Sorted kernel matrices as cluster validity indexes

Francisco A. A. Queiroz[1] Antonio P. Braga[1] Witold Pedrycz[2]

1 Department of Electronics Engineering, Federal University of Minas Gerais
Belo Horizonte, Brazil
2 University of Alberta, Department of Electrical and Computing Engineering
Edmonton, Canada
Email: faaq@ufmg.br, apbraga@ufmg.br, pedrycz@ece.ualberta.ca

***Abstract**—Two basic issues for data analysis and kernel-machines design are approached in this paper: determining the number of partitions of a clustering task and the parameters of kernels. A distance metric is presented to determine the similarity between kernels and FCM proximity matrices. It is shown that this measure is maximized, as a function of kernel and FCM parameters, when there is coherence with embedded structural information. We show that the alignment function can be maximized according FCM and kernel parameters. The results presented shed some light on the general problem of setting up the number of partitions in a clustering task and in the proper setting of kernel parameters according to structural information.*

Keywords— Affinity matrix, Clustering, Fuzzy C-Means (FCM), Kernel matrix, Reordering, Sorting.

1 Introduction

The importance of kernels has been much emphasized in the literature as the basic construct of learning machines like SVMs [6]. Kernel parameters play, therefore, a major role in achieving good performance of these classifiers. Setting up kernel parameters is usually accomplished by an exhaustive search carried out in the space of parameters. Quite often this search is completed without a full understanding of how kernels represent relations in the input space. The proper set-up of kernels is only expected to capture the input to feature space mapping that results on an acceptable overall machine performance, without any direct constraints imposed on the internal representation of relationships between patterns.

However, like Fuzzy C-Means (FCM) [7] proximity matrices (FPM) [8], kernels may incorporate important structural information about the data. In fact, data clustering can be accomplished by using information already contained in kernels, by finding the proper order of columns and rows of the kernel matrix [2, 3, 4, 5]. A similar affinity matrix [1] representation can also be obtained for FPMs, which may suggest that kernels and FPMs embody similar information about the data if their parameters are set accordingly. This observation also suggests that supervised and unsupervised learning may exhibit a closer relationship by the exchange of information between kernels and FPMs.

The problem of approximating kernels and FPMs relies, therefore, on setting their parameters so that they yield matrices that are close to each other or, in other words, that represent the same structural information. The parameters to be set are basically the width *r* of the Gaussian functions, for RBF kernels, and the number of partitions c, the basic parameter of FPMs. In this paper, we investigate how these two parameters are related to the alignment between RBF kernels and FPMs.

Experimental results on synthetic data yielded maximum alignment on values of *c* that coincide with the number of clusters of the data generator functions. This suggests that, for a given kernel matrix, the proper number of clusters can be induced by maximizing the similarity of FPM and the kernel matrix. In addition, it is also shown that, for the same problems, SVMs [6] can also be designed by setting the kernel parameter to the corresponding value of *r* that maximizes the similarity.

The paper is organized as follows. We start with a general overview of the main concepts including kernels and FPM. The concept of affinity matrix representation of kernels and FPMs is then described. Another important link between these two representations of data is presented in section 5 where kernels are shown to incorporate the same structural information the FPMs. The metric for similarity measure is then presented, which is followed by the description of the optimization algorithm, the main results, discussions and conclusions.

2 Mercer Kernels

The main characteristic of (Mercer) kernels [9] in the supervised learning context is that they allow implicit mapping of input data into feature space without actually computing the mapping itself. Instead, the mapping is accomplished by computing the internal product located in the input space. The kernel matrix **K**=[k(x_i,x_j] is regarded as a Mercer kernel if it is symmetric and positive semi-definite. In general, the kernel matrix can be considered as a matrix capturing similarity between all pairs of points of a data set. Kernels can be implemented by polynomials, linear and sigmoidal functions as well as Gaussian functions. A Gaussian, or RBF, kernel is described as follows

$$k(x_i, x_j) = e^{\frac{-(x_i - x_j)^2}{2r^2}} \qquad (1)$$

where *r* is the radius of the Gaussian function.

