Kernel Entropy Component Analysis

Robert Jenssen, *Member*, *IEEE*

Abstract—We introduce kernel entropy component analysis (kernel ECA) as a new method for data transformation and dimensionality reduction. Kernel ECA reveals structure relating to the Renyi entropy of the input space data set, estimated via a kernel matrix using Parzen windowing. This is achieved by projections onto a subset of entropy preserving kernel principal component analysis (kernel PCA) axes. This subset does not need, in general, to correspond to the top eigenvalues of the kernel matrix, in contrast to the dimensionality reduction using kernel PCA. We show that kernel ECA may produce strikingly different transformed data sets compared to kernel PCA, with a distinct angle-based structure. A new spectral clustering algorithm utilizing this structure is developed with positive results. Furthermore, kernel ECA is shown to be an useful alternative for pattern denoising.

Index Terms—Spectral data transformation, Renyi entropy, Parzen windowing, kernel PCA, clustering, pattern denoising.

1 Introduction

Data transformation is of fundamental importance in pattern analysis and machine intelligence. The goal is to transform the potentially high-dimensional data into an alternative and typically lower dimensional representation, which reveals the underlying structure of the data. For example, a method which transforms nonlinear class structure into linear dependencies may be greatly beneficial for clustering purposes.

There exists a vast literature on data transformation methods, see, for example, the textbooks [1], [2]. One dominant direction of research in this area is exemplified by the so-called *spectral methods*, which is based on the *top or bottom* eigenvalues (spectrum) and eigenvectors of specially constructed data matrixes. The most well known such method is principal component analysis (PCA) [3], which is based on the data correlation matrix. It is a *linear* method ensuring that the transformed data are uncorrelated and preserve maximally the second order statistics of the original data. Another linear method is metric multidimensional scaling (MDS) [4], which preserves inner products. It is based on the inner product matrix of the data. Metric MDS and PCA can be shown to be equivalent.

In recent years, a number of advanced *nonlinear* spectral data transformation methods have been proposed. A very influential method is kernel PCA [5]. Kernel PCA performs traditional PCA in a so-called kernel feature space, which is nonlinearly related to the input space. This is enabled by a positive semidefinite (psd) kernel function, which computes inner products in the kernel feature space. An inner product matrix, or *kernel matrix*, can thus be constructed. Performing metric MDS on the kernel matrix, based on the top eigenvalues of the matrix, provides the kernel PCA data transformation. Kernel PCA has been used in many contexts. A spectral clustering method was, for example, developed in [6], where C-means [7] is performed on the kernel PCA eigenvectors. Kernel PCA has also been used for pattern denoising [8], [9], [10] and in classification [11].

Many other spectral methods exist, differing in the way the data matrixes are constructed. In [12], for example, the data transformation is based on the normalized Laplacian matrix, and clustering is performed using C-means on unit norm spectral data. Manifold learning and other clustering variants using the Laplacian matrix exist, see, e.g., [13] and [14]. Other recent spectral methods include locally linear embedding [15], isometric mapping [16], and maximum variance unfolding [17], to name a few. See the recent review papers [18], [19] for thorough reviews of several spectral methods for dimensionality reduction.

In this paper, we develop a new spectral data transformation method, which is fundamentally different from other spectral methods in two very important ways as follows:

- The data transformation reveals structure related to the *Renyi entropy of the input space data set*.
- The method does *not* necessarily use the *top* eigenvalues and eigenvectors of the kernel matrix.

The new method is appropriately called kernel entropy component analysis, or kernel ECA for short. In order to develop kernel ECA, we show that an estimator of the Renyi (quadratic) entropy may be expressed in terms of the spectral properties of a kernel matrix induced by Parzen windowing for density estimation. Assuming that the window, or kernel, function is psd, the Renyi entropy estimator thus obtained is expressed in terms of *projections onto principal axes in kernel feature space*. These principal axes are the kernel PCA axes. However, in kernel ECA, the data transformation and dimensionality reduction are obtained by projecting onto those kernel PCA axes that contribute to the entropy estimate of the input space data set, and these axes will, in general, not necessarily correspond to the top eigenvalues and eigenvectors of the kernel matrix, as is the case for kernel PCA dimensionality reduction.

Indeed, kernel ECA may produce strikingly different data transformations than kernel PCA, as illustrated both theoretically and on several toy data sets and real data sets. In particular, we show that kernel ECA typically produces a transformed data set with a distinct *angular* structure, in the

• *The author is with the Department of Physics and Technology, University of Tromsø, N-9037 Tromsø, Norway. E-mail: robert.jenssen@uit.no.*

Manuscript received 3 Nov. 2008; revised 13 Mar. 2009; accepted 23 Apr. 2009; published online 29 Apr. 2009.
Recommended for acceptance by O. Chapelle.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-11-0759.
Digital Object Identifier no. 10.1109/TPAMI.2009.100.

sense that even nonlinearly related input space data clusters are distributed in different angular directions with respect to the origin of the kernel feature space. Kernel ECA thus reveals cluster structure and in a sense, therefore information about the underlying labels of the data.

Another main contribution of this paper is to take advantage of the angular structure by developing a new angle-based spectral clustering algorithm using kernel ECA for data representation. Clustering results comparable or even better in some cases than Laplacian matrix-based spectral clustering are obtained.

Finally, we illustrate kernel ECA as a reasonable alternative to kernel PCA with respect to pattern denoising.[1]

The remainder of the paper is organized as follows: Section 2 provides examples of some spectral data transformation methods of importance. In Section 3, kernel ECA is introduced. A spectral clustering algorithm using kernel ECA is developed in Section 4. Kernel ECA for pattern denoising is illustrated in Section 5. Finally, Section 6 concludes the paper.

2 Examples of Spectral Data Transformations

PCA is perhaps the best-known spectral data transformation method. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, where $\mathbf{x}_t \in \mathcal{R}^d$, $t = 1, \ldots, N$. In PCA, a *linear* transformation $\mathbf{Y}_{pca} = \mathbf{A}\mathbf{X}$ is sought, where $\mathbf{Y}_{pca} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$. At this point, we assume $\mathbf{A}$ is $(d \times d)$ such that $\mathbf{y}_t \in \mathcal{R}^d$, $t = 1, \ldots, N$. The sample correlation matrix of $\mathbf{Y}_{pca}$ is given by

$$\frac{1}{N}\mathbf{Y}_{pca}\mathbf{Y}_{pca}^T = \frac{1}{N}\mathbf{A}\mathbf{X}(\mathbf{A}\mathbf{X})^T = \mathbf{A}\frac{1}{N}\mathbf{X}\mathbf{X}^T\mathbf{A}^T, \tag{1}$$

where $\frac{1}{N}\mathbf{X}\mathbf{X}^T$ is the sample correlation matrix of $\mathbf{X}$. The goal is to determine $\mathbf{A}$ such that $\frac{1}{N}\mathbf{Y}_{pca}\mathbf{Y}_{pca}^T = \mathbf{I}$, the identity matrix, yielding uncorrelated data $\mathbf{Y}_{pca}$. By eigendecomposition, $\frac{1}{N}\mathbf{X}\mathbf{X}^T = \mathbf{V}\boldsymbol{\Delta}\mathbf{V}^T$, where $\boldsymbol{\Delta}$ is a diagonal matrix storing the eigenvalues $\delta_1, \ldots, \delta_d$ in decreasing order, with the corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ as the columns of $\mathbf{V}$. Now, by resubstitution into (1), it is easily verified that the choice $\mathbf{A} = \boldsymbol{\Delta}^{-\frac{1}{2}}\mathbf{V}^T$ achieves the goal such that $\mathbf{Y}_{pca} = \boldsymbol{\Delta}^{-\frac{1}{2}}\mathbf{V}^T\mathbf{X}$. Note that the jth element, or row, of $\mathbf{Y}_{pca}$ corresponds to a projection of $\mathbf{X}$ onto the vector, or principal axis $\mathbf{v}_j$, that is, $P_{\mathbf{v}_j}\mathbf{X} = \delta_j^{-\frac{1}{2}}\mathbf{v}_j^T\mathbf{X}$.

A dimensionality reduction from d to $l < d$ is often performed by projecting onto a subspace spanned by the principal axes corresponding to the top l eigenvalues. Hence, $\mathbf{A} = \boldsymbol{\Delta}_l^{-\frac{1}{2}}\mathbf{V}_l^T$, where $\boldsymbol{\Delta}_l$ consists of the l largest eigenvalues with the corresponding eigenvectors as the columns of $\mathbf{V}_l$. It is well known that the resulting l-dimensional $\mathbf{Y}_{pca_l} = \boldsymbol{\Delta}_l^{-\frac{1}{2}}\mathbf{V}_l^T\mathbf{X}$ preserves the maximum amount of *second order statistics*[2] in the l-dimensional data compared to the original d-dimensional data.

1. This paper is an extended version of the work presented in the two conference papers [20] and [21].

2. Often, the data points in $\mathbf{X}$ are assumed to be zero mean, in which case $\frac{1}{N}\mathbf{Y}\mathbf{Y}^T$ ($\frac{1}{N}\mathbf{X}\mathbf{X}^T$) becomes the *covariance* matrix of the output (input) data. Hence, $\frac{1}{N}\mathbf{Y}\mathbf{Y}^T = \mathbf{I}$ corresponds to uncorrelated output data with unit *variance* in each dimension. Dimensionality reduction by PCA therefore directly preserves variance in the zero mean case.

