Towards Automatic Generation of Hardware Classifiers

Flora Amato, Mario Barbareschi, Valentina Casola,
Antonino Mazzeo, and Sara Romano

Università degli Studi di Napoli "Federico II",
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione,
Via Claudio 21, 80125, Napoli, Italia
{flora.amato,mario.barbareschi,casolav,mazzeo,sara.romano}@unina.it

Abstract. Nowadays, in a broad range of application areas, the daily data production has reached unprecedented levels. This data origins from multiple sources, such as sensors, social media posts, digital pictures and videos and so on. The technical and scientific issues related to the data booming have been designated as the "Big Data" challenges. To deal with big data analysis, innovative algorithms and data mining tools are needed in order to extract information and discover knowledge from the continuous and increasing data growing. In most of data mining methods the data volume and variety directly impact on computational load. In this paper we illustrate a hardware architecture of the decision tree predictor, a widely adopted machine learning algorithm. In particular we show how it is possible to automatically generate a hardware implementation of the predictor module that provides a better throughput that available software solutions.

1 Introduction

In many application areas the daily data production has reached unprecedented levels. According to recently published statistics, in 2012 every day 2.5 EB (Exabyte) were created, with 90% of the data created in the last two years [7]. This data origins from multiple sources: sensors used to gather climate information, social networks, digital pictures and video streaming, and so on. Moreover, the size of this data is growing exponentially due to not expensive media (smartphones and sensors), and to the introduction of big Cloud Datacenters. The technical and scientific issues related to the data booming have been designated as the "Big Data" challenges and have been identified as highly strategic by major research agencies. Most definitions of big data refer on the so-called three Vs: *volume*, *variety* and *velocity*, referring respectively to the size of data storage, to the variety of source and to the frequency of the data generation and delivery. To deal with big data analysis, innovative approaches for data mining and processing are required in order to enable process optimization and enhance decision making tasks. To achieve this, an increment on computational power is needed and dedicated hardware can be adopted. There are two main classes of solutions: 1) using general purpose CPUs as multi-core processors and/or computer

J. Kołodziej et al. (Eds.): ICA3PP 2013, Part II, LNCS 8286, pp. 125–132, 2013.

clusters to run the data mining software; 2) using dedicated hardware (special purpose) to compute specific parts of an algorithm, reducing the computational effort. Indeed special purpose machines may not be suitable as they are not programmable and many classification systems need tuning and reprogramming to achieve high accuracy. Nevertheless Field Programmable Gate Array (FPGA) can be adopted for the low costs and the re-configuration properties.

At this aim, in this paper, we have focused our attention on FPGA for the classification task. In particular, we adopt an FPGA architecture implementing a predictor, built by the C4.5 decision tree algorithm, widely adopted for classification tasks in many application areas [1]. Exploiting hardware characteristics, this architecture is designed to maximize the throughput, making the classification task suitable for very large amount of data characterized by an high number of features in input and a high number of classes in output. Indeed in literature some FPGA based solutions exist, nevertheless our focus is on the automatic generation of an optimized hardware description that provides higher throughputs than available solutions. In particular we show how it is possible the automation of the hardware accelerator generation process, starting from the data model. We illustrate its applicability in two real case studies (URL malicious detection and event detection with sensor networks) obtaining performance that are 4 orders of magnitude greater than software implementation.

The reminder of the paper is structured as follows: in Section 2 we describe related work to both C4.5 algorithm and data mining algorithm implemented on FPGAs; in Section 3 we introduce some details on the C4.5 algorithm describing the learning and prediction phases and illustrating the improvements achieved with hardware implementation. In Section 4 we present the process that automatize the hardware synthesis. In Section 5 two case studies are presented to demonstrate the advantages of the proposed solution. Finally, in Section 6 some conclusion and future work are drawn.

