Decision Tree based Classifiers for Large Datasets

Anilu Franco-Arcega[1,2], Jesús Ariel Carrasco-Ochoa[2], Guillermo Sánchez-Díaz[3], and José Francisco Martínez-Trinidad[2]

[1] Universidad Autónoma del Estado de Hidalgo, Hidalgo, Mexico

[2] Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, Mexico

[3] Universidad Autónoma de San Luis Potosí, San Luis Potosí, Mexico

afranco@uaeh.edu.mx, {anifranco6, ariel, fmartine}@inaoep.mx, guillermo.sanchez@uaslp.mx

Abstract. In this paper, several algorithms have been developed for building decision trees from large datasets. These algorithms overcome some restrictions of the most recent algorithms in the state of the art. Three of these algorithms have been designed to process datasets described exclusively by numeric attributes, and the fourth one, for processing mixed datasets. The proposed algorithms process all the training instances without storing the whole dataset in the main memory. Besides, the developed algorithms are faster than the most recent algorithms for building decision trees from large datasets, and reach competitive accuracy rates.

Keywords. Decision trees, supervised classification, large datasets.

Clasificadores basados en arboles de decisión para grandes conjuntos de datos

Resumen. En este artículo se desarrollaron varios algoritmos de generación de árboles de decisión a partir de grandes conjuntos de datos, los cuales resuelven algunas de las limitaciones de los algoritmos más recientes del estado del arte. Tres de estos algoritmos permiten procesar conjuntos de datos descritos exclusivamente por atributos numéricos; y otro puede procesar conjuntos de datos mezclados.
Los algoritmos propuestos procesan todos los objetos del conjunto de entrenamiento sin necesidad de almacenarlo completo en memoria. Además, los algoritmos desarrollados son más rápidos que los algoritmos más recientes para la generación de árboles de decisión para grandes conjuntos de datos, obteniendo resultados de clasificación competitivos.

Palabras clave. Árboles de decisión, clasificación supervisada, grandes conjuntos de datos.

1 Introduction

Decision Trees [12] are among the most used algorithms for solving supervised classification problems. A decision tree (DT) is a structure consisting of internal nodes, edges and leaves. An internal node has one or more test attributes associated with it and two or more edges which lead to other nodes. A leaf includes a class label which is assigned to new instances arriving to the leaf.

Currently, it is a very common task in computation to operate datasets with a big amount of instances [3]. Nevertheless, building DTs from large datasets requires long time for processing all the training instances and, moreover, the available memory may be not sufficient for storing the whole training set. Therefore, implementation of conventional algorithms for building DTs becomes very time and space consuming, and in some cases not applicable. For this reason in the literature, a number of authors have proposed algorithms for

[1] Extended abstract of PhD thesis. Graduated: Anilu Franco-Arcega. Advisors Jesús Ariel Carrasco Ochoa, Guillermo Sánchez-Díaz, and José Francisco Martínez-Trinidad. Graduation date: 14/07/2010.

building DTs from large datasets (See Section 2). However, these algorithms have some drawbacks. Therefore, in this PhD thesis we introduced DT induction algorithms which overcome some of these drawbacks. The main characteristics of our algorithms are as follows. Firstly, they process the whole training set for building a DT without storing the set in the main memory, and secondly, they are faster than the most recent algorithms for building DTs from large datasets, maintaining a competitive accuracy rate.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 introduces the proposed algorithms. Section 4 shows some experimental results. Finally, Section 5 presents our conclusions and future work.

2 Related Work

Many algorithms have been developed for building decision trees such as ID3 [10], C4.5 [13], ID5R [18], ITI [19], CART [2], ModelTrees [16], CTC [11], UFFT [5], FDT [8], etc. However, all these algorithms have to keep in memory the whole training set for building a DT. Therefore, they cannot be applied to large training sets.

Other algorithms have been developed for building DTs from large training sets, for example, SLIQ [9], SPRINT [15], CLOUDS [1], RainForest [7] and BOAI [20]; however, all of them use lists for keeping a dataset in the main memory. For each attribute in the dataset, these algorithms assign a list. The problem is that some of these lists require more space than the one required to store the whole training set. Other algorithms, like BOAT [6], ICE [21] and VFDT [4], are incremental algorithms. Both BOAT and ICE use a subset of training instances for building a DT; but for large datasets, to search this subset of instances may be too expensive. In VFDT, the user needs to define values for three parameters before building a DT, which could be very difficult in practice.