Another well-known data transformation method is known as metric, or classical, MDS. In MDS, the goal is to preserve inner products in the l-dimensional data compared to the original d-dimensional data. The inner product matrix (Gram matrix) of the data is given by $\mathbf{G} = \mathbf{X}^T\mathbf{X}$. By eigendecomposition, we have $\mathbf{G} = \mathbf{E}'\mathbf{D}'\mathbf{E}'^T$, where $\mathbf{D}'$ stores the eigenvalues $\lambda'_1, \ldots, \lambda'_N$ and $\mathbf{E}'$ stores the corresponding eigenvectors $\mathbf{e}'_1, \ldots, \mathbf{e}'_N$. A data transformation that exactly preserves inner products is given by $\mathbf{Y}_{mds} = \mathbf{D}'^{\frac{1}{2}}\mathbf{E}'^T$ because $\mathbf{Y}_{mds}^T\mathbf{Y}_{mds} = (\mathbf{D}'^{\frac{1}{2}}\mathbf{E}'^T)^T\mathbf{D}'^{\frac{1}{2}}\mathbf{E}'^T = \mathbf{G}$. The l-dimensional MDS data transformation which best preserves inner products in a least-square sense is given by $\mathbf{Y}_{mds_l} = \mathbf{D}_l'^{\frac{1}{2}}\mathbf{E}_l'^T$, where $\mathbf{D}'_l$ consists of the top l eigenvalues of $\mathbf{G}$ with the corresponding eigenvectors as the columns of $\mathbf{E}'_l$ because it is the solution to the minimization problem

$$\mathbf{Y}_{mds_l} = \mathbf{D}_l'^{\frac{1}{2}}\mathbf{E}_l'^T : \min_{\lambda'_1, \mathbf{e}'_1, \ldots, \lambda'_N, \mathbf{e}'_N} \mathbf{1}^T(\mathbf{G} - \mathbf{G}_{mds}).^2\mathbf{1}. \tag{2}$$

Here, $\mathbf{G}_{mds} = \mathbf{Y}_{mds_l}^T\mathbf{Y}_{mds_l} = \mathbf{E}'_l\mathbf{D}'_l\mathbf{E}'^T_l$, $(\cdot).^2$ denotes element-wise squaring and $\mathbf{1}$ is an $(N \times 1)$ vector where each element equals one. Note that, by picking the top l eigenvalues and corresponding eigenvectors to constitute $\mathbf{Y}_{mds_l}$, the Frobenius norm, given by (2), of $\mathbf{G} - \mathbf{G}_{mds}$ is minimized.

There is a direct correspondence between PCA and metric MDS. Consider an eigenvalue-eigenvector pair $\lambda', \mathbf{e}'$ of $\mathbf{X}^T\mathbf{X}$. We have

$$\frac{1}{N}\mathbf{X}\mathbf{X}^T(\mathbf{X}\mathbf{e}') = \lambda'(\mathbf{X}\mathbf{e}') \tag{3}$$

since $\mathbf{X}^T\mathbf{X}\mathbf{e}' = \lambda'\mathbf{e}'$. Thus, $\delta = \lambda'$ is an eigenvalue of $\frac{1}{N}\mathbf{X}\mathbf{X}^T$ and $\mathbf{v} = \mathbf{X}\mathbf{e}'$ is a corresponding eigenvector. If we assume $d < N$, then $r = rank(\frac{1}{N}\mathbf{X}\mathbf{X}^T) = rank(\mathbf{X}^T\mathbf{X}) \leq d$ since (3) must hold. Therefore, the first r columns of $\mathbf{V}$ are $\mathbf{V}_r = \mathbf{X}\mathbf{E}'_r$ and $\boldsymbol{\Delta}_r = \mathbf{D}_r$. Moreover,

$$\begin{aligned}\mathbf{Y}_{pca_r} &= \mathbf{D}_r'^{-\frac{1}{2}}\left(\mathbf{X}\mathbf{E}'_r\right)^T\mathbf{X} = \mathbf{D}_r'^{-\frac{1}{2}}\mathbf{E}_r'^T\mathbf{X}^T\mathbf{X} \\ &= \mathbf{D}_r'^{-\frac{1}{2}}\mathbf{E}_r'^T\mathbf{E}'_r\mathbf{D}'_r\mathbf{E}_r'^T = \mathbf{D}_r'^{\frac{1}{2}}\mathbf{E}_r'^T = \mathbf{Y}_{mds_r}.\end{aligned}$$

Similarly, $\mathbf{Y}_{pca_l} = \mathbf{Y}_{mds_l}$ for $l < r$. As a result of this analysis, the projection of $\mathbf{X}$ onto the jth principal axis $\mathbf{v}_j$ may also be expressed as $P_{\mathbf{v}_j}\mathbf{X} = \sqrt{\lambda'_j}\mathbf{e}_j'^T$ (see [22] for a related derivation).

Schölkopf et al. [5] proposed a nonlinear version of PCA operating implicitly in a so-called kernel feature space, which is nonlinearly related to the input space. The nonlinear map from input space to feature space is given by $\phi : \mathcal{R}^d \rightarrow \mathcal{F}$ such that $\mathbf{x}_t \rightarrow \phi(\mathbf{x}_t)$, $t = 1, \ldots, N$. Let $\boldsymbol{\Phi} = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N)]$. To perform PCA in $\mathcal{F}$, we need to find an expression for the projection $P_{\mathbf{u}_i}\boldsymbol{\Phi}$ of $\boldsymbol{\Phi}$ onto a feature space principal axes $\mathbf{u}_i$, or onto a subspace U_l spanned, for example, by the top l such principal axes. This may be achieved implicitly via the kernel function.

A positive semidefinite kernel function, or Mercer kernel [23], [24], $k_\sigma : \mathcal{R}^d \times \mathcal{R}^d \rightarrow \mathcal{R}$ computes an inner product in the Hilbert space $\mathcal{F}$:

$$k_\sigma(\mathbf{x}_t, \mathbf{x}_{t'}) = \langle \phi(\mathbf{x}_t), \phi(\mathbf{x}_{t'}) \rangle. \tag{4}$$

We may define the $(N \times N)$ *kernel matrix* $\mathbf{K}$ such that element (t, t') of $\mathbf{K}$ equals $k_\sigma(\mathbf{x}_t, \mathbf{x}_{t'})$. Hence, $\mathbf{K} = \mathbf{\Phi}^T\mathbf{\Phi}$ is an inner product (Gram) matrix in $\mathcal{F}$. The kernel matrix may be eigendecomposed as $\mathbf{K} = \mathbf{E}\mathbf{D}\mathbf{E}^T$, where $\mathbf{D}$ is a diagonal matrix storing the eigenvalues $\lambda_1, \ldots, \lambda_N$ and $\mathbf{E}$ is a matrix with the corresponding eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_N$ as columns.

Williams [25] pointed out that the equivalence between PCA and MDS discussed earlier also holds in a kernel feature space. Therefore, the resulting *kernel PCA* expression becomes

$$\mathbf{\Phi}_{pca} = P_{U_l}\mathbf{\Phi} = \mathbf{D}_l^{\frac{1}{2}}\mathbf{E}_l^T, \tag{5}$$

where $\mathbf{D}_l$ consists of the top l eigenvalues of $\mathbf{K}$ and $\mathbf{E}_l$ stores the corresponding eigenvectors as columns. This means that the projection $P_{\mathbf{u}_i}\mathbf{\Phi}$ of $\mathbf{\Phi}$ onto a feature space principal axes $\mathbf{u}_i$ is given by $P_{\mathbf{u}_i}\mathbf{\Phi} = \sqrt{\lambda_i}\mathbf{e}_i^T$.

Using the analogy with (2), $\mathbf{\Phi}_{pca} = \mathbf{D}_l^{\frac{1}{2}}\mathbf{E}_l^T$ is the solution to the minimization problem

$$\mathbf{\Phi}_{pca} = \mathbf{D}_l^{\frac{1}{2}}\mathbf{E}_l^T : \min_{\lambda_1, \mathbf{e}_1, \ldots, \lambda_N, \mathbf{e}_N} \mathbf{1}^T(\mathbf{K} - \mathbf{K}_{pca}).^2\mathbf{1}, \tag{6}$$

where $\mathbf{K}_{pca} = \mathbf{\Phi}_{pca}^T\mathbf{\Phi}_{pca} = \mathbf{E}_l\mathbf{D}_l\mathbf{E}_l^T$. Hence, the kernel PCA procedure minimizes the Frobenius norm of $\mathbf{K} - \mathbf{K}_{pca}$.

In analogy to the relation $\mathbf{v} = \mathbf{X}\mathbf{e}'$ following (3), we can obtain an explicit expression for a principal axis $\mathbf{u}_i$ in the kernel feature space. This is helpful for deriving a formula for projecting an out-of-sample data point $\phi(\mathbf{x})$ onto that axis. Requiring $\|\mathbf{u}_i\|^2 = 1$, we thus have $\mathbf{u}_i = \lambda^{-\frac{1}{2}}\mathbf{\Phi}\mathbf{e}_i$. Hence,

$$\begin{aligned} P_{\mathbf{u}_i}\phi(\mathbf{x}) = \mathbf{u}_i^T\phi(\mathbf{x}) &= \left\langle \lambda_i^{-\frac{1}{2}} \sum_{t=1}^{N} \mathbf{e}_{i,t}\phi(\mathbf{x}_t), \phi(\mathbf{x}) \right\rangle \\ &= \lambda^{-\frac{1}{2}} \sum_{t=1}^{N} \mathbf{e}_{i,t} k_\sigma(\mathbf{x}_t, \mathbf{x}), \end{aligned} \tag{7}$$

where $\mathbf{e}_{i,t}$ denotes the tth element of $\mathbf{e}_i$. Let $\mathbf{\Phi}'$ refer to a collection of out-of-sample data points, and define the inner product matrix $\mathbf{K}' = \mathbf{\Phi}'^T\mathbf{\Phi}$. Then, by (7), $\mathbf{\Phi}'_{pca} = P_{U_l}\mathbf{\Phi}' = \mathbf{D}_l^{-\frac{1}{2}}\mathbf{E}_l^T\mathbf{K}'^T$.

In the kernel PCA literature, $\mathbf{K}$ is often centered. We do not center in this exposition unless otherwise stated.

3 Kernel Entropy Component Analysis

The Renyi quadratic entropy is given by [26]

$$H(p) = -\log \int p^2(\mathbf{x}) d\mathbf{x}, \tag{8}$$

where $p(\mathbf{x})$ is the probability density function generating the data set, or sample, $\mathcal{D} = \mathbf{x}_1, \ldots, \mathbf{x}_N$. Since the logarithm is a monotonic function, we may concentrate on the quantity $V(p) = \int p^2(\mathbf{x}) d\mathbf{x}$. Alternatively, this expression may be formulated as $V(p) = \mathcal{E}_p(p)$, where $\mathcal{E}_p(\cdot)$ denotes expectation w.r.t. the density $p(\mathbf{x})$.

In order to estimate $V(p)$, and hence $H(p)$, we may invoke a Parzen window density estimator, given by [27],

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{\mathbf{x}_t \in \mathcal{D}} k_\sigma(\mathbf{x}, \mathbf{x}_t). \tag{9}$$

Here, $k_\sigma(\mathbf{x}, \mathbf{x}_t)$ is the so-called Parzen window, or kernel, centered at $\mathbf{x}_t$ and with a width governed by the parameter σ. In order for $\hat{p}(\mathbf{x})$ to be a proper density function, $k_\sigma(\mathbf{x}, \cdot)$ must be a density function itself [28]. We choose to denote the Parzen kernel by the same symbol $k_\sigma(\cdot, \cdot)$ as the Mercer kernel discussed above, for reasons that will become apparent.

