Pattern Recognition 45 (2012) 1436–1454

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Cohort-based kernel visualisation with scatter matrices

Enrique Romero [a,*], Tingting Mu [b], Paulo J.G. Lisboa [c]

[a] Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Spain
[b] School of Computing, Informatics and Media, University of Bradford, UK
[c] School of Computing and Mathematical Sciences, Liverpool John Moores University, UK

ARTICLE INFO

Article history:
Received 11 January 2010
Received in revised form
15 September 2011
Accepted 21 September 2011
Available online 19 October 2011

Keywords:
Visualisation
Discriminant analysis
Kernel method

ABSTRACT

Visualisation with good discrimination between data cohorts is important for exploratory data analysis and for decision support interfaces. This paper proposes a kernel extension of the cluster-based linear visualisation method described in Lisboa et al. [15]. A representation of the data in dual form permits the application of the kernel trick, so projecting the data onto the orthonormalised cohort means in the feature space. The only parameters of the method are those for the kernel function. The method is shown to obtain well-discriminating visualisations of non-linearly separable data with low computational cost. The linearity of the visualisation was tested using nearest neighbour and linear discriminant classifiers, achieving significant improvements in classification accuracy with respect to the original features, especially for high-dimensional data, where 93% accuracy was obtained for the Splice-junction Gene Sequences data set from the UCI repository.

1. Introduction

Low-dimensional visualisation methods generally fall into three categories. Purely linear methods frequently utilise singular values spanning the largest variance in the data or preserving pairwise inner products, such as the widely used Principal Component Analysis-based bi-plots [8] or classical Multi-Dimensional Scaling [25]. A second approach is to relax the linearity restriction and to define a non-linear projection to optimise the correspondence between distances in the original input space and distances in the projected space, such as in generalised (metric and non-metric) Multi-Dimensional Scaling [4] (including for example Sammon's mapping [20] and Kruskal's approach [11]) or Isomap [24]. A third approach generates latent variable models over the data space to induce a visualisation based on the density distribution of the data. This can be achieved directly with smoothing maps such as Gaussian Processes [12,13] and with topographic methods including Self-Organizing Maps [9] and Generative Topographic Mapping [3]. Alternatively, graph-partitioning methods can be used to avoid the curse of dimensionality by resorting to domain specific similarity relationships, followed by the application of clustering methods to rank the data by similarity [23].

Most visualisation methods construct the mapping to low-dimensional spaces without using the cluster or class labels, and then represent differently objects from different cohorts for comparison purposes [14]. Although this property makes the method generically applicable, it does not always use all of the information that is available, for instance clustering labels or prior knowledge about class membership. This is particularly important if the objective is to visualise the separation between cohorts. The most common supervised visualisation methods are based on Linear Discriminant Analysis (LDA) (see, for example, [5]) and its non-linear (kernel) version, Kernel Linear Discriminant Analysis (KLDA) [21], although other supervised non-linear visualisation approaches can be found (see [7,10,17] or [16], for example). While KLDA searches for optimal directions in feature space for which separation between classes is maximal, this method does not directly attempt to perform dimensionality reduction, which is important in data visualisation. It is possible to combine KLDA with feature space selection by optimising the kernel parameters directly [27] or, alternatively, to map the data onto a high-dimensional kernel space and then optimise a scatter-matrix separation index similar to that used in this paper, based on the Kernel PCA transformation [26]. Some relationships between KLDA, Kernel PCA (KPCA) [22] and LDA can be found in Yang et al. [28]. These methods have been further extended to kernel quadratic discriminant analysis [19].

This paper takes advantage of a recent result derived for linear visualisation of labelled data cohorts, typically defined by cluster membership or class labels, in the context of scatter-matrix separation measures. In the Cluster-based Visualisation with Scatter Matrices (CVSM) method, described in Lisboa et al. [15], it was shown that the space spanned by the cohort means forms a useful basis for dimensionality reduction while retaining much of the cohort separation measured by a quadratic index.

* Corresponding author. Tel.: +34 934137796.
E-mail address: eromero@lsi.upc.edu (E. Romero).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.09.025

In particular, when the covariance matrix of the original data is non-singular, it was proven that this data compression, preceded by a sphering of the data, exactly preserves the value of the separation index, thus preserving the cohort separation at the level of second-order statistics. This is a surprising amount of separation for what can be a drastic reduction in dimensionality. For non-linearly separable data cohorts, the natural extension is to use kernel methods, raising the possibility that a similarly efficient compression can be obtained. The application of kernels to CVSM may lead to insights about the data, by generating linearly separable views of non-linearly separable data.

The main contribution of this paper is to develop a kernel extension of CVSM [15], and to demonstrate its value for the visualisation of non-linearly separable data: class or cluster labels are used to show how kernels can linearise the visualisation of non-linearly separable data. Although the kernel trick cannot be directly applied, this drawback can be avoided by representing the data in dual form. With this representation, sphering in the feature space, orthonormalisation in the feature space and the projections of the data onto the sub-space spanned by the (orthonormalised) class means in the feature space can be easily and uniformly computed. This is the second contribution of the paper. This method is different from KLDA because it usually requires only the inversion of an $N_c \times N_c$ matrix, where N_c is the number of classes, rather than the solution of a convex quadratic optimisation problem [21] or an eigenvalue problem of size N (N is the number of examples) [2] in KLDA. In principle, the proposed method has an advantage over the non-supervised methods PCA and KPCA, since it explicitly makes use of the cohort labels to maximise the cohort separation in the low-dimensional projective space, using the non-linear properties of kernel features to resolve non-linearly separable data. KPCA must also solve an eigenvalue problem of size N. The only parameters of the whole method are the parameters of the kernel function.

The goal of this paper is data visualisation, in particular to visualise the separation between cohorts. These data cohorts can be either different clusters (after performing a cluster analysis, for example) or classes (if this information is available). Correspondingly, the proposed algorithm is aimed at either preserving the multimodal distribution of the data or the separability between classes, by incorporating the cluster or class labels, respectively. The result is a visualisation of the kernel-defined feature space, which can facilitate to find the most appropriate kernels for visualisation and, potentially, for kernel-based classification of a given data set. If we accept that the role of the kernel is to project the data onto a space where the projections are linearly separable, then it follows that measuring the extent of linear separation with the proposed method helps to short-list the most useful kernels for a given classification task. Since the proposed method can induce a linearly separable visualisation for data with non-linear decision boundaries between population cohorts, it provides a direct visualisation of complex data sets in a feature space that is relevant to their categorisation into labelled groups, be they clusters or classes. The linearity of the visualisation was tested using nearest neighbour and linear discriminant classifiers, achieving significant improvements in classification accuracy with respect to the original features, specially for high-dimensional data (93% compared with 87% in the Gene data set from the UCI repository), and with similar performance to KLDA.

2. Cluster-based linear visualisation with scatter matrices

The method proposed in Lisboa et al. [15] is linear in nature and it is based on the decomposition of the invariant scatter matrix after projecting the data onto the subspace spanned by the class means. In the interest of making the paper as self-contained as possible, this section will summarise the main mathematical equations required to reproduce our results.

To fix notation, suppose we are given a data matrix $\mathbf{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ comprising N rows with d-dimensional data points of overall mean $\boldsymbol{m}$, where the data are partitioned into N_c groups, each group $j \in \{1, \ldots, N_c\}$ with N_j points and mean $\boldsymbol{m}_j$. It is well-known that the overall variance of the data, $\mathbf{S}_T$, can be decomposed into the sum of scatter matrices calculated within and between labelled cohorts [5,6] thus generating a within-cluster matrix, $\mathbf{S}_W$, and a between-cluster matrix, $\mathbf{S}_B$, such that $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$. This decomposition generates a natural scalar index for the separation between the data cohorts by taking the trace of the scatter matrix $\mathbf{M} = \mathbf{S}_W^{-1}\mathbf{S}_B$ leading to the class separation index $\mathrm{J} = \mathrm{tr}(\mathbf{M})$. Note that $\mathbf{S}_T$, $\mathbf{S}_W$, and $\mathbf{S}_B$ are $d \times d$ matrices.

A strength of the index J is its invariance to affine transformations of the data matrix, which makes it insensitive to co-linearities in the data and to changes in relative scaling of the covariates, both of which are useful properties for exploratory analysis of high dimensional data as in bioinformatics. Furthermore, if the covariance matrix is non-singular, then this invariance can be exploited by applying a Mahalanobis rotation to de-correlate the covariates, or sphering the data, thus rendering $\mathbf{S}_T$ is diagonal. The scatter matrix decomposition now indicates that the information contained in the within-cluster scatter matrix is implicit in the between-cluster matrix, which uses only the values of the class means $\{\boldsymbol{m}_j\}_{j=1}^{N_c}$ as representatives for the classes.

This suggests that the class means form a natural basis to project the data with minimal loss in class separation as measured by second-order statistics, as shown in Lisboa et al. [15]. However, in cases where the classes are non-linearly separable, then the scatter-matrix based separation index is not a reliable measure of class separation since this is no longer well represented by the second-order statistics of the data. A better low-dimensional projection may then be obtained by resorting to non-linear features, for instance through the use of kernels. To see how this can be done we briefly review the linear projective method described in Lisboa et al. [15].

The compression onto the subspace of the class means is readily achieved by defining an orthonormal set of basis vectors $\mathbf{B}^T = \{\boldsymbol{b}_j\}_{j=1}^{N_c}$, for instance by Gram–Schmidt orthogonalisation, generating the projection of $\mathbf{X}$ onto the space spanned by the set of orthonormalised cluster mean vectors $\mathbf{X}^c = \mathbf{X} \cdot \mathbf{B}$. Note that $\mathbf{X}^c$ and $\mathbf{B}$ are $N \times N_c$ and $d \times N_c$ matrices, respectively. Scatter matrices for $\mathbf{X}^c = \{\boldsymbol{x}_i^c\}_{i=1}^{N}$ can be calculated in the space of class means, namely: $\mathbf{S}_W^c = \sum_{j=1}^{N_c} \sum_{i=1}^{N_j} \{(\boldsymbol{x}_i^c - \boldsymbol{m}_j^c)^T (\boldsymbol{x}_i^c - \boldsymbol{m}_j^c)\}$, $\mathbf{S}_B^c = \sum_{j=1}^{N_c} N_j \{(\boldsymbol{m}_j^c - \boldsymbol{m}^c)^T (\boldsymbol{m}_j^c - \boldsymbol{m}^c)\}$ and, similarly, an invariant scatter matrix $\mathbf{M}^c = (\mathbf{S}_W^c)^{-1}\mathbf{S}_B^c$ and an invariant class separation index $\mathrm{J}^c = \mathrm{tr}(\mathbf{M}^c)$ can be defined. Note that $\mathbf{S}_W^c$ and $\mathbf{S}_B^c$ are $N_c \times N_c$ matrices, so that the computation of $(\mathbf{S}_W^c)^{-1}$ is computationally fast.

In Lisboa et al. [15] it is shown that the invariant separation measure J is exactly preserved (i.e., $\mathrm{J} = \mathrm{J}^c$) when the projection is preceded by a sphering of the data. The paper also shows that when the covariance matrix is singular, then some loss is induced by this dimensionality reduction, but most of the class separation is maintained. A diagonalisation of the new scatter matrix $\mathbf{M}^c$ shows, typically, that the trace of the matrix is contained in the largest few eigenvalues. Their correspondent eigenvectors form the basis for a 2D or 3D visualisation of the data. The whole visualisation procedure can be summarised in Algorithm 1. Note that the method is parameter-free.

As previously said, a natural extension of this approach is to investigate the use of kernel transformations to further separate the clusters, or class-labelled cohorts, in the low-dimensional projective space. The next section describes how this can be done.

Algorithm 1. Linear visualisation algorithm proposed in Lisboa et al. [15].

Given a labelled data set $\mathbf{X}=\{\boldsymbol{x}_i\}_{i=1}^N$,

1. Optionally, sphere the data: $\mathbf{X}=\mathbf{X}\cdot\boldsymbol{\Sigma}^{-1/2}$ with the covariance matrix $\boldsymbol{\Sigma}$
2. Compute and orthonormalise the class means: $\mathbf{B}^T$
3. Project data onto the orthonormalised class means: $\mathbf{X}^c=\mathbf{X}\cdot\mathbf{B}$
4. Compute scatter matrices for $\mathbf{X}^c$: $\mathbf{S}_W^c$, $\mathbf{S}_B^c$ and $\mathbf{M}^c$
5. Project $\mathbf{X}^c$ onto the eigenvectors of $\mathbf{M}^c$ with the largest eigenvalues

3. Cohort-based kernel visualisation with scatter matrices

A deeper analysis of the method described in Algorithm 1 reveals that, in order to construct $\mathbf{X}^c$, only inner products are needed. This allows us to develop a non-linear extension of the visualisation method in Section 2 by employing the kernel trick, leading to what we term Cohort-based Kernel Visualisation with Scatter Matrices (CKVSM). The core idea is to map the data into a kernel-based feature space $\mathcal{F}$, where the method could be applied. This kernel-based procedure has been widely used to define non-linear versions of classical linear procedures, such as KPCA (for PCA) or KLDA (for LDA) [21].

Based on the Mercer's theorem [18], a (positive definite) kernel function $K(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^d\mapsto\mathbb{R}$ defines a dot product $\langle\cdot,\cdot\rangle:\mathcal{F}\times\mathcal{F}\mapsto\mathbb{R}$ in a (maybe infinite-dimensional) feature space $\mathcal{F}$. Let $\phi:\mathbb{R}^d\mapsto\mathcal{F}$ denote the nonlinear mapping that transforms the original d-dimensional data to this infinite kernel-based feature space $\mathcal{F}$, and $\mathbf{K}=[k_{ij}]$ denote the $N\times N$ kernel matrix between N data points. The ijth entry of $\mathbf{K}$ is $k_{ij}=K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\langle\boldsymbol{\phi}(\boldsymbol{x}_i),\boldsymbol{\phi}(\boldsymbol{x}_j)\rangle$. Using the kernel function K, the mapping ϕ becomes implicit.