The algorithms proposed here solve some of the restrictions highlighted above. Our algorithms process the whole training set without storing it in memory and their parameters can be easily defined by the user. Besides, our algorithms are faster than the most recent algorithms reported in the literature for building DTs from large datasets.

3 Proposed Algorithms

This section introduces the main features of the proposed algorithms for building DTs from large training sets. In order to reduce memory requirements and handle large training sets, our algorithms process the training instances one by one in an incremental way, updating the DT with each instance. For expanding a node, our algorithms employ only a small amount of instances, so the expansion of the nodes is faster than the common expansion process used in previous algorithms reported in the literature. Besides, in our algorithms, the instances used for expanding a node will be deleted once the expansion is done, in order to avoid storing the whole training set in memory.

The structure of a DT built by our algorithms is similar to the structure of a conventional DT. A DT has a root node, internal nodes and leaves. Each internal node has one or more test attributes associated with it (according to the algorithm) and each leaf includes a class label used for classifying new instances.

In all the algorithms developed in this PhD thesis, the building step starts with creating an empty root node. Then, each training instance traverses the DT beginning in the root node and descending through internal nodes, until it reaches the leaf in which the instance is stored. When a leaf has *s* instances (*s* is a parameter of our algorithms), it is expanded or updated, using only the *s* instances stored in the leaf.

Depending on the type of *s* instances stored in a node, the node is expanded or updated. If these instances belong to two or more classes, the node is expanded. For expanding a node, one or more test attributes are obtained and for each class, using instances belonging to that class, a test value for each test attribute is computed thus creating a set of test values for each class. Each one of these sets is assigned to the corresponding edge (one edge per class); this process is presented graphically in Figure 1. The way of obtaining test attributes and sets of test values depends on a particular algorithm (see Section 3.1).

On the other hand, if a node has instances only of a single class, this node is updated. For updating a node, a set of test values is obtained

from *s* instances stored in the node, and this set is combined with the set of test values assigned to the input edge of the node. The new set of test values replaces the set assigned to the input edge of the node; Figure 2 shows the scheme of this process.

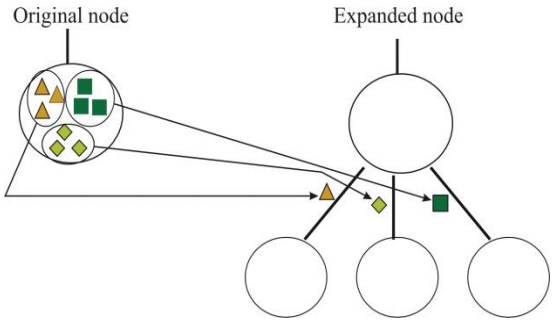

Fig. 1. Expansion process for proposed algorithms

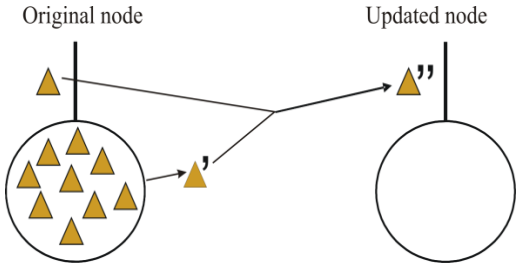

Fig. 2. Updating process for proposed algorithms

Once a node is expanded or updated, the *s* instances stored in it are deleted.

Finally, once all training instances have been processed, a class label is assigned to each leaf in the DT (the majority class of the instances stored in the leaf or the class associated to the input edge if the leaf is empty), and the DT induction finishes.

Since at the beginning the DT has only the root node (a leaf), the first *s* instances are stored in that node. In order to avoid having instances from only one class in the root node, we propose to reorganize the training set before starting to build a DT. This reorganization consists in alternating instances from each class. The first instance will be from class 1, the second instance from class 2, and so on. If there are *c* classes, the $(c+1)^{th}$ instance will be from class 1, the $(c+2)^{th}$ instance will be from class 2, and so on.

The classification step using the generated DT is similar to the one used in traditional algorithms. A new instance traverses the DT starting at the root node and descends through internal nodes, until it arrives to the leaf with the relevant class label. This label is assigned to the new instance.