2 Related Work

As mentioned above, FPGAs are very suitable to implement classification algorithm as they need deep re-programming during the tuning phase. Several FPGA-based classification architectures have been proposed in order to *speed up* the classification processes [11,12]. In [13] two FPGA architectures were proposed to classify packets traffic. Both architectures use a programmable classifier exploiting the C4.5 algorithm. Using NetFPGA, in [9] the authors proposed a traffic classifier by using C4.5. The main architectural characteristic is the programmability by the software, without loss of service, using memories that store the classifier.

The C4.5 classification algorithm is very promising for big data analysis, it performs very well in many application domains as the predictor works on the tree structure built during the learning phase. It was presented by Quinlan in [10] and soon was adopted in a broad range of applications such as image recognition, medical diagnosis, fraud detection and target marketing.

In [4] and [8] the C4.5 algorithm was adapted within the framework of differential privacy in distributed environments. In [5] the authors show the power of C4.5 for classifying the Internet application traffic, due to the discretization of input features done by the algorithm during classification operations. The authors of [14] extended the traditional decision tree based classifiers to work with uncertain data.

The results presented in these papers are very interesting, but they did not provide any automatic tool to automatically generate the hardware architecture from the prediction model. In this paper we intend to propose an automatic generation tool and we will describe the results of two case studies.

3 The Classification Algorithm

To characterize and classify big data, we exploit the C4.5 decision tree algorithm widely adopted for classification tasks in several application areas. C4.5 is based on decision-tree induction approach, which directly constructs a classification model from the training database, following a top-down and divide-and-conquer paradigm. This classifier is made of a learner for building the predictive model, named training phase, and a predictor for performing data classification.

Algorithm 1. Description of the algorithm for building a decision tree model Classifier

Data: Training Set T, Output Class $C = C_1, ..., C_k$
Result: Prediction Model outputted by learner
`/* Base Case */`
if *T f T contains one or more examples, all belonging to the same class C_j* **then**
  Create a single leaf in which all the sample having label C. `/* The decision tree for T is a leaf identifying class` C_j `*/`
if $T = \emptyset$ **then**
  Creates a single leaf that has the label of the most frequent class of the samples contained in the parent node `/* heuristic step leading to misclassification rates */`
`/* Recursive Case */`
if *T contains samples that belong to several classes* **then**
  foreach *Feature f* **do**
    Find the normalized information gain by splitting on f, based on an Entropy Test Value
  Let f_i be the attribute with the highest normalized information gain; Create a decision node that splits on f_j;`/* node associated with the test */`
  Create a branch for each possible outcome of the test; Execute the algorithm on each branch (corresponding to a subset of sample)
  `/* partitioning of T in more subsets according to the test on the attribute obtained by splitting on` f_j `*/`

During the training phase, a domain expert builds a training set, i.e. a subset of data records that have been previously classified, which will be used to tune the predictor and define the predictor parameters. The predictor decision rules are modeled as a tree, the nodes represent classes associated to events to be detected and branches represent conjunctions of conditions that define a path that leads to a class. Periodically, or when some samples are mis-classified, the learner can recalculate the parameters on the basis of a new training set, in order to refine the behavior of the classifier, excluding in this way the predictor behaviors that leads to the mis-classifications. As this activity requires re-programming the classifier, we have implemented the predictor with a re-programmable hardware. As illustrated in Algorithm 1, the algorithm recursively works on data taken from the training set. At each step, the data set is split, based on conditions of a chosen feature that are defined by thresholds. The selection of the feature is performed by using an *entropy test*.