Using the sample mean approximation of the expectation operator, we have

$$\begin{aligned} \hat{V}(p) &= \frac{1}{N} \sum_{\mathbf{x}_t \in \mathcal{D}} \hat{p}(\mathbf{x}_t) = \frac{1}{N} \sum_{\mathbf{x}_t \in \mathcal{D}} \frac{1}{N} \sum_{\mathbf{x}_{t'} \in \mathcal{D}} k_\sigma(\mathbf{x}_t, \mathbf{x}_{t'}) \\ &= \frac{1}{N^2} \mathbf{1}^T\mathbf{K}\mathbf{1}, \end{aligned} \tag{10}$$

where element (t, t') of the $(N \times N)$ kernel matrix $\mathbf{K}$ equals $k_\sigma(\mathbf{x}_t, \mathbf{x}_{t'})$ and $\mathbf{1}$ is an $(N \times 1)$ vector where each element equals one. Hence, the Renyi entropy estimate obtained based on the available sample fully resides in the elements of the corresponding kernel matrix. This was first noticed by Girolami [29].

Moreover, the Renyi entropy estimator may be expressed in terms of the eigenvalues and eigenvectors of the kernel matrix, which may be eigendecomposed as $\mathbf{K} = \mathbf{E}\mathbf{D}\mathbf{E}^T$, where, as before, $\mathbf{D}$ is a diagonal matrix storing the eigenvalues $\lambda_1, \ldots, \lambda_N$ and $\mathbf{E}$ is a matrix with the corresponding eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_N$ as columns. Rewriting (10), we then have

$$\hat{V}(p) = \frac{1}{N^2} \sum_{i=1}^{N} \left(\sqrt{\lambda_i}\mathbf{e}_i^T\mathbf{1}\right)^2. \tag{11}$$

Each term in this expression will contribute to the entropy estimate. This means that certain eigenvalues and eigenvectors will contribute more to the entropy estimate than others since the terms depend on different eigenvalues and eigenvectors.

In the analysis that follows, we assume that the Parzen window is psd. This is true, for instance, for the most widely used Parzen window, namely the Gaussian. This means that the matrix $\mathbf{K}$ in (10) equals the familiar kernel matrix used in kernel PCA. We are now able to interpret the Renyi entropy estimator in terms of projections in a Mercer kernel feature space.

3.1 Renyi Entropy and Projections onto Principal Axes

Using kernel PCA, the projection of $\mathbf{\Phi}$ onto the ith principal axis $\mathbf{u}_i$ in the kernel feature space is defined as $P_{\mathbf{u}_i}\mathbf{\Phi} = \sqrt{\lambda_i}\mathbf{e}_i^T$. Equation (11) therefore reveals that *the Renyi entropy estimator is composed of projections onto all the kernel PCA axes*, given a psd window function. However, only a principal axis $\mathbf{u}_i$ for which $\lambda_i \neq 0$ *and* $\mathbf{e}_i^T\mathbf{1} \neq 0$ contributes to the entropy estimate. In fact, a large eigenvalue λ_i does not guarantee that $\mathbf{u}_i$ contributes to the entropy estimate at all.

Those principal axes contributing most to the Renyi entropy estimate clearly carry most of the information regarding the shape of the pdf generating the input space data set. Note that kernel PCA performs dimensionality reduction by selecting l eigenvalues and eigenvectors solely based on the size of the eigenvalues, and the resulting transformation may be based on uninformative eigenvectors from an entropy perspective.

3.2 Defining the Kernel ECA Transformation

We define kernel entropy component analysis as a k-dimensional data transformation obtained by projecting $\mathbf{\Phi}$ onto a subspace U_k spanned by those k kernel PCA axes contributing most to the Renyi entropy estimate of the data. Hence, U_k is composed of a subset of the kernel PCA axes, but not necessarily those corresponding to the top k eigenvalues. The transformation is denoted by

$$\mathbf{\Phi}_{eca} = P_{U_k}\mathbf{\Phi} = \mathbf{D}_k^{\frac{1}{2}}\mathbf{E}_k^T, \tag{12}$$

and it is the solution to the minimization problem

$$\mathbf{\Phi}_{eca} = \mathbf{D}_k^{\frac{1}{2}}\mathbf{E}_k^T : \min_{\lambda_1,\mathbf{e}_1,\ldots,\lambda_N,\mathbf{e}_N} \hat{V}(p) - \hat{V}_k(p), \tag{13}$$

where the entropy estimate associated with $\mathbf{\Phi}_{eca}$ is expressed by

$$\hat{V}_k(p) = \frac{1}{N^2}\mathbf{1}^T\mathbf{E}_k\mathbf{D}_k\mathbf{E}_k^T\mathbf{1} = \frac{1}{N^2}\mathbf{1}^T\mathbf{K}_{eca}\mathbf{1}, \tag{14}$$

where $\mathbf{K}_{eca} = \mathbf{E}_k\mathbf{D}_k\mathbf{E}_k^T$. An alternative expression yields

$$\mathbf{\Phi}_{eca} = \mathbf{D}_k^{\frac{1}{2}}\mathbf{E}_k^T : \min_{\lambda_1,\mathbf{e}_1,\ldots,\lambda_N,\mathbf{e}_N} \frac{1}{N^2}\mathbf{1}^T(\mathbf{K} - \mathbf{K}_{eca})\mathbf{1}. \tag{15}$$

The minimum value obtained as the solution to (15) is given by

$$\min_{\lambda_1,\mathbf{e}_1,\ldots,\lambda_N,\mathbf{e}_N} \frac{1}{N^2}\mathbf{1}^T(\mathbf{K} - \mathbf{K}_{eca})\mathbf{1} = \frac{1}{N^2}\sum_{j=k+1}^{N}\psi_j, \tag{16}$$

where ψ_j corresponds to the jth largest term of (11).

Since kernel ECA selects components from the kernel PCA space, out-of-sample data points represented by $\mathbf{\Phi}'$ are projected onto U_k based on (7) over the selected components, yielding $\mathbf{\Phi}'_{eca} = P_{U_k}\mathbf{\Phi}' = \mathbf{D}_k^{-\frac{1}{2}}\mathbf{E}_k^T\mathbf{K}'^T$.

3.3 Interpretation in Terms of Input Space

Note that kernel ECA outputs the new kernel matrix $\mathbf{K}_{eca}$. Hence, $-\log$ of the quantity $\hat{V}_k(p) = \frac{1}{N^2}\mathbf{1}^T\mathbf{K}_{eca}\mathbf{1}$ may be interpreted as the Renyi entropy estimator of some data set that we may denote by $\mathbf{x}'_1,\ldots\mathbf{x}'_N$. Since $\hat{V}_k(p)$ preserves as much as possible of the Renyi entropy estimate $\hat{V}(p)$ of the original data $\mathbf{x}_1,\ldots,\mathbf{x}_N$, we may associate $\mathbf{K}_{eca}$ with an *input space transformation* $\mathbf{x} \rightarrow \mathbf{x}'$ such that the entropy of $\mathbf{x}'_1,\ldots\mathbf{x}'_N$ is maximally similar to the entropy of $\mathbf{x}_1,\ldots,\mathbf{x}_N$. This yields an intuitive input space interpretation of kernel ECA.

To the best of our knowledge, such an input space interpretation is missing with regard to kernel PCA.

3.4 Interpretation in Terms of MDS

The inner products associated with the kernel ECA data transformation $\mathbf{\Phi}_{eca}$ is contained in the kernel feature space inner product matrix $\mathbf{K}_{eca} = \mathbf{\Phi}_{eca}^T\mathbf{\Phi}_{eca}$. By (15), we therefore see that kernel ECA is a form of inner product preserving MDS transformation, but not in the least-squares sense, as is the case with kernel PCA. In kernel ECA, inner products are preserved in an absolute sense, by the minimization of $\mathbf{1}^T(\mathbf{K} - \mathbf{K}_{eca})\mathbf{1}$ over the eigenvalues and eigenvectors.

3.5 Interpretation in Terms of Kernel Space Mean Vectors

Kernel ECA may also be interpreted as minimizing the difference between the squared euclidean length of the mean vector of the kernel feature space data and the squared euclidean length of the mean vector of the transformed data.

We have that

$$\begin{aligned}\hat{V}(p) &= \frac{1}{N^2}\mathbf{1}^T\mathbf{K}\mathbf{1} = \frac{1}{N^2}\mathbf{1}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{1}\\ &= \left\langle \frac{1}{N}\sum_{\mathbf{x}_t\in\mathcal{D}}\phi(\mathbf{x}_t), \frac{1}{N}\sum_{\mathbf{x}_{t'}\in\mathcal{D}}\phi(\mathbf{x}_{t'})\right\rangle\\ &= \|\mathbf{m}\|^2,\end{aligned} \tag{17}$$

where $\mathbf{m} = \frac{1}{N}\sum_{\mathbf{x}_t\in\mathcal{D}}\phi(\mathbf{x}_t)$ is the mean vector of the kernel feature space data set. Let $\mathbf{\Phi}_{eca} = [\phi_{eca}(\mathbf{x}_1),\ldots\phi_{eca}(\mathbf{x}_N)]$. Then,

$$\hat{V}_k(p) = \frac{1}{N^2}\mathbf{1}^T\mathbf{K}_{eca}\mathbf{1} = \|\mathbf{m}_{eca}\|^2, \tag{18}$$

where $\mathbf{m}_{eca} = \frac{1}{N}\sum_{\mathbf{x}_t\in\mathcal{D}}\phi_{eca}(\mathbf{x}_t)$ is the mean vector of the transformed data $\mathbf{\Phi}_{eca}$. Hence, (13) may alternatively be expressed as

$$\mathbf{\Phi}_{eca} = \mathbf{D}_k^{\frac{1}{2}}\mathbf{E}_k^T : \min_{\lambda_1,\mathbf{e}_1,\ldots,\lambda_N,\mathbf{e}_N}[\|\mathbf{m}\|^2 - \|\mathbf{m}_{eca}\|^2]. \tag{19}$$

Kernel ECA is therefore a data transformation which tries to preserve the squared euclidean length of the mean vector of the data in the kernel feature space.

3.6 No Centering in Kernel ECA

We have not assumed centered data throughout the analysis, in contrast to most expositions on kernel PCA. Centering does not, however, make sense in kernel ECA. Centering the kernel feature space data would mean that $\mathbf{m} = \frac{1}{N}\sum_{\mathbf{x}_t\in\mathcal{D}}\phi(\mathbf{x}_t) = 0$. Hence, $\hat{V}(p) = \|\mathbf{m}\|^2 = 0$ such that the Renyi entropy of the input data would be estimated to be $\hat{H}(p) = -\log\hat{V}(p) = \infty$. Centering the kernel feature space data corresponds to assuming infinite entropy input space data.