As explained in Section 2, two levels of projection are pursued in the cluster-based visualisation: (1) projecting the sphered data points into the N_c-dimensional space spanned by the orthonormalised class means; (2) further projecting the obtained N_c-dimensional data points into an n-dimensional space spanned by the eigenvectors of the scatter matrix $\mathbf{M}_c$. Working in the infinite feature space $\mathcal{F}$, we are looking for an $\infty\times k$ transformation matrix to project the infinite kernel-based features to a k-dimensional space. In this kernelised version, both the sphering procedure and the construction of the projection onto the space spanned by the orthonormalised cohort means are conducted in the kernel-based feature space, as explained in the next sections.

3.1. Cohort-based kernel visualisation with scatter matrices and dual-form representation

The kernel trick cannot be directly applied in the method described in Algorithm 1, because $\mathbf{B}^T$ would represent points in the feature space, that may be unknown. This affects to steps 2 and 3 of Algorithm 1. However, this drawback can be avoided by representing the data in dual form, which is one of the key points of the proposed method. Let us define the dual space $\mathcal{D}=\{\sum_{i=1}^N a_i\boldsymbol{\phi}(\boldsymbol{x}_i)|a_1,a_2,\ldots,a_N\in\mathbb{R}^N\}\subseteq\mathcal{F}$ (recall that $\{\boldsymbol{x}_i\}_{i=1}^N$ is the original data). Let $\boldsymbol{a}=(a_1,a_2,\ldots,a_N)\in\mathbb{R}^N$ represent the point $\hat{\boldsymbol{a}}=\sum_{i=1}^N a_i\boldsymbol{\phi}(\boldsymbol{x}_i)\in\mathcal{D}$ in dual form. With this representation:

1. Vector space operations in $\mathbb{R}^N$ have a direct correspondence in $\mathcal{D}$.
2. The mean of class C_j in the feature space is represented as $\boldsymbol{m}_j=(m_{j1},m_{j2},\ldots,m_{jN})$, where
$$m_{ji}=\begin{cases}1/N_j & \text{if } \boldsymbol{x}_i \text{ belongs to } C_j,\\ 0 & \text{otherwise.}\end{cases}\tag{1}$$
3. Inner products in the feature space between two vectors $\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}\in\mathcal{D}$ without sphering can be computed as usual:
$$\langle\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}\rangle=\sum_{i,j=1}^N a_i b_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{a}^T\mathbf{K}\boldsymbol{b},\tag{2}$$
where $\mathbf{K}$ is the kernel matrix. Inner products between two vectors $\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}\in\mathcal{D}$ after sphering can be computed as
$$\langle\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}\rangle=\boldsymbol{a}^T\mathbf{K}_s\boldsymbol{b},\tag{3}$$
where $\mathbf{K}_s$ is defined in Eq. (9) (see Section 3.2). Let
$$\boldsymbol{\Upsilon}=\begin{cases}\mathbf{K} & \text{if no sphering in the feature space is performed,}\\ \mathbf{K}_s & \text{otherwise.}\end{cases}\tag{4}$$
4. The Gram–Schmidt orthonormalisation procedure can be applied as usual, since only inner products and vector spaces operations are needed. The set of orthonormalised class means $\hat{\mathbf{B}}$ can also be represented in dual form by a $N\times N_c$ matrix (see Section 3.3). Abusing of notation, we will also denote $\hat{\mathbf{B}}$ to this matrix.
5. Using Eq. (4), the projection of the data onto the orthonormalised cohort means in the feature space can be obtained as: $\mathbf{X}^c=\boldsymbol{\Upsilon}\hat{\mathbf{B}}$ (recall that the projection of $\phi(\boldsymbol{x}_j)$ can be computed as $\boldsymbol{u}_j^T\boldsymbol{\Upsilon}\hat{\mathbf{B}}$, where $\boldsymbol{u}_j^T$ is the N-dimensional vector with a 1 in position j and 0 elsewhere).
6. Once $\mathbf{X}^c$ has been obtained, steps 4 and 5 of Algorithm 1 can be performed.

The computation of the sphered inner product matrix $\mathbf{K}_s$ and the Gram–Schmidt orthonormalisation procedure in dual form are described in Sections 3.2 and 3.3, respectively.

3.2. Sphering in the feature space

In the input space, given an $N\times d$ matrix of centered data points $\mathbf{X}=\{\boldsymbol{x}_i\}_{i=1}^N$, the covariance matrix can be defined as $\boldsymbol{\Sigma}=(1/N)\mathbf{X}^T\mathbf{X}$. Since $\boldsymbol{\Sigma}$ is symmetric, it can be decomposed as $\boldsymbol{\Sigma}=\mathbf{V}_\Sigma\mathbf{D}_\Sigma\mathbf{V}_\Sigma^T$, where $\mathbf{D}_\Sigma$ is a diagonal matrix with the non-zero eigenvalues of $\boldsymbol{\Sigma}$, and $\mathbf{V}_\Sigma$ is an orthonormal matrix whose columns are the corresponding eigenvectors of $\boldsymbol{\Sigma}$. The sphering of the data consists of a rotation of the data by applying a linear transformation $\mathbf{Y}=\mathbf{X}\mathbf{R}_\Sigma$, where $\mathbf{R}_\Sigma=\boldsymbol{\Sigma}^{-1/2}=\mathbf{V}_\Sigma\mathbf{D}_\Sigma^{-1/2}\mathbf{V}_\Sigma^T$, so that the covariance matrix of $\mathbf{Y}$ is the identity matrix. The inner product matrix between the transformed features is $\mathbf{Y}\mathbf{Y}^T=\mathbf{X}\mathbf{V}_\Sigma\mathbf{D}_\Sigma^{-1}\mathbf{V}_\Sigma^T\mathbf{X}^T=\mathbf{X}\boldsymbol{\Sigma}^{-1}\mathbf{X}^T$.

Suppose now that we have a set of centered data points $\boldsymbol{\Phi}=\{\phi(\boldsymbol{x}_i)\}_{i=1}^N$ in the feature space induced by the kernel function K. We can represent $\boldsymbol{\Phi}$ as an $N\times\infty$ matrix. For sphering the data in the feature space the kernel trick cannot be directly applied simply changing $\mathbf{X}$ by $\boldsymbol{\Phi}$, because $\mathbf{D}_\Sigma$ and $\mathbf{V}_\Sigma$ represent matrices in the feature space, that may be unknown. Even if they were known, they have infinite dimension (recall that $\mathbf{D}_\Sigma$ and $\mathbf{V}_\Sigma$ have d columns, where d is the dimension of the space). However, as we have seen (see Section 3.1), to work with sphered data in the feature space we only need to know the inner product of any two sphered data points, instead of the representation of the sphered data itself. The rest of the section explains how to compute the inner product matrix of sphered data in the feature space, by studying the relationship between the eigenvectors of the covariance matrix in the feature space and the eigenvectors of the kernel matrix.

3.2.1. Eigendecomposition of the covariance matrix in the feature space

The inner product matrix of $\mathbf{\Phi}$ can be computed as $\mathbf{K}=\mathbf{\Phi}\mathbf{\Phi}^T$, where $\mathbf{K}$ is the $N\times N$ kernel matrix. Since $\mathbf{K}$ is symmetric, it can be decomposed as $\mathbf{K}=\mathbf{V}_K\mathbf{D}_K\mathbf{V}_K^T$ or, equivalently

$$\mathbf{K}\mathbf{V}_K=\mathbf{V}_K\mathbf{D}_K, \tag{5}$$

where $\mathbf{D}_K$ is a diagonal matrix with the non-zero eigenvalues of $\mathbf{K}$, and $\mathbf{V}_K$ is an orthonormal matrix whose columns are the corresponding eigenvectors of $\mathbf{K}$. Note that $\mathbf{V}_K$ may not be a square matrix, since only contains the set of eigenvectors of $\mathbf{K}$ that correspond to non-zero eigenvalues. The covariance matrix in the feature space $\mathbf{\Sigma}'$ can be defined as usual $\mathbf{\Sigma}'=(1/N)\mathbf{\Phi}^T\mathbf{\Phi}$, and can also be decomposed as

$$\mathbf{\Sigma}'\mathbf{V}_{\Sigma'}=\mathbf{V}_{\Sigma'}\mathbf{D}_{\Sigma'}, \tag{6}$$

where $\mathbf{D}_{\Sigma'}$ is a diagonal matrix with the non-zero eigenvalues of $\mathbf{\Sigma}'$, and $\mathbf{V}_{\Sigma'}$ is an orthonormal matrix whose columns are the corresponding eigenvectors of $\mathbf{\Sigma}'$.

We will express the eigenvectors of $\mathbf{\Sigma}'$ as a function of $\mathbf{V}_K$ to compute the inner product of two sphered points in the feature space.

As shown in Schöolkopf et al. [22], the eigendecomposition of the covariance matrix in the kernel-based feature space can be solved by computing the eigendecomposition of the kernel matrix: $\mathbf{\Phi}\mathbf{V}_{\Sigma'}$ are eigenvectors of $\mathbf{K}$ with the corresponding eigenvalues $N\mathbf{D}_{\Sigma'}$ and, conversely, $\mathbf{\Phi}^T\mathbf{V}_K$ are eigenvectors of $\mathbf{\Sigma}'$ with corresponding eigenvalues $(1/N)\mathbf{D}_K$. As a consequence, $\mathbf{\Sigma}'$ and $\mathbf{K}$ have the same number of eigenvectors with non-zero eigenvalue. Let $n\le N$ be the number of eigenvectors of $\mathbf{K}$ with non-zero eigenvalue. Then, $\mathbf{V}_{\Sigma'}$ has n columns and $\mathbf{D}_K=N\mathbf{D}_{\Sigma'}$ (note that since $\mathbf{D}_K$ and $\mathbf{D}_{\Sigma'}$ only contain the non-zero eigenvalues, both are $n\times n$ matrices).

When the eigenvectors of the covariance matrix $\mathbf{\Sigma}'$ are computed from $\mathbf{V}_K$, the orthogonality condition, given as $\mathbf{V}_{\Sigma'}^T\mathbf{V}_{\Sigma'}=\mathbf{I}$, requires to be imposed. Let $\mathbf{S}$ be a $n\times n$ scaling diagonal matrix of $\mathbf{V}_{\Sigma'}$, such that $\mathbf{V}_{\Sigma'}=\mathbf{\Phi}^T\mathbf{V}_K\mathbf{S}$. Then we have

$$\mathbf{I}=\mathbf{V}_{\Sigma'}^T\mathbf{V}_{\Sigma'}=\mathbf{S}^T\mathbf{V}_K^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{V}_K\mathbf{S}=\mathbf{S}^T\mathbf{V}_K^T\mathbf{K}\mathbf{V}_K\mathbf{S}. \tag{7}$$

By incorporating Eq. (5) and $\mathbf{V}_K^T\mathbf{V}_K=\mathbf{I}$ into Eq. (7), we simply have $\mathbf{S}^T\mathbf{D}_K\mathbf{S}=\mathbf{I}$. Therefore, $\mathbf{S}=\mathbf{D}_K^{-1/2}$, and equivalently,

$$\mathbf{V}_{\Sigma'}=\mathbf{\Phi}^T\mathbf{V}_K\mathbf{D}_K^{-1/2}. \tag{8}$$

3.2.2. Sphered inner product matrix in the feature space

Let us return to the problem of computing the inner product of two sphered data points in the feature space. Similar to the input space, we can define the rotation matrix $\mathbf{R}_{\Sigma'}=\mathbf{V}_{\Sigma'}\mathbf{D}_{\Sigma'}^{-1/2}\mathbf{V}_{\Sigma'}^T$, and apply the linear transformation $\mathbf{\Phi}'=\mathbf{\Phi}\mathbf{R}_{\Sigma'}$. By incorporating Eq. (8), the new kernel matrix of inner products after sphering in the feature space can be computed as $\mathbf{K}_s=\mathbf{\Phi}'\mathbf{\Phi}'^T=\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{V}_K\mathbf{D}_K^{-1/2}\ \mathbf{D}_{\Sigma'}^{-1}\mathbf{D}_K^{-1/2}\ \mathbf{V}_K^T\mathbf{\Phi}\mathbf{\Phi}^T$. Since $\mathbf{D}_K^{-1/2}\mathbf{D}_{\Sigma'}^{-1}\mathbf{D}_K^{-1/2}=N\mathbf{D}_K^{-2}$ and using Eq. (5), this leads to a sphered inner product $N\times N$ matrix, finally computed as

$$\mathbf{K}_s=N\mathbf{V}_K\mathbf{V}_K^T \tag{9}$$

in the kernel-based feature space. If $n=N$, then $\mathbf{K}_s$ is N times the identity matrix. If $n<N$, then $\mathbf{K}_s$ is rank deficient and usually is not a diagonal matrix.

3.3. Kernelised Gram–Schmidt orthonormalisation

Algorithm 2. Gram–Schmidt orthonormalisation algorithm in the feature space.