As an example, let us consider Fig. 1, where data were sampled from two Gaussian distributions. The corresponding RBF kernel where *r=0.25* is shown in Fig. 2. As patterns are not ordered according to the generator functions, no structural information can be directly visualized in the kernel matrix. In order to observe pattern relation properties directly from the kernel matrix, it is necessary to order the patterns accordingly, as it will be described in the next sections.

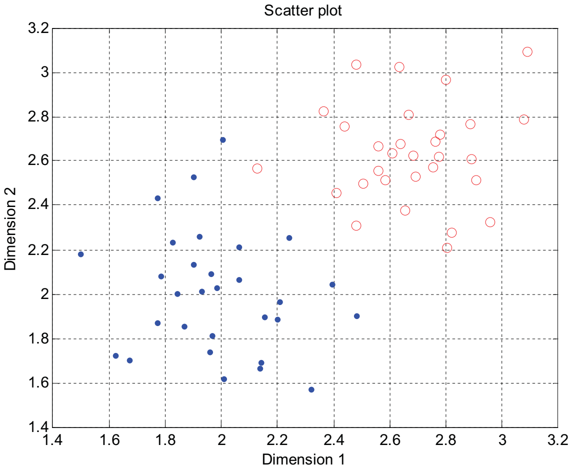

Figure 1: Data sample from two Gaussian distributions.

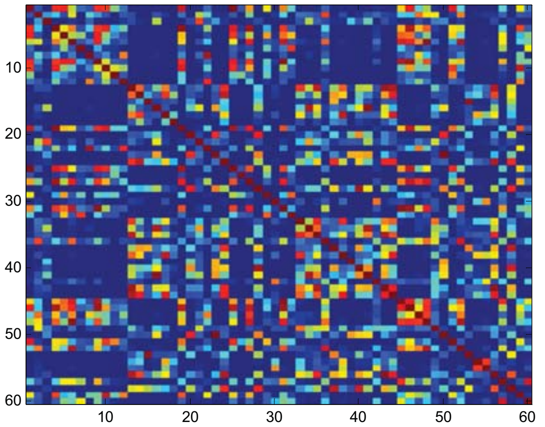

Figure 2: Unordered RBF kernel matrix (r=0.25) for the data of Fig. 1.

3 FCM Proximity Matrix

Fuzzy clustering realized by the well-known FCM algorithm [7] is formulated as a constrained optimization problem, where the partition matrix **U**=[u$_{ik}$] satisfies the constraint $\sum_{i=1}^{c} u_{ik} = 1$, while the objective function to be minimized reads as follows

$$J = \sum_{i=1}^{c} \sum_{k=1}^{N} u_{ik}^{m} \| \mathbf{x}_k - \mathbf{v}_i \|^2 \qquad \mathbf{(2)}$$

where *c* is the number of partitions (clusters) in the data set, *N* is the data set size, *m* is the fuzziness coefficient (fuzziness factor) and **v**$_i$ is the prototype of cluster i.

The obtained partition matrix **U=[u$_{ij}$]** can be used to generate the *NxN* proximity matrix **P**=[p$_{kl}$], according to the known relationship available in the literature [8] that is presented in (3). Likewise kernels, the proximity matrix **P** embodies relationships between patterns according to the clusters represented in the partition matrix **U**.

$$p_{kl} = \sum_{i=1}^{c} \min(u_{ik}, u_{il}) \qquad (3)$$

4 Affinity matrices

Given a data set $\Gamma_u = \{x_k\}_{k=1}^{N}$, where N is the number of patterns, the elements s$_{ij}$ of the Affinity Matrix **S**=[s_{ij}] contain a measurement or estimation of the affinity of the pair of patterns (**x**$_i$, **x**$_j$), where affinity is defined as a likeness based on relationship or causal connection [1]. For reflexive affinities, **S** is symmetrical, what implies on $s_{ij}=s_{ji}$. In general, the Affinity Matrix can be represented as a block diagonal symmetrical matrix as the one of (4).

$$S = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1k} \\ S_{21} & S_{22} & \cdots & S_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ S_{k1} & S_{k2} & \cdots & S_{kk} \end{bmatrix} \qquad (4)$$

where $\boldsymbol{S}_{ij}=\boldsymbol{S}_{ji}$ are sub-matrices of **S** and *k* is the total number of sub-groups of **S**.