3.7 "Ideal" Case

We illustrate that kernel ECA and kernel PCA are equivalent in the "ideal" case. Assume that the data set $\mathcal{D} = \mathbf{x}_1,\ldots,\mathbf{x}_N$ consists of two clusters with N_1 and N_2 members, respectively, such that $\mathcal{D}_1 = \mathbf{x}_1,\ldots,\mathbf{x}_{N_1}$ and $\mathcal{D}_2 = \mathbf{x}_{N_1+1},\ldots,\mathbf{x}_N$, where $N_2 = N - N_1$. Assume that the clusters are maximally compact, $k_\sigma(\mathbf{x}_i,\mathbf{x}_{i'}) = 1$ for $\mathbf{x}_i,\mathbf{x}_{i'} \in \mathcal{D}_1(\mathcal{D}_2)$ and infinitely distant as far as the kernel function is concerned, that is, $k_\sigma(\mathbf{x}_i,\mathbf{x}_j) = 0$ for $\mathbf{x}_i \in \mathcal{D}_1$ and $\mathbf{x}_j \in \mathcal{D}_2$, making $\mathbf{K}$ block-diagonal

$$\mathbf{K} = \begin{bmatrix}\underline{\underline{\mathbf{1}}}^{N_1\times N_1} & \underline{\underline{\mathbf{0}}}^{N_1\times N_2}\\ \underline{\underline{\mathbf{0}}}^{N_2\times N_1} & \underline{\underline{\mathbf{1}}}^{N_2\times N_2}\end{bmatrix}, \tag{20}$$

where $\underline{\underline{\mathbf{1}}}^{M\times M}$ ($\underline{\underline{\mathbf{0}}}^{M\times M}$) is the $(M\times M)$ all ones (zero) matrix. The eigenvalues and eigenvectors of such a matrix will be the union of the eigenvalues and eigenvectors of the nonzero blocks. Since the blocks are rank one, there will be only one nonzero eigenvalue associated with each block, and the

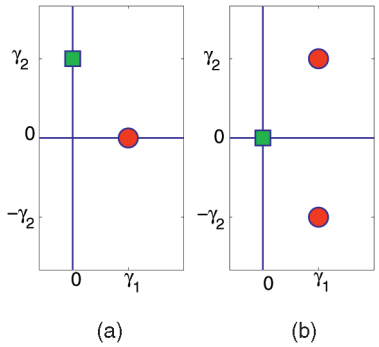

Fig. 1. (a) "Ideal" case data transformation. The groups (marked by different symbols) are mapped into points separated by a 90 degree angle. (b) Suboptimal case. We have $\gamma_1 = \sqrt{\frac{\lambda_1}{N_1}}$ and $\gamma_2 = \sqrt{\frac{\lambda_2}{N_2}}$ in general. In the leftmost panel, in particular, $\gamma_1 = \gamma_2 = 1$.

eigenvalues are in this case given by $\mathbf{D} = diag(N_1, N_2)$. The orthonormal eigenvectors are given by

$$\mathbf{E} = \begin{bmatrix} \frac{1}{\sqrt{N_1}}\mathbf{1}_{N_1} & \mathbf{0}_{N_1} \\ \mathbf{0}_{N_2} & \frac{1}{\sqrt{N_2}}\mathbf{1}_{N_2} \end{bmatrix}, \tag{21}$$

where the subscript denotes the length of the vectors. Now it is easily verified that $\hat{V}(p) = \frac{1}{N^2}\mathbf{1}^T\mathbf{K}\mathbf{1} = \frac{1}{N^2}(\sqrt{\lambda_1}\mathbf{e}_1^T\mathbf{1})^2 + (\sqrt{\lambda_2}\mathbf{e}_2^T\mathbf{1})^2 = \frac{1}{N^2}[N_1^2 + N_2^2]$.

We seek a two-dimensional data transformation. Since there are only two nonzero eigenvalues in this case, kernel ECA and kernel PCA are equivalent $\mathbf{\Phi}_{eca} = \mathbf{\Phi}_{pca} = \mathbf{D}^{\frac{1}{2}}\mathbf{E}^T$. In both cases, therefore, $\mathbf{x}_i \in \mathcal{D}_1 \rightarrow [1\ 0]^T$ and $\mathbf{x}_j \in \mathcal{D}_2 \rightarrow [0\ 1]^T$ (see also [12] for a related analysis). Fig. 1a shows that the clusters are mapped into orthogonal points on the unit sphere in the kernel space, thus spread by 90 degree *angles*. The eigenvectors carry in this case information about the cluster structure (cluster memberships can be assigned by a proper thresholding). This kind of "ideal" analysis has been used as a justification for the kernel PCA mapping based on the C top eigenvalues, hence, assuming C clusters. Such a situation will correspond to maximally concentrated eigenvalues of the kernel matrix, which is highly unrealistic in practice.

3.8 Suboptimal Case

Kernel ECA and kernel PCA may yield strikingly different data transformations. Assume that the two clusters are not compact, but still infinitely distant. A simplified example may correspond to a block-diagonal kernel matrix $\mathbf{K} = diag(\mathbf{K}', \mathbf{K}'')$, where $\mathbf{K}'$ corresponds to $\mathcal{D}_1$ and $\mathbf{K}''$ corresponds to $\mathcal{D}_2$ such that

$$\mathbf{K}' = \begin{bmatrix} \underline{\underline{\mathbf{1}}}^{\frac{N_1}{2}\times\frac{N_1}{2}} & (\underline{\underline{\mathbf{1}}} - \gamma)^{\frac{N_1}{2}\times\frac{N_1}{2}} \\ (\underline{\underline{\mathbf{1}}} - \gamma)^{\frac{N_1}{2}\times\frac{N_1}{2}} & \underline{\underline{\mathbf{1}}}^{\frac{N_1}{2}\times\frac{N_1}{2}} \end{bmatrix}, \tag{22}$$

and $\mathbf{K}'' = \beta\underline{\underline{\mathbf{1}}}^{N_2\times N_2}$. Here, γ is a relatively small number. We have $\mathbf{D}'' = \beta N_2$ and $\mathbf{E}'' = \frac{1}{\sqrt{N_2}}\mathbf{1}_{N_2}$. However, $\mathbf{K}'$ is a rank-two matrix such that $\mathbf{D}' = diag(N_1(1-\frac{\gamma}{2}), \frac{N_1}{2}\gamma)$ and

$$\mathbf{E}' = \begin{bmatrix} \frac{1}{\sqrt{N_1}}\mathbf{1}_{N_1} & -\frac{1}{\sqrt{N_1}}\mathbf{1}_{\frac{N_1}{2}} \\ & \frac{1}{\sqrt{N_1}}\mathbf{1}_{\frac{N_1}{2}} \end{bmatrix}. \tag{23}$$

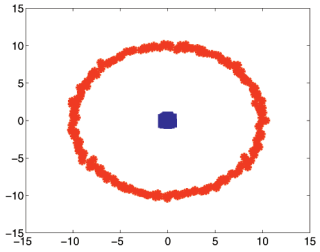

Fig. 2. Toy data set consisting of two clusters, used in Example 1.

If, for example, $\frac{N_1}{2} = N_2$ and $\gamma > \beta$, then $\lambda_1 = N_1(1-\frac{\gamma}{2})$ is the largest eigenvalue, the second largest eigenvalue is $\lambda_2 = \frac{N_1}{2}\gamma$, and the smallest eigenvalue is $\lambda_3 = \beta N_2$. Since β necessarily is quite small, $\mathcal{D}_2$ is not a very compact cluster.

Overall, we then have $\mathbf{D} = diag(N_1(1-\frac{\gamma}{2}), \frac{N_1}{2}\gamma, \beta N_2)$ and

$$\mathbf{E} = \begin{bmatrix} \frac{1}{\sqrt{N_1}}\mathbf{1}_{N_1} & -\frac{1}{\sqrt{N_1}}\mathbf{1}_{\frac{N_1}{2}} & \mathbf{0}_{N_1} \\ & \frac{1}{\sqrt{N_1}}\mathbf{1}_{\frac{N_1}{2}} & \\ \mathbf{0}_{N_2} & \mathbf{0}_{N_2} & \frac{1}{\sqrt{N_2}}\mathbf{1}_{N_2} \end{bmatrix}. \tag{24}$$

Kernel PCA, based on $\lambda_1, \mathbf{e}_1$ and $\lambda_2, \mathbf{e}_2$, then produces the two-dimensional transformed data illustrated in Fig. 1b. Note that the group $\mathcal{D}_2$ is collapsed into $(0,0)$. The group $\mathcal{D}_1$ is represented by two points. This data transformation does not resemble the "ideal" case discussed above and does not posses the same kind of angular structure.

Now we may compute the contribution to the Renyi entropy estimate associated with each of the three eigenvalues by $\hat{V}(p) = \frac{1}{N_2}\sum_{i=1}^{3}(\sqrt{\lambda_i}\mathbf{e}_i^T\mathbf{1})^2$, yielding

$$\begin{aligned} \hat{V}(p) &= \frac{1}{N^2}\left[\left(\sqrt{\lambda_1}\frac{N_1}{\sqrt{N_1}}\right)^2 + (\sqrt{\lambda_2}\ 0)^2 + \left(\sqrt{\lambda_3}\frac{N_2}{\sqrt{N_2}}\right)^2\right] \\ &= \frac{1}{N^2}\left[\left(1-\frac{\gamma}{2}\right)N_1^2 + \beta N_2^2\right]. \end{aligned} \tag{25}$$

Importantly, note that $\lambda_2, \mathbf{e}_2$ contributes *zero* to the entropy estimate. Therefore, kernel ECA is based on $\lambda_1, \mathbf{e}_1$ and $\lambda_3, \mathbf{e}_3$, and the resulting data transformation becomes exactly similar to the "ideal" case transformation, up to a constant in each eigenvalue, corresponding to Fig. 1a. We note the angular structure of the transformed data, where the groups are separated by a 90 degree angle.

3.9 Examples

In this section, we provide some examples of kernel ECA as a data transformation method. In all examples, we use a Gaussian kernel and we choose to use C eigenvalues and eigenvectors in the data transformation, assuming that the number of clusters C is known.

