An algorithm for high-density n-dimensional data categorization in FPGA

Pablo Huerta[1] Javier Castillo[2] Javier M. Moguerza[3] Javier Cano[4] and José Ignacio Martínez[5]

Resumen— **This paper presents a new method for high density n-dimensional data categorization, which has been specially designed to be implemented in hardware. This method allows to easily identifying groups with similar characteristics (category) from a large set of individuals with a great number of associated variables. Once the categories are established, any new individual can be easily categorized and get an event probability ("score") assigned based on the group to whom it belongs to. This method's best quality is its very hardware-oriented design, as can be seen in the hardware implementation in a Nallatech PCIe-280 FPGA board, processing gigabytes of data in seconds. To name a few, some interesting application fields are medical diagnosis and credit risk assessment.**

Palabras clave— **FPGA, Categorization, ImpulseC**

I. Introduction

A "score" is a number associated to a given individual and computed based on its profile. We understand by "scoring" as the system or mathematical method (usually based on statistical or operational research techniques) used to assign scores to individuals. In general, these scores measure the potential risk associated to individuals. This risk is related to the final result of making some decisions on the individual. For instance, in the banking sector a credit score is a number obtained after an analysis of a person's bank profile. Based on this score, the bank will decide on the approval of a credit or the credit limits on credit cards, see Thomas [7], Martens et al. [4] or Lee and Chen [6]. In Health Sciences scoring is used to determine the risk of patients regarding certain pathologies or diseases. The score may be calculated based on the patient medical tests or his clinical record. In this case for example, the score could measure the risk for a patient of having a certain disease, see Conroya et al. [2] or Rassi et al. [5], among others.

The calculus of scores is based on the analysis of historical data. But, as nowadays the amount of available data is so huge, there is a need for specific techniques. So that, data mining is becoming a relevant source of statistical techniques in order to design scoring procedures, see for instance Hand et al. [3], or Huang et al. [1].

[1]Dpto. DATCCCIA, Univ. Rey Juan Carlos, e-mail: pablo.huerta@urjc.es.
[2]Dpto. DATCCCIA, Univ. Rey Juan Carlos, e-mail: javier.castillo@urjc.es.
[3]Dpto. de Estadística, Univ. Rey Juan Carlos, e-mail: javier.moguerza@urjc.es.
[4]Dpto. DATCCCIA, Univ. Rey Juan Carlos, e-mail: javier.cano.montero@urjc.es.
[5]Dpto. DATCCCIA, Univ. Rey Juan Carlos, e-mail: joseignacio.martinez@urjc.es.

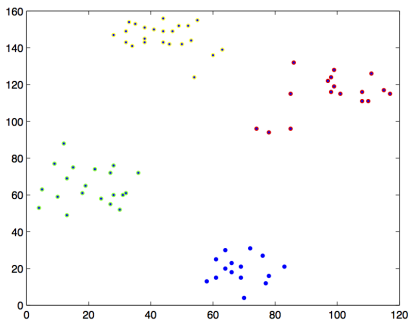

Fig. 1. Ruspini with 4 categories.

This work is focused on the development of a new algorithm and associated hardware architecture useful for real-time scoring with large amount of data. In these circumstances, the software procedures are not efficient enough to generate real-time results.

The proposed categorization algorithm is described in section 3, whilst the Impulse-C language and the hardware architecture developed to speed the algorithm execution up are detailed in Sections 4 and 5, respectively. The paper ends with the experimental results and the conclusions.

II. Related Work

In order to assign a score to a set of data is it needed to separate them in clusters. The clustering problem has been studied during many years due its high number of applications. Given a set of samples, the problem of deciding how many clusters exist and assigning each sample to a cluster is a NP-hard problem.

One of the most commonly used clustering algorithms is the K-means [12] method. For example, figure 1 shows the performance of the K-Means clustering algorithm over the Ruspini set of data.

K-means is an iterative algorithm. In a first iteration, a predefined number of random centroids are generated. Then, the distance from each centroid to every point in the sample is calculated, in order to assign each point to the cluster corresponding to the nearest centroid. In the following iteration, the algorithm recalculates the centroids by solving an optimization problem. Again, the distances from each point to the new centroids are calculated. The method converges when two consecutive iterations provide the same results.