Given $\{\boldsymbol{i}_j\}_{j=1}^{N_c}$,
$\boldsymbol{i}_1=\boldsymbol{i}_1/\sqrt{\langle\hat{\boldsymbol{m}}_1,\hat{\boldsymbol{m}}_1\rangle}$
for $j=2\ldots N_c$ **do**
$\boldsymbol{b}_j=\boldsymbol{m}_j-\sum_{i=1}^{j-1}\langle\hat{\boldsymbol{m}}_j,\hat{\boldsymbol{b}}_i\rangle\boldsymbol{b}_i$
$\boldsymbol{b}_j=\boldsymbol{b}_j/\sqrt{\langle\hat{\boldsymbol{b}}_j,\hat{\boldsymbol{b}}_j\rangle}$
end for

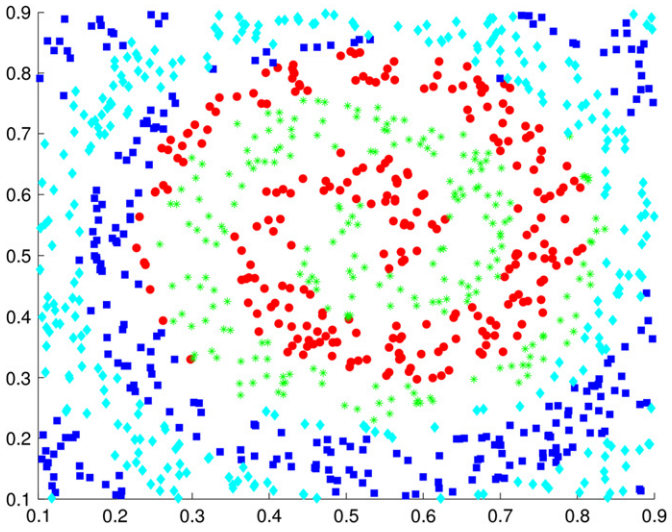

Fig. 1. Distribution of the four classes in the original artificial data set.

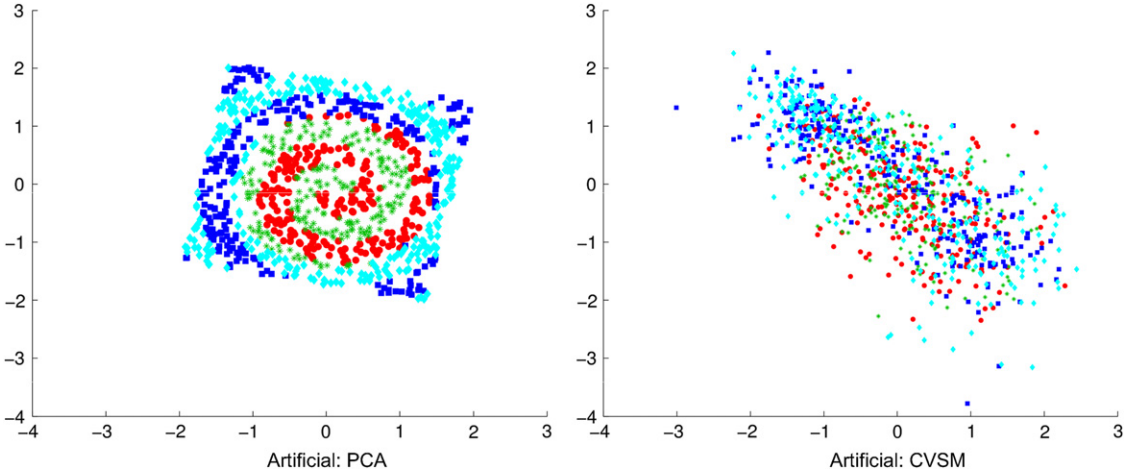

Fig. 2. 2D visualisations of the projections of PCA (left) and CVSM (right) for the four classes of the data set in Fig. 1.

Here, we describe the extension of the Gram–Schmidt orthonormalisation in the feature space of vectors represented in dual form. Recall that inner products $\langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}} \rangle$ in the feature space are computed with Eq. (2) or (3). Given a set $\{\boldsymbol{m}_j\}_{j=1}^{N_c}$ of vectors in $\mathbb{R}^N$ representing vectors $\{\hat{\boldsymbol{m}}_j\}_{j=1}^{N_c} \in \mathcal{D}$ in dual form, the set $\{\boldsymbol{b}_j\}_{j=1}^{N_c}$ of vectors in $\mathbb{R}^N$ defined in Algorithm 2 represent a set of orthonormal vectors $\{\hat{\boldsymbol{b}}_j\}_{j=1}^{N_c} \in \mathcal{D}$ in dual form that span the same subspace than $\{\hat{\boldsymbol{m}}_j\}_{j=1}^{N_c}$.

3.4. Pseudo code of the proposed algorithm for CKVSM

For the sake of simplicity, we have made the assumption that the data are centered. If this was not the case, the previously showed results are still valid changing $\mathbf{K}$ by $\overline{\mathbf{K}} = \mathbf{K} - \mathbf{K}\,\mathbf{1}_N - \mathbf{1}_N\,\mathbf{K} + \mathbf{1}_N\mathbf{K}\,\mathbf{1}_N$, where $\mathbf{1}_N$ is an $N \times N$ matrix such that $(\mathbf{1}_N)_{ij} = 1/N$ (see [22] for details).

Since $\boldsymbol{a}^T\overline{\mathbf{K}}\boldsymbol{b} = (\boldsymbol{a}^T - \boldsymbol{a}^T\mathbf{1}_N)\mathbf{K}(\boldsymbol{b} - \mathbf{1}_N\boldsymbol{b}) = \boldsymbol{a}^T(\mathbf{I} - \mathbf{1}_N)\mathbf{K}(\mathbf{I} - \mathbf{1}_N)\boldsymbol{b}$, using $\overline{\mathbf{K}}$ with the original data is equivalent to using $\mathbf{K}$ with centered data. Centering data in dual form can be done by subtracting $1/N$ from

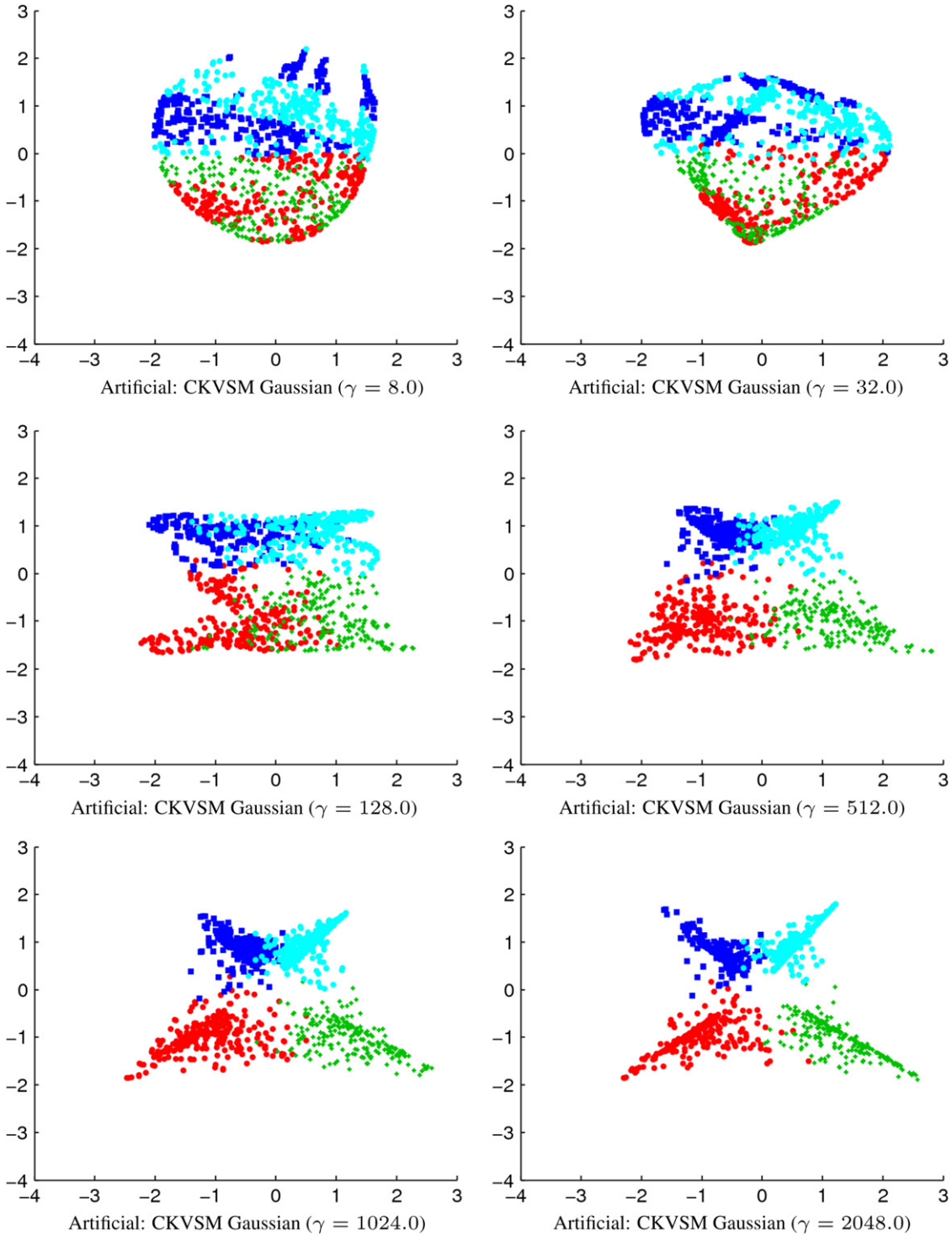

Fig. 3. 2D visualisations of the projections of CKVSM with Gaussian kernel for the data set in Fig. 1 without sphering in the feature space for increasing values of the γ parameter.

each component: a centered data point $\overline{\phi(\boldsymbol{x}_i)}$ is represented as $(\overline{a}_{i1},\overline{a}_{i2},\ldots,\overline{a}_{iN})$ where

$$\overline{a}_{ij}=\begin{cases}1-1/N & \text{if } i=j,\\ -1/N & \text{otherwise.}\end{cases}$$

This claim is also valid for Eq. (3).

Algorithm 3. Algorithm for CKVSM.

Given a labelled data set $\mathbf{X}=\{\boldsymbol{x}_i\}_{i=1}^{N}$,

0. Represent the mean of class C_j in the feature space $\boldsymbol{m}_j$ in dual form with Eq. (1)

1. If sphering, compute the final inner product matrix Υ with Eq. (3), otherwise Eq. (2)

2. Orthonormalise the class means in the feature space: $\hat{\mathbf{B}}^T$

3. Project data onto the orthonormalised cohort means in the feature space: $\mathbf{X}^c=\Upsilon\hat{\mathbf{B}}$

4. Compute scatter matrices for $\mathbf{X}^c$: $\mathbf{S}_W^c$, $\mathbf{S}_B^c$ and $\mathbf{M}^c$

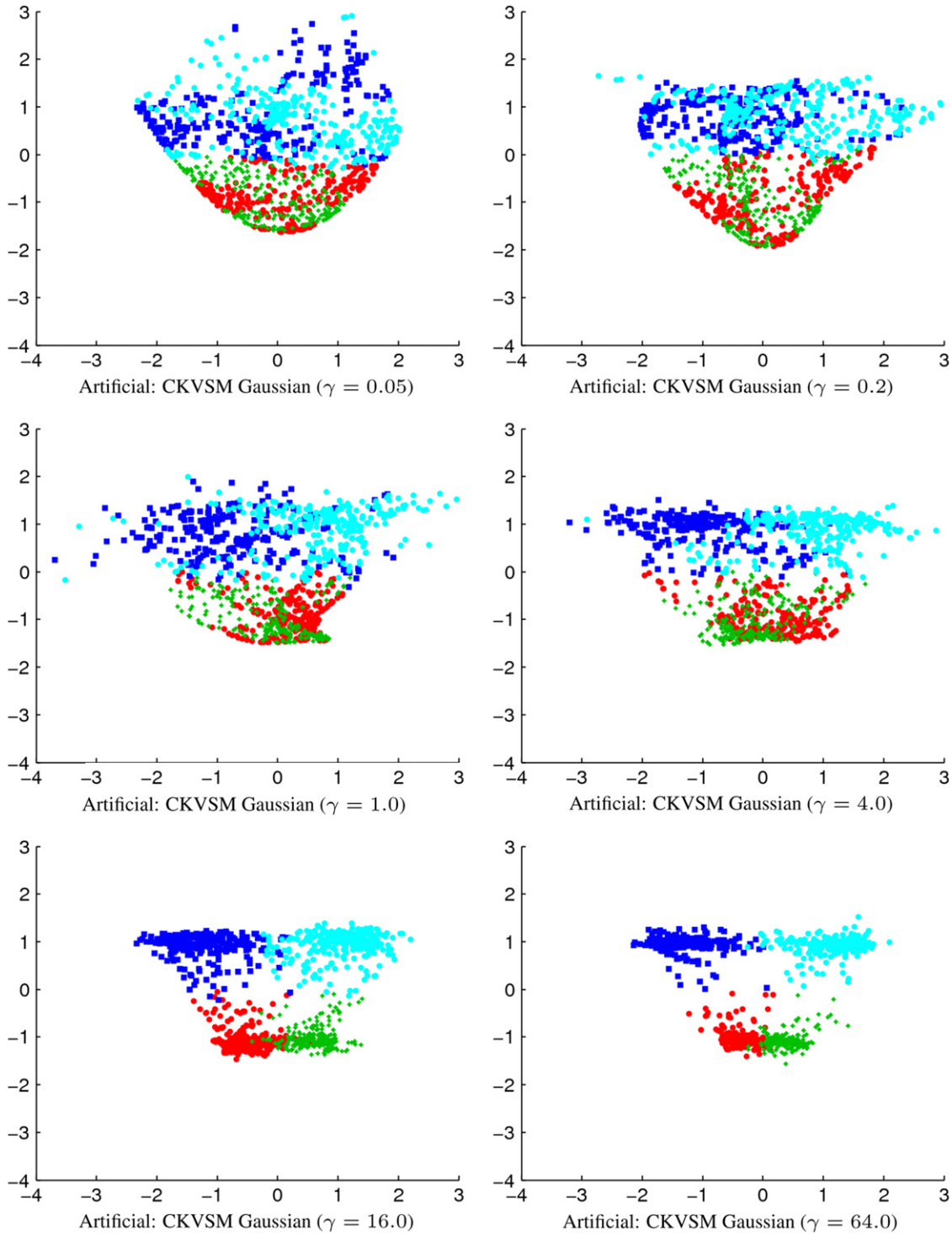

Fig. 4. 2D visualisations of the projections of CKVSM with Gaussian kernel for the data set in Fig. 1 after sphering in the feature space for increasing values of the γ parameter.