3.1 Expansion Process

We have defined three ways to expand a node for our algorithms designed to process numeric datasets; each one defines a different algorithm as follows:

1. Using all the attributes as test attributes in the node (IIMDT algorithm). For each attribute and each class, a test value is computed in order to form the set of test values corresponding to each of the output edges of the node. Therefore, the number of output edges is the same as the number of classes.
2. Using n attributes (n is another parameter) as test attributes (IIMDTS algorithm). This algorithm expands a node in a similar way as in the IIMDTS algorithm, with the difference that IIMDTS uses only n test attributes for computing the sets of test values for the output edges.
3. Using only one attribute as test attribute (DTFS algorithm), computing only the test value for this attribute, one for each class.

These options are used when a training set is described exclusively by numeric attributes, since the test values for each attribute are computed as the mean of the values of the corresponding attributes in the instances belonging to the corresponding class.

The way to expand a node when we have mixed training sets is as follows:

4. Using a single test attribute (DTLT algorithm).

The rule for expanding a node using a numeric attribute is $X \leq V$ (creating two output edges) as in the C4.5 algorithm [13]. Applying a categorical attribute, the number of edges will be the same as the number of possible values in the attribute.

In all cases, once a node has been expanded, the *s* instances stored in the node are deleted, in order to save memory space.

4 Experimental Results

To demonstrate the behavior of our algorithms, we evaluated the processing time and the accuracy rate. We compared the obtained results of the proposed algorithms with ICE, VFDT and BOAI algorithms. For all experiments, we employed 10-fold cross validation, showing in the graphs the 95% confidence intervals. Our experiments were performed on a Pentium 4 processor at 3.06 GHz, with 2 GB of RAM, running Linux Kubuntu 7.10.

Based on experimental analysis, we used *s*=100 for all algorithms and *n*=1,2,5 for IIMDTS, since these values were the best for our algorithms. Such values assure the best trade-off between processing time and accuracy rate. Several real and synthetic datasets were used for evaluating our algorithms. However, in this paper we give results only from one dataset for each algorithm.

First, we present the results of IIMDT, IIMDTS and DTFS using the GalStar dataset [14]. This is a real-world dataset with 2 classes, 30 numeric attributes and 4,000,000 instances. In this experiment, several training sets from 500,000 to 4,000,000 instances were created from GalStar.

Figures 3 and 4 show the processing time and accuracy rates obtained for this dataset, respectively. BOAI was not included in this experiment because it is not able to process training sets bigger than 200,000 instances. As one can observe, our algorithms are the best, since they are faster than ICE and VFDT, obtaining competitive accuracy rates.

We utilize the KDD dataset [17] for showing the behavior of DTLT for a mixed dataset. This is a real-world dataset with 2 classes, 41 mixed attributes and 4,800,000 instances. We also created several training sets from this dataset, from 500,000 to 4,500,000 instances. BOAI cannot process any of these training sets; therefore it was not included in this experiment. It can be observed in Figures 5 and 6 that DTLT is faster than ICE and VFDT, being better than ICE and similar to VFDT in accuracy.

We also included some experiments for showing the behavior of our algorithms when the number of attributes is increased. A synthetic dataset with two classes and 4,000,000 instances