Let T be the set of samples, $freq(C_j, T)$ the number of samples in T that belong to class C_j, the entropy of T is defined as:

$$Info(T) = -\sum((freq(C_j, T)/T) \cdot log_2(freq(C_j, T)/T))$$

The algorithm computes the value of *Info* for all T_i partitioned in accordance with n conditions on a feature f_i:

$$Info_{f_{i(T)}} = \sum((T_i/T) \cdot Info(T_i))$$

The features that maximize the following Gain value are selected for the splitting:

$$Gain(f_i) = Info(T) - Info_{f_{i(T)}}$$

In order to be properly processed, the input information must be represented by a proper data model. For the selected dataset, proper features must be chosen, in order to obtain effective classification for the data of the targeted applications. In the proposed case studies, the candidate features will be selected in order to obtain high accuracy in the classification results, even if the huge dimension of dataset requires that the features are computed with low computationally cost.

4 The Synthesis Process to Generate the Classifier

The predictor module is based on the tree-like model defined by the learner and the tree visit is usually executed in a sequential way. Each tree node contains a predicate that represents a condition, while leaves are labeled with classes the samples belong to. The predictor hardware implementation, intrinsically concurrent, is able to elaborate in parallel the conditions of the nodes of the tree.

We designed a fully automatic process to synthesize the decision tree hardware starting from the training set. The process flow is reported in Figure 1; it is made of three different steps that produce in output three standard artifacts. The goal of the *Data Processing* is to automatically structure data coming from heterogeneous sources into a common schema, a data-preprocessing tool extracts

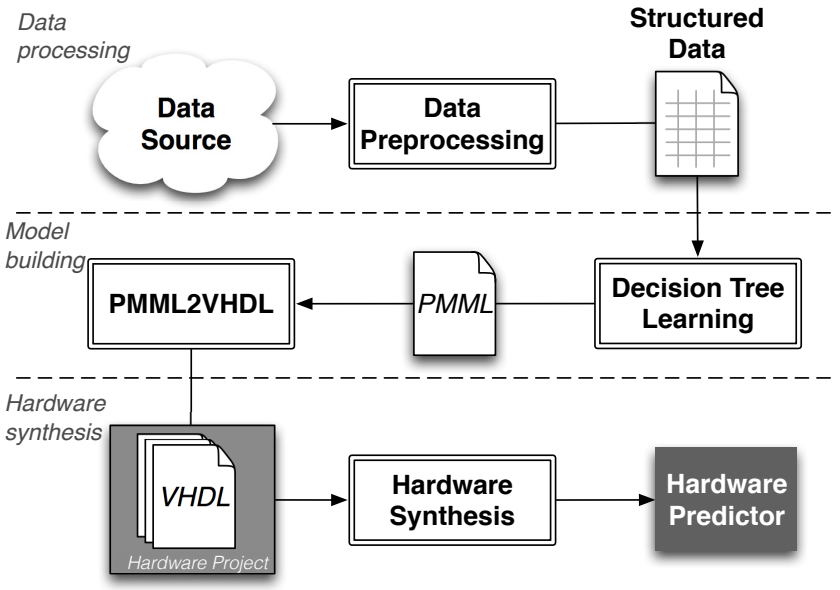

Fig. 1. Process flow from data source to the hardware implementation of decision tree

the only relevant information to build a common schema and it implements the methodology proposed by Amato et al. in [2,3]. This artifact, stored in a tabular format, is given in input to the *Model Building* step. It implements C4.5 learning algorithm, the output of this step is the predictor model coded into PMML (Predictor Model Markup Language), a standard XML schema for predictors. At this step, exploiting the PMML formalization, we developed a tool, PMML2VHDL, that parses the decision tree predictor model and generates an optimized hardware description for the predictor [1]. We formalized it in VHDL but other descriptions can be adopted. Finally, in the last step, the *Hardware Synthesis*, we used the VHDL as input for the hardware synthesizer in order to obtain a working version of the predictor.

As for the hardware architecture, we implemented the tree visiting algorithm as a multi-output boolean function: 1) we pre-evaluated all conditions (nodes) in parallel using configurable components called *Decision Boxes*, that are implemented as comparators; 2) we performed the visiting as a boolean function, implemented with a component called *Boolean Net*.