The K-Means algorithm presents two main dis-

advantages, especially when it is going to be used within high dimensional settings. The first one is that the number of clusters, which coincides with the number of centroids, has to be predefined in advance. The second drawback, given that it is an iterative algorithm, is that the number of iterations needed to converge depends on the quality of the initial centroids. This becomes a problem mainly when dealing with large datasets, and therefore the main reason why there are no FPGA implementations of the method for real data.

The only work in FPGA is [9], where the K-Means algorithm is used to colour an image in order to find regions on it.

There are extensions of the K-Means algorithms and other methods to calculate the number of clusters [13], but they have the same problem: they are not suitable for large sets of data, and they are even more complex and take longer execution times than the original K-Means method. In this work, we propose a new algorithm that, for a n-dimensional dataset, is able to calculate the number of clusters and assign the data to them in a linear time. The algorithm has been designed so that its FPGA hardware implementation is possible. The implementation is suitable to have a quick estimation of the clusters. This is particularly important in real time environments, for instance, when a scoring has to be assigned to new individuals, as the scoring depends on the cluster the individual belongs to.

III. Proposed Categorization Algorithm

The main goal of this work is to design and implement a new algorithm for n-dimensional data categorization that can easily assign scores to every category based on event probabilities. From the raw data it is possible to find similarities between the data and create categories so that the individuals can be classified. New individuals are analysed and assigned to existing groups with similar properties (disease probability, credit risk index, etc.). These characteristics make the algorithm a powerful tool for early event detection. It is also important to mention that the algorithm complexity is linear, therefore the algorithm is very appropriate to process large data sets.

The proposed algorithm is not a general solution to the clustering problem. The clustering problem is a NP-hard problem and it has not got a unique solution. Standards techniques have two major drawbacks, first one they are very slow and not suitable for high dimensional data. The second one is that the number of clusters must de defined by the user prior to the algorithm execution. In a real scenario with many Gigabytes or even Terabytes of data is it impossible to the user to set the number of clusters. In this context an algorithm that returns the number of clusters and a possible classification in a linear time is very relevant.

One good example of application for this scoring algorithm is early cancer detection. It should be very useful for the medical staff to be able to classify their big number (thousand or millions of patients with hundreds of variables) of patients into groups (knowing not only if they have previously suffered the disease, but also from the big number of medical parameters in their records) in order to know in advance to what category a new patient should be assigned to and consequently treat the patient.

When we want to find the probability of a new patient to develop a cancer, we have to compute the distance between this new patient and the already computed categories, assigning the new patient to the closest category and, as a consequence, its probability.

The distance function is the key factor for a good data categorization; therefore, it is compulsory to involve the area experts to choose the proper function for each problem.

For instance, disease detection has to manage patient information such as address, age, height, weight, blood type, blood pressure, genetic predisposition, other diseases previously suffered, etc.

An accurate distance function should be able to find patterns about the disease occurrence, for example: people living close to each other with similar age will belong to the same category.

Many problems can be solved in terms of the Euclidean distance computation, but many others might have very complex distance functions (analytical or non-linear table based). Generally speaking, any function should help to know how close an individual is to each of the groups.

The algorithm is divided into two different processes: 1) category creation: where similarities are detected from the raw data, and 2) categorization of new individuals: where an incoming individual is assigned to an existing category.

A. Category creation

The category creation problems consist on: given a set of P individuals calculate the different categories that may exist based on the statistical characteristics of the individuals, and assign each individual to one of the categories. The characteristics of an individual are defined by a set of N variables. A category exists when many individuals share some similar characteristics. An example with individuals with 2 variables (dimension 2) is shown in figure 2, where 4 different categories can be seen.

The process of deciding on how many categories exist is an iterative process of N iterations, where N is the number of variables defining the individuals. On each iteration new categories may be detected and individuals are accordingly assigned to them.

In order to detect the different existing categories the process uses a different distance function $FDist_i$ in each iteration. After N iterations, all the categories have been detected and every individual X has been assigned to one of them. The algorithm is shown below.

for $i = 1 \rightarrow Num\_Dimensions$ **do**
  for $k = 1 \rightarrow Num\_Categories$ **do**

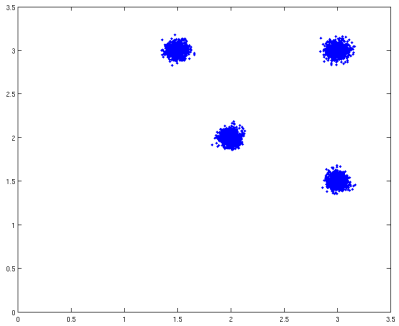

Fig. 2. Example with 4 categories.