5. Project $\mathbf{X}^c$ onto the eigenvectors of $\mathbf{M}^c$ with the largest eigenvalues

The whole cohort-based kernel visualisation procedure is described in Algorithm 3. After computing the orthonormalised cohort means in the feature space $\hat{\mathbf{B}}$ (step 2), the data is then projected onto $\hat{\mathbf{B}}$ as $\mathbf{X}^c = \boldsymbol{\Upsilon}\hat{\mathbf{B}}$ (step 3), where $\boldsymbol{\Upsilon}$ is computed with Eq. (3) or (2) depending on whether the data is sphered or not. Once $\mathbf{X}^c$ has been obtained, steps 4 and 5 are the same than those of Algorithm 1. Regarding the dimensions of the matrices involved in the algorithm, recall that $\boldsymbol{\Upsilon}$ is an $N \times N$ matrix, $\hat{\mathbf{B}}$ and $\mathbf{X}^c$ are $N \times N_c$ matrices, and $\mathbf{S}_W^c$, $\mathbf{S}_B^c$ and $\mathbf{M}^c$ are $N_c \times N_c$ matrices. Note that the only parameters of the method are the parameters of the kernel.

The computational cost depends on whether sphering in the feature space is performed or not. When sphering is performed, the computational cost is dominated by the computation of $\mathbf{K}_s$ in Eq. (9), which involves solving an eigenvalue problem of size N.

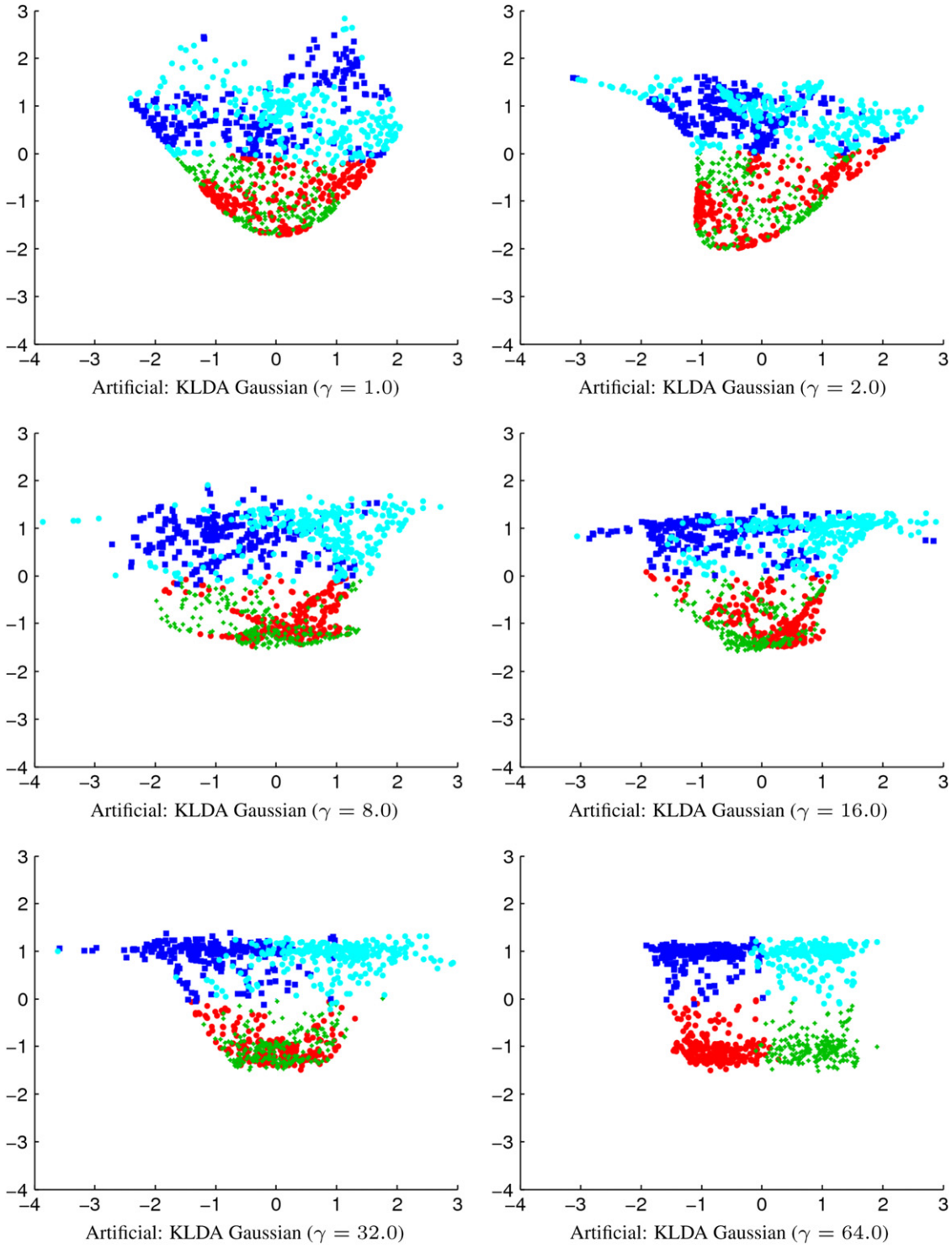

Fig. 5. 2D visualisations of the projections of KLDA with Gaussian kernel for the data set in Fig. 1 for increasing values of the γ parameter.

Otherwise, the computation is dominated by the orthonormalisation and projection steps, which are $O(N^2 \cdot N_c)$. Note that the eigenvalue problem involved in the computation of the eigenvectors of $\mathbf{M}^c$ has size N_c.

The projection of new data $\mathbf{Y} = \{\boldsymbol{y}_i\}_{i=1}^{M}$ (a test set, for example) onto the projection space defined by a data set $\mathbf{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ (a training set, for example) is described in Algorithm 4. In essence, after computing $\hat{\mathbf{B}}$ and $\mathbf{M}^c$ with algorithm 3, the new data is projected first onto $\hat{\mathbf{B}}$ and then onto the eigenvectors of $\mathbf{M}^c$ with the largest

Table 1
Description of the benchmark data sets.

Data set	#Variables	#Classes	#Examples
Gene	120	3	3175
Glass	9	6	214
Vehicle	18	4	846
Wine	13	3	178

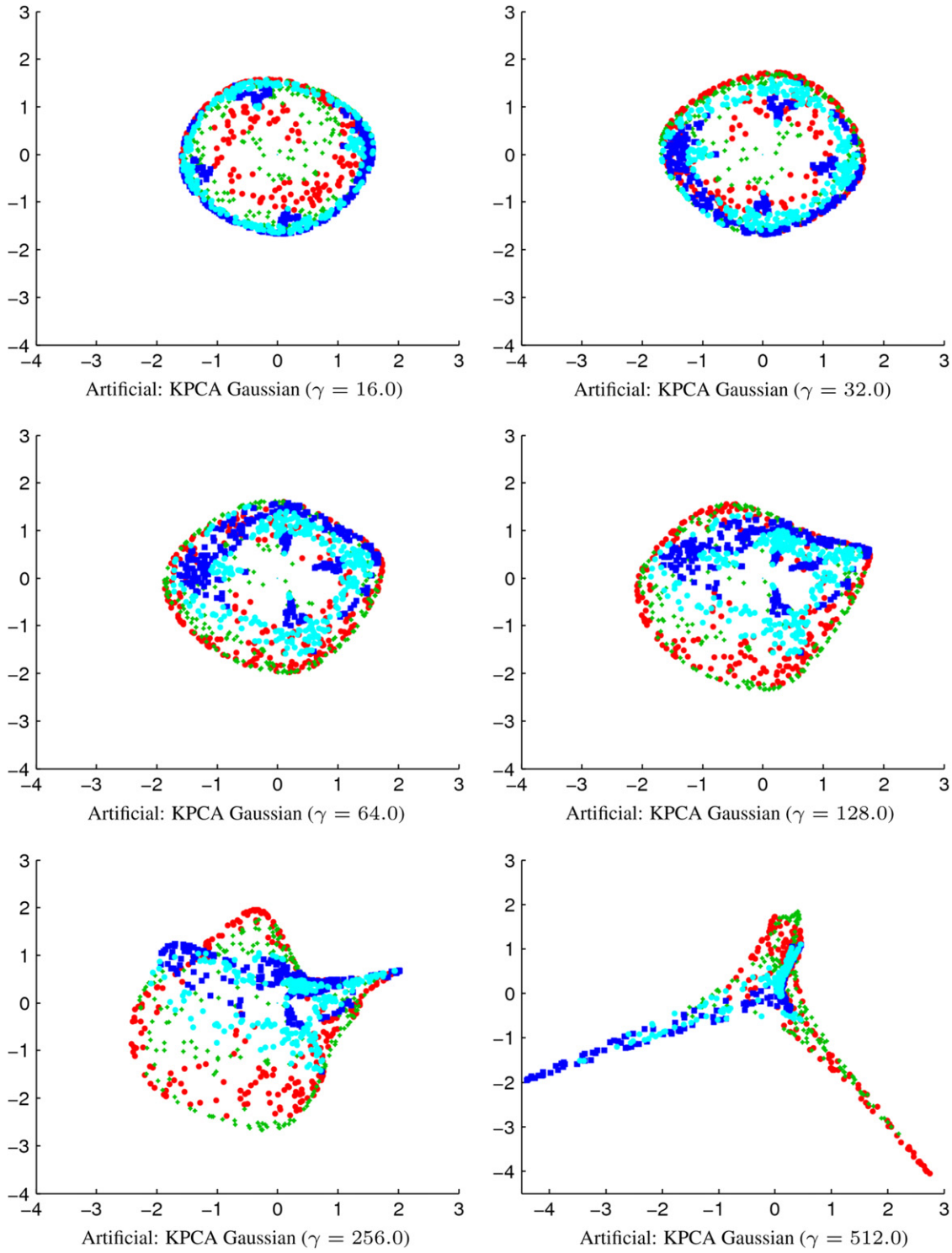

Fig. 6. 2D visualisations of the projections of KPCA with Gaussian kernel for the data set in Fig. 1 for increasing values of the γ parameter.

eigenvalues. If sphering, the whole data is used to compute the inner product matrix. Anyway, only the data set **X** is used to compute the projection space.

Algorithm 4. Projection of new data with CKVSM.

Given a labelled data set $\mathbf{X}=\{\boldsymbol{x}_i\}_{i=1}^{N}$ and a data set $\mathbf{Y}=\{\boldsymbol{y}_i\}_{i=1}^{M}$,
If sphering **then**

1. Define $\mathbf{K}=\begin{pmatrix}\mathbf{K}_{XX} & \mathbf{K}_{XY}\\ \mathbf{K}_{YX} & \mathbf{K}_{YY}\end{pmatrix}$, where $\mathbf{K}_{AB}$ is the kernel matrix for data sets **A** and **B**
2. Compute the final $(N+M)\times(N+M)$ inner product matrix Υ with Eq. (3)
3. Use **X** and Υ_{XX} to obtain $\hat{\mathbf{B}}$ and $\mathbf{M}^c$ with steps 0, 2, 3 and 4 of Algorithm 3
4. Compute $\mathbf{Y}^c=\Upsilon_{YX}\hat{\mathbf{B}}$