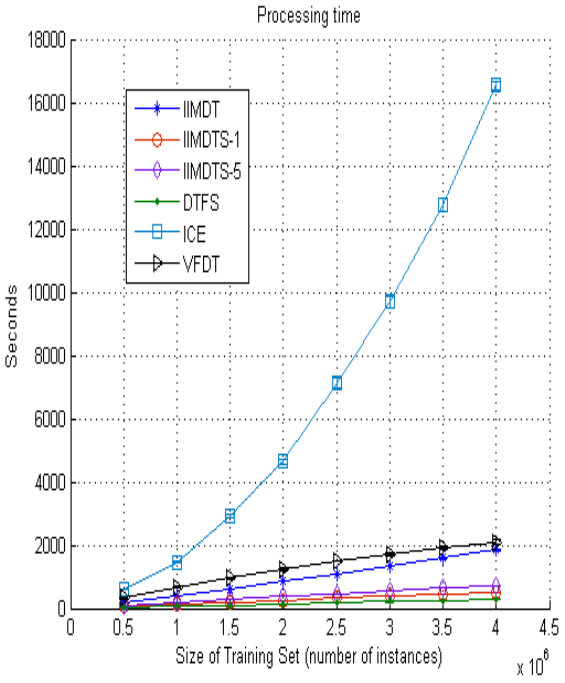

Fig. 3. Processing time for the GalStar dataset

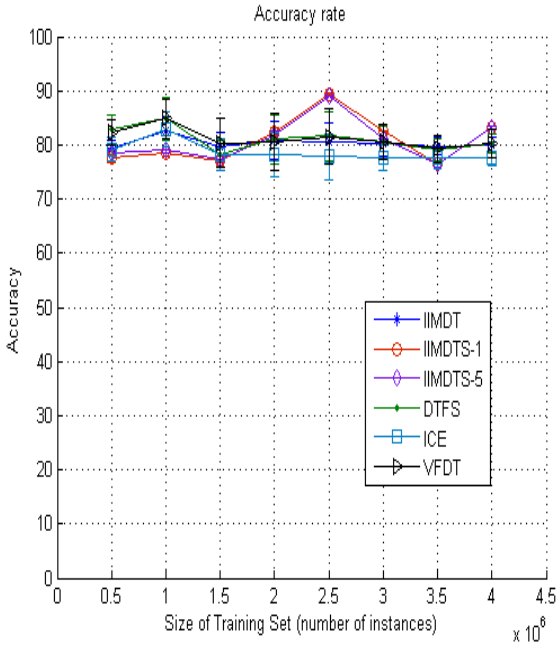

Fig. 4. Accuracy rate for the GalStar dataset

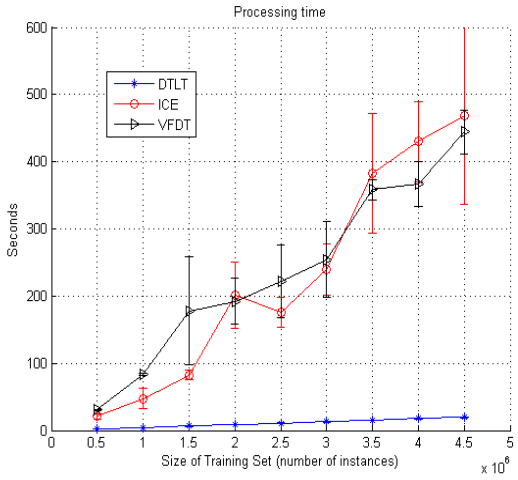

Fig. 5. Processing time for the KDD dataset

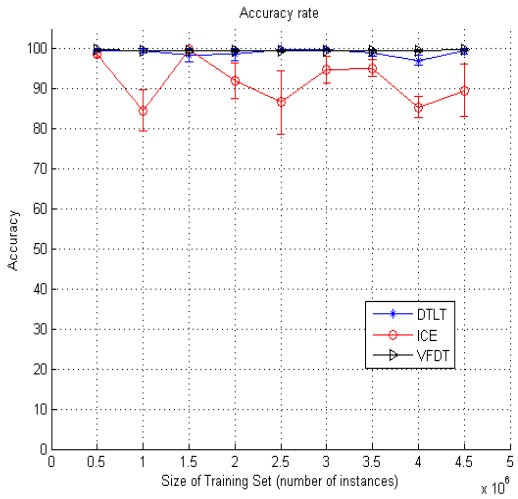

Fig. 6. Accuracy rate for the KDD dataset

was created with different number of attributes. Several training sets were created, from 5 to 40 attributes, with increments of 5. The datasets were randomly generated following the normal distribution with different mean and standard deviation for each class and each attribute.

Figure 7 shows the processing time obtained from these experiments; in this figure it can be observed that the processing time used by our

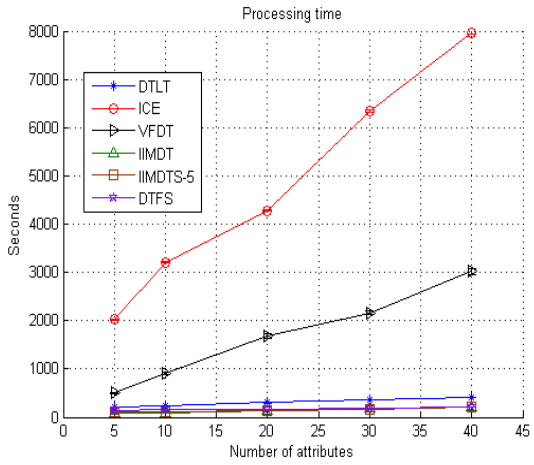

Fig. 7. Processing time when the number of attributes is increased

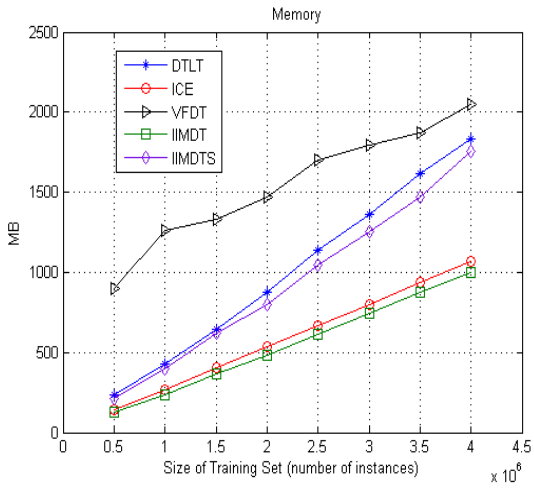

Fig. 8. Amount of memory used for the GalStar dataset

algorithms increases slightly, while ICE and VFDT processing time increases quickly.