Example 1. In Fig. 2, a toy data set consisting of two clusters is shown (marked by different symbols for illustration purpose). The first 210 points correspond to the ring-shaped cluster while the last 500 points correspond to the tight Gaussian cluster in the middle. Being a low-dimensional data set, Silverman's rule [28] for Parzen window width selection may be employed, yielding $\sigma = 1.2$.

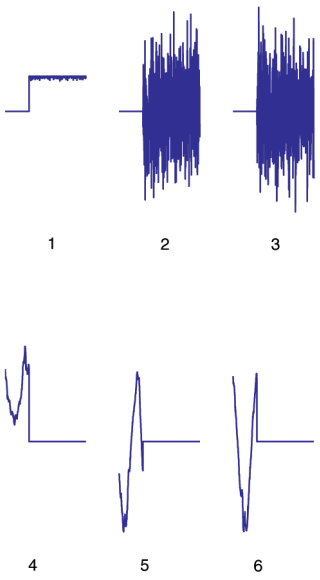

Fig. 3. First six eigenvectors in decreasing order of the eigenvalues.

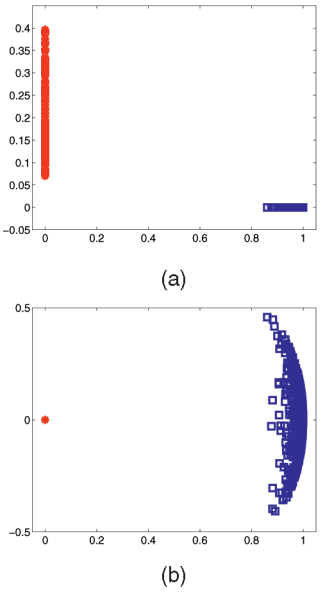

Fig. 4. Example 1: (a) kernel ECA and (b) kernel PCA.

Fig. 3 shows the first six eigenvectors of the kernel matrix, in decreasing order of the eigenvalues. For $\mathbf{e}_1$, the elements corresponding to the ring are basically zero. The remaining elements have a fairly constant positive value. This eigenvector thus carries information about the cluster structure by a proper thresholding. Moreover, $(\sqrt{\lambda_1}\mathbf{e}_1^T\mathbf{1})^2$ is the largest of all the entropy terms. However, $\mathbf{e}_2^T\mathbf{1} \approx \mathbf{e}_3^T\mathbf{1} \approx 0$ since the elements corresponding to the Gaussian now oscillate around zero, hence *not* contributing to the entropy. Note the similarity to the second eigenvector in the example analyzed in Section 3.8.

For $\mathbf{e}_4$, the elements corresponding to the Gaussian are basically zero. The remaining elements are all positive. This eigenvector also carries information about the cluster structure. In fact, $(\sqrt{\lambda_4}\mathbf{e}_4^T\mathbf{1})^2$ contributes second most to the entropy of the data. Furthermore, $\mathbf{e}_5^T\mathbf{1} \approx \mathbf{e}_6^T\mathbf{1} \approx 0$ since the elements corresponding to the ring oscillate around zero such that the entropy associated with these eigenvectors is negligible. Likewise, w.r.t. the remaining eigenvectors.

Hence, the kernel ECA transformation is based on $\lambda_1, \mathbf{e}_1$ and $\lambda_4, \mathbf{e}_4$, resulting in the data set is shown in Fig. 4a. Since the elements of $\mathbf{e}_1$ that correspond to the ring are zero, and the elements of $\mathbf{e}_4$ that correspond to the Gaussian are zero, the transformed data set has an angular structure in the sense that the clusters are distributed along lines in different angular directions with respect to the origin. This angular structure, in turn, reflects the fact that each of these eigenvectors carries information about the cluster structure of the data set.

The kernel PCA transformation, based on $\lambda_1, \mathbf{e}_1$ and $\lambda_2, \mathbf{e}_2$, by definition, is shown in Fig. 4b, yielding a totally different structure than kernel ECA. The ring-shaped cluster is collapsed into a cluster near $(0, 0)$. The Gaussian cluster is spread out on both the positive and negative sides of the horizontal axis. Note the similarity with Fig. 1b.

The data transformations depend on the kernel size. Kernel sizes below $\sigma = 0.5$ are, in this example, too small to capture the structure of the data. For $0.5 \leq \sigma \leq 1.4$, kernel ECA and kernel PCA are different, but they produce the same results for $\sigma > 1.4$ (results not shown).

Example 2. Fig. 5a shows a toy data set with three clusters, ordered from innermost ring to outermost ring with 63, 126, and 126 elements, respectively. Silverman's rule yields $\sigma = 3.7$. Fig. 5b shows the normalized eigenvalues λ_i in decreasing order as the vertical lines. The bars illustrate the normalized entropy terms $(\sqrt{\lambda_i}\mathbf{e}_i^T\mathbf{1})^2$ corresponding to λ_i. The three largest terms are the first, the second, and the ninth.

The kernel ECA transformed data set based on $\lambda_1, \mathbf{e}_1$, $\lambda_2, \mathbf{e}_2$, and $\lambda_9, \mathbf{e}_9$ is shown in Fig. 6a. The data set exhibits a

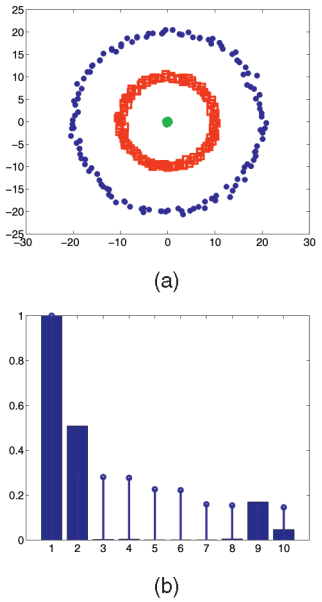

Fig. 5. (a) Toy data set used in Example 2. (b) Plot of entropy terms.

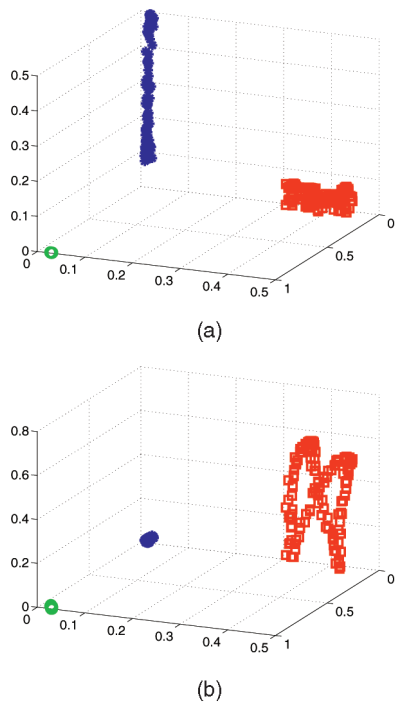

Fig. 6. Example 2: (a) kernel ECA and (b) kernel PCA.

clear angular structure. The mean vectors of the clusters are separated by 90 degree angles. The innermost ring is mapped to a dense representation. The two other rings are more spread out in their respective angular directions.

Kernel PCA, on the other hand, produces the data set shown in Fig. 6b. The same angular structure cannot be observed. The outermost ring has been collapsed near $(0, 0, 0)$. The mapping of the middle ring is not very compact.

Fig. 7 shows the first nine eigenvectors of the kernel matrix. Note that only $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_9$ provide information about the cluster structure, by having relatively constant nonzero elements corresponding to the innermost, middle, and outermost rings, respectively.

Fig. 8a shows the kernel matrix $\mathbf{K}$. It does not exhibit the blockwise structure one would expect in the "ideal" case. In

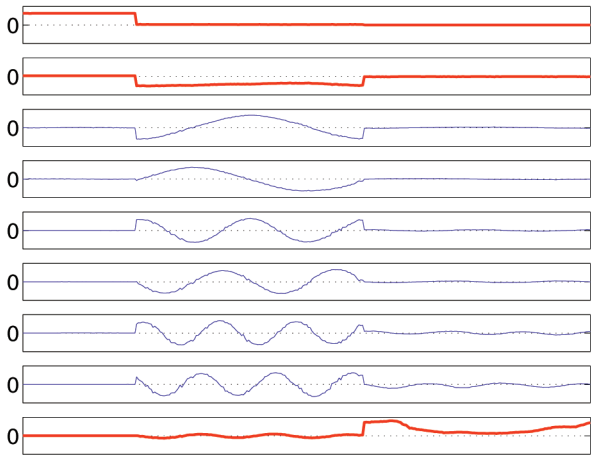

Fig. 7. Eigenvectors of $\mathbf{K}$ corresponding to the eight largest eigenvalues.

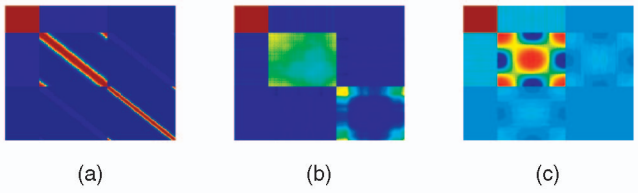

Fig. 8. Kernel matrices: (a) original data, (b) $\mathbf{K}_{eca}$, and (c) $\mathbf{K}_{pca}$.

(b), $\mathbf{K}_{eca}$ is shown. Note that a blockwise structure has appeared, in a sense approximating the "ideal" case. This is not the case for $\mathbf{K}_{pca}$, as shown in (c).

Fig. 9 illustrates the three entropy components used in kernel ECA over a range of kernel sizes in steps of 0.1. At no point are the entropy components based on the top three eigenvalues, hence kernel ECA and kernel PCA never coincide for this kernel range. This also holds for $\sigma > 10$.

Example 3. We illustrate kernel ECA using digits 6 and 9 of the USPS handwritten digits data set, obtained from the UCI repository [30]. The (16×16) gray-scale images are represented by 256-dimensional vectors of gray-scale values.

Since this is a high-dimensional data set, using Silverman's rule for kernel size selection is no longer reliable. We instead show the two-dimensional kernel ECA data transformation in Fig. 10a (a reduction by 254 dimensions) obtained over a range of kernel sizes. We let $2.5 \leq \sigma \leq 3.9$, in steps of 0.1 (top-left to bottom-right). The typical angle-based structure of kernel ECA is present for all kernel sizes, with the possible exception of $\sigma = 3.7$. As the kernel size increases, the clusters become less tight in the angular direction.

Fig. 10b shows the result using kernel PCA. One class is more or less collapsed into $(0, 0)$ for most kernel sizes. However, for $\sigma \geq 3.7$, kernel PCA and kernel ECA are equivalent, both being based on the two top eigenvalues.

