Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 34 (2007) 3154–3161

computers & operations research

www.elsevier.com/locate/cor

K-T.R.A.C.E: A kernel k-means procedure for classification[☆]

C. Cifarelli[a], L. Nieddu[a], O. Seref[b], P.M. Pardalos[b,*]

[a]Dipartimento di Statistica, Probabilità e Statistiche Applicate Università di Roma "La Sapienza", Italy
[b]Center for Applied Optimization, University of Florida, Gainesville, FL 32611-6595, USA

Available online 30 January 2006

Abstract

In a computational context, classification refers to assigning objects to different classes with respect to their features, which can be mapped to qualitative or quantitative variables. Several techniques have been developed recently to map the available information into a set of features (feature space) that improve the classification performance. Kernel functions provide a nonlinear mapping that implicitly transforms the input space to a new feature space where data can be separated, clustered and classified more easily. In this paper a kernel revised version of the Total Recognition by Adaptive Classification Experiments (T.R.A.C.E) algorithm, an iterative k-means like classification algorithm is presented.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Classification; k-means; Kernel functions

1. Introduction

There has been a recent increasing interest in applying kernel methods to pattern recognition problem. Kernel methods represent the current state-of-the-art for classification algorithms. Kernel functions were first introduced in 1964 [1]. With the development of support vector machines (SVMs) [2] they have become one of the most popular and powerful tools in classification. Kernel methods allow classification and clustering algorithms to work in a nonlinearly embedded higher-dimensional feature space in order to obtain nonlinear partitions of the input space, which provide greater separability [3–5].

The computation of the coordinates of each vector in a high-dimensional feature space can become computationally intractable, considering that the dimension of the feature space can extend to infinity. This problem can be overcome by the so-called "kernel trick" [6], which maps the data to the kernel induced feature space without explicitly computing the features. The only computation needed is the pairwise inner products, which can be computed efficiently using kernel functions.

The aim of this study is to present a new version of a k-means-based supervised classification technique, where the data points in the input space are mapped to a high-dimensional feature space using kernel functions. The presented algorithm has already been used in several applications, both using benchmark and real data sets (mostly medical), yielding comparable results in all cases. Some of the latest results have been obtained in medical diagnosis, particularly

☆ This work is partially supported by NSF and NIH.
* Corresponding author. Tel.: +1 352392 9011; fax: +1 352392 3537.
E-mail address: pardalos@cao.ise.ufl.edu (P.M. Pardalos).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.11.023

in the field of human-assisted reproduction [7], and in protein secondary structure classification [8]. Our algorithm is a revised version of the Total Recognition by Adaptive Classification Experiments (T.R.A.C.E) [9,10] algorithm, which partitions the input space into subspaces such that each subspace belongs to one and only one class, and thus the algorithm obtains a complete classification of the training set.

Given all of the quantifiable features of an object listed in a specific order as a vector, each object can be represented as a point in a space whose dimension is equal to the number of features. The standard form of the T.R.A.C.E algorithm generates barycenters, i.e., centers of mass for each class until any data point is closest to a barycenter of its own class compared to the barycenters of the other classes, with respect to Euclidean distance. In order to provide better separability of the classes, the kernel implementation modifies the standard T.R.A.C.E algorithm to map the data points that are originally in the *input space* in a higher dimensional space called the *feature space*. In particular, it uses the "kernel trick" to calculate the distances between the data points, and the distances of the data points from the barycenters, using only their inner products. Similar to several other nonlinear clustering and classification techniques such as kernel k-means and kernel nearest-neighborhood algorithms [4,3], the kernel implementation of the T.R.A.C.E algorithm achieves higher accuracy in the classification rate and computational improvements compared to the standard implementation.

The rest of the paper is organized as follows. In Section 2 the T.R.A.C.E algorithm in its standard version and in its new kernel implementation are presented. In Section 3 experimental results on benchmark data set are shown. Finally, in Section 4, conclusions are drawn and future work directions are addressed.

2. The T.R.A.C.E algorithm

The classification algorithm presented in this study is a kernel version of the T.R.A.C.E. algorithm, which has been applied in several classification problems in biomedical research [7,11]. In this section, the standard version of the T.R.A.C.E. algorithm and its kernel implementation are presented.