For ordered data sets, kernels and proximity matrices can be represented in the form of (4), where sub-matrix **S**$_{ii}$ represents within-cluster affinities and **S**$_{ij}$ represents inter-cluster affinities, for *i≠j*. Consider, for instance, the data set of Fig. 1 and the corresponding kernel presented in Fig. 2, which is now ordered according to the generator distributions and presented in Fig. 3. The block-diagonal affinity matrix form of the kernel presented in Fig. 3 shows clearly the structure of the data set. The block-diagonal matrices **S**$_{11}$ and **S**$_{22}$ represent the two clusters of Fig. 1 with 30 elements each, whereas the matrices **S**$_{21}$ and **S**$_{12}$ represent the relationships between data of matrices **S**$_{11}$ and **S**$_{22}$. Matrices **S**$_{21}$ and **S**$_{12}$ have in fact very small values, indicating that the two clusters are not quite related to each other, as can be confirmed in Fig. 1. A similar result to the one presented in Fig. 3 would have been obtained if the proximity matrix **P** were obtained and ordered according to the partition matrix **U** for c=2.

ISBN: 978-989-95079-6-8

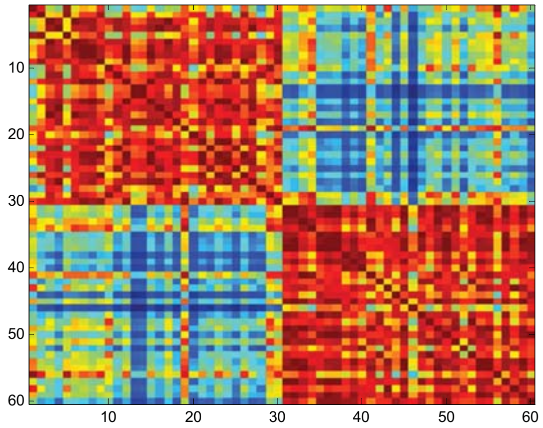

Figure 3: Ordered RBF kernel matrix (*r=0.25*) for the data set of Fig. 1.

5 Clustering by Sorting Kernels

In the previous section we have pointed out to the close relationship between kernels and proximity matrices as represented by affinities of the data. Clustering algorithms can be described with the objective of sorting kernel matrices. There are in fact clustering algorithms in the literature that work by sorting rows and columns of the kernel matrix [2, 3, 4, 5] in order to obtain affinity matrices like the one presented in Fig. 3. Once the affinity matrix has been formed, embedded cluster information can be directly extracted from the kernels.

The eigenvector ordering method [1, 2] was applied to the data set presented in Fig. 4, which was generated from 6 Gaussian generator functions. The corresponding ordered RBF kernel is presented in Fig. 5. As it can be observed, the ordered kernel shows clearly the presence of 6 groups of data in the input space. The affinity matrix representation of kernels simply reveal the information that is already present in the kernel, since only permutation operations were accomplished in the rows and columns of the original matrix. This indicates that (properly set) kernels already contain information about data structure that is equivalent to that obtained by clustering methods.