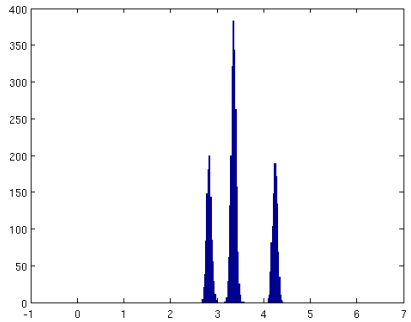

Fig. 3. Distance histogram count.

for $j = 1 \rightarrow Length(Category_k)$ **do**
$D_j \leftarrow FDist_i(X_j)$
end **for** $histo \leftarrow histogram(D_j)$
$histofiltered \leftarrow filter(histo)$
$modalcategories\_detected \leftarrow maxs(histofiltered)$ *Create new detected categories*
for $j = 1 \rightarrow Length(Category_k)$ **do**
add X_j *to the category where* D_j *is closest to the maximun*
end for
end for
end for

The different steps for a two-dimensional example are explained below. The two-dimensional example is used is this work to graphically show the algorithm operation. However the extension to n dimensions is trivial.

In the first step the process computes the distance of each individual to a reference point, usually the Cartesian origin. On extensions of the algorithm we are working selecting different origin points and calculating them in parallel to find out the point that gives the best view of the data.

Depending on the nature of the data a different distance function must be selected. The election of this function is a key decision because the final results strongly depend on it. Many real problems can be solved with the Euclidean distance; therefore this work focuses on a hardware architecture that solves the problem using this distance. Several other problems can be solved using the same hardware architecture just modifying the distance calculation block with the appropriate function for each specific problem.

For a two dimensional problem the Euclidean distance is:

$$D_i = \sqrt{x^2 + y^2} \tag{1}$$

The obtained distances are discretized and used for computing the histogram. The number of statistical modes in the histogram corresponds with the statistical modes and it is also the number of categories in the set of individuals. Figure 3 shows the histogram count for the data set example where three categories have been detected.

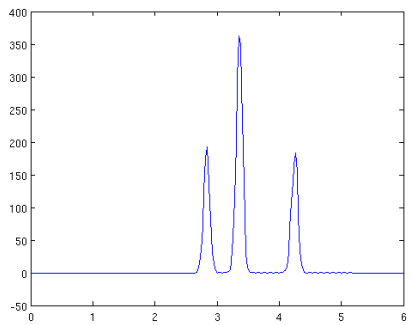

Fig. 4. Filtered histogram signal.

The number of statistical modes of the histogram is computed in terms of the local maximums of the histogram, but before that, a low pass filter eliminates the noise in the histogram (figure 4) so that the local maximums are cleared.

Once the categories have been detected and created, every individual must be assigned to one of them. This is done by comparing the distance of the individual to the centroid of each of the statistical modes found previously, given by the maximums in the histogram, and assigning the individual to the closest one. Figure 5 shows the categorization obtained in this first step, where 3 categories have been detected. Once every individual has been categorized in one of the categories, the second step begins.

The second step is similar to the first one, but this time each category of the first step is separately analyzed in order to find out if can be split into more that one category.

Now, the analysis is carried out with the angle of each individual as the distance function instead of the distance to the origin. Figure 6 shows the angle analysis for the green category of figure 5 whilst Figure 7 shows the final categorization of the initial set of individuals.

Once all the initial set of individuals has been processed and all of the individuals have been assigned to a category, the mean individual value of each category is computed so that can be used in the categorization of new individuals.

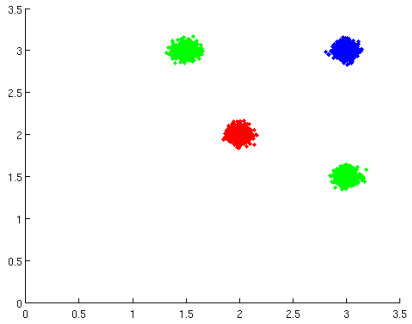

Fig. 5. Individuals assigned to each category.