The T.R.A.C.E algorithm is a supervised learning algorithm [12]. This means that the algorithm needs to be trained using a set of data points (training set) that has previously been classified by an expert or in any other appropriate way. Once trained, new data points can be classified whose classes are unknown to the classifier. The classification rate can be found by comparing the actual class labels of each data point and the labels assigned by the classifier. Usually, in order to asses the performance of the algorithm, a cross validation technique is used. In an n-fold cross validation, the entire data set, whose actual class membership information is known, is divided into n random subsets of equal cardinality. One of these subsets, say subset i, is chosen as the test set and the classifier is trained using all of the data points in the remaining subsets, $s = 1, \dots, n$, $s \neq i$. Then, all of the data points in the test set i are classified with the trained classifier. This procedure is repeated for each subset $i = 1, \dots, n$ and all of the cumulative classification results are compared to the actual labels to find the overall classification rate.

The T.R.A.C.E algorithm first computes the barycenter $\mathbf{b}_c^0$ of each class $c = 1, \dots, k$. Then it computes the Euclidean distance of each data point from each barycenter. If each data point is closer to the barycenter of its class than to the barycenter of any other class, the algorithm stops. Otherwise, there is a non-empty set $\mathscr{M}$ of data points that are closer to a barycenter of a different class than any barycenter of their own class. The algorithm selects the data point $x_j \in \mathscr{M}$ that is the farthest from the barycenter of its class. This data point is used as a seed for a new barycenter for the class of x_j. Then, a k-means algorithm [13] is performed until convergence, considering only the data points in the class of x_j and using the barycenter of that class and the new seed as starting points. After the execution of k-means algorithm, the set of barycenters has $k + 1$ elements, since a barycenter of the class of the seed is added to the initial set. The barycenters at the new iterations do not need to be computed for all classes, but only for the considered class, since the barycenters of the other classes remain unchanged.

In the following step, the distances of each data point from all of the barycenters $\mathbf{b}_c^p$ are computed anew, and set $\mathscr{M}$ is updated. The algorithm stops when set $\mathscr{M}$ becomes empty. The convergence of the algorithm in a finite number of steps is proved in a number of ways [10]. The pseudo-code for the T.R.A.C.E algorithm is given in Fig. 1.

After convergence, this algorithm produces a set of barycenters, whose cardinality is bounded below by the number of classes, and above by the number of data points. The aim of this algorithm is to find subclasses in the data set which can be used to classify new data points of unknown class. Occurrence of subclasses with only one data point is possible when the data set is not representative of the population that it is generated from, or the data set is undersampled, or the data

```
Step1 Let
        – $\mathbf{x}_j$, $j = 1, \ldots, n$ be the data points in the training set;
        – $\mathbf{B}_0$ be the set of $k$ initial barycenters $\mathbf{b}_c^{t(c)}$, $t(c) = 0$, $c = 1, \ldots, k$;
        – $D(\cdot, \cdot)$ be the distance between any two points;
        – $\psi(\mathbf{x}_j)$ return the class label for data point $\mathbf{x}_j$;
Step2 Compute $D(\mathbf{x}_j, \mathbf{b}_c^p)$ for all data points $\mathbf{x}_j$ and barycenters $\mathbf{b}_c^0 \in \mathbf{B}_0$;
      Let $\mathcal{M}$ be the set of data points $\mathbf{x}_j$ such that there exist a barycenter $\mathbf{b}_{c^*}^{p^*}$,
          $c^* \neq \psi(\mathbf{x}_j)$, for which
              $D(\mathbf{x}_j, \mathbf{b}_{c^*}^{p^*}) < \min_p \{D(\mathbf{x}_j, \mathbf{b}_{\psi(\mathbf{x}_j)}^p)\}$;
      $t \leftarrow 0$;
Step3 while $\mathcal{M} \neq \emptyset$
        – Let $\mathbf{x}_s \in \mathcal{M}$, where
              $\mathbf{x}_s : \max_{\mathbf{x}_j \in M} \{\min_p \{D(\mathbf{x}_j, \mathbf{b}_{\psi(\mathbf{x}_j)}^p)\}\}$;
        – Let $t(c) \leftarrow t(c) + 1$; $\mathbf{b}_c^{t(c)} \leftarrow \mathbf{x}_s$; $B_{t+1} \leftarrow B_t \cup \mathbf{x}_s$;
        – for all $\mathbf{x}_j \in \psi(\mathbf{x}_s)$, $j = 1, ..., n$, perform a k-means routine using barycenters
          $\mathbf{b}_{\psi(x_s)}^p \in B_{t+1}$ as starting points;
        – Compute $D(\mathbf{x}_j, \mathbf{b}_c^p)$ for all data points $\mathbf{x}_j$ and barycenters $\mathbf{b}_c^p \in \mathbf{B}_{t+1}$;
        – Update $\mathcal{M}$ as in Step 2;
        – $t \leftarrow t + 1$;
      end
```

Fig. 1. Pseudo-code for the T.R.A.C.E algorithm.

set has outliers. In this case, the barycenter, which is the single data point itself, is called a singleton. The significance of singletons and some important theoretical results are discussed next, followed by the kernel implementation of the T.R.A.C.E algorithm.