In Figure 2 we report an example of decision tree and the corresponding hardware scheme. As illustrated, each tree path leads to a class, so the AND of decisions along the path can be represented by a boolean minterm, and the multitude of paths that lead to the same class may be represented as a boolean function in SOP form. Due to space limitations, we cannot report the details of the hardware implementation, in next section we will present some preliminary experimental results on different case studies.

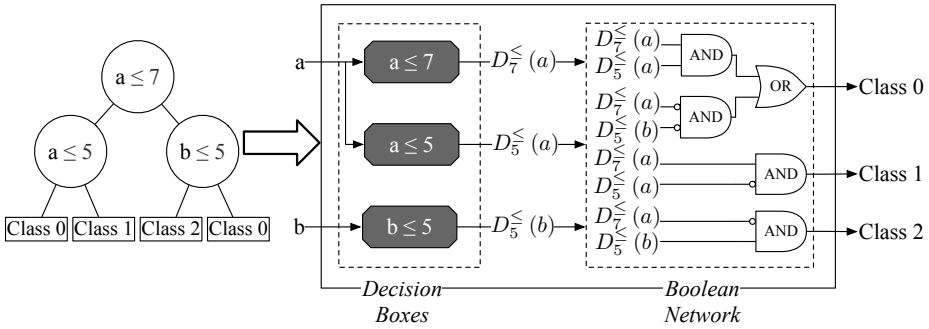

Fig. 2. Predictor Hardware architecture

5 Preliminary Experimental Results

We evaluated the architecture by means of two experimental campaigns, one related to URL reputation classification and the other one related to sensor networks data classification. The first one is characterized by a high number of features, while the second one is an interesting case study with a high number of classes.

Malicious URL Detection: we chose the dataset proposed by [6], created to detect the malicious web sites from the URLs. The dataset contains about 2.4 million records and 3.2 million host-based features, coded in floating point. In order to handle this data we performed a pre-processing activity and we applied a feature filter based on the entropy test. We discarded features having a value less than a threshold defined for this case study. Thanks to this filtering process, we reduced the number of significant features to 51. From this data collection we built the C4.5 predictor tree model. In order to evaluate the accuracy of the predictor we applied a cross validation technique which breaks the dataset in 10 sets: 7 sets are used as training set and 3 sets are used for testing. The overall accuracy of the system was evaluated as the average accuracy over 10 iterations, in which samples for training and testing were randomly chosen. Using KNIME framework, we obtained a predictor tree with 624 nodes, 32 levels and 98.647% of accuracy. Once built the predictor model, we classified 4 million records with the KNIME software version in order to compare it with our proposal. As for the proposed architecture, we automatically obtained the VHDL project from the KNIME PMML file by using PMML2VHDL tool. The experiments were done using a Xilinx Virtex5 LX110T-2 and the results are reported in the first column of Table 1.

The interesting result is that the computation time is 11.28 ms, while the classification of the same 4 million records with KNIME on an Intel Core $i7-3770$ ($3.40GHz$), with 16Gb RAM requires, in average, about 124,834.45 ms ($\sim$ 2 min). The gain obtained in terms of elaboration time by introducing the hardware classifier is about of 10,000 times the software implementation.

Table 1. Classification results for malicious URL detection and sensor network dataset

	malicious URL detection	sensor network event detection
Features	51	12
Classes	2	24
Accuracy (%)	98.647	98.994
Throughput (Gbps)	578.93	126.48
Slices	2777	1546
Computation Time (ms)	11.28	7.59

Event Detection over Sensor Network: this case study is focused on sensor networks data classification to detect different events, for instance temperature and acceleration exceeding nominal values. We referred to a sensor network deployed in [1] and running for several days, we collected data and manually labeled the training set in 24 thresholds, representing 24 alarm classes. Following the same steps described for the first case study, we obtained a predictor tree with 329 nodes and 20 levels with an accuracy of 98.994%. The computation time for the classification of 2.5 million samples was in average 75,770 ms using KNIME. The hardware synthesis results of this decision tree are reported in the second column of Table 1. Even in this case we obtained a gain of 10,000. Furthermore we can see that the throughput of the second case is lower than the first one, although we have less nodes and less levels.