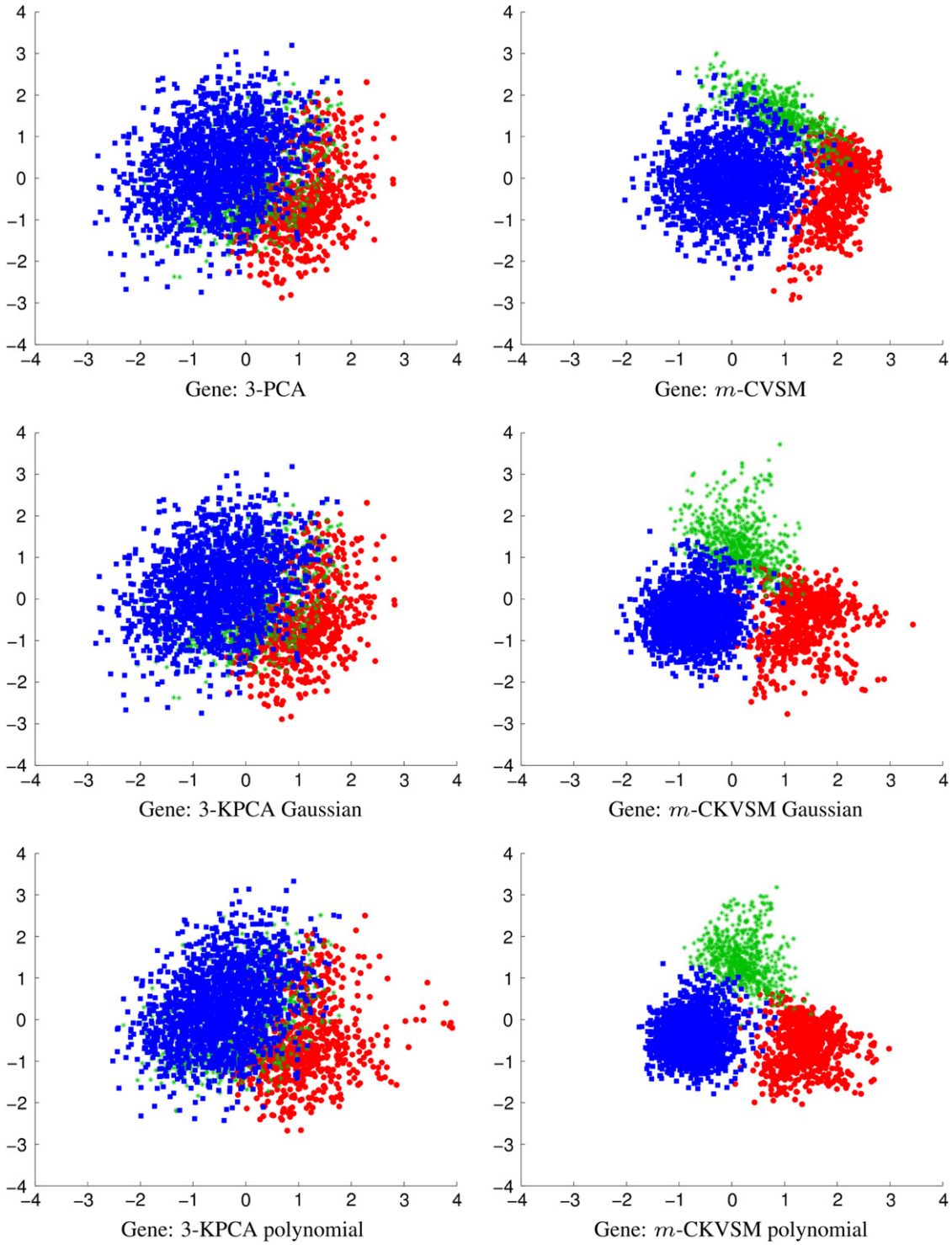

Fig. 7. 2D visualisations of the projections in the "Projection+1NN" models for the Gene data set. Top row: 3-PCA (left) and m-CVSM (right). Middle row: 3-KPCA Gaussian (left) and m-CKVSM Gaussian (right). Bottom row: 3-KPCA polynomial (left) and m-CKVSM polynomial (right).

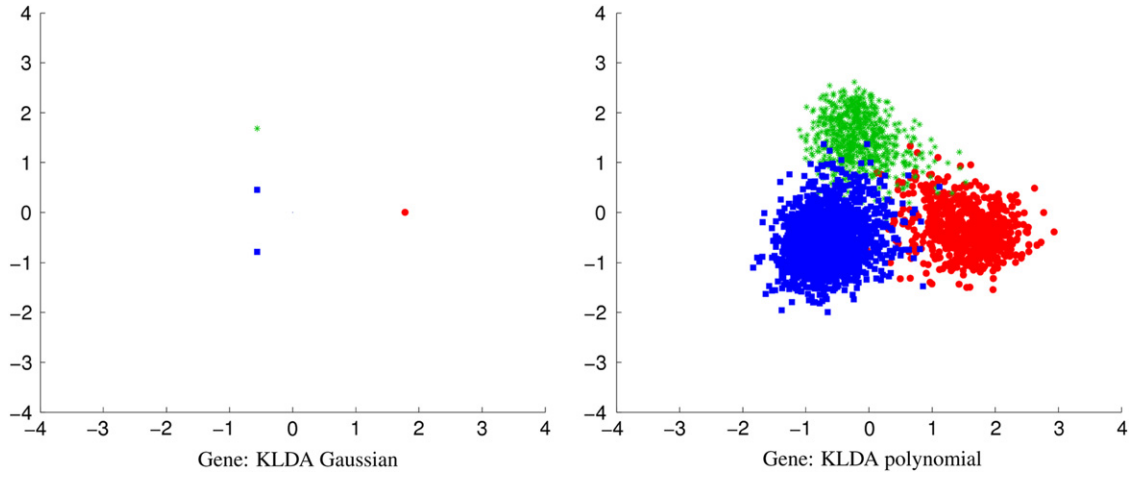

Fig. 8. 2D visualisations of the projections of KLDA for the Gene data set. Left: Gaussian kernel. Right: polynomial kernel.

else

1. Define $\mathbf{K}=\mathbf{K}_{XX}$
2. Compute the final $N\times N$ inner product matrix Υ with Eq. (2) (i.e., $\Upsilon=\mathbf{K}$)
3. Use $\mathbf{X}$ and Υ to obtain $\hat{\mathbf{B}}$ and $\mathbf{M}^c$ with steps 0, 2, 3 and 4 of Algorithm 3
4. Compute $\mathbf{Y}^c=\mathbf{K}_{YX}\hat{\mathbf{B}}$

end if

Project $\mathbf{Y}^c$ onto the eigenvectors of $\mathbf{M}^c$ with the largest eigenvalues

Note that the labels of the new points in $\mathbf{Y}$ may be unknown, since they are not used in the projection. This allows to validate the goodness of the projection not only by measuring the invariant index J but also by measuring the accuracy of a classifier on the projected data. In this case we can run any standard classifier in the projected space, as follows. Given a training set $\mathbf{X}=\{\boldsymbol{x}_i\}_{i=1}^N$ and a test set $\mathbf{Y}=\{\boldsymbol{y}_i\}_{i=1}^M$:

1. Construct $\hat{\mathbf{B}}$ and $\mathbf{M}^c$ with Algorithm 4.
2. Compute $\mathbf{X}^c$ and $\mathbf{Y}^c$, and project them onto the eigenvectors of the largest eigenvalues of $\mathbf{M}^c$.
3. Construct a standard classifier using the projection of $\mathbf{X}^c$ as training set.
4. Test the classifier with the projection of $\mathbf{Y}^c$.

We will include this scheme in the "Projection+Classifier" experiments (see Section 4).

4. Experiments

Several visualisations and performance measures, together with the computational cost, are benchmarked for the proposed method, and compared to other models, such as PCA, CVSM, KPCA or KLDA. In the visualisations, data was projected onto the two most important directions of the final models and the numbers in the axes are the reference values of the projected data.

4.1. Validation of the model on an artificial data set

An artificial data set, consisting of 1000 2D examples and four classes, was constructed to validate the proposed model. The input values were randomly generated in $[0.1,0.9]\times[0.1,0.9]$. The class value was assigned in a similar way to the well-known *Two Spirals* problem, as shown in Fig. 1.

As expected, linear projections do not show useful information for this data set, as shown in Fig. 2: both PCA and CVSM give visualisation without any structure. It is worth pointing out that if the spirals were not coplanar, then PCA would mix them up, as does CVSM. In contrast, the visualisations obtained by CKVSM allow to imagine more clearly the structure of the data and, for some values of the kernel parameter, separate the classes. This can be seen in Figs. 3 and 4, where the visualisations of CKVSM with Gaussian kernel are shown for increasing values of the γ parameter. Fig. 3 are the visualisations obtained without sphering in the feature space and Fig. 4 are those obtained after sphering in the feature space. Note that it is quite difficult to decide, only by visual inspection, whether it is better sphering or not.

Fig. 5 displays the equivalent visualisations for KLDA showing a similar behaviour to CKVSM. The visualisations results of KPCA indicate a certain structure of the data but do not allow to separate the classes (see Fig. 6).

4.2. Experiments on benchmark data sets

A quantitative evaluation of the visualisation plots is difficult to define. For this reason, we opted to measure the separability of the kernel mapped data. For labelled data, it may be measured by the invariant index J or by the classification accuracy of a classifier that takes as input the projected data. In our experiments, some J-indexes were abnormally large or showed numerical problems. Therefore, we decided to measure the separability of the kernel mapped data by the classification accuracy of a classifier on the projected data. We will call this scheme "Projection+Classifier". That is the idea followed in our experiments with benchmark data sets, and used for the comparison of CKVSM[1] with other projection schemes, such as CVSM, PCA and KPCA: the "Projection" part was used for the comparison of the visualisations and the computation of the J-index values; the "Classifier" part was used to compare the accuracies of the resulting classifiers. In addition, since KLDA can also be decomposed into a first step of projection and a second step of classification, its projections can also be

[1] For CKVSM, recall that not labelled data can be projected onto the space defined by a labelled data set, so that the classifier can be constructed with a labelled projected training set and tested on an unlabelled projected data set, as pointed out in Section 3.4. The same happens for CVSM and, obviously, for pure unsupervised projection methods such as PCA or KPCA.

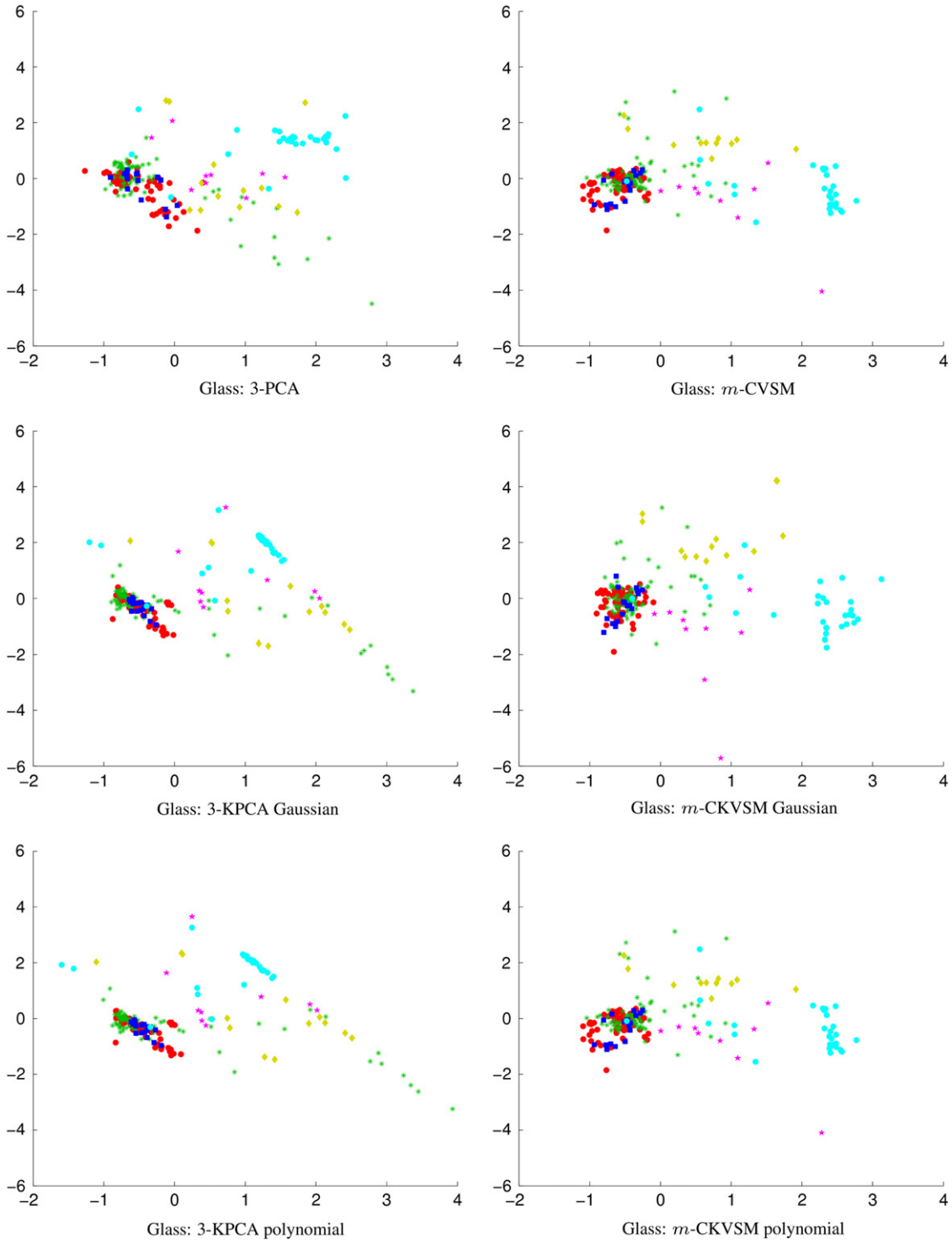

Fig. 9. 2D visualisations of the projections in the "Projection + 1NN" models for the Glass data set. Top row: 3-PCA (left) and m-CVSM (right). Middle row: 3-KPCA Gaussian (left) and m-CKVSM Gaussian (right). Bottom row: 3-KPCA polynomial (left) and m-CKVSM polynomial (right).

visualised and compared to those of CKVSM and the rest of projection methods. Since the inclusion of supervised information in the mapping process may introduce potential sample bias, we measured the out-of-sample generalisation of the classifiers in response to different kernels.

The intention of the quantitative results (Section 4.2.5) that complement the subjecting reading of the figures that follow (Section 4.2.4) is to assess how well the kernel element in CKVSM aggregates together examples from the same labelled cohort and how well the cohorts are separated. The use of classifiers in the evaluation is not intended to suggest that CKSVM is an alternative design for the metric of nearest neighbour classifiers, since kernel methods can be specifically designed to optimise their value for that purpose. Instead, the use of LDA is to measure the extent of linear separation between cohorts in the projective space and k-Nearest Neighbours is used to measure the uniformity, by cohort, of the neighbourhood structure around each example when projected.