Additionally, we analyzed the amount of memory which the algorithms use for building a DT. Figures 8 and 9 show the results using GalStar (numeric dataset) and KDD (mixed dataset). As one can observe, for the GalStar dataset, IIMDT and ICE use a similar amount of memory which is less than for IIMDTS and DTLT, which in their turn use even less memory than

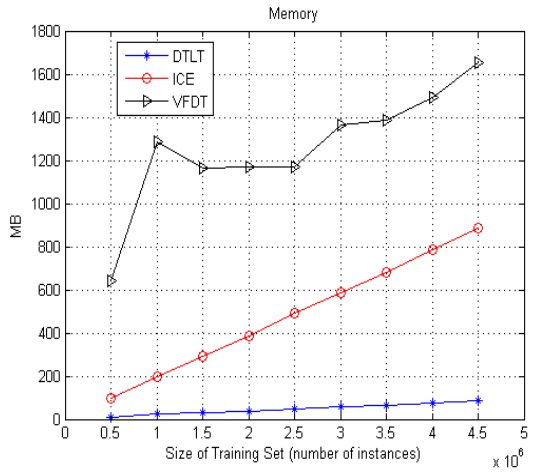

Fig. 9. Amount of memory used for the KDD dataset

VFDT. However, it is important to highlight that ICE uses only a subset of instances for building a DT. For the KDD dataset, our algorithm DTLT uses less memory than ICE and VFDT. The BOAI algorithm does not appear in these figures, because it cannot not process training sets bigger than 200,000 instances, otherwise it produces memory failures.

5 Conclusions and Future Work

In this PhD thesis we have proposed new algorithms for building decision trees from large training sets. These algorithms solve some restrictions of previous algorithms reported in the state of the art. Our algorithms process the whole training set in an incremental way without storing it in memory. Besides, the algorithms use only a small amount of instances for expanding a node which allows them to fulfill a fast selection of test attributes. Taking advantage of this characteristic, the proposed algorithms are able to process large training sets.

The obtained experimental results show that our algorithms IIMDT, IIMDTS, DTFS and DTLT display good behavior when the number of instances increases. Besides, they are faster than the most recent algorithms for building DTs from large datasets (ICE, VFDT and BOAI), maintaining competitive accuracy rates. Also, a fast selection of the test attributes makes our algorithms stable in terms of processing time, when the number of attributes increases.

As future work, we will propose methods for automatic selection of values for the parameters in our algorithms.

Acknowledgements

Authors wish to thank CONACyT for its support with the grant 165151 given to the first author of this paper, and the project grants CB2008-106443 and CB2008-106366.

References

1. **Alsabti, K., Ranka, S., & Singh, V. (1998).** CLOUDS: A decision tree classifier for large datasets. *Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, New York, USA, 2–8.
2. **Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984).** *Classification and Regression Trees*. Belmont, Calif.: Wadsworth International Group.
3. **Chakrabarti, S., Cox, E., Frank, E., Güting, R.H., Han, J., Jiang, X., Kamber, M., Lightstone, S.S., Nadeau, T.P., Neapolitan, R.E., Pyle, D., Refaat, M., Schneider, M., Teorey, T.J., & Witten, I.H. (2009).** *Data Mining: Know it all*. Burlington, MA: Elsevier/ Morgan Kaufmann Publishers.
4. **Domingos, P. & Hulten, G. (2000).** Mining high-speed data streams. *Sixth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (KDD'00),* Boston, MA, USA, 71–80.
5. **Gama, J. & Medas, P. (2005).** Learning decision trees from dynamic data streams. *Journal of Universal Computer Science*, 11(8), 1353–1366.
6. **Gehrke, J., Ganti, V., Ramakrishnan, R., & Loh, W.Y. (1999).** BOAT – Optimistic decision tree construction. *ACM SIGMOD Record*, 28(2), 169–180.
7. **Gehrke, J., Ramakrishnan, R., & Ganti, V. (2000).** Rainforest – A framework for fast decision tree construction of large datasets. *Data Mining and Knowledge Discovery*, 4(2-3), 127–162.
8. **Janikow, C.Z. (1998).** Fuzzy decision trees: Issues and methods. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 28(1), 1–14.