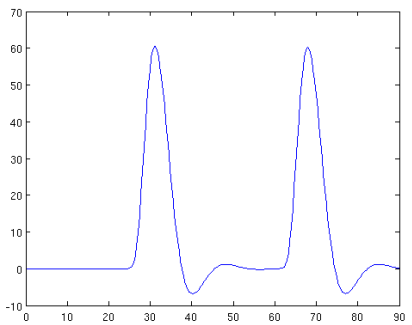

Fig. 6. Angle analysis for one of the categories.

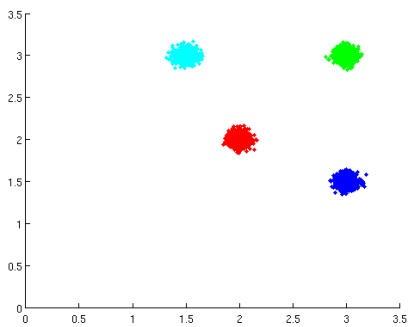

Fig. 7. Final categorization of the set of individuals.

B. Categorization of new individuals

When a new individual arrives to the system it has to be categorized in one of the categories, but it does not mean to repeat the whole recently described process. Instead of that, the distance of the new individual is compared with the mean individual's value of each category and then assigned to the category with the smallest distance value. When the arrival of new individuals alters the statistical characteristics of the global population the whole process of category creating have to be repeated to establish the new categories.

IV. Impulse C

The hardware architecture has been designed using Impulse-C from Impulse Technologies. ImpulseC is a C-based language that allows the design of hardware systems using a full codesign methodology. The system is described using C language and a set of new functions and pragmas to describe the intrinsic parallelism of the hardware. Once the system is validated with the simulations and a target platform is selected, ImpulseC generates the whole set of files (VHDL, C, and platform configuration files) needed to automatically synthetize and compile the design for the platform. This work is part of a much broader project and, although the very powerful Nallatech PCIe-280 board was not supported by ImpulseC, we developed a full BSP (Board support Package) in order to complete the design flow. This package is now officially available for all ImpulseC users.

V. Hardware Architecture

The prototype platform is made up of a Nallatech PCIe-280 board attached to a host PC. The Nallatech PCIe-280 is a PCI-Express accelerator card with a large Xilinx FPGA directly coupled to 2 different types of on-board memory: two 500 Mbyte banks of DDR2 SDRAM and two 9 Mbyte banks of QDR-II SRAM. The board is connected to the host through a PCIe x8 interface that sends and receives data up to 500 MB/s (effectively measured). The hardware architecture implemented in the Nallatech PCIe-280 was optimized to work with bidimensional data.

The system implemented in the board is shown in figure 8.

All the memory banks can be accessed by the host through the PCI-Express interface, and by the accelerator through the memory controller. The two available DDR2 memory banks are used by the host to send the data of the individuals to the accelerator that can access this memory in 128 bit chunks. When one of the banks is filled up by the host, the accelerator starts to process the individuals whilst the host writes more individuals in the second bank, therefore hiding some memory latency.

The data is read from the DDR in bursts so that almost 1 data per cycle can be processed. The data is converted from floating-point to fixed point to be able to use the CORDIC core, because the distances can be normalized for the required precision. There is no problem attached to this precision change because the output is just used to generate a histogram.

The two available SRAM blocks are written and read by the accelerator to get the algorithm results. The results are one entry with each individual and its category.

The blocks labeled step 1 and step 2 in figure 8 implement the distance functions and histogram computations needed in each of the steps.

Each block can calculate several distances in parallel while keeping the histogram updated (Figure 9).

For the example with two variables per individual, the distance function in step 1 is a square root and in step 2 is *arctan*, with also the multipliers and adders that compute the Euclidean distance. Both functions have been implemented using a pipelined CORDIC core. Because the DDR memory data width is 128 bits, a total number of up to 4 values can be read in parallel from the memory, therefore 4 CORDIC

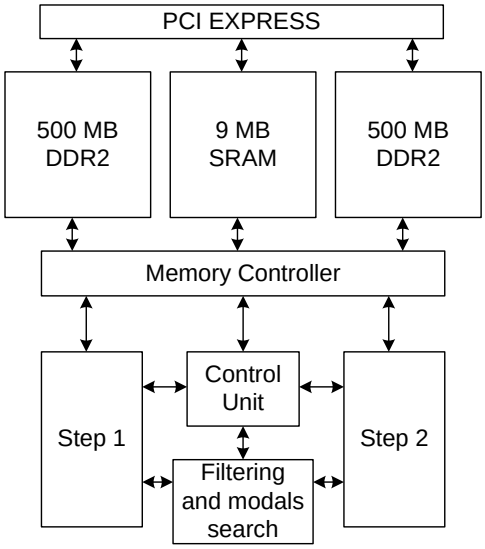

Fig. 8. System architecture.

hardware cores are implemented to speed the computation up.