2.1. Theoretical results

First, we discuss the properties and convergence results for the T.R.A.C.E algorithm. We give the definition of coherent data, and show that the algorithm converges in a finite number of steps if the data set is coherent.

Definition 2.1. A data set is *coherent* if the classes define a partition on the data set.

Definition 2.1 implies that if the data set is coherent, then two elements belonging to different classes cannot occupy the same point in the input space.

Remark 2.1. If the data set is coherent the T.R.A.C.E algorithm converges in a finite number of steps [10].

Note that the previous remark is not an "if and only if" property, i.e., it is still possible to achieve convergence even if the data set is incoherent. It is also likely that two identical elements which belong to different classes move back and forth from one cluster to another. Therefore this anomalous iteration can continuously create new barycenters and new clusters with no elements inside, making the algorithm iterate indefinitely. In the case of incoherent data, coherency can be gained by increasing the dimension of the input space by kernel functions until any two data points belonging to different classes occupy different points in the new feature space. After the algorithm converges, the sets of barycenters can be used to classify new data points of unknown classes simply by assigning the new elements to the class whose barycenter is the closest. Finally, note that the convergence of the algorithm implies that each point in the training set can be classified correctly by the barycenters created using the training set itself.

One of the important properties of the T.R.A.C.E algorithm is its capability in averaging out additive white noise on the data while computing the barycenters. Moreover, outliers can be detected from the training results. If an outlier is added to the data set, it most likely forms a singleton during training. Therefore an accurate study of the barycenters with few elements (one, two or three elements) may give further insights to the problem. In fact, this evaluation can help in detecting some errors of the expert (supervisor) that has previously classified the data (recognition with imperfect supervisor). Various techniques can be applied to correct outliers [7,10].

2.2. Kernel implementation

Here, we introduce a kernel implementation of the T.R.A.C.E algorithm. First, the definition of kernel and the widely used kernel types are presented. Then, the formulation of kernel-induced distances between two points, and between a point and a barycenter are given. Finally, results regarding the kernel implementation versus standard implementation are shown on a two-dimensional, two-class example.

For a given data set, if a suitable kernel exists, then the initial data can be mapped to a higher-dimensional embedded feature space with increased separability. As a result, the number of barycenters decreases, and the less number of barycenters yields a more robust classifier. In order to achieve this objective, the classification problem has to be reformulated using kernels. It must be pointed out that a kernel implementation does not explicitly uses a mapping ϕ of a data point x_i from the input space to $\phi(x_i)$ in the feature space. The mapping is achieved by different functions applied to inner products between the data points. The matrix that includes all of the inner products between the data points in the feature space is called the *kernel or Gram matrix* [1,14].

Definition 2.2. A kernel is a function H that satisfies

$$H(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \quad \text{for all } \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X},$$

where $\mathbf{X}$ is the set of all data points, and ϕ is a mapping from $\mathbf{X}$ to a feature space $\mathbf{F}$ induced by the inner products of the data points, that is, $\mathbf{x} \in \mathbf{X} \longrightarrow \phi(\mathbf{x}) \in \mathbf{F}$.

In Eqs. (1–3) some common kernel functions used to embed the data points x_i, $i = 1, \ldots, N$ in the feature space are given. These kernels are called the *linear* kernel (1), the *polynomial* kernel (2) and the *Gaussian* kernel (3).

$$H_l(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j), \tag{1}$$

$$H_p(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d, \tag{2}$$

$$H_e(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma}\right). \tag{3}$$

Although these kernel functions are among the most frequently used, there are many other kernel functions available [15]. A kernel implementation of a classification technique could also benefit from the modularity of these kernel methods in general. Appropriate kernel functions allow learning algorithms to be applied in many data domains [6]. For example, string kernels can be used to cluster and classify string data [16] for information retrieval and bioinformatics problems as well as for protein folding and DNA data sequences [17].