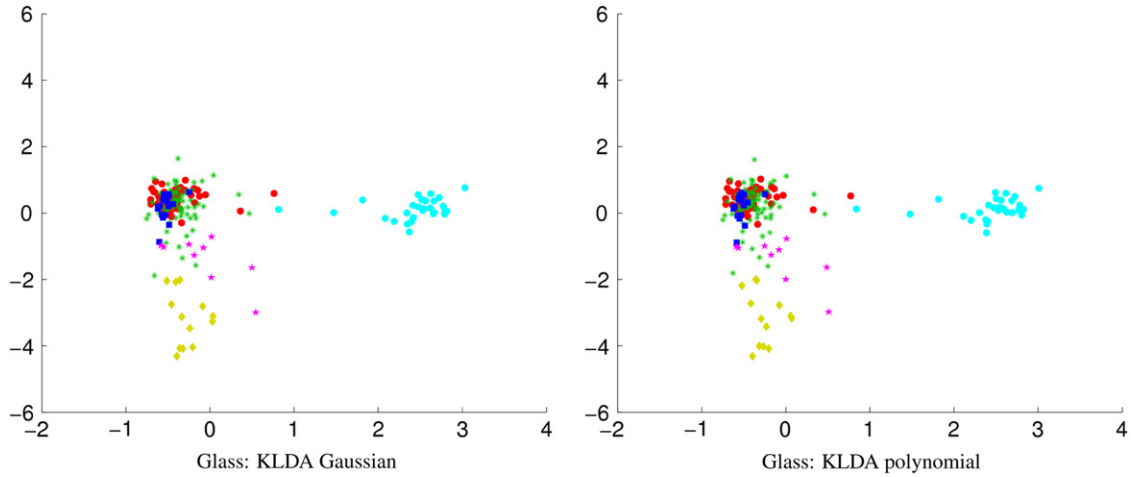

Fig. 10. 2D visualisations of the projections of KLDA for the Glass data set. Left: Gaussian kernel. Right: polynomial kernel.

Therefore, we expect that a projective structure that reflects well the cohort labels in near neighbour relationships will have optimal k-Nearest Neighbours classification for a small number of neighbours. If it generates linear separating boundaries then it will perform well with LDA also, while the J-index provides an overall merit figure that balances intra-cohort homogeneity with inter-cohort separation.

4.2.1. Data sets

Several benchmark data sets from the UCI repository [1] were used for the experiments: Splice-junction Gene Sequences (Gene), Glass Identification (Glass), Vehicle Silhouettes (Vehicle) and Wine. A brief description of these data sets is provided in Table 1. The motivation of the selection of these different data sets was to find representative examples with different dimensionality and different sample sizes, within public domain repositories, so that our work can be replicated and benchmarked in the future.

4.2.2. Performed experiments

The "Projection+Classifier" experiments were performed as follows:

1. The projections performed were
   (a) No projection (the classifier was applied in the input space).
   (b) Linear projections:
      (i) 3-PCA (three first components of PCA)
      (ii) m-CVSM (m first components of CVSM), where $m=\min(3,N_c-1)$.
   (c) Non-linear projections:
      (i) 3-KPCA (three first components of KPCA)
      (ii) m-CKVSM (m first components of CKVSM), where $m=\min(3,N_c-1)$.
2. The classifiers applied were
   (a) 1NN: 1-Nearest Neighbour.
   (b) LDA: Linear Discriminant Analysis.

Finally, a series of experiments were performed to compare m-CKVSM with KLDA. For these models, several issues were compared: visualisation of the projected data, accuracy, J-index and execution times.

4.2.3. Experimental setting

Data preprocessing: Two different preprocessings were applied to the input data: normalisation to zero mean and unit variance and sphering. For m-CKVSM, the data was (as in Algorithm 1) optionally sphered in the feature space.

Kernel and kernel parameter: The kernel functions used were

1. The Gaussian kernel $k(x,y)=e^{-\gamma\|x-y\|^2}$.
2. The polynomial kernel $k(x,y)=(\gamma x'y+1)^2$.

The tested values of the parameter γ were 0.000001, 0.000002, 0.000005, 0.00001, 0.00002, 0.00005, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, and 4.

Software: The experiments were performed in MATLAB. For m-CKVSM and m-CVSM we used our own implementation. The software used for 3-KPCA and KLDA was the "SVM and Kernel Methods Matlab Toolbox".[2] The KLDA version used was that of Baudat and Anouar [2].[3] For the rest of models, we used the PRTools software.[4]

Selection of the parameters and final models: For every combination of preprocessing and parameters, a stratified 10-fold cross-validation was performed. For the Glass data set, a fivefold cross-validation was performed, because one of the six original classes contained only nine examples, inadequate for a stratified 10-fold cross-validation. The final models were those with the best 10-fold cross-validation accuracies, obtained with the "one-versus-all" scheme. Note that the labels of new data are not used for the projection (see Algorithm 4).

Normalisation of the plots: In order to obtain similar ranges of the axes for the different plots, the projections were normalised subtracting the mean and dividing by the standard deviation of every dimension. Subsequently, the figures were plotted in fixed ranges for every data set. A different approach would be to scale the projective axes using the corresponding eigenvalues of the scatter matrix. However, this form of preprocessing proved less effective as it resulted in too different ranges among the plots.

4.2.4. Visualisation results

Two-dimensional visualisations of the projected data for the final models can be seen in Figs. 7–14. The visualisations

[2] Available at http://asi.insa-rouen.fr/enseignants/~arakotom/toolbox/index.html.
[3] Also available at http://www.kernel-machines.org/software.
[4] Available at http://www.prtools.org.

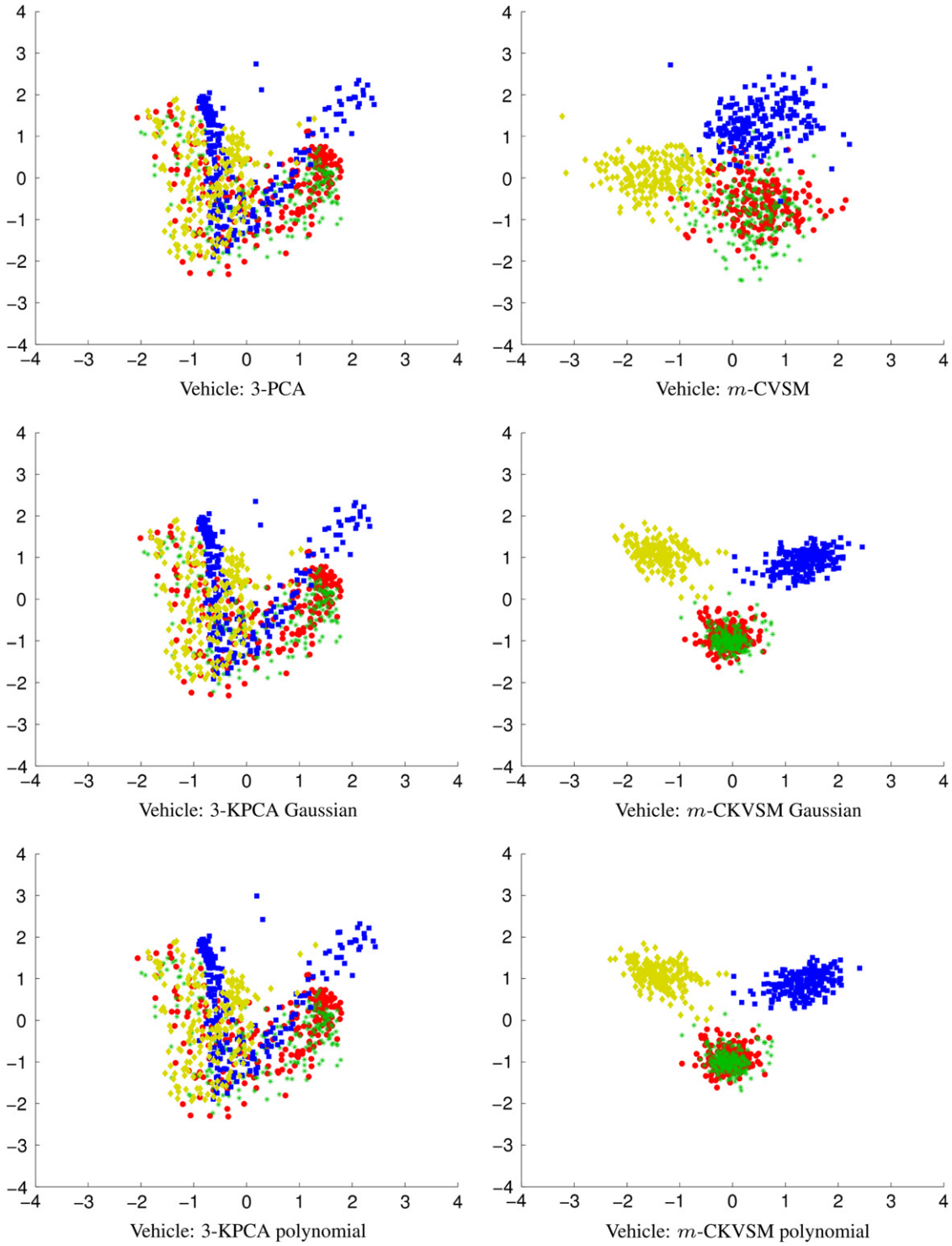

Fig. 11. 2D visualisations of the projections in the "Projection+1NN" models for the Vehicle data set. Top row: 3-PCA (left) and m-CVSM (right). Middle row: 3-KPCA Gaussian (left) and m-CKVSM Gaussian (right). Bottom row: 3-KPCA polynomial (left) and m-CKVSM polynomial (right).

were computed on the whole data set. Figs. 7, 9, 11, 13 are the visualisations of the projected data in the "Projection+1NN" scheme (see Section 4.2.2), and Figs. 8, 10, 12, 14 are the projected data of KLDA. As already mentioned, data was projected onto the two most important directions of the final models. Although data was projected (when possible) to 3D previous to the classification, 2D visualisations are shown for the sake of clarity. In general, visualisations in 3D show a similar behaviour, except for the Vehicle data set (see below). The relations between colours and classes can be found in Table 2.

In the case of the Gene data set the first thing to note from Fig. 7 is that m-CVSM enhances a better separation between cohorts, compared with 3-PCA. This is as expected. The effect of involving a non-linear kernel in the cluster-based method is to compress the data from each cohort and so further separate them. This is not achieved using 3-KPCA but it is with the m-CKVSM

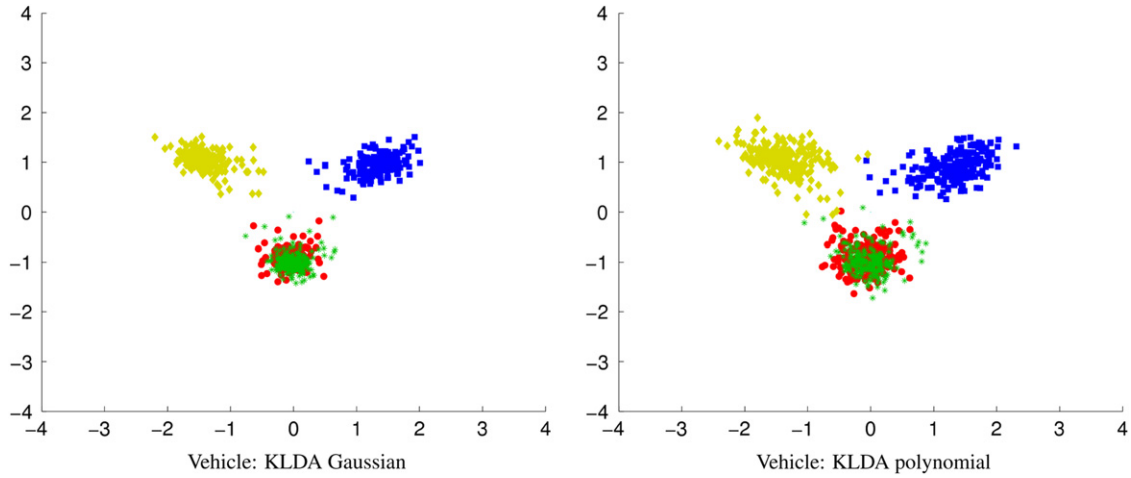

Fig. 12. 2D visualisations of the projections of KLDA for the Vehicle data set. Left: Gaussian kernel. Right: polynomial kernel.

approach showing that the visualisation space corresponds to a non-linear distortion of the original data space in the directions along class separating boundaries. The visualisations with the KLDA polynomial approach in Fig. 8 are similar to those of *m*-CKVSM polynomial. The visual artefacts for KLDA with Gaussian kernel in Fig. 8 are due to singularities in the scatter matrix (see results of KLDA in Table 3, Section 4.2.5). This pathological behaviour was not observed with *m*-CKVSM.

The number of different classes in the Glass data set means that there is greater loss in inter-class separation when projecting onto a low-dimensional visualisation space. In general, the similarities between plots are more defined by the method than by the kernel (for example, *m*-CKVSM Gaussian is more similar to *m*-CKVSM polynomial than to 3-KPCA Gaussian or KLDA Gaussian, see Figs. 9 and 10). The cohort based methods *m*-CVSM and *m*-CKVSM are more stable in the visual representation of the data, compared with 3-KPCA, although noticeable class mixing remains. The kernel approach spreads out the data better than its linear equivalent and the stability of the visualisation arguably points in the direction of the Gaussian kernel for this data set.

For the Vehicle data set, the visualisations of *m*-CKVSM and KLDA clearly show the best separation between cohorts (see Figs. 11 and 12). Similar to the Gene data set, 3-PCA and 3-KPCA in Fig. 11 do not obtain good separation, and in this case there exists a remarkable mixing between the classes. The linear *m*-CVSM projects the classes in different directions, although does not separate them. Again, the effect of using a non-linear kernel in the cluster-based method is to further separate the cohorts. In addition, the projections to 3D with *m*-CKVSM and KLDA are the only ones that separate classes red and green (see Fig. 15).