The histogram calculation needs to process those 4 elements in parallel to be optimal. For this purpose an architecture similar to the one described in [8] has been used. This architecture also works in a pipelined fashion. Once all the data has been processed it is time for the filtering and modals search block to detect which categories may exist in the data. The filtering has been implemented as a FIR filter of order 3.

After detecting the existing categories, all the individuals are read again from memory and the distance of each individual is compared with each category, assigning the individual to the category with the smallest difference. For this, the same blocks are reused and all the comparisons are made in parallel (Figure 9). Because of the parallel comparisons and the hardware architecture and constraints, the maximum number of different categories the system can deal with is 8. The results are stored in the SRAM memory for being read by the host. When one step is finished, the host starts reading the results and copying the data needed for the following step of the algorithm. A control unit synchronizes all the communications within the blocks, and also within the system and the host PC. Extending the system for a higher number of dimensions is trivial: only requires minor changes in the control unit, and implementing the appropriate distance function for each new step added.

VI. Experimental Results

For the sake of comparison, the K-Means algorithm was executed for the same data sets to evaluate the speedup of the proposed algorithm. As said in the Introduction, the K-Means algorithm is not deterministic and has to converge to a solution in a number of iterations that depends of the initially random selected centroids. For a bidimensional data, like the one used in the experiments, the number of

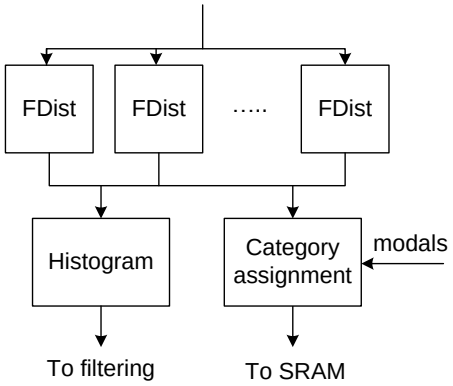

Fig. 9. Step architecture.

Size	Proposed	K-Means	Speed Up
125MB	3,69s	7,92s	2,14x
250MB	7,44s	20,16s	2,71x
500MB	15,58s	55,57	3,56x
1000MB	29,37s	58,44s	1,98x
2000MB	61,26s	145,18s	2,37x

TABLE I
K-Means execution time for data sets compared against software implemetation of the proposed algorithm

iterations to solve the problem goes for 4 to 31, with a statistical mode of 16 iterations.

The execution time for K-Means in this case is:

The proposed algorithm is not only up to 3 times faster than the K-Means algorithm, but also calculates the number of clusters (not calculated by the K-Means algorithm).

The algorithm was implemented and test in the real hardware and compared with the software implementation to evaluate the performance. The computer host for the Nallatech board was an Intel i7 processor with 4GB of RAM memory. The whole system spends a 8% of the Virtex5 330LXT FPGA resources and currently runs at 100 MHz. Random data sets with Gaussian distribution were generated as input to the system and saved to files with several sizes (125MB,250MB, 500MB, 1000MB and 2000MB) constrained to the 500MB size value of the DDR RAM on board.

The same set of experiments was test with the Impulse-C model, i.e. the C golden model of the system showing no performance loss with the ANSI-C model.

Size	Software	Hardware	Speed Up
125MB	3,69s	1,96s	1,87x
250MB	7,44s	3,93s	1,89x
500MB	15,58s	7,71s	2,02x
1000MB	29,37s	16,9s	1,73x
2000MB	61,26s	32,17s	1,90x

TABLE II
Total execution times for hardware and software.

Size	Software	Hardware	Speed Up
125MB	2,10	0,29s	7,08x
250MB	4,27s	0,59s	7,18x
500MB	9,39s	1,19s	7,88x
1000MB	15,48s	2,38s	6,5x
2000MB	32,07s	4,73s	6,78x

TABLE III
PROCESSING TIMES FOR HARDWARE AND SOFTWARE.

The results prove a 2x speedup of the FPGA hardware implementation when compared with the software implementation for a bi-dimensional problem.