For completeness, we also show in Fig. 10c the data transformation obtained using centered kernel PCA. The structure of the resulting transformation is different from both uncentered kernel PCA and kernel ECA.

4 Kernel ECA Spectral Clustering Algorithm

In this section, we demonstrate kernel ECA as an important component in a new spectral clustering algorithm. Kernel ECA often leads to a data set with a distinct angular structure,

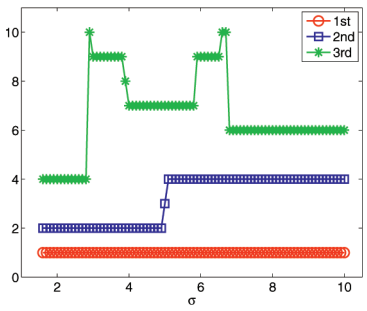

Fig. 9. Illustrating the three largest entropy terms as a function of σ.

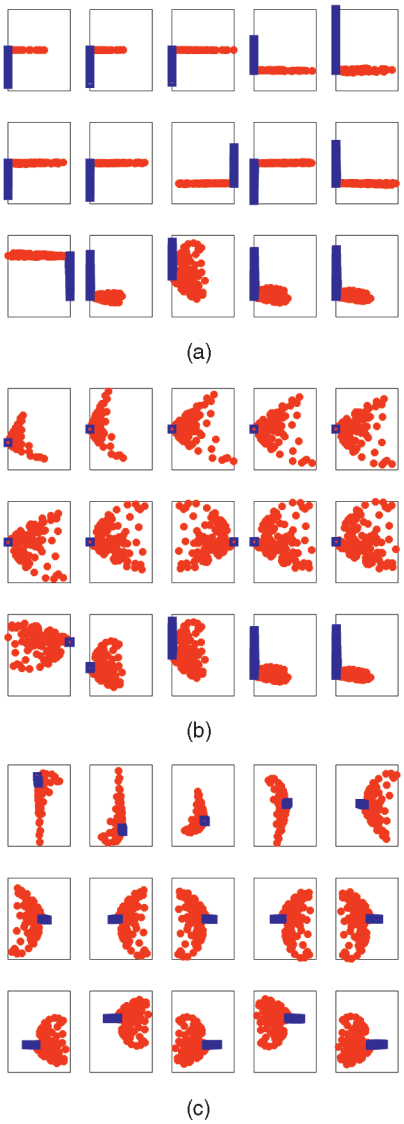

Fig. 10. Data transformations using kernel ECA and kernel PCA.

where clusters are distributed more or less in different angular directions with respect to the origin of the kernel feature space. A clustering cost function capable of capturing this angular structure thus seems like a reasonable choice.

It has previously been shown [31] that a Cauchy-Schwarz (CS) divergence measure between pdfs corresponds to a measure of the cosine of the angle between kernel feature space mean vectors. For example, the CS divergence between the pdf of the ith cluster $p_i(\mathbf{x})$ and the overall pdf of the data $p(\mathbf{x})$ is given by $D_{CS}(p_i, p) = -\log V_{CS}(p_i, p)$, where

$$V_{CS}(p_i, p) = \frac{\int p_i(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}}{\sqrt{\int p_i^2(\mathbf{x}) d\mathbf{x} \int p^2(\mathbf{x}) d\mathbf{x}}}. \tag{26}$$

Assume that the N_i data points $\mathbf{x}_n \in \mathcal{D}_i$ are generated from $p_i(\mathbf{x})$ corresponding to the cluster C_i. These data points are a subset of the N data points $\mathbf{x}_t \in \mathcal{D}$ generated from $p(\mathbf{x})$. Via Parzen windowing, we have

$$\hat{V}_{CS}(p_i, p) = \cos \angle(\mathbf{m}_i, \mathbf{m}). \tag{27}$$

Here, $\mathbf{m}_i = \frac{1}{N_i} \sum_{\mathbf{x}_n \in \mathcal{D}_i} \phi(\mathbf{x}_n)$ is the kernel feature space cluster mean vector corresponding to C_i, and $\mathbf{m} = \frac{1}{N} \sum_{\mathbf{x}_t \in \mathcal{D}} \phi(\mathbf{x}_t)$ is the overall kernel space mean vector of the data. The CS divergence basically measures the cosine of the angle between these mean vectors.

An angle-based clustering cost function in terms of the kernel feature space data set $\mathbf{\Phi}$ may be defined as

$$J(C_1, \ldots, C_C) = \sum_{i=1}^{C} N_i \cos \angle(\mathbf{m}_i, \mathbf{m}). \tag{28}$$

In [31], it was shown theoretically that this is the kernel C-means clustering objective using an *angular* distance measure, instead of a euclidean distance-based measure as used in, e.g., [6]. Here, we optimize this angle-based cost function over $C_1, \ldots, C_C$ using the kernel ECA transformed data $\mathbf{\Phi}_{eca}$ to represent $\mathbf{\Phi}$ because of its angular structure. The optimization procedure is simply the well-known C-means algorithm using angular distances in kernel ECA space and is guaranteed to converge to a local optimum for that reason. When put together, this constitutes a new spectral clustering algorithm, given by

Kernel ECA Spectral Clustering Algorithm:

1. obtain $\mathbf{\Phi}_{eca}$ by kernel ECA;
2. initialize means $\mathbf{m}_i,\ i = 1, \ldots, C$;
3. for all t:

$$\mathbf{x}_t \rightarrow C_i : \max_i \cos \angle(\phi_{eca}(\mathbf{x}_t), \mathbf{m}_i);$$

4. update mean vectors;
5. repeat steps 3 and 4 until convergence.

In summary, we assign a kernel feature space data point $\phi_{eca}(\mathbf{x}_t)$ to the cluster represented by the closest mean vector $\mathbf{m}_i$ in terms of angular distance.

In step 2, we initialize the mean vectors such that the two data points $\phi_{eca}(\mathbf{x}_t)$ and $\phi_{eca}(\mathbf{x}_{t'})$ having the property that $\min_{t,t'} \cos \angle(\phi_{eca}(\mathbf{x}_t), \phi_{eca}(\mathbf{x}_{t'}))$ are selected to represent $\mathbf{m}_1$ and $\mathbf{m}_2$. Furthermore, $\mathbf{m}_i$, $i > 2$, is represented by $\phi_{eca}(\mathbf{x}_{t''})$ such that $\min_{t''} \sum_{j=1}^{i-1} \cos \angle(\phi_{eca}(\mathbf{x}_{t''}), \mathbf{m}_j)$. This we have experienced to be a very robust initialization, providing convergence within a few iterations. Convergence is measured by the change in the CS cost function from one iteration to the next.

In the following, we show clustering results using both kernel ECA and (uncentered) kernel PCA for data representation. We also compare with the state-of-the-art NJW method [12]. All permutations between cluster labels and the "ground truth" labels are compared, and we report as clustering results the combination resulting in the smallest error. The number of clusters is assumed to be known, equaling the dimensionality of the transformed data. A Gaussian kernel is used in all experiments.

Clustering example 1. We show in Fig. 11 the clustering results in terms of the percentage of errors over a range of σ-values for the data set shown in Fig. 5a. Kernel ECA leads to much better results compared to kernel PCA. For kernel sizes greater than $\sigma \approx 2.5$, kernel ECA performs without errors in almost all cases, typically requiring only two iterations for convergence. Hence, the entropy components

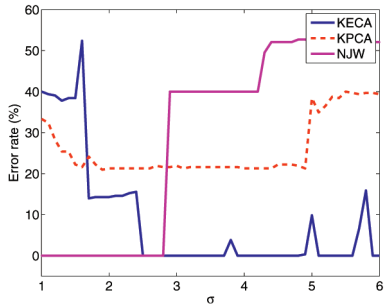

Fig. 11. Clustering results for ring-shaped clusters as a function of σ.

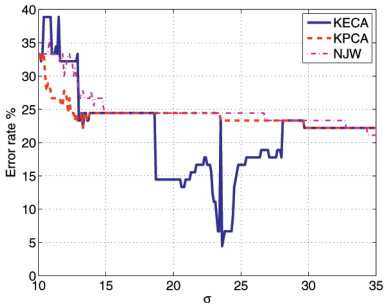

Fig. 13. Image segmentation clustering results as a function of σ.

(Fig. 9) over this range yield an angular structure. Compared to NJW, kernel ECA performs better for larger kernel sizes, while NJW performs better for smaller kernel sizes. Altogether, kernel ECA-based CS clustering clearly performs comparable in overall performance to NJW, if not better.

Clustering example 2. The `thyroid` data set is one of the two-class IDA benchmark data sets (see [11] and references therein). There are 100 realizations of this five-dimensional data set. In Fig. 12, we show the mean clustering results over a range of σ-values. For "large" kernel sizes, all methods perform equally well at an error rate of about 20 percent. Kernel PCA and NJW perform at their best in this region. However, kernel ECA performs even better for smaller kernel sizes, achieving the best error rate below 10 percent. Interestingly, kernel ECA uses $\lambda_1, \mathbf{e}_1$ and $\lambda_6, \mathbf{e}_6$ when achieving the best performance.

Clustering example 3. In this experiment, we use the `image segmentation` data set, here obtained from the UCI repository [30]. A database of seven images was hand segmented to create a classification for every pixel. A total of 19 features were created for each pixel, based on a (3×3) region surrounding the pixel.

We extract the classes "cement," "sky," and "grass." The clustering results are shown in Fig. 13, where σ is increased in steps of 0.1. Very small kernel sizes produce varying but not very good results for all of the methods. For $13.0 \leq \sigma \leq 18.6$, kernel ECA and kernel PCA are both based on the top three eigenvalues and eigenvectors yielding identical results in this range. In the region $18.7 \leq \sigma \leq 23.4$, entropy components three, two, and four are used. The clustering results for kernel ECA are better in this range. This is also the case for $23.6 \leq \sigma \leq 25.0$, where components three, one, and four are used, and for $25.1 \leq \sigma \leq 28.0$, where components two, one, and four are used. The best clustering result using kernel ECA corresponds to an error rate of 4 percent, compared to 20-25 percent errors for both the two other methods. Hence, kernel ECA and kernel PCA are different in the range $18.7 \leq \sigma \leq 28.0$. The only exception is for $\sigma = 23.5$, where, perhaps surprisingly, the top three eigenvalues and eigenvectors are used in both these two methods, producing equal clustering results.

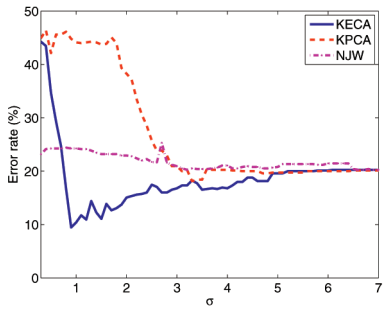

Fig. 12. Thyroid clustering results as a function of σ.

