Kernel-based outlier preserving clustering

Marie-Jeanne Lesot, Rudolf Kruse

Faculty of Computer Science, University of Madgeburg

Universitätsplatz 2, D-39106 Magdeburg, Germany

E-Mail: {lesot, kruse}@iws.cs.uni-magdeburg.de

Abstract

We propose a kernel-based method to identify both major and marginal behaviours present in a dataset, so as to provide a complete and accurate simplified description of it. The kernel framework makes it possible to take into account more flexible metrics than the euclidian one and in particular to handle non-vectorial data, independently of the data nature. We consider an application to specific XML data representing student results to several exams and present results obtained on a real dataset.

Keywords: clustering, outlier handling, kernel methods

1 Introduction

Kernel methods [7, 6] constitute a set of learning algorithms which make it possible to extend classic learning algorithms: on the one hand, this extension leads to an implicit data representation enrichment, which allows to address tasks requiring a richer framework than the linear approach and simultaneously preserving this simple formalism; on the other hand, it makes it possible to apply classic algorithms to non-vectorial data, represented by more complex objects, such as sequences, trees or more generally graphs.

These properties are achieved by means of a specific formulation of the algorithms, that only depends on scalar products: kernel methods take as input a matrix containing

these scalar product values for each data couple. Thus, they do not depend on the data nature but only on the comparison between points and can be applied to any data for which a comparison function having the properties of a scalar product can be defined.

In this paper, we consider a kernel approach for the Outlier Preserving Clustering Algorithm (OPCA) [5], which is a clustering algorithm that identifies in a dataset the major trends, as any clustering method, but also the atypical behaviours, corresponding to outliers or small outlying groups: the latter are associated to linguistic labels such as "abnormal" and contribute to the data characterisation; they should not be overlooked, but should be present in the final data description as any other cluster since they are necessary to a simplified and still accurate representation of the initial dataset. A kernel variant of this algorithm makes it possible to consider other scalar products than the classic euclidian one, and to apply it to non-vectorial data.

In the first section, we recall the principles of the OPCA algorithm; we then present its extension to the kernel case and illustrate its application on artificial data. Lastly, we consider its application to XML data representing student results to several exams.

2 Outlier Preserving Clustering Algorithm

The Outlier Preserving Clustering Algorithm [5] provides a complete characterisation of a dataset by identifying both the major and the marginal trends present in the data: it identifies subgroups as any clustering algorithm, but also one-point clusters, corresponding to outliers and lastly intermediate clusters corresponding to small sets of similar outliers, which can be overlooked by both clustering techniques and outlier detection methods.

To that aim, it explicitely takes into account the double objective of clustering, namely compactness and separability of the extracted subgroups: it is based on the combination, in an iterative process, of the single linkage agglomerative hierarchical clustering algorithm, denoted AHC_{min} in the following, and the fuzzy c-means algorithm, denoted *fcm*. This combination exploits the algorithms respective advantages: *fcm* build especially compact clusters, and AHC_{min} is sensitive to the minimal distance between subgroups and thus identifies well-separated clusters. The iterative process makes it possible to take into account several distance scales and to adapt locally to the data density. OPCA can be seen as a divisive method which selects at each step the most appropriate clustering algorithm

and modifies its parameters according to local criteria.

More precisely, it considers a data subgroup G and tests its separability. Indeed, AHC_{min} is applied if G distribution presents a significant gap, which is determined by computing the quotient between the maximal merging cost of AHC_{min}, i.e. the cost of considering G as a non-divisible group, and a locally minimal significant distance. If the group is not separable, its compactness is measured, to determine whether it corresponds to a chaining effect [2] which cannot be handled by AHC_{min}: if the compactness is low, G is split by *fcm*, whose optimal c value is determined according to a stability criterion. The compactness is measured as a function of G diameter and its variance, compared to the average variance of its possible subgroups (see [5] for more details).