9. **Mehta, M., Agrawal, R., & Rissanen, J. (1996).** SLIQ: A fast scalable classifier for data mining. *Fifth International Conference Extending Database Technology (EDBT)*, Avignon, France, 18–32.
10. **Mitchell, T.M. (1997).** *Machine Learning*. New York: McGraw Hill.
11. **Pérez, J.M., Muguerza, J., Arbelaitz, O., Gurrutxaga, I., & Martin, J.I. (2007).** Combining multiple class distribution modified subsamples in a single tree. *Pattern Recognition Letters*, 28(4), 414–422.
12. **Quinlan, J.R. (1986).** Induction of decision trees. *Machine Learning*, 1(1), 81–106.
13. **Quinlan, J.R. (1993).** *C4.5: Programs for Machine Learning*. San Mateo, Calif.: Morgan Kaufmann Publishers.
14. **SDSS - Adelman-McCarthy, J., Agueros, M. A., Allam, S.S., et al. (2008).** The Sixth Data Release of the Sloan Digital Sky Survey. *The Astrophysical Journal Supplement*, 175(2), 297–313.
15. **Shafer, J.C., Agrawal, R., & Mehta, M. (1996).** SPRINT: A scalable parallel classifier for data mining. *22nd International Conference on Very Large Data Bases (VLDB '96),* Mumbai, India, 544–555.
16. **Hsing-Kuo, P., Shou-Chih, C., & Yuh-Jye, L. (2005).** Model trees for classification of hybrid data types. *6th international conference on Intelligent Data Engineering and Automated Learning (IDEAL'05),* Queensland, Australia, 32–39.
17. Kdd cup 1999 data (1999). Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
18. **Utgoff, P.E. (1989).** Incremental induction of decision trees. *Machine Learning*, 4(2), 161–186.
19. **Utgoff, P.E., Berkman, N.C., & Clouse, J.A. (1997).** Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1), 5–44.
20. **Yang, B., Wang, T., Yang, D., & Chang, L. (2008).** BOAI: Fast alternating decision tree induction based on bottom-up evaluation. *12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'08),* Osaka, Japan, 405–416.
21. **Yoon, H., Alsabti, K., & Ranka, S. (1999).** *Tree-based incremental classification for large datasets*. Technical Report (TR-99-013), Gainesville, FL.: University of Florida.

Anilu Franco Arcega received her B.S. and M.Sc. degrees in Computer Science from Autonomous University of Hidalgo State in 2003 and 2006, respectively. Her Ph.D. degree was obtained from the National Institute of Astrophysics, Optics and Electronics (INAOE), Puebla, Mexico.

Jesús Ariel Carrasco Ochoa received his Ph.D. degree in Computer Science from the Center for Computing Research of the National Polytechnic Institute (CIC-IPN), Mexico, in 2001. At present, he is a full time researcher at the National Institute for Astrophysics, Optics and Electronics (INAOE) of Mexico. His current research interests include Sensitivity Analysis, Logical Combinatorial Patter Recognition, Testor Theory, Feature Selection, Prototype Selection and Clustering.

Guillermo Sánchez Díaz received his B.S. degree in Computer Science from Autonomous University of Puebla (BUAP), Mexico, in 1995; his M.Sc. degree in Computer Science from Autonomous University of Puebla, Mexico, in 1997, and his Ph.D. degree in the Center for Computing Research of the National Polytechnic Institute (CIC-IPN), Mexico, in 2001. His current research interests include Pattern Recognition, Data Mining, Image Processing, Testor Theory.

José Francisco Martínez Trinidad received his B.S. degree in Computer Science from Physics and Mathematics School of the Autonomous University of Puebla (BUAP), Mexico, in 1995; his M.Sc. degree in Computer Science from the Faculty of Computer Science of the Autonomous University of Puebla, Mexico, in 1997; his Ph.D. degree from the Center for Computing Research of the National Polytechnic Institute (CIC-IPN), Mexico, in 2000. Professor Martinez-Trinidad edited/authored seven books and over one hundred and twenty journal and conference papers on subjects related to Pattern Recognition.

Article received on 21/09/2011; accepted on 25/09/2011.