Clustering example 4. We have shown clustering results in terms of error rates over a range of kernel sizes. However, in practice, we either need an automated method for selecting the kernel size (such as Silverman's rule, which is unreliable in higher dimensions) or we need to select a limited range of kernel sizes for which the clustering is performed accompanied by a criterion for selecting the final clustering.

Shi and Malik [14] proposed using a kernel size in the order of 10-20 percent of the total range of the euclidean distances between the feature vectors. Using the mean range or the median range may be more robust choices. On the `image segmentation` data set, for example, 10-20 percent of the median range corresponds to $18.2 \leq \sigma \leq 36.4$.

In the following, we perform clustering experiments on some UCI data sets for 80 linearly spaced kernel sizes in the region corresponding to 10-20 percent of the median range of the euclidean distances of the feature vectors. The result corresponding to the minimum value of the CS cost function is used. For NJW and kernel PCA, the minimum value of the C-means cost function is used since we choose in this experiment to combine kernel PCA with C-means.

Table 1 shows that using kernel ECA, reasonable clustering results in terms of error rates are achieved on the `Iris`, `Wine`, and `Pen` (based on digits 0, 1, and 2) data sets. Also note that the corresponding entropy components are not in any case based on the top eigenvalues and eigenvectors. The data transformations associated with

TABLE 1
Clustering of Three UCI Data Sets with Three Clusters

	KECA	NJW	KPCA/C-means
Iris	Errors: 10.7% $\sigma = 0.36$ Comp.: 1,3,4	Errors: 31.3% $\sigma = 0.23 - 0.46$	Errors: 38.7% $\sigma = 0.41 - 0.46$
Wine	Errors: 5.1% $\sigma = 0.91$ Comp.: 1,3,4	Errors: 38.2% $\sigma = 0.61 - 0.99$	Errors: 32.6% $\sigma = 0.99$
Pen	Errors: 16.2% $\sigma = 0.98$ Comp.: 1,2,6	Errors: 27.4% $\sigma = 0.61 - 1.11$	Errors: 27.2% $\sigma = 1.11$

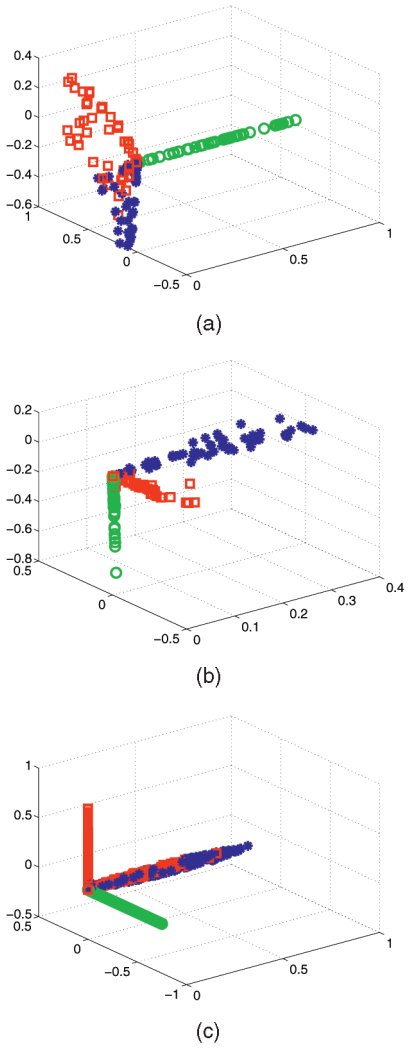

Fig. 14. Examples of best kernel ECA data transformations in terms of clustering errors.

these kernel ECA results are shown in Figs. 14a, 14b, and 14c. The data points are marked by symbols corresponding to the ground truth. The typical angular structure is present. Over the range of kernel sizes for which kernel ECA performs well, neither NJW nor kernel PCA using C-means performs very well, although the C-means cost is low. Both of these methods in fact achieve much better results for larger kernel sizes. However, the value of the C-means cost function is not smaller for better error rate results. For example, the NJW cost of the `Wine` clustering associated with $\sigma = 0.9$ is 0.051 (error rate 38.2 percent), while the cost associated with $\sigma = 5$ is 0.104 (error rate 2.8 percent).

Clustering example 5. The clustering experiments conducted so far have for the most part dealt with relatively clustered data, in the sense that we know in advance that some clustering structure should be present. In this experiment, we apply kernel ECA to a nonclustered data set, namely the 1,965 `Frey` faces, previously used to illustrate manifold learning [15]. These (20×28) face images are taken from sequential frames of a small video and differ in the way the faces are tilted, whether they are smiling or not, etc.

A kernel size corresponding to 15 percent of the total median range of the euclidean distances between the feature

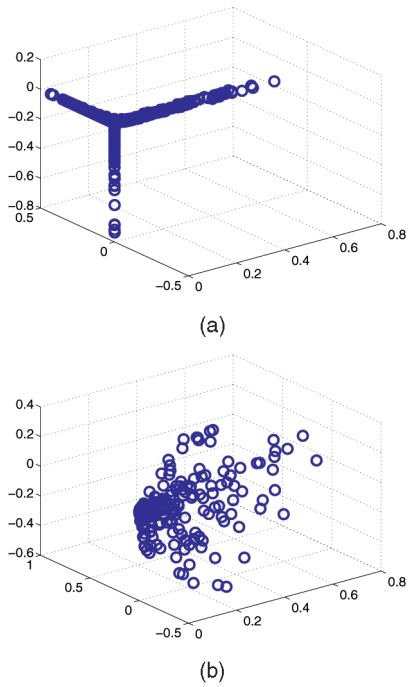

Fig. 15. "Frey faces": (a) kernel ECA data set and (b) kernel PCA data set.

vectors is used, yielding $\sigma = 130$. Kernel ECA seeks clusters, so we need to specify the number of clusters the method should look for. Specifying $C = 3$, for example, results in the data set shown in Fig. 15a. Also in this case, an angular structure is present, based on $\lambda_1, \mathbf{e}_1$, $\lambda_{10}, \mathbf{e}_{10}$, and $\lambda_4, \mathbf{e}_4$.

Kernel PCA produces the data set shown in Fig. 15b, resulting in a cloud-like collection of data points, with no clear structure present. In fact, even though λ_2 is the second largest eigenvalue, the entropy term corresponding to $\lambda_2, \mathbf{e}_2$ is only the 444th largest, and $\lambda_3, \mathbf{e}_3$ only corresponds to the 5th largest entropy term.

Due to space limitations, we cannot show all the kernel ECA clustered faces, assigned to groups consisting of 540, 807, and 618 data points, respectively. The panels above the vertical line in Fig. 16 show the 10 largest-norm (w.r.t. kernel ECA) faces in decreasing order as they are assigned

Fig. 16. Kernel ECA clustering result for "Frey faces."

TABLE 2
KECA Confusion Matrix for Internet Newsgroups

298	54
66	200

to clusters one, two, and three, respectively. The panels below the vertical line show randomly selected members of the three clusters. Cluster three, in particular, stands out, since all of the faces are smiling. The faces in cluster one seem for the most part to look straight into the camera, eyes showing. The eye region for the faces in cluster two seems darker overall, probably due to a shadowing effect, and some faces are tilted. It may appear as if kernel ECA has unveiled some cluster structure even in this apparently nonclustered data set, focusing on differences in the eye region, the mouth region, and with respect to the face tilt.

When clustering the kernel PCA data set, no reasonable results are obtained due to the lack of structure in the data. This is in fact also the case for NJW (results not shown).

Clustering example 6. In all examples and clustering experiments, we have had a data set available for them to create the kernel matrix using a Parzen window, or kernel function. It is the kernel matrix which is the actual input parameter to kernel ECA. In fact, given some kernel matrix, as far as kernel ECA is concerned, it corresponds to the Renyi entropy of some data set and need not even be psd, even though the connection to kernel PCA in that case is lost. It could therefore be interesting to see how kernel ECA reacts to a kernel matrix, which is not created directly from the Parzen window estimator, but which is assumed to correspond to a Parzen window Renyi entropy estimate. As a final clustering example along these lines, we provide kernel ECA with a kernel matrix generated as $\mathbf{K} = (\mathbf{X}^T\mathbf{X})^2$, where each column of $\mathbf{X}$ is a 100-dimensional sparse vector with elements one or zero, indicating whether or not a word from a "bag" consisting of 100 words is present in an article from the Newsgroups data set.[3] We concentrate on the groups `rec` and `sci`. The `rec` cluster is characterized by "recreational" words such as "car," "baseball," "games," "hockey," etc., while `sci` is characterized by "scientific" words such as "e-mail," "insurance," "ftp," "technology," etc. There are in this case 352 articles in `rec` and 266 articles in `sci`.

Kernel ECA selects $\lambda_4, \mathbf{e}_4$ and $\lambda_1, \mathbf{e}_1$ to be used. The resulting clustering yields the confusion matrix shown in Table 2. This corresponds to a rate of 80.6 percent correct clustering. Fig. 17 shows the difference between the distribution of the 54 articles incorrectly assigned to `sci` according to the "ground thruth" and the distribution of the 66 articles incorrectly assigned to `rec`. Note that scientific words like "e-mail," "insurance," "number," "technology," and "version" do appear more often in the 54 articles assigned by kernel ECA to `sci` than in the 66 articles assigned to `rec`, where recreational words like "hockey," "nhl," "players," "season," and "team" dominate. This suggests that the clustering performed by kernel ECA has revealed a reasonable structure of the data, even though this clustering corresponds to 19.6 percent error rate compared to the "ground truth." Clustering using kernel PCA corresponds to 28.6 percent errors, significantly worse than kernel ECA. NJW performs even worse in this case, with an error rate of 33.3 percent.

3. Sam Roweis' version of the Newsgroups data and the video images of Brendan Frey (Clustering Example 6) are used, downloaded from http://www.cs.toronto.edu/~roweis.

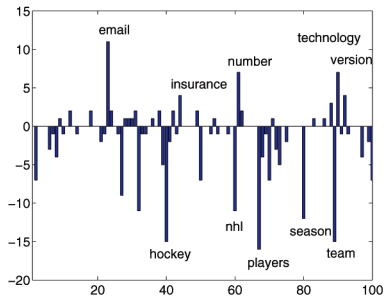

Fig. 17. Difference between distributions of incorrectly assigned articles.