3 OPCA kernel extension

Kernel methods are based on an *implicit* data transformation: $\phi : \mathcal{X} \rightarrow \mathcal{F}$ where $\mathcal{X}$ denotes the input space and $\mathcal{F}$ is called the *feature space*; $\mathcal{F}$ is usually of high or even infinite dimension and is only constrained to be a Hilbert space. The second principle of kernel methods is that data are not handled directly in the feature space but only through their scalar products, which are computed using the initial representation, through the *kernel function* $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, defined such that $\forall x, y \in \mathcal{X}, \langle \phi(x), \phi(y) \rangle = k(x, y)$. Thus the function ϕ needs not be known explicitely, scalar products in the feature space only depend on the initial representation. This *kernel trick* first makes it possible to implicitly enrich a vectorial data representation, as the learning method is implicitly applied in the high dimensional feature space; second it allows to apply algorithms designed for vectorial data to non-vectorial data, as only the matrix of kernel values is involved and not data points as such.

3.1 Hierarchical clustering with kernel

For hierarchical clustering algorithms, the kernel extension is straightforward as data points are not involved as such, but only through their distances. Now the latter are functions of the scalar products: $d(x, y) = \sqrt{\langle x - y, x - y \rangle} = \sqrt{\langle x, x \rangle - 2\langle x, y \rangle + \langle y, y \rangle}$ Thus replacing the scalar product with the kernel function, one can implicitly apply

hierarchical algorithms in the feature space, considering the kernel-based distance:

$$d_K(x,y) = \sqrt{k(x,x) - 2k(x,y) + k(y,y)} \quad (1)$$

3.2 Kernel fuzzy c-means

The fuzzy c-means transposition to the kernel framework is less straightforward as the algorithm is based on the computation of barycenters, depending on the data themselves.

Fuzzy shell clustering Some variants of the fuzzy c-means have been explicitely proposed to take into account other metrics than the euclidian one, for instance so as to detect lines, ellipsoids or quadrics [4]. They aim at extracting prototypes which have a different nature than the data points, and thus they modify the distance definition between points and cluster prototypes. The kernel approach differs from this fuzzy shell point of view: the similarity is computed between data point couples, and does not involve the cluster centres; the kernel influences more directly which points are to be grouped in the same cluster and the prototypes themselves usually have no explicit representation as they belong to the feature space.

Kernel approach There have been several propositions for kernelizing the fuzzy c-means. Zhang and Chen [11] specifically consider the Gaussian kernel, which makes it possible to reduce the computational cost of the method. Wu et al. [10] consider the more general case, their algorithm can be applied for any kernel type and in particular for structured data. It consists in transposing the *fcm* cost function to the feature space, i.e. applying it to the transformed data $\phi(x)$. Provided the cluster centres are looked for as linear combinations of the transformed data, which is consistent with the centre expression obtained in the non-kernel case, it can be shown that the update equations for the cluster centres and the membership degrees only depend on scalar products and thus on the kernel [10].

3.3 Kernel OPCA definition

Algorithm presentation The OPCA kernel variant, denoted kOPCA and detailed in Table 1, is based on the combination, in an iterative process, of the kernel single linkage hierarchical clustering algorithm ($kAHC_{min}$) and the kernel fuzzy c-means (*kfcm*) [10]:

as its non-kernel equivalent, it considers a data subgroup G and tests its separability, but the latter is considered in the feature space $\mathcal{F}$. If G contains well-separated clusters, $kAHC_{min}$ is applied; otherwise, its compactness in $\mathcal{F}$ is measured, to determine whether it still requires a subdivision (e.g. in chaining effect cases). If the compactness is low, G is split by the kernel *fcm*. This process is then iterated on the obtained subgroups.

The criteria to select the algorithms and their parameters are transposition to the feature space of the criteria used in OPCA: for the hierarchical algorithm, they only depend on distances, and thus can be computed in $\mathcal{F}$. Indeed, $kAHC_{min}$ is applied if G distribution in $\mathcal{F}$ presents a significant gap, which is determined by computing the quotient between the maximal merging cost of $kAHC_{min}$ and a locally minimal significant distance, defined as the distance between points in G (eq. (2)). Considering the kernel distance, as defined in eq. (1), instead of the euclidian one in the input space, this criterion can be computed in the feature space.