The visualisations for the Wine data set in Figs. 13 and 14 show, for all models, a good separation of the cohorts. This is an expected result, since the Wine data set has well defined class structures [1]. Note, however, that the visualisations of 3-PCA and 3-KPCA in Fig. 13 show a greater mixing between the classes than *m*-CVSM, *m*-CKVSM and KLDA. In the case of 3-KPCA, in addition, the classes are separated in a clearly different way that will be reflected in the classification results (see Table 3, Section 4.2.5).

In general, visualisations of *m*-CKVSM are more informative than those of 3-PCA, 3-KPCA, LDA and *m*-CVSM, and similar to KLDA visualisations, although KLDA tends to produce visualisations (previous to the normalisation) in a small range of values.

4.2.5. Classification results

Table 3 show the accuracies and J-indexes of the final "Projection+Classifier" (top) and KLDA models (bottom). The J-indexes were those of the projected data for the final models, and were computed on the whole data set. Figures in boldface underlined indicate the best results, and figures in boldface not underlined indicate the second best results, both for the accuracies and J-indexes. J-indexes and accuracies are positively correlated showing better accuracies models with higher J-indexes. Some J-indexes are abnormally large or show numerical problems (see, for example, the values for KLDA with Gaussian kernel).

For the Gene data set, 3-PCA and 3-KPCA obtain poor accuracies and low J-index values. Results of *m*-CVSM are slightly worse than those of *m*-CKVSM and KLDA. The best results are obtained by *m*-CKVSM and KLDA with J-index values in a similar range and likewise for the classification accuracies of both models. This is also reflected in the visual appearance of the *m*-CKVSM polynomial projection that is comparable to that obtained with the KLDA polynomial approach (see Figs. 7 and 8). Results for this data set indicate that the data may be moved toward being almost linearly separable, even for this large sample size. This is consistent with the relatively high classification accuracy for the linear method *m*-CVSM, which also retains, in 3D, the full separation J-index value from the original 120D space. However, the application of kernels improves both classification accuracy and J-index further, significantly exceeding the values obtained in the original data space.

In the case of the Glass data set, the classification accuracies do not return to the value obtained with the original data, coming closest for the *m*-CKVSM Gaussian projection (note that the visualisation results point in the direction of the Gaussian kernel for this data set, see Section 4.2.4). The Glass data set has the least number of variables and the largest number of classes of the studied data sets (see Table 1). Although it has also a small number of examples to fit the model, results in Table 3 suggest that the classes are highly non-linearly separable since the data set favours the 1NN classifier and a local kernel, poor accuracies are obtained using the LDA projections and, although the application of KLDA achieves clearly better cohort separation measured by the J-index, it does not improve the classification accuracy. The last observation indicates that the unexpected linear envelope in the visualisation introduced by KLDA may be an artefact.

The results of the Vehicle data set show several similarities with those of the Gene data set. On the one hand, 3-PCA and 3-KPCA have poor accuracies and low J-index values. On the other hand, *m*-CKVSM and KLDA have similar behaviour. In the Vehicle

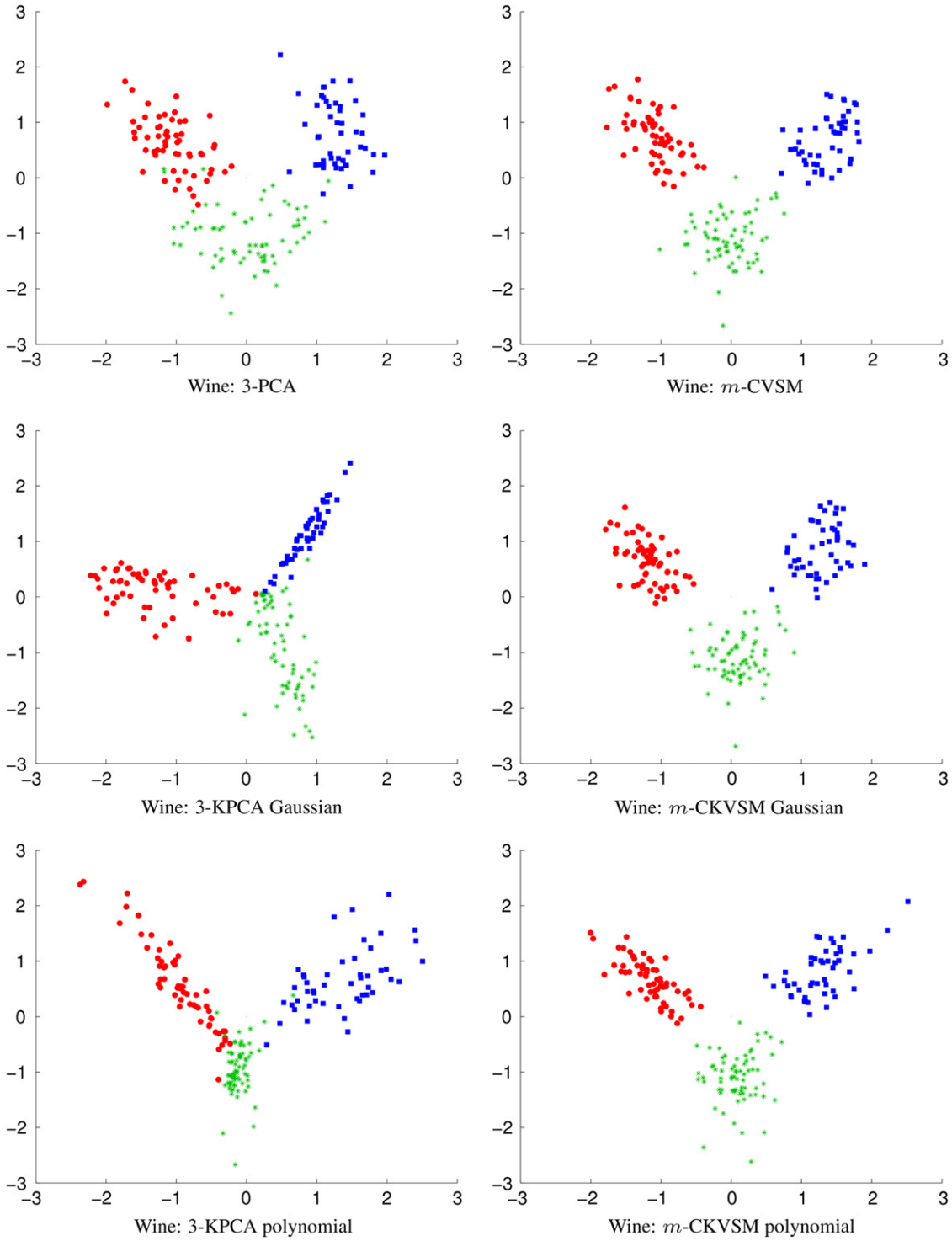

Fig. 13. 2D visualisations of the projections in the "Projection+1NN" models for the Wine data set. Top row: 3-PCA (left) and m-CVSM (right). Middle row: 3-KPCA Gaussian (left) and m-CKVSM Gaussian (right). Bottom row: 3-KPCA polynomial (left) and m-CKVSM polynomial (right).

data set, however, results of m-CVSM show larger differences with those of m-CKVSM and KLDA. The fact that the projections to 3D with m-CKVSM and KLDA are the only ones that separate classes red and green (see Fig. 15) is consistent with the high accuracies and J-index values for this data set.

The well defined class structures observed in the visualisations of the Wine data set are confirmed by the classification results. Accuracies and J-index values are very high for m-CVSM, m-CKVSM and KLDA. The clear linear separability of the classes means that both the LDA and 1NN classifiers perform similarly well. The worse results of 3-PCA and 3-KPCA can be explained by looking at Figs. 13 and 14, where it is apparent that 3-PCA and 3-KPCA do not separate the classes in the same way the rest of methods. This is reflected in the respective values of the classification accuracies.

In summary, results in Table 3 indicate that "m-CKVSM+Classifier" outperforms "3-PCA+Classifier", "3-KPCA+Classifier", "m-CVSM+Classifier", 1NN and LDA, and has similar performance

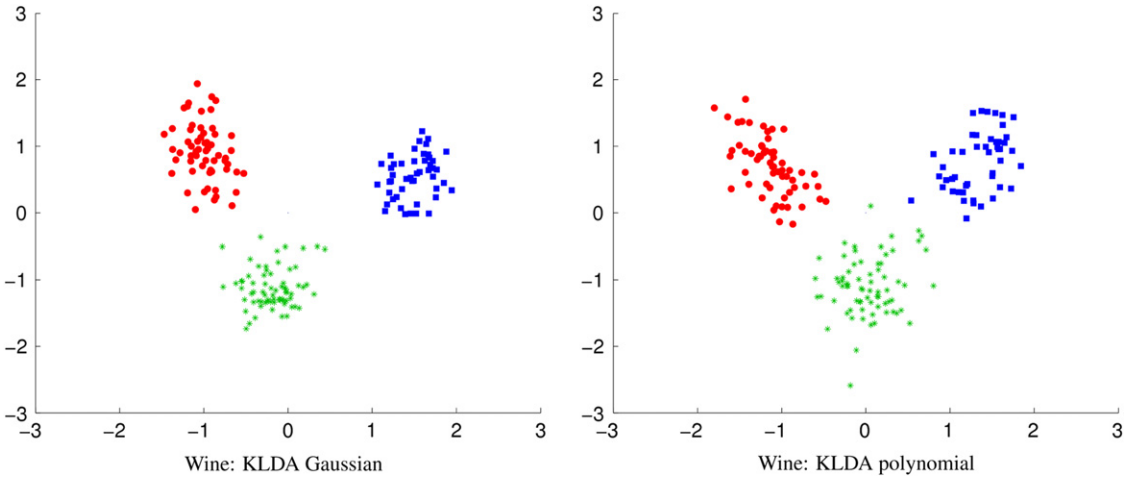

Fig. 14. 2D visualisations of the projections of KLDA for the Wine data set. Left: Gaussian kernel. Right: polynomial kernel.

Table 2
Relations between colours and classes in the visualisations of the benchmark data sets.

Colour	Gene	Glass	Vehicle	Wine
Red	EI	Building windows float	Opel	Wine of cultivar 1
Green	IE	Building windows non-float	Saab	Wine of cultivar 2
Blue	Neither	Vehicle windows float	Bus	Wine of cultivar 3
Yellow		Containers	Van	
Magenta		Tableware		
Cyan		Headlamps		

Table 3
Results of the "Projection+Classifier" (top) and KLDA final models (bottom).

Model		Gene		Glass		Vehicle		Wine	
Projection	Classifier	Accuracy	J-index	Accuracy	J-index	Accuracy	J-index	Accuracy	J-index
Original features	1NN	73.51	2.83	**72.43**	5.49	77.78	4.62	95.49	13.21
3-PCA	1NN	74.71	1.70	65.42	3.86	54.85	0.35	93.24	7.50
m-CVSM	1NN	86.55	2.83	66.37	4.62	75.29	4.62	98.30	13.21
3-KPCA Gaussian	1NN	74.68	1.70	69.68	3.84	55.20	0.35	97.78	7.31
3-KPCA polynomial	1NN	76.44	1.85	66.84	3.54	55.09	0.34	95.49	6.90
m-CKVSM Gaussian	1NN	87.81	4.77	**71.05**	6.00	81.09	22.76	**99.44**	**15.49**
m-CKVSM polynomial	1NN	90.14	**7.21**	67.76	4.62	81.57	22.19	**99.44**	12.98
Original features	LDA	87.47	2.83	14.03	5.49	77.41	4.62	**98.89**	13.21
3-PCA	LDA	77.99	1.70	13.19	3.86	41.96	0.39	96.11	7.50
m-CVSM	LDA	87.47	2.83	27.97	1.03	77.41	4.62	**99.44**	13.19
3-KPCA Gaussian	LDA	74.27	1.61	29.44	0.33	33.95	0.52	96.83	7.52
3-KPCA polynomial	LDA	75.42	1.68	24.22	0.03	27.54	0.01	95.32	7.16
m-CKVSM Gaussian	LDA	90.96	4.77	26.17	7.10	83.10	22.76	**99.44**	15.01
m-CKVSM polynomial	LDA	**93.13**	7.01	28.85	7.68	**83.56**	**24.05**	**99.44**	13.85
KLDA Gaussian		**91.31**	NumP	65.90	**19.26**	**86.40**	**48.95**	**98.89**	**25.02**
KLDA polynomial		87.53	**7.82**	65.90	**19.44**	83.44	21.02	**98.89**	14.58

than KLDA. In general, the lowest accuracies have been obtained by "3-PCA+Classifier" and "3-KPCA+Classifier". They can be justified because of the unsupervised nature of these models that do not profit from the known labels. Note that these models also show low values of the J-indexes. As expected, the linear nature of "*m*-CVSM+Classifier" allows to obtain better results when the data is almost linearly separable, as for the Wine data set.

Table 4 shows a comparison of the accuracies of the "*m*-CKVSM+1NN" models between sphering and not sphering in the feature space. Figures in boldface indicate the best results. Although better results are usually obtained without sphering, it will probably be dependent on the problem at hand. Anyway, the goodness of the results without sphering is encouraging, since sphering in the feature space is computationally more expensive (see Sections 3.4 and 4.2.6).

