INTEGRATION, the VLSI journal xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Quasi-exact logic functions through classification trees

Valerio Tenace, Andrea Calimera*

Politecnico di Torino, Torino, Italy

ARTICLE INFO

Keywords:
Machine learning
Classification trees
CART
Logic synthesis
Decision diagrams

ABSTRACT

Over the last few years machine learning (ML) has evolved becoming pervasive in many scientific and industrial fields. Such success is ignited by the possibility of applying algorithms that can solve complex problems through generic rule formulations. With such premise, ML surely represents a new opportunity to improve existing design paradigms.

In this paper we propose a novel ML-driven synthesis methodology that allows to describe generic Boolean functions through a representative subset of core expressions using Classification Trees (CTs). Obtained circuits are able to mimic Boolean functions to a certain degree of accuracy, hence the name *quasi-exact* logic functions. The proposed synthesis flow enables a smart hardware mapping of quasi-exact logic functions by means of reduced and ordered decision diagrams. Experiments conducted on a subset of open-source benchmarks demonstrate that CTs are indeed able to cover rather complex Boolean functions with a very high degree of accuracy, 88% on average, still requiring 3× less area over standard multi-level circuit counterparts.

1. Introduction

Machine learning (ML) has emerged as a powerful tool that enables human-inspired computational models. The key feature is to allow solving complex problems through inductive reasoning built upon previous knowledge. More specifically, ML is a paradigm where hardware or software systems replicate a few simple learning/reasoning mechanisms proper of the human brain [1]. Although the first evidence of such techniques could be set back to the mid 1950's [2], ML has become pervasive in the last few years, with contaminations in several commercial and scientific areas, including the field of Electronic Design Automation (EDA). This leap has been made possible due to the recent improvements of computing platforms [3–5].

Most of the EDA research is focused on efficient hardware mappings of those human-inspired paradigms, whilst little effort was spent on investigating how to take advantage of those biological mechanisms to solve EDA problems. Pioneering works include those of Li Wang et al. [6,7] and Guzey et al. [8]. The former described how ML techniques could be shaped as to address testing and verification of digital circuits using both supervised and unsupervised learning methodologies. Guzey was the first to introduce the concept of a real-time Boolean function learning mechanism where an *ad-hoc* statistical model evaluates the output of the Boolean function in order to reconstruct the original logic. More recently, ML has been used as a design-time utility that forecasts critical issues in complex digital design [9,10]. Although such contributions represent a significant seminal work in the context of ML-driven EDA, very little effort has been spent in terms of design strategies, with the result that a proper and complete brain-inspired synthesis flow is still currently missing. In our view, the potentiality brought by ML techniques should be employed to generate logic circuits that are able to *accelerate* the evaluation of a logic function by recreating the inferential processes proper of the human-brain, where yields are *inferred* rather than being evaluated. As an example, when people are asked to perform simple maths they will first identify some useful characteristic of the problem, e.g., a multiplication involving a power-of-ten multiplicand, or the possibility of discarding less significant digits as to preserve the order of magnitude of the answer; afterwards they will infer the result leveraging some logic relationship suggested by previous experience, e.g., append "0" to the multiplier if the multiplicand is "10". As a matter of fact, such inferential processing of the information is used to skip the actual maths as to provide a reasonably accurate answer using as few resources as possible. The same concept can be extended to any kind of human-related reasoning, e.g., decision making and predictions. In this work, we face the challenge of transferring this rationale to the Boolean domain, where a logic computation is replaced by an educated guess of the yield. As a consequence, this would push the limit of edge computing [11] to new borders where smarter and more energy efficient ubiquitous devices will be capable of processing large amount of data at a fraction of the power that is currently required to run integrated circuits designed with standard

* Corresponding author.
E-mail address: andrea.calimera@polito.it (A. Calimera).

https://doi.org/10.1016/j.vlsi.2018.06.007
Received 20 December 2017; Received in revised form 23 May 2018; Accepted 28 June 2018