The $kAHC_{min}$ algorithm provides a whole nested sequence of clustering result possibilities, one of which must be selected as the desired result. To that aim, one must indicate which of the proposed merges are carried out, which is done through a cost threshold s^*: only the merges whose associated cost is below s^* are performed. The threshold s^* is defined depending on the merging cost average and its standard deviation (eq. (3)); this is equivalent to performing a fixed proportion of the proposed merges at each step, the proportion being determined by a user-defined parameter α.

The kernel *fcm* application depends on G compactness. The latter is measured by the group diameter (eq. (4)), which again corresponds to a distance. A second criterion is used (eq. (5)), based on the gain provided by *fcm*, measured as the average standard deviation of G candidate subgroups compared to that of G; this quantity can be computed in the feature space as indicated in eq. (6). Lastly the optimal cluster number c is determined according to a stability criterion which depends on the kernel *fcm* cost function (eq. (7)).

Algorithm parameters kOPCA depends on a single parameter, α involved in eq. (3), that determines the proportion of $kAHC_{min}$ proposed merges performed at each step. It indirectly rules the final number of clusters and should be defined according to the expected outlier proportion: outliers correspond to the most expensive merges which should not be performed; thus their cost should be over the threshold determined by eq. (3). We usually choose $\alpha \in [2, 5]$.

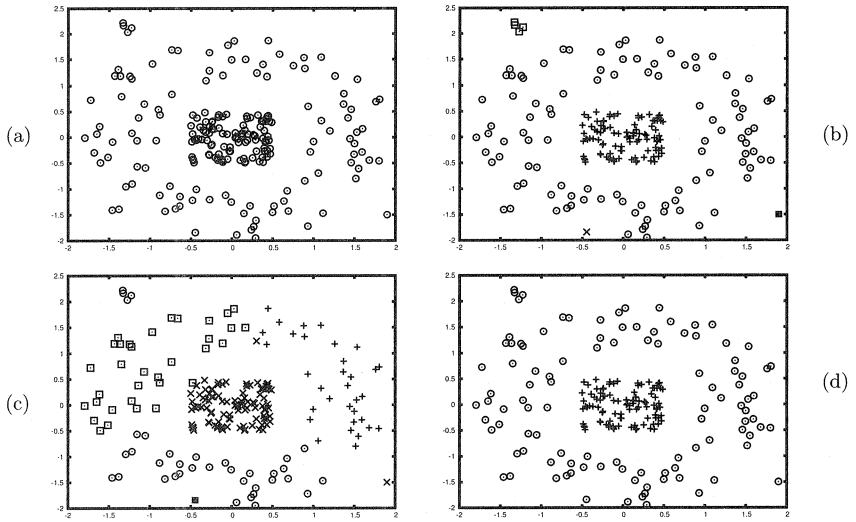


Figure 1: Clustering results using several clustering algorithms: (a) Considered dataset; (b) kOPCA (5 clusters); (c) OPCA (7 clusters); (d) kernel *fcm* (2 clusters).

It is to be noticed that the results obtained by kOPCA also depend on the kernel function choice and on its parameters. This dependence is equivalent to the influence the data representation has on clustering results in the non-kernel case.

3.4 Experimental results on an artificial dataset

We tested kOPCA on a 2D dataset, illustrated on Figure 1a, composed of a noisy ring surrounding data generated by a uniform distribution, one outlier located in (1.9, -1.5), and a small outlying group, located around (-1.4, 2.1), generated following a uniform distribution. We used $\alpha = 3$ and a gaussian kernel with standard deviation 0.5. Figure 1b shows that kOPCA identifies 5 clusters, among which the ring and the internal uniform distribution; it isolates the outlier and a locally isolated point located in (-0.5, -1.9). Lastly it identifies the small outlying group, providing a result compatible with the expectations.