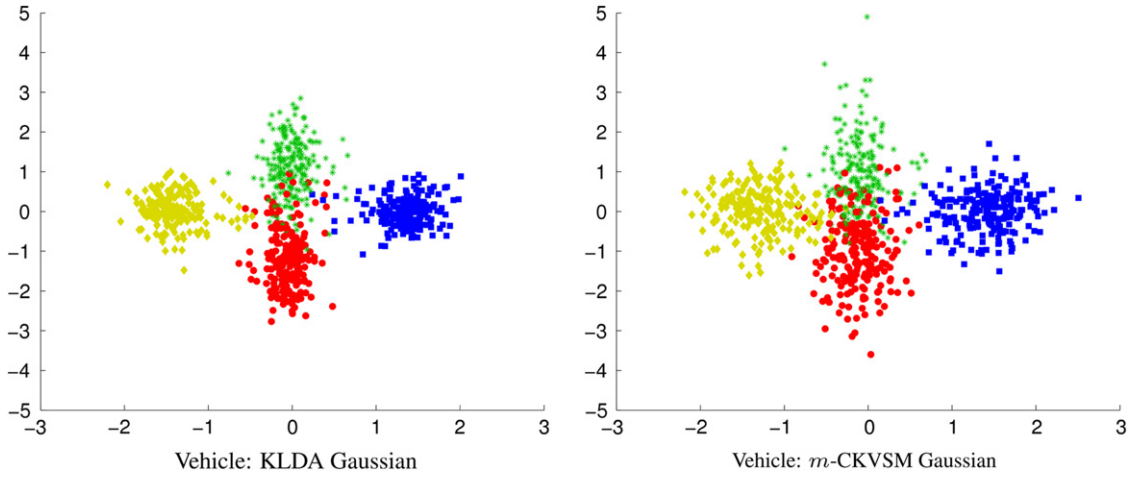

Fig. 15. Vehicle data set: 2D visualisations of the projections to 3D showing the separation between classes red and green. Left: KLDA Gaussian. Right: m-CKVSM Gaussian. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Comparison of accuracies between sphering and not sphering in the feature space ("m-CKVSM + 1NN" models).

Model	Sphering	Gene	Glass	Vehicle	Wine
m-CKVSM Gaussian	Yes	79.80	60.31	81.09	**99.44**
m-CKVSM polynomial	Yes	78.66	60.27	**81.57**	98.33
m-CKVSM Gaussian	No	87.81	**71.05**	79.52	98.89
m-CKVSM polynomial	No	**90.14**	67.76	76.47	**99.44**

Fig. 16 shows the accuracies of the "3-KPCA + 1NN", "m-CKVSM + 1NN" and KLDA models as a function of the parameter γ for the four studied data sets. The behaviour of "3-KPCA + LDA" and "m-CKVSM + LDA" was similar to that of "3-KPCA + 1NN" and "m-CKVSM + 1NN". In general, it has the typical appearance with an area of parameters that lead to good results, getting worse as we move away from that area, sometimes smoothly and sometimes abruptly. The first thing to note from Fig. 16 is that, for m-CKVSM and KLDA, the polynomial kernel is more stable than the Gaussian one. It can be clearly seen in the plots for the Gene and Vehicle data sets. Regarding the models, it seems clear that KLDA has a wider area of good parameters than m-CKVSM, specially for the Gaussian kernel. The shape of 3-KPCA is similar to those of m-CKVSM and KLDA, but in a lower level, confirming the lowest accuracies obtained by "3-KPCA + Classifier". Finally, note that the worst results are always in the largest values of the parameter suggesting to test small values of γ first.

4.2.6. Execution times

Table 5 shows several execution times of different models. As it can be seen, m-CKVSM is faster than 3-KPCA and KLDA. This is particularly significant when no sphering is performed, and it can be explained because 3-KPCA and KLDA must solve an eigenvalue problem of size N (N in the number of examples), whereas m-CKVSM solves an eigenvalue problem of size N_c (N_c is the number of classes). When sphering is performed, m-CKVSM also must solve an eigenvalue problem of size N. Note that results are many times better without sphering (see Table 4), and in this case the execution times of m-CKVSM are of the same order than those of m-CVSM.

Table 5
Execution times (in seconds) of the projection models and KLDA.

Model	Sphering	Gene	Glass	Vehicle	Wine
m-CVSM	–	1.64	0.81	0.84	0.80
3-KPCA Gaussian	–	949.26	1.09	20.99	0.92
3-KPCA polynomial	–	1090.81	1.05	17.34	0.92
m-CKVSM Gaussian	Yes	912.74	1.07	15.90	0.94
m-CKVSM Gaussian	No	2.79	0.79	0.97	0.80
m-CKVSM polynomial	Yes	1043.61	1.06	13.37	0.93
m-CKVSM polynomial	No	1.72	0.79	0.95	0.80
KLDA Gaussian	–	1720.09	1.63	21.26	1.38
KLDA polynomial	–	1246.28	1.60	14.43	1.42

5. Conclusions

This paper combined kernel methods with dimensionality reduction using class means to obtain accurate data classification in low-dimensional feature spaces that are suitable for direct visualisation of the data. A projective framework defined for linearly separated data in Lisboa et al. [15] was extended using the kernel trick and empirically shown to generate well-separated low-dimensional visualisations of benchmark externally-labelled data of moderate and high dimensions, resulting in a novel tool for data exploration and for class-specific data visualisation in human–computer interfaces.

The method of cohort-based kernel projection improves the visualisation of the linear method when the cohorts are not linearly separable. It is competitive with the best of Fisher and KPCA for cohort separation, classification, and data visualisation, provided a suitable kernel is used, and was found to be numerically more stable than KPCA and KLDA. It is proposed as a novel tool for the exploration of different kernels through data visualisation, and for obtaining linearly separable renditions of non-linearly separable data sets.

The classification results serve to measure the extent of near-neighbour uniformity with respect to classes and the extent of linearity in the separation between classes that the use of the proposed algorithm achieves. These results provide evidence for the effectiveness of the algorithm to linearise non-linearly separable data and so to aid visualisation of the separation between classes.

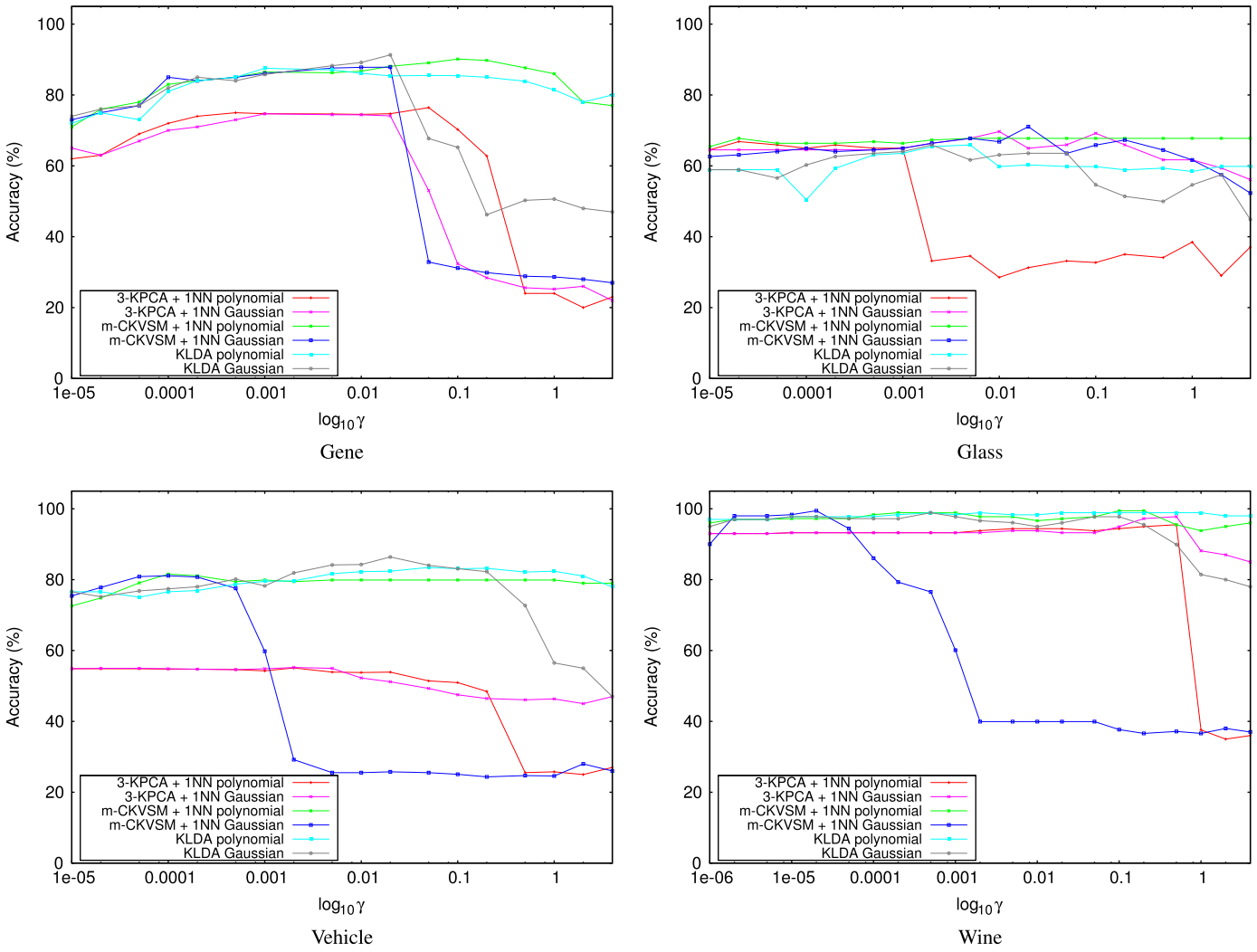

Fig. 16. Evolution of the accuracy as a function of the parameter γ (in $\log_{10}$ scale).

References

[1] A. Asuncion, D.J. Newman, UCI machine learning repository, University of California, Irvine, School of Information and Computer Science, ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩, 2007.

[2] G. Baudat, F. Anouar, Generalized discriminant analysis using a Kernel approach, Neural Computation 11 (2) (2000) 483–497.

[3] C.M. Bishop, M. Svensén, C.K.I. Williams, GTM: the generative topographic mapping, Neural Computation 10 (1) (1998) 215–234.

[4] T.F. Cox, M.A.A. Cox, Multidimensional Scaling, Chapman and Hall, UK, 2001.

[5] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, John Wiley, NY, 1973.

[6] H.P. Friedman, J. Rubin, On some invariant criteria for grouping data, Journal of the American Statistical Association 62 (320) (1967) 1159–1178.

[7] K. Fukunaga, Introduction to Statistical Pattern Recognition, second ed., Academic Press, NY, 1990.

[8] K.R. Gabriel, The biplot graphical display of matrices with applications to Principal Component Analysis, Biometrika 58 (3) (1971) 453–467.

[9] T. Kohonen, Self-organized formation of topologically correct feature maps, Biological Cybernetics 43 (1) (1982) 59–69.

[10] W.L.G. Koontz, K. Fukunaga, A nonlinear feature extraction algorithm using distance transformation, IEEE Transactions on Computers C-21 (1) (1972) 56–63.

[11] J.B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika 29 (1) (1964) 1–27.

[12] N.D. Lawrence, Gaussian process latent variable models for visualisation of high dimensional data. in: Advances in Neural Information Processing Systems, vol. 16, MIT Press, 2004, pp. 329–336.

[13] N.D. Lawrence, A.J. Moore, Hierarchical Gaussian Process latent variable models, in: International Conference on Machine Learning, 2007, pp. 481–488.

[14] J.A. Lee, M. Verleysen, Nonlinear Dimensionality Reduction, Springer-Verlag, NY, 2007.

[15] P.J.G. Lisboa, I.O. Ellis, A.R. Green, F. Ambrogi, M.B. Dias, Cluster-based visualisation with scatter matrices, Pattern Recognition Letters 29 (13) (2008) 1814–1823.

[16] D. Lowe, M. Tipping, Feed-forward neural networks and topographic mappings for exploratory data analysis, Neural Computing and Applications 4 (2) (1996) 83–95.

[17] J. Mao, A.K. Jain, Artificial neural networks for feature extraction and multivariate data projection, IEEE Transactions on Neural Networks 6 (1995) 296–317.

[18] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, Philosophical Transactions of the Royal Society London A 209 (1909) 415–446.

[19] E. Pękalska, B. Haasdonk, Kernel discriminant analysis for positive definite and indefinite kernels, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (6) (2009) 1017–1032.

[20] J.W. Sammon, A non-linear mapping for data structure analysis, IEEE Transactions on Computers C-18 (1969) 401–408.

[21] B. Schölkopf, A.J. Smola, Learning with Kernels, MIT Press, 2002.

[22] B. Schölkopf, A.J. Smola, K.R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (5) (1998) 1299–1319.

[23] A. Strehl, J. Ghosh, Relationship-based clustering and visualization for high-dimensional data mining, INFORMS Journal on Computing 15 (2) (2003) 208–230.

[24] J.B. Tenenbaum, V. de Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.

[25] W.S. Torgerson, Multidimensional scaling: I. Theory and method, Psychometrika 17 (4) (1952) 401–419.
[26] J.G. Wang, Y.S. Lin, W.K. Yang, J.Y. Yang, Kernel maximum scatter difference based feature extraction and its application to face recognition, Pattern Recognition Letters 29 (13) (2008) 1832–1835.
[27] L. Wang, K.L. Chan, P. Xue, L. Zhou, A kernel-induced space selection approach to model selection in KLDA, IEEE Transactions on Neural Networks 19 (12) (2008) 2116–2131.
[28] J. Yang, Z. Jin, J.Y. Yang, D. Zhang, A.F. Frangi, Essence of kernel fisher discriminant: KPCA plus LDA, Pattern Recognition 37 (10) (2004) 2097–2100.