form of real numbers, not natural numbers. This type of data, so far, has shown itself to be very difficult for multi-valued functional decomposition. There have been several attempts at this, though. In [9,5], the authors map real-valued variables onto natural numbers by discretization. In [4, 3, 6] the data is actually left in the form of real values which causes problems in the decomposition process when trying to combine two values. A way around this is to combine values that are some **ä** distance apart from another. For example, 2.46 and 2.55 could be combined if **ä** = 0.1 (Ð2.46 – 2.55Ð)). Combining the two values may be done by using one of the following: the average of the two values, one of the two values, a don't know, etc. This seems to be a topic that needs further research. Given that existing tools (AIM, C4.5, neural networks) can be applied to continuous data, a possible solution is to use them as pre-processors for functional decomposition. This approach is applicable because machine learning tools are based on experience gained in years of experimentation on machine learning problems.

4. CONCLUSION

In this paper is presented the basis for using multi-valued logic synthesis in the field of machine learning. While the problems inherent in machine learning are not exactly the same as the problems found in multi-valued logic synthesis, there still exists a strong correlation between the two areas. The important aspect of this is that bridging the gap between the two areas could prove beneficial to both logic synthesis and machine learning. There are many theories to explore, especially in the context of manipulating machine learning data.

As the size of databases grows, it is obvious that the need for machine learning is going to be necessary to process all the data. This is a tough problem, but by using multi-valued functional decomposition, hopefully, the problems can still be solved. It is difficult to predict the future, but the possibility exists that machine learning based tools may be used in circuit design as well.

REFERENCES

[1] C. M. Files, "New Functional Representation for the Decomposition of Machine Learning Problem", *Third Symposium on Logic Design and Learning, Conference Proceedings*, pp. Oregon, USA, May 2000.

[2] M. Perkowski, M. Marek-Sadowska, L. Jozwiak, T. Luba, S. Grygiel, M. Nowicka, R. Malvi, Z. Wang, J.S. Zhang, "Decomposition of Multiple-Valued Relations", *Proc. IEEE International Symposium on Multiple-Valued Logic*, pp.13–18, Nova Scotia, Canada, May 1997.

[3] J. A. Goldman, M. L. Axtell, " On Using Logic Syntesis for Knowledge Discovery", *Tools with AI conference*, 1997

[4] B. Zupan, M. Bohanec, I. Bratko, J. Demsar, "Machine Learning by Function Decomposition, *Proceedings of the Fourteenth International Conference Machine Learning* (ICML'97), pp. 421 – 429, Nashville, Tennessee, July 1997.

[5] B. Zupan, M. Bohanec, J. Demsar, I. Bratko, "Feature transformation by function decomposition", *IEEE Intelligent Systems & Their Applications*, vol. 13, pages 38-43, 1998.

[6] B. Zupan, M. Bohanec, I. Bratko, B. Cestnik, "A data set decomposition approach to data mining and machine discovery", *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 229 – 302, Newport Beach, Canada, August 1997.

[7] C. M. Files, M. A. Perkowski, " Multi-Valued Functional Decomposition as s Machine Learning method", *Proc. on 28th IEEE International Symposium on Multiple-Valued Logic*, pp. 173 – 178, Fukuoko, Japan, May 27 –29, 1998.

[8] C. M. Files, M. A. Perkowski, " An Error Reducing Approach to Machine Learning Using Multi-Valued Functional Decomposition", *Proc. on 28th IEEE International Symposium on Multi-Valued Logic*, pp. 167 – 172, Fukuoko, Japan, May 27 – 29, 1998.

[9] B. Zupan, *Machine Learning Based on Functional Decomposition*, PhD thesis, University of Ljubljana, Slovenia, 1997.

[10] C. M. Files, *A New Functional Decomposition Method As Applied to Machine Learning and VLSI Layout*, Ph.D. Dissertation, Portland State University, Portland Oregon, June 2000.

[11] S. Russell, Machine Learning, Handbook of Perception and Cognition, chapter IVion to FPGA synthesis, *Design Automation Conference*, pp. 642 – 647, 1993.

[12] J.R. Quinlan, "Unknown Attribute Values in Induction", *Proc. on 28th IEEE International Symposium on Multiple-Valued Logic*, pp. 173 – 178, Fukuoko, Japan, May 27 –29, 1998.

[13] J. R. Quinlan, C4.5: Programs for Machine Learning, San Mateo, California: Morgan Kaufmann, 1993.

[14] J. R. Quinlan, "A Case Study in Machine Learning", *Proceedings 16th Australian Computer Science Conference*, pp., 731-737, Brisbane, Australia, 1993.