Figure 1c shows the OPCA results and highlights the kernel advantage: according to the euclidian distance, the ring cluster cannot be considered as compact, thus it is split into four subgroups, which misrepresents the data distribution. Still, the outlying data

are successfully identified. Figure 1d illustrates the kernel *fcm* results with the stability-based selection criterion for the optimal c value: as it is also based on the gaussian kernel, it identifies the ring cluster, but it cannot isolate the outliers, be it the small outlying group or the outlier point. It assigns them to the noisy ring, loosing information about the specificity of these data points.

4 Application to structured data

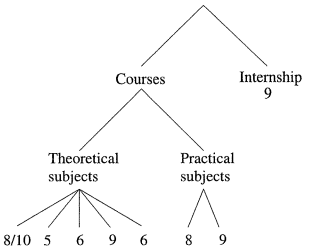
In this section, we consider the application of kOPCA to specific XML data, representing student results to several exams: the XML structure expresses the relationships between the exams, e.g. opposing theoretical and practical subjects; an example is provided on the left part of Figure 2, where internship results are opposed to the course ones, which are divided into theoretical subjects (5 components) and practical ones (2 components).

These data are to be considered as trees to represent the structural information. It is to be noticed that they can also be seen as vectors, having one attribute per leaf, i.e. vectors of the field values. Yet the information provided by the structure must be taken into account as it makes it possible to enrich the data point comparison.

4.1 Tree kernels

Many kernels have been proposed to compare data for which a tree structure is known, taking into account different objectives. Collins and Duffy [1] consider parse trees, and propose a similarity based on the common subtrees in both data points: the kernel is equivalent to the scalar product of vectors representing the data, defined as having one component for each possible subtree; the associated attribute stores the number of occurrences of the subtree in each data. This kernel has been extended by Kashima and Koyanagi [3] to any ordered tree and for a loosened definition of a subtree occurrence, allowing label mutations.

Vert [8] considers a different case, in the context of phylogenetic trees that come from biological data and represent phylogenetic relationships between species: leaves correspond to current organisms, internal nodes to their ancestors in the evolution. The considered objects are phylogenetic protein profiles, represented by vectors having as many components as the tree has leaves, and for which each attribute indicates whether the



Examples of depths of deepest common node (tree leaves, corresponding to fields, are numbered from left to right) :

$p(1, 8) = 0$

$p(1, 2) = 2$

Figure 2: Left: example of a considered data point: XML structure and field values; right: examples of depths of deepest common node for two field couples.

corresponding organism expresses the protein or not. The comparison between the proteins must exploit the available structure, i.e. the common history of the organisms. Thus the kernel does not aim at comparing two trees one with another: a single structure is available, whose internal node content is unknown and which is exploited to compare the vectors of the leaf contents. It is based on a probabilistic reconstruction of the internal node content which determines the probability of the simultaneous observation of two protein profiles.

4.2 Considered data

The data we consider, illustrated on Figure 2, correspond to trees whose leaves contain the student results and whose internal nodes represent the XML structure indicating the relationships between the exams. They are specific insofar as the structure is identical for all data points: structure is not to be seen as a discriminative feature, as is for instance the case in [1, 3]; it only corresponds to an additional source of information that can enrich the comparison of the leaf content vectors.

Thus it is more similar to the case considered by Vert [8]: structure provides information about the attributes. As opposed to the latter, in our case, internal nodes are not associated to unknown information which must be recovered, consequently the semantics associated to the structure is different. Therefore the exploitation of the XML structure must also take a different form: one can simply consider it as defining an ontology on the attributes describing the students, providing information about their relationships. Still, this information is interesting and must be exploited in the data similarity measure.

4.3 Proposed kernel

Kernel definition The XML structure indicates that attributes are not independent one from another, which suggests to enrich the attribute by attribute euclidian metric so as to take into account their relationships: two attributes belonging to the same branch convey similar meaning and their values should also be compared. For instance, students having similar results for two theoretical subjects could have a higher similarity value than that provided by a euclidian comparison which considers each attribute separately.