In the kernel implementation of the T.R.A.C.E algorithm, data points are implicitly mapped to the feature space by their inner products. Let the mapped point for $\mathbf{x}_i$ in the input space be $\mathbf{u}_i = \phi(\mathbf{x}_i)$. Similarly, let the mapped point for barycenter $\mathbf{b}_c^p$ in the input space be $\mathbf{z}_c^p = \phi(\mathbf{b}_c^p)$ in the feature space. We use two measurements of distances in the feature space: the Euclidean distance $D(\mathbf{u}_i, \mathbf{u}_j)$ between two data points and the Euclidean distance $D(\mathbf{u}_i, \mathbf{z}_c^p)$ between a data point u_i and a barycenter $\mathbf{z}_c^p$. Let the set of points that define barycenter p of class c be β_c^p. Then, this barycenter can be represented as

$$\mathbf{z}_c^p = \frac{1}{|\beta_c^p|} \sum_{j \in \beta_c^p} \mathbf{u}_j.$$

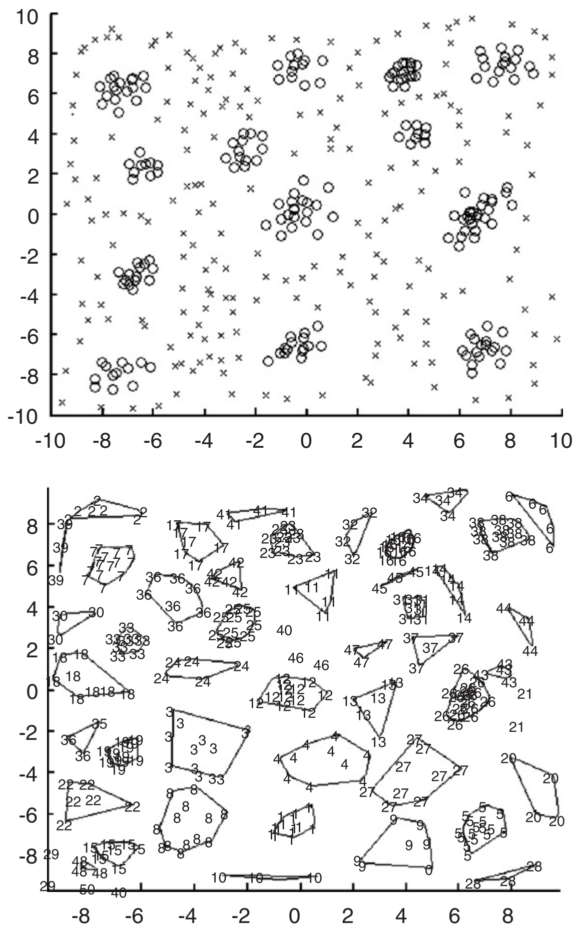

Fig. 2. Subclasses needed to obtain the correct classification of the training set. The first figure shows the toy data set used, the second the subclasses obtained using the standard version of the algorithm (linear kernel).

We replace the distances in the input space with the distances in the feature space, and we use the kernel functions to calculate these distances without explicitly using the mapped points in the feature space. The Euclidean distances between two data points, and between a point and a barycenter are given as follows.

1. The Euclidean distance between two data points:

$$\begin{aligned} D(\boldsymbol{u}_i, \boldsymbol{u}_j) &= \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\| \\ &= \langle \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) \rangle = H(\mathbf{x}_i, \mathbf{x}_i) + H(\mathbf{x}_j, \mathbf{x}_j) - 2H(\mathbf{x}_i, \mathbf{x}_j). \end{aligned}$$

2. The Euclidean distance between a data point and a barycenter:

$$\begin{aligned} D(\mathbf{u}_i, \mathbf{z}_c^p) &= \left\| \mathbf{u}_i - \frac{1}{|\beta_c^p|} \sum_{j \in \beta_c^p} \mathbf{u}_j \right\| \\ &= H(\mathbf{x}_i, \mathbf{x}_i) + \frac{2}{|\beta_c^p|} \sum_{j \in \beta_c^p} H(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{|\beta_c^p|^2} \sum_{j \in \beta_c^p} \sum_{l \in \beta_c^p} H(\mathbf{x}_j, \mathbf{x}_l). \end{aligned}$$

These distance functions are used both in the training and classification phase to transform the standard k-means procedure to a kernel k-means procedure. This kernel implementation is referred to as K-T.R.A.C.E. for the rest of the paper.