In conclusion we can observe that in the both case studies the computation time gain is very high, and the proposed architecture seems very suitable for big data applications.

6 Conclusion and Future Work

The adoption of FPGA in big data analysis seems very promising. In this paper we proposed a process to implement in hardware a reconfigurable decision tree classifier. In particular, we proposed an innovative architecture to fasten the classificator that is made of *Decision Boxes* to compute in parallel all decisions and a *Boolean Net*, to effectively compute the classification. We evaluated the proposal with two different case studies, putting in evidence the great performance obtained with the hardware implementation. Many optimization are still possible, in future work we intend to enhance the proposal by introducing pipelining mechanism into the available hardware implementation, furthermore we want to prove the scalability of the proposed approach.

Acknowledgments. This work was partially supported by FARO 2012 project "Un sistema elettronico di elaborazione in tempo reale per l'estrazione di informazioni da video ad alta risoluzione, alto frame rate e basso rapporto segnale rumore" founded by the University of Naples "Federico II".

References

1. Amato, F., Barbareschi, M., Casola, V., Mazzeo, A.: An FPGA-based smart classifier for decision support systems. In: Zavoral, F., Jung, J.J., Badica, C. (eds.) Intelligent Distributed Computing VII. SCI, vol. 511, pp. 289–299. Springer, Heidelberg (2014)
2. Amato, F., Casola, V., Mazzeo, A., Romano, S.: A semantic based methodology to classify and protect sensitive data in medical records. In: 2010 Sixth International Conference on Information Assurance and Security (IAS), pp. 240–246. IEEE (2010)
3. Amato, F., Casola, V., Mazzocca, N., Romano, S.: A semantic-based document processing framework: A security perspective. In: International Conference on Complex, Intelligent and Software Intensive Systems, pp. 197–202 (2011)
4. Friedman, A., Schuster, A.: Data mining with differential privacy. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 493–502. ACM, New York (2010)
5. Lim, Y.S., Kim, H.C., Jeong, J., Kim, C.K., Kwon, T.T., Choi, Y.: Internet traffic classification demystified: on the sources of the discriminative power. In: Proceedings of the 6th International Conference on Emerging Networking EXperiments and Technologies, pp. 9:1–9:12. ACM, New York (2010)
6. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Identifying suspicious URLs: an application of large-scale online learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 681–688. ACM (2009)
7. Mayer-Schönberger, V., Cukier, K.: Big Data: A Revolution That Will Transform How We Live, Work, and Think. Houghton Mifflin Harcourt (2013)
8. Mohammed, N., Chen, R., Fung, B.C., Yu, P.S.: Differentially private data release for data mining. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 493–501. ACM, New York (2011)
9. Monemi, A., Zarei, R., Marsono, M.N.: Online NetFPGA Decision Tree Statistical Traffic Classifier. Computer Communications (2013)
10. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
11. Schadt, E.E., Linderman, M.D., Sorenson, J., Lee, L., Nolan, G.P.: Computational solutions to large-scale data management and analysis. Nature Reviews Genetics 11(9), 647–657 (2010)
12. Skoda, P., Medved Rogina, B., Sruk, V.: FPGA implementations of data mining algorithms. In: MIPRO, 2012 Proceedings of the 35th International Convention, pp. 362–367. IEEE (2012)
13. Tong, D., Sun, L., Matam, K., Prasanna, V.: High throughput and programmable online traffic classifier on FPGA. In: Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 255–264. ACM (2013)
14. Tsang, S., Kao, B., Yip, K., Ho, W.S., Lee, S.D.: Decision trees for uncertain data. IEEE Transactions on Knowledge and Data Engineering 23(1), 64–78 (2011)