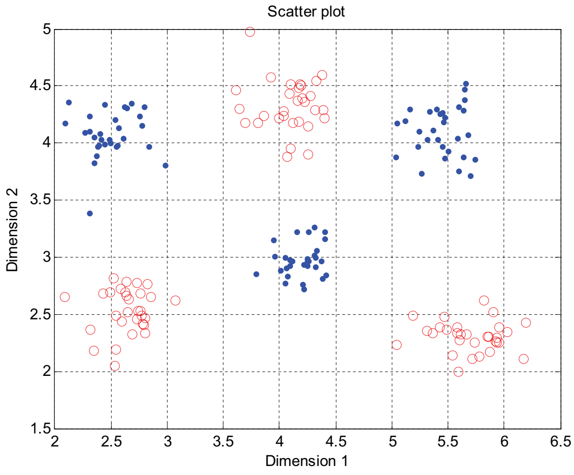

Figure 4: Data set obtained from 6 Gaussian functions.

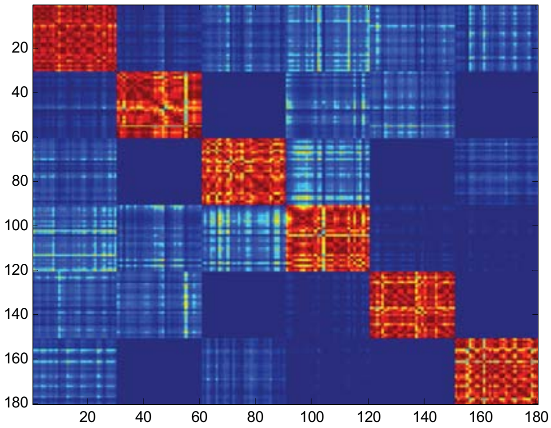

Figure 5: Affinity matrix representation of the ordered RBF kernel matrix of the data set of Fig. 4.

6 Alignment of Kernels and FPMs

In order to quantify the similarity between kernels and proximity matrices, the Empirical Alignment described in [2] was adopted. The alignment quantity A is described in the form

$$A(K,P)=\frac{\langle K,P\rangle_F}{\sqrt{\langle K,K\rangle_F\langle P,P\rangle_F}} \tag{3}$$

where K and P, respectively, are the kernel and proximity matrices and $\langle .,. \rangle_F$ is the Frobenius inner product [2],

$$\langle K,P\rangle_F=\sum_{i=1}^{N}\sum_{j=1}^{N}K(i,j)P(i,j) \tag{4}$$

7 Alignment as an Optimization Problem

As discussed in the previous sections, depending on their parameters, kernels and FPM may incorporate analogous information about patterns and cluster relationships which hold in the input space. For the RBF kernels, structural information may be revealed by the proper setting of the width *r*, whereas FPM ability to describe pattern relations depends on the previously set number of clusters *c*. Nevertheless, these parameters are usually blindly configured in advance by the user. In the SVM design, the value of *r* can be fine-tuned according to the overall learning machine performance, while further cluster analysis can also give a hint for fine tuning of the value of *c*. When the setting of (*c*, *r*) results on affinity matrices that are coherent with the actual data structure, the corresponding kernel and FPM will be aligned to each other, according to the metric presented in (3). These arguments suggest that A(K,P) has a maximum on (c^*, r^*), where c^* and r^* are, respectively, the number of partitions and RBF width that best describe the original data structure. This characterizes the optimization problem presented in (5). The objective is therefore to obtain the values of the parameters *c* and *r* that maximize the alignment, which are expected to be the ones that describe better the data set for both kernel and proximity matrix.

$$\underset{(c,r)}{\arg\max}\; A(K,P) \tag{5}$$

In order to demonstrate the arguments above, the values of A(K,P) were computed as a function of *c* and *r* for the data sets coming from Fig. 1 and 4, respectively. The obtained results are shown in Fig. 6 and 7. The same calculations were also completed for another data set with 4 original Gaussian generator functions. The obtained graph is presented in Fig. 8. As can be observed in Fig. 6, 7 and 8, the maximum occurs for *c=2*, *c=6* and *c=4*, respectively, which correspond to the original number of generator functions used in each data set. These results confirm the principle that the alignment function (3) has a maximum corresponding to the number of generator functions and that (5) could be optimized in order to obtain the values of *c* and *r* that result on the best representation of the data set. In the next section, the optimization of (5) will be implemented with a simple Genetic Algorithm (GA) [10] approach.