It is important to notice that the hardware performance is limited by the time needed to read the data from the hard disk. Table 2 presents the actual processing time when removing the hard disk read delay, assuming that the data is already loaded in the RAM memory. The hardware processing time includes the data transfer from the host to the board using the DMA and the FPGA processing time. It is very important to notice that the speedup in the processing time comes for the use of the *arctan* function in the distance calculation, which is much faster in the CORDIC engine of the FPGA than in the host processor. If more sophisticated distance functions are needed to solve more complex problems the increase of performance (and speedup) will be substantially higher.

Due to the algorithm, and the transfer and processing times are linear, the speedup will keep constant for more dimensional computations. This means that for high execution times the speedup results in a significant execution time reduction.

VII. Conclusion

A novel algorithm for high-density n-dimensional data categorization was presented. The algorithm linear complexity is very appropriate to process large amounts of data. Although the algorithm performance is good enough in software, the hardware architecture implemented in the Nallatech PCIe-280 board results in a substantial speedup increase when compared to the software version. As a future work, the algorithm will be test in a real problem environment (disease preventive diagnose or credit risk evaluation) with emphasis in finding the best distance functions to properly split big data sets into categories. We hoped that more complex distance functions in n-dimensional data will prove the benefits of the hardware implementation because of the much higher and better speedups.

VIII. Acknowledgment

We must particularly thank Impulse Technologies, Nallatech and Xilinx for their support.

Referencias

[1] Cheng-Lung Huang, Mu-Chen Chen, Chieh-Jen Wang, *Credit scoring with a data mining approach based on support vector machines*, Expert Systems with Applications 33 (2007), 847-856

[2] R.M. Conroya, K. Pyöräläb, A.P. Fitzgeralda, S. Sansc, A. Menottid, G. De Backere, D. De acquere, P. Ducimetièref, P. Jousilahtig, U. Keilh, I. Njølstadi, R.G. Oganovj, T. Thomsenk, H. unstall-Pedoel, A. Tverdalm, H. Wedeln, P. Whincupo, L. Wilhelmsen, Estimation of ten-year risk of fatal cardiovascular disease in Europe: the SCORE project, European Heart Journal (2003) 24 (11): 987-1003. Nan-Chen Hsieh, *An integrated data mining and behavioral scoring model for analyzing bank customers*, Expert Systems with Applications 27 (2004), 623-633

[3] D. J. Hand, H. Mannila, P. Smyth, *Principles of data mining*, The MIT Press, 2001.

[4] D. Martens, B. Baesens, T. Van Gestel, J. Vanthienen, *Comprehensible credit scoring models using rule extraction from support vector machines*, European Journal of Operational Research 183 (2007), 1466-1476

[5] Anis Rassi, Jr., Anis Rassi, William C. Little, Sergio S. Xavier, Sergio G. Rassi, Alexandre G. Rassi, Gustavo G. Rassi, Alejandro Hasslocher-Moreno, Andrea S. Sousa and Mauricio I. Scanavacca, *Development and Validation of a Risk Score for Predicting Death in Chagas' Heart Disease*, New England Journal of Medicine; 355 (2006), 799-808

[6] Tian-Shyug Lee, I-Fei Chen, *A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines*, Expert Systems with Applications 28 (2005), 743-752

[7] Lyn C. Thomas, *A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers*, International Journal of Forecasting 16 (2000), 149-172

[8] Cadenas, J., Sherratt, R. S. and Huerta, P., *Parallel pipelined histogram architectures*. Electronics Letters, 47 (20) (2011), 1118-1120

[9] Wang, Xiaojun and Leeser, Miriam, *K-means Clustering for Multispectral Images Using Floating-Point Divide*. Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM '07, (2007), 151-162

[10] MacQueen, J. B., *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. In Some Methods for Classification and Analysis of MultiVariate Observations, University of California Press, (1967), 281-297

[11] Pelleg, Dan and Moore, Andrew W., *X-means: Extending K-means with Efficient Estimation of the Number of Clusters*, Proceedings of the Seventeenth International Conference on Machine Learning, (2000), 727-734

[12] Mardia K.V., Kent J.T. and Bibby J.M., *Multivariate Analysis*, Academic Press, 1979

[13] Moguerza J.M., Muñoz A. and Martín-Merino M., *Detecting the number of clusters using a Support Vector Machine Approach*, Lecture Notes in Computer Science, Vol. 2415, pp. 763-768, Springer Verlag, 2002.