Therefore we propose a kernel based on attribute cross comparisons, that are weighted according to their proximity indicated in the XML structure: considering two data points x and y, it is defined as

$$k(x, y) = \sum_{ij=1}^{d} \lambda_{ij} x_i y_j$$

where d denotes the number of fields, x_i the value of field i for point x and $(\lambda_{ij})_{ij=1..d}$ the weighting coefficients; if $\lambda_{ij} = \delta_{ij}$, the kernel equals the euclidian scalar product. We propose to define

$$\lambda_{ij} = \delta_{ij} + \frac{l}{P - p(i, j)}$$

which corresponds to an enrichment as compared to the euclidian scalar product: it increases the weight associated to the cross comparison of two attributes if they are similar according to the XML structure. In this equation, l denotes a user-defined parameter which rules the influence given to the cross combinations, P is the tree depth ($P = 3$ for our example) and $p(i, j)$ the depth of the deepest common node of fields i and j: attributes belonging to the same branch play a more important role in the similarity than attributes having little in common. For instance (see right part of Figure 2) for a theoretical subject and an internship, the deepest common node is the tree root, whose depth is 0, and thus they are associated with a low coefficient, $\lambda = l/P = l/3$; on the contrary, the depth of the deepest common node for two different theoretical subjects is 2, and thus they get a higher influence, $\lambda = l/(P - 2) = l$.

Kernel properties It is to be noticed that the kernel matrix can be computed as $K = X^T \Lambda X$ where X is the matrix representing the student results as vectors of their field values, and Λ the weighting coefficient matrix. Thus, provided Λ is semi-definite positive, which depends on the considered structure and applies in the considered case, k is indeed a kernel.

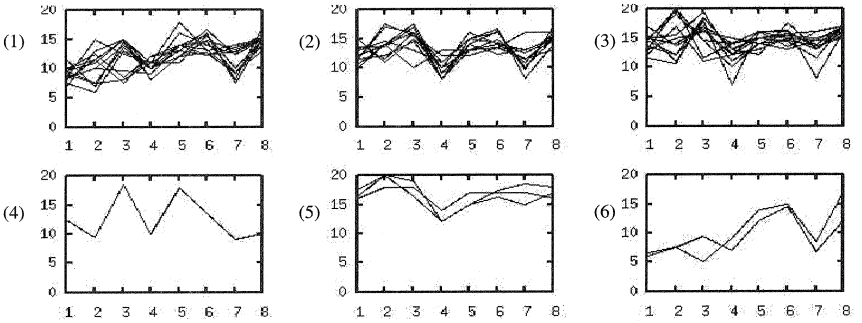


Figure 3: Clustering results obtained with a real dataset representing student results in an XML form. Data are represented as sequences of their values for the different fields.

This formulation highlights the fact that the proposed kernel is associated to a linear transformation of the data, the transformation being derived from the available structure information. The kernel advantage as compared to a vectorial representation is that the additional information is much easier to encode in the similarity measure than in the transformation function: it is simpler to define the role a cross comparison between attributes should play, and the weight it should have (through the λ coefficients), than to define the new relevant attributes. Indeed a property of kernels is that one does not need to determine explicitely the most relevant data representation but only the way data points should be compared [9].

Moreover, the kernel framework implies that it is not necessary to recompute the update equations for the cluster centres or to modify the algorithm, even if the considered scalar product is changed (as compared to the euclidian scalar product or to the kernel used with the artificial dataset in section 3.4).

4.4 Experimental results

We applied the previous methodology to a real dataset describing 42 students according to the XML structure represented on Figure 2, corresponding to 8 fields, with $l = 0.9$. The algorithm with $\alpha = 2.5$ built 6 clusters represented on Figure 3, as sequences of their field values (although this representation may lead a visual bias as it suggests an attribute by attribute interpretation).