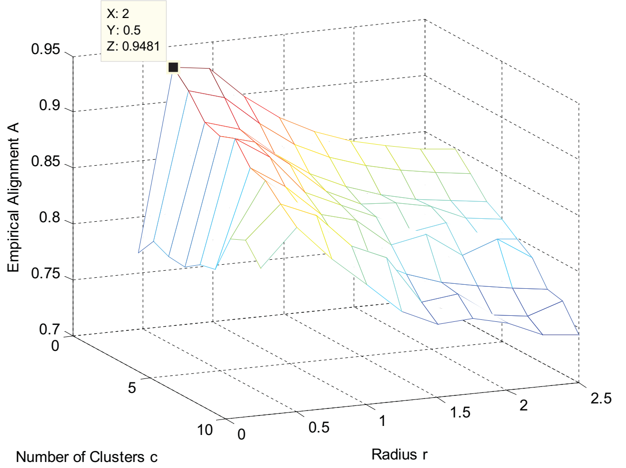

Figure 6: Alignment A(K,P) as a function of c and r for the data set of Fig. 1.

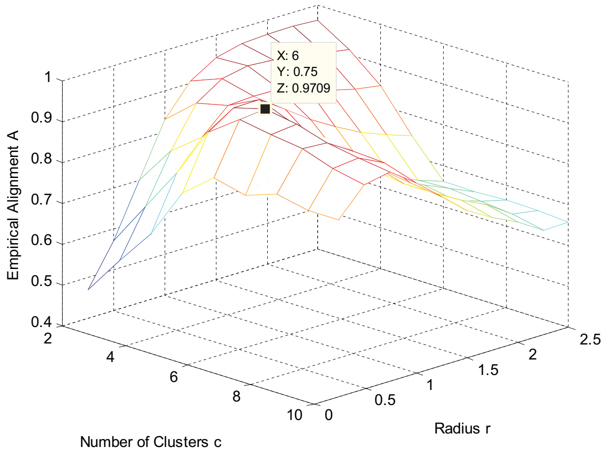

Figure 7: Alignment A(K,P) as a function of c and r for the data set of Fig. 4.

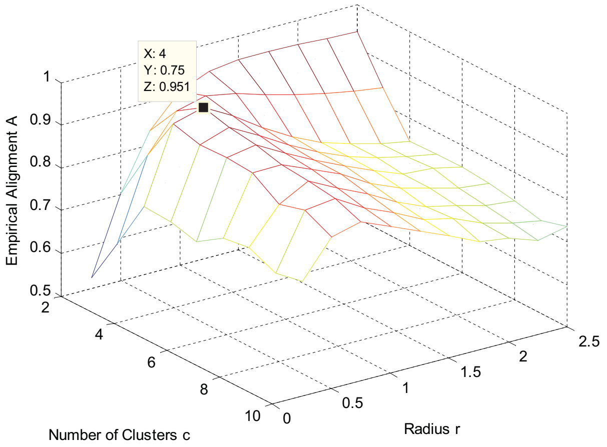

Figure 8: Alignment A(K,P) as a function of c and r for a data set with 4 Gaussian generator functions.

8 Evolutionary Optimization

A simple Genetic Algorithm (SGA) [10] was implemented in order to optimize (5) for the three problems represented by the objective functions of Fig. 6, 7, and 8. GA parameters are presented in Table 1 and the obtained results in Table 2. As can be observed, the maximum obtained for the three objective functions are quite close to the ones observed in the figures. As for the number of clusters, the optimization yielded exactly the same number of generator functions.

Table 1: Parameters of the SGA for optimization of the objective functions of Fig. 6, 7 and 8.

Number of variables in search space	2
Population size	50
Maximum number of generations	65
Crossover rate	0.6
Mutation rate	0.03
Biased linear cross-over coefficient	0.9
Linear cross-over coefficient	0.5

Table 2: SGA solutions for the objective functions of Fig. 6, 7 and 8.