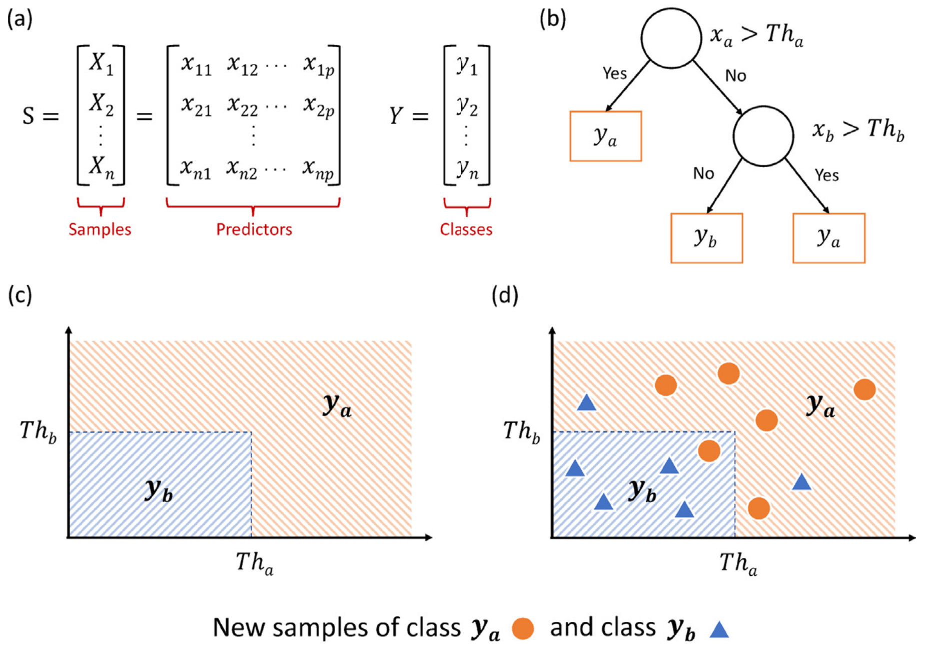

Fig. 1. Classification problems. Training sample description (a), abstract model representation through a CT structure (b), input space partitioning (c), and classification of new samples (d).

synthesis flows [12]. Needless to say, achieving this goal implies a clear departure from classical Boolean logic circuits that evaluate exact, or approximate, logic rules. This represents the motivation of this work. We envision a new class of *inferential* circuits that predict the result of a logic function by considering the characteristics of the problem (a power-of-ten multiplicand, as in the previous example), and what logic rules they require to achieve the desired goal (append a “0” digit to the multiplier). In this regard, a new dedicated ML-driven logic synthesis methodology is required to: (*i*) learn the most representative expressions of a given Boolean function as to build an efficient abstract data representation model that “replicates”, or “mimics”, the logic behavior of the Boolean function with the highest degree of accuracy and the minimum complexity; and (*ii*) perform *ad-hoc* hardware mapping strategies of the obtained data model representation, as to enable smart on-chip logic function transpositions. We refer to these functions as *quasi-exact* logic functions, where logic circuits are obtained by solving a learning problem that makes use of Classification Trees (CTs). It is worth to underline that the proposed solution should not be confused with approximate logic synthesis algorithms [13], nor with previous techniques that make use of approximate decision tree construction [14]. Indeed, differences with those solutions are substantial, and can be summarized as follows. First, approximate synthesis is based on the assumption that, given a predefined error tolerance, it is possible to tweak classical Boolean logic synthesis processes by relaxing some constraints on minterm minimization, as to generate logic networks that match the desired quality constraints [13,15,16]. On the contrary, quasi-exact logic functions are built with statistical rules, where the ML-driven synthesis flow enables the understanding of what decision tree topology better replicates the logic behavior of the whole Boolean function. Second, approximate logic synthesis solutions leverage pre-defined heuristics and rules specific for an application domain [17]. On the other hand, the proposed solution is orthogonal to the application since it is aimed at building such rules by recreating an inferential process during logic synthesis stages. Last, but not least, unlike previous approximate tree construction algorithms [14], the proposed technique does not constrain the exploration of the solution space to tree structures with specific topological characteristics. As a result, the ML-driven synthesis flow illustrated in this work allows to achieve much smaller, faster, and less power hungry quasi-exact logic circuits, thus enabling more resource efficient error-resilient computations [18].

Quasi-exact logic functions have been briefly introduced in a recent work [19]. In this paper, we extend such initial analysis with a more comprehensive description of the problem. The contributions of this paper can be summarized as follows. In the first place, we thoroughly describe the training and validation phases of CTs, with a focus on Boolean training set populations. We then show how CTs can be learned to represent any Boolean logic function, either single or multiple-output. Lastly, an automated optimization strategy, leveraging reduced & ordered decision trees, is described as to enable compact on-chip representations of CTs. Experimental results collected on several open-source combinational benchmarks demonstrate that logic circuits obtained with the proposed synthesis flow achieve a remarkable accuracy of 88%, still requiring $3\times$ less area w.r.t. standard multi-level circuit counterparts.

2. Background

Machine learning techniques can be classified into two big families: those for supervised learning and those for unsupervised learning. Whilst the latter imply a *blind* search for proper rules that allow to group objects with similar characteristics, the former are used when prior knowledge of the problem is available, e.g., the population under analysis is properly labeled and packed into a training set. Supervised learning models allow the extraction of common key features among samples composing the training data-set. More formally, it is possible to split a generic classification problem in two main phases: (*i*) *training*, during which a training data-set consisting of n samples labeled with one of the m available classes $y_i \in Y = \{y_1, \ldots y_m\}$ and described through p predictor variables $X = \{x_1, \ldots x_p\}$ (Fig. 1-a) is used to learn an efficient data model representation (Fig. 1-b); (*ii*) *validation*, during which a data-set made up of a new set of samples labeled with Y and described through X is used to quantify the accuracy of the obtained data model representation. In this regard, Fig. 1-c describes the input space partitioning due to the rules imposed by the model, whereas Fig. 1-d shows how samples are classified. Although several options for building data models are available, classification trees represent a solution that combines high quality-of-result (QoR) with a reasonable simple decision tree structure [20] which enables complex partitioning of the input