Note that K-T.R.A.C.E with linear kernel (1) is equivalent to the algorithm in its original version because the distances in the feature space are equivalent to the Euclidean distances in the input space. Therefore the kernel version should be considered as a generalization of the original T.R.A.C.E algorithm.

In order to graphically demonstrate how the algorithm works, a toy example is created as shown in Fig. 2. This data set is composed of 200 elements and 2 classes. The first plot shows the binary data set considered, and the second figure shows the 47 subclasses generated by the algorithm with a linear kernel when convergence is achieved, i.e., that every element is closer to the barycenter of its own subclass. The clusters generated with the Gaussian kernel cannot be represented graphically since the feature space is high-dimensional, and the clusters are not convex in the two-dimensional input space due to the nonlinear mapping.

The kernel version of the T.R.A.C.E algorithm generates only 6 clusters for this problem. The less number of generated subclasses not only leads to a better generalization and classification, but also manages to cluster non-linearly related cases that cannot be clustered together in the input space. This result shows the robustness of the kernel implementation of T.R.A.C.E and the improvements over the standard version.

3. Experimental results

The proposed algorithm is evaluated on several benchmark data sets. These data sets are obtained from UCI repository [18] and are widely used to compare the classification performance of new algorithms with the existing ones. Our algorithm can easily be applied to multi-class classification problems. However, the majority of the widely used classifiers involve two-class classification such as SVMs. Therefore, we used the two-class version of the data sets from the UCI repository. We also present the results for the toy example used in the previous section.

We compare K-T.R.A.C.E with *SVMs*, *nearest neighbor* and *classification trees* techniques, all of which are supervised classification methods. The comparison results are presented in Table 1. All of the results for the compared techniques are obtained from published results, except for two data sets, on which we applied LIBSVM [19], one of the most widely used SVM implementations. SVMs and K-T.R.A.C.E are trained using Gaussian kernel, which is the most widely used kernel in the literature and in practice. For each data set the best parameters for the Gaussian kernel are obtained through a grid search. K-T.R.A.C.E with Gaussian kernel consistently achieves higher accuracy in less number of iterations compared to the linear kernel case, which is equivalent to the standard version of T.R.A.C.E.. This result confirms the capabilities of kernel functions to obtain feature spaces that provide better separation of the data. The classification accuracies for each method are obtained using ten fold cross-validation, and the results for the best kernel are shown. In general, SVMs and K-T.R.A.C.E achieve comparable results and dominate the other methods in every data set.

Another important result is the difference between the number of barycenters generated for each data set during the execution of K-T.R.A.C.E versus the standard T.R.A.C.E algorithms. In Table 2, the average number of barycenters generated from the ten training sets are given. In the first column, the number of elements of each data set is shown to be compared with the number of barycenters generated by the T.R.A.C.E and the K-T.R.A.C.E algorithms. K-T.R.A.C.E

Table 1
Experimental results obtained with K-T.R.A.C.E. compared with the standard version of the algorithm other widely used algorithms

Data	K-T.R.A.C.E.	T.R.A.C.E.	SVM	1-NN	C&RT
Wave	92.7	85.4	90.1 (3)	84.7	84.7
Wpbc	74.8	66.0	75.3 (*)	67.5	68.1
Example	99.1	95.0	98.7 (*)	95.5	85.2
Ionosphere	94.2	88.3	95.4 (2)	86.6	86.3
Bupa	67.1	65.8	70.4 (2)	63.1	67.0
Cleveland	81.9	73.7	80.8 (1)	75.4	68.4
German	73.6	69.5	75.9 (2)	70.1	70.4
Wbc	96.6	95.7	96.4 (2)	95.3	95.2

For the SVM column we cited the results obtained in (1) Auer et al. 2002 [20], (2) Gestel et al. 2004 [21] and (3) Muller et al. 2001 [22]. (*) we used results obtained using LIBSVM [19].

Table 2
Average of the number of barycenters computed to obtain the classification of the training set

Data	Elements	Number of baricenters	
		T.R.A.C.E.	K-T.R.A.C.E.
Wave	5000	630.7	372.8
Wpbc	194	67.7	4.5
Boubble	400	48.6	6.3
Ionosphere	351	62.2	25.7
Bupa	345	153.8	122.8
Cleveland	297	83.6	71.7
German	1000	321.9	170.1
Wbc	683	54.8	33.5

dramatically reduces the overall number of barycenters in most of the data sets. This result shows the improvement in the computational burden over the standard algorithm.