Number of generator functions	***Number of iterations***	***r***	***c***	**A**
Two	52	0.5894	2	0.9540
Four	19	0.6808	4	0.9537
Six	63	0.6551	6	0.9758

The SGA was also applied to the data set of Fig. 9, represented as a binary classification problem. In order to distinguish between structural information and data set labels for classification problems, one of the classes was sampled from two Gaussian distributions, as can be observed in Fig. 9. In such a situation, there are three generator functions and two classes. This can be visualized by the ordered kernel of Fig. 10, which suggests the existence of the three clusters, in spite of the number of labels. The SGA optimization resulted on *r=0.66959*, *c=3* and *A=0.92225*,

ISBN: 978-989-95079-6-8

what is consistent with the number of clusters of the original distribution.

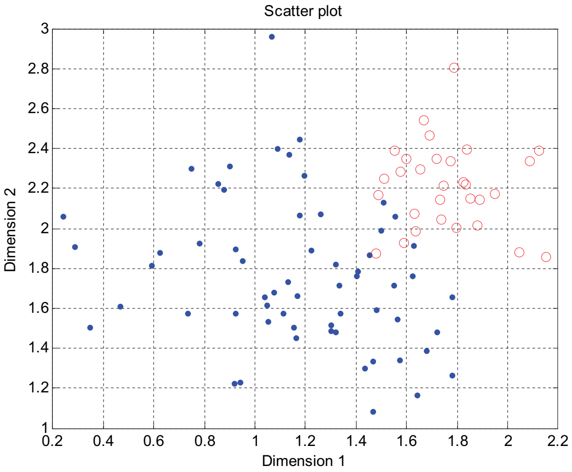

Figure 9: Binary classification problem sampled from three generator functions.

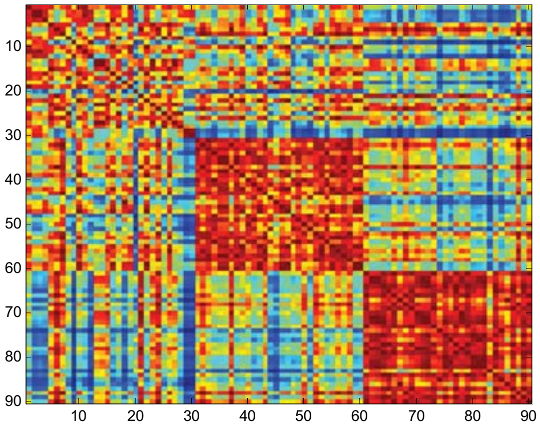

Figure 10: Affinity matrix representation of the ordered RBF kernel matrix of the data set of Fig. 9.

In order to show the consistency of the RBF width resulted from SGA optimization for classification purposes, a SVM was designed with the obtained value of *r=0.66959*. The corresponding separating surface is presented in Fig. 11 and, as can be observed, the resulting surface is coherent with the classification problem (the margin parameter was set to *C=2*). This suggests that the maximum obtained from (5) also points out to a proper tuning of kernel parameters for SVM design. Similar procedures were applied to the Iris data set [12], which resulted on *c=2* in the maximum alignment. This result is consistent with the literature and with statistical analysis of the data set [13].

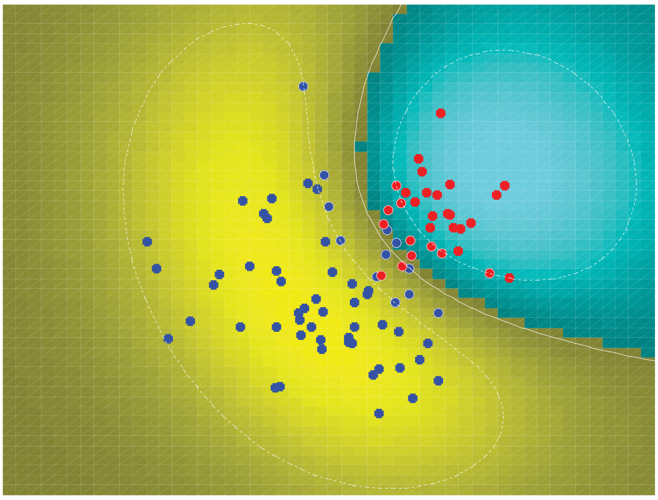

Figure 11: SVM separation surface for Fig. 9. RBF kernel was selected with and *r=0.66959* obtained from SGA optimization.