4. Conclusion and future work

The method introduced in this paper is a powerful classification tool whose performance is comparable to the state-of-the-art methods. With its multi-class classification capability, it can easily be used in a wide range of applications.

Future work on the current algorithm has three main aspects. The first main aspect is regarding the subclasses with one or two data points. Patrizi and Nieddu [10] have shown that using various sampling methods on the data, it is possible to achieve a classification rate that can be used as an estimation of the T.R.A.C.E algorithm. This method, which is called *labeling*, is developed to account for the cases where the data is under-represented and to prevent consequent misclassifications. The results obtained with this procedure are reported in the previous paper. Labeling is not considered in the current work, however it can easily be included in K-T.R.A.C.E algorithm.

The second main aspect includes the application of statistical tools such as novelty detection and cluster quality considerations in order to analyze the subclasses generated in detail. These subclasses may identify particular behaviors in the data.

The last main aspect, which is currently under study, is the development of an optimized version of K-T.R.A.C.E algorithm in order to decrease the computational burden of the algorithm on massive data sets.

Acknowledgements

We acknowledge Prof. Giacomo Patrizi for his useful comments and suggestions.

References

[1] Aizerman M, Braverman E, Rozonoer L. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control 1964;25:821–37.
[2] Vapnik VN. The nature of statistical learning theory. New York: Springer; 1995.
[3] Yu K, Ji L, Zhang X. Kernel nearest-neighbor algorithm. Neural Processing Letters 2002;15:147–56.
[4] Zhang R, Rudnicky AI. A large scale clustering scheme for kernel k-means. ICPR02, vol. IV, 2002. p. 289–92.
[5] Mika S, Ratsch G, Weston J, Scholkopf B, Muller KR. Fisher discriminant analysis with kernels. Neural Networks for Signal Processing IEEE 1999;IX:41–8.
[6] Shawe-Taylor J, Cristianini N. Kernel methods for pattern analysis. Cambridge, UK: Cambridge University Press; 2004.
[7] Manna C, Moscatelli C, Nieddu L, Patrizi G. Pattern recognition methods in human-assisted reproduction. International Transactions in Operational Research 2004;11(4):365–79.
[8] Patrizi G, Cifarelli C. Solving large protein folding problem by a linear complementarity algorithm with 0–1 variables. Optimization Methods and Software 2005; accepted.
[9] Patrizi G. Optimal clustering properties. Ricerca Operativa 1979;10:41–64.

[10] Nieddu L, Patrizi G. Formal methods in pattern recognition: a review. European Journal of Operational Research 2000;120:459–95.
[11] Patrizi G, Nieddu L, Mingazzini P, Paparo F, Patrizi G, Provenza C. et al. Algoritmi di supporto alla diagnosi istopatologica delle neoplasie del colon. A.I * I.A 2002;2:4–14.
[12] Watanabe S. Pattern recognition: human and mechanical. New York: Wiley; 1985.
[13] Gordon AD. Classification. London: Chapman & Hall; 1999.
[14] Cristianini N, Shawe-Taylor J. An introduction to support vector machines. Cambridge, UK: Cambridge University Press; 2000.
[15] Gartner T. A survey of kernels for structured data. SIGKDD Exploration Newsletters 2003;5(1):49–58.
[16] Joachims T, Cristianini N, Shawe-Taylor J. Composite kernels for hypertext categorisation. In: Proceedings of the international conference on machine learning, ICML'01. 2001.
[17] Leslie C, Eskin E, Noble WS. The spectrum kernel: a string kernel for SVM protein classification. In: Russ B, Altman A, Dunker K, Hunter L, Lauerdale K, Klein TE, editors. In: Proceedings of the pacific symposium on biocomputing 2002. Singapore: World Scientific; 2002. p. 564–75.
[18] Blake CL, Merz CJ. UCI repository of machine learning databases; 1998.
[19] Chang CC, Lin CJ. LIBSVM: a library for support vector machines; 2001 (http://www.csie.ntu.edu.tw/cjlin/libsvm).
[20] Auer P, Burgsteiner H, Maass W. Reducing communication for distributed learning in neural networks. Artificial neural networks, ICANN; 2001. Berlin: Springer.
[21] Gestel TV, Suykens J, Baesens B, Viaene S, Vanthienen J, Dedene G. et al. Benchmarking least squares support vector machine classifiers. Machine Learning 2004;54:5–32.
[22] Muller KR. An introduction to kernel-based learning algorithms. IEEE Transaction on neural networks 2001;12(2):181–201.