9 Discussions and conclusions

Kernel and clustering design was presented in this paper in the perspective of function optimization. Our main argument to support this principle is that when both kernel and FPM are coherent with the embedded structural information contained in the data set, they should be aligned to each other. Alignment is measured in this paper according to the Frobenius inner product [2] between kernel and FPM, computed as a function of the parameters *c* and *r*. Therefore, we argue that maximum alignment should result on a proper setting of the parameters *c* and *r*. This approach differs from other kernel FCM algorithms [11], since the original objective functions are used.

This principle sheds some light on the discussion of determining the number of partitions of a clustering task. It has been shown throughout the paper that coherent partitions can be inferred according to the maximum alignment principle. Likewise, proper kernel design can also be obtained from the alignment. This suggests that kernel can be designed by exploring the structural data properties instead of performing an exhaustive blind search in the space of parameters with the aim of solely accomplishing the input-output mapping. According to these arguments, kernels should be more representative of the underlying data instead of serving uniquely as a mapping engine. This also suggests that kernels and FCM clustering should be co-designed and not described independently. According to the same principle, supervised and unsupervised learning may exhibit a closer relationship by collaborative interaction between kernels and FPMs.

References

[1] Y. Weiss, Segmentation using eigenvectors: a unifying view, *Proceedings of the International Conference on Computer Vision*, Volume 2, p. 975, May 20-25, 1999.

[2] N. Cristianini, J. Kandola, A. Elisseeff and J. Shaw-Taylor, On kernel target alignment, *Journal of Machine Learning Research*, 1, 2002.

[3] L. Võhandu, R. Kuusik, A. Torim, E. Aab, G. Lind, Some algorithms for data table (re) ordering using Monotone Systems, *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases: 5th WSEAS International*

Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED'06) , 417-422, Madrid, Spain, Feb 15-17, 2006.

[4] D. Tsafrir, I. Tsafrir, L. Ein-Dor, O. Zuk, D. A. Notterman and E. Domany, Sorting points into neighborhoods (SPIN): data analysis and visualization by ordering distance matrices. *Bioinformatics*, 21:2301-8, 2005.

[5] P. Arabie, L. J. Hubert, The bond energy algorithm revisited, *IEEE Transactions on Systems, Man and Cybernetics*, 20 (1): 268-274, Jan-Feb 1990.

[6] C. Cortes, V. Vapnik, Support Vector Network, *Machine Learning*, vol.20, pp.273-297.

[7] J. C. Dunn, A Fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, *Journal of Cybernetics* 3: 32-57. 1973.

[8] V. Loia, W. Pedrycz and S. Senatore, Proximity fuzzy clustering for web context analysis, *Proceedings of EUSFLAT 2003*, pp. 59-62.

[9] N. Cristianini and J. Shawe-Taylor, Support vector machines and other kernel-based learning methods. Cambridge University Press, 2000.

[10] D. E. Goldberg, Genetic algorithms in search, optimization and machine learning. Kluwer Academic Publishers, Boston, MA, 1989.

[11] W. Cai, S. Chen and D. Zhang, (2007). Robust fuzzy relational classifier incorporating the soft class labels**,** *Pattern Recognition Letters*, vol. 28(16), pp 2250-2263.

[12] A. Asuncion, D.J. Newman, UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science, 2007.

[13] S. R. Gunn, Support vector machines for classification and regression (Technical Report). Image Speech and Intelligent Systems Research Group, University of Southampton, 1997.