4.1 Kernel ECA Revealing Cluster Structure

In the clustering experiments, the distinct angular structure of kernel ECE clearly is key to the positive clustering performance when using an angle-based cost function. Thorough theoretical investigations regarding this relationship between the Renyi entropy estimator and the angle structure are deferred to future work. We give some preliminary explanations here.

If we assume a clustered data set, the kernel matrix will be a permutation of a blockwise matrix and the eigenvectors will be the union of the eigenvectors of the permuted blocks (see, e.g., Sections 3.7 and 3.8). Hence, the eigenvectors corresponding to the permuted zero blocks will have elements close to zero. Furthermore, if one of the eigenvectors of a permuted nonzero block has strictly positive (negative) elements, then all of the other eigenvectors of that block will have elements that are both positive and negative or zero because of orthogonality, regardless of the associated eigenvalues.

If there exists a strictly positive (negative) eigenvector corresponding to a nonzero block, then the overall eigenvector of the kernel matrix will be composed of strictly positive (negative) elements corresponding to the cluster comprising the nonzero block and near-zero elements corresponding to the elements of the $C-1$ other clusters comprising the permuted zero blocks. Moreover, if C eigenvectors are selected by kernel ECA, where each of the eigenvectors has positive (negative) elements corresponding to one of the clusters, and near-zero elements corresponding to the other clusters, *then the angular structure of the transformed data set will appear*. Equivalently, this means that *the selected eigenvectors convey information about the clustering structure*. The reason kernel ECA selects eigenvectors with clustering structure can be seen from (11). An eigenvector $\mathbf{e}_i$ contributes to the entropy via $(\mathbf{e}_i^T\mathbf{1})^2$. If $\mathbf{e}_i$ happens to have such a clustering structure, then the inner product will be relatively large, and this eigenvector will likely be picked by kernel ECA. If, on the other hand, $\mathbf{e}_i$ is composed of another eigenvector from the same nonzero block and near-zero eigenvectors from the other blocks, then

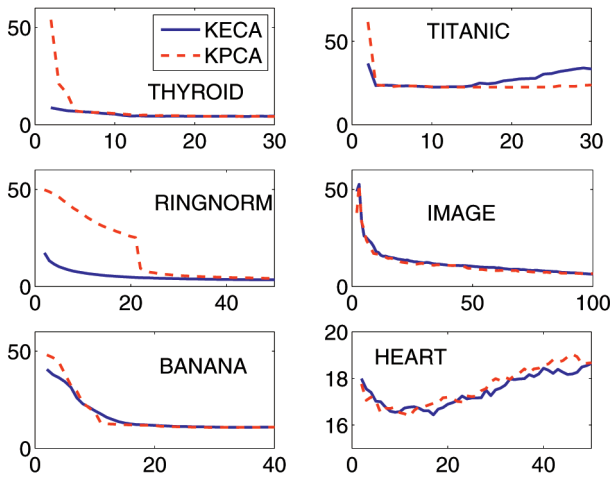

Fig. 18. Classification using kernel ECA and kernel PCA regressions.

the elements corresponding to the nonzero block will oscillate or vary around zero, and $\mathbf{e}_i^T\mathbf{1} \approx 0$. In that case, $\mathbf{e}_i$ will likely *not* be picked by kernel ECA because it contributes nothing to the entropy, as shown also in Section 3.8.

In order to illustrate that kernel ECA selects eigenvectors relevant to cluster, or class, structure also for very *nonclustered* data sets, we reproduce some of the experiments in [11], using kernel ECA regression instead of kernel PCA regression for supervised classification. In kernel ECA regression, we order the kernel PCA eigenvalues and eigenvectors according to their contribution to the Renyi entropy estimate of the input data. The two-class IDA data sets used here are normalized such that each feature is zero mean with unit standard deviation. In order to make a fair comparison, we use the same kernel size for kernels ECA and PCA and plot in Fig. 18 the mean error rate (vertical axis) over the resamples as a function of the number of eigenvalues and eigenvectors, i.e., the dimension (horizontal axis). For these normalized data sets, we use a kernel size equal to the median range of the euclidean feature vector distances. In several cases, the mean error rate drops more quickly using kernel ECA than kernel PCA. This illustrates that kernel ECA orders the eigenvalues and eigenvectors such that the top components reveal class structure better than the top kernel PCA components, which is the message we would like to convey here. On some data sets, the kernel ECA error rates are, in practice, equal to kernel PCA error rates. The *best* possible performance is for all data sets close to equal for the two methods. For completeness, we list the best performance obtained for kernel ECA and kernel PCA, respectively: `thyroid`: $4.2, 4.1 \pm 2.2, 2.1$, `titanic`: $22.6, 22.5 \pm 1.2, 1.2$, `ringnorm`: $3.1, 3.2 \pm 0.4, 0.5$, `image`: $2.9, 2.8 \pm 0.5, 0.4$ (15 resamples of 20), `banana`: $10.8, 10.8 \pm 0.5, 0.5$, `heart`: $16.4, 16.4 \pm 3.4, 3.5$. Perhaps surprisingly, this shows that, for these data sets, the choice of kernel size used here is adequate, providing results on par with the state of the art. In practice, a method for selecting an appropriate dimension is however needed.

5 Pattern Denoising

In this section, we illustrate briefly the use of kernel ECA as an alternative to kernel PCA in a supervised approach to denoising. The approach is supervised since clean training data are available. The method is an adaptation of Kwok and Tsang's [8] pre-image-based kernel PCA denoising algorithm. We do not use centered kernel PCA in our implementation.

Given clean training data, the kernel ECA subspace U_k may be found. A noisy out-of-sample data point $\mathbf{x}$ is projected onto U_k, resulting in $P_{U_k}\phi(\mathbf{x})$. If the subspace U_k represents the clean data appropriately, this operation will remove the noise. The next step is the computation of the preimage $\hat{\mathbf{x}}$ of $P_{U_k}\phi(\mathbf{x})$, yielding the input space denoised pattern. The method in [8] assumes that the preimage lies in the span of its n nearest neighbors. The nearest neighbors of $\hat{\mathbf{x}}$ will be equal to the kernel feature space nearest neighbors of $P_{U_k}\phi(\mathbf{x})$, which we denote by $\phi(\mathbf{x}_n) \in \mathcal{D}_n$. The algebraic method for finding the preimage needs euclidean distance constraints between $\hat{\mathbf{x}}$ and the neighbors $\mathbf{x}_n \in \mathcal{D}_n$. These constraints are obtained via euclidean distances in the kernel feature space, using an invertible kernel such as the Gaussian. The pseudocode for kernel ECA pattern denoising is summarized as:

- Based on noise-free training data $\mathbf{x}_1, \ldots, \mathbf{x}_N$, determine $\mathbf{K}$ and the kernel ECA projection $\mathbf{\Phi}_{eca} = \mathbf{D}_k^{\frac{1}{2}}\mathbf{E}_k^T$ onto the subspace U_k.
- For a noisy pattern $\mathbf{x}$, do the following:
  1. Project $\phi(\mathbf{x})$ onto U_k by $P_{U_k}\phi(\mathbf{x})$.
  2. Determine the feature space euclidean distances from $P_{U_k}\phi(\mathbf{x})$ to its n nearest neighbors $\phi(\mathbf{x}_n) \in \mathcal{D}_n$.
  3. Translate the feature space euclidean distances into input space euclidean distances.
  4. Find the preimage $\hat{\mathbf{x}}$ using the input space euclidean distances.

Projecting onto kernel ECA subspace. As explained in (7), the principal axis $\mathbf{u}_i$ in the kernel feature space defined by $\mathbf{K}$ is given by $\mathbf{u}_i = \lambda_i^{-\frac{1}{2}}\mathbf{\Phi}\mathbf{e}_i$, where $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N)]$. Moreover, the projection of $\phi(\mathbf{x})$ onto the direction $\mathbf{u}_i$ is given by $P_{\mathbf{u}_i}\phi(\mathbf{x}) = \mathbf{u}_i^T\phi(\mathbf{x}) = \lambda_i^{-\frac{1}{2}}\mathbf{e}_i^T\mathbf{k}_\mathbf{x}$, where $\mathbf{k}_\mathbf{x} = [k_\sigma(\mathbf{x}, \mathbf{x}_1), \ldots, k_\sigma(\mathbf{x}, \mathbf{x}_N)]^T$. The projection $P_{U_k}\phi(\mathbf{x})$ of $\phi(\mathbf{x})$ onto the subspace U_k spanned by the k principal axes as determined by kernel ECA can therefore be expressed as

$$P_{U_k}\phi(\mathbf{x}) = \sum_{i=1}^{k} P_{\mathbf{u}_i}\phi(\mathbf{x})\mathbf{u}_i = \mathbf{\Phi}\mathbf{M}\mathbf{k}_\mathbf{x}, \tag{29}$$

where $\mathbf{M} = \sum_{i=1}^{k} \frac{1}{\lambda_i}\mathbf{e}_i\mathbf{e}_i^T$ is symmetric.

Translating euclidean distances. We need the euclidean distances between $P_{U_k}\phi(\mathbf{x})$ and $\phi(\mathbf{x}_n) \in \mathcal{D}_n$. These are obtained by $\tilde{d}^2[P_{U_k}\phi(\mathbf{x}), \phi(\mathbf{x}_n)] = \|P_{U_k}\phi(\mathbf{x})\|^2 + \|\phi(\mathbf{x}_n)\|^2 - 2P_{U_k}^T\phi(\mathbf{x})\phi(\mathbf{x}_n)$, where $\|\phi(\mathbf{x}_n)\|^2 = K_{nn} = k_\sigma(\mathbf{x}_n, \mathbf{x}_n)$. Using the results above, we have

$$\|P_{U_k}\phi(\mathbf{x})\|^2 = (\mathbf{\Phi}\mathbf{M}\mathbf{k}_\mathbf{x})^T(\mathbf{\Phi}\mathbf{M}\mathbf{k}_\mathbf{x}) = \mathbf{k}_\mathbf{x}^T\mathbf{M}\mathbf{K}\mathbf{M}\mathbf{k}_\mathbf{x}, \tag{30}$$

since $\mathbf{M}^T = \mathbf{M}$ and $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{K}$. Moreover,

$$P_{U_k}^T\phi(\mathbf{x})\phi(\mathbf{x}_n) = (\mathbf{\Phi}\mathbf{M}\mathbf{k}_\mathbf{x})^T\phi(\mathbf{x}_n) = \mathbf{k}_\mathbf{x}^T\mathbf{M}\mathbf{k}_{\mathbf{x}_n}, \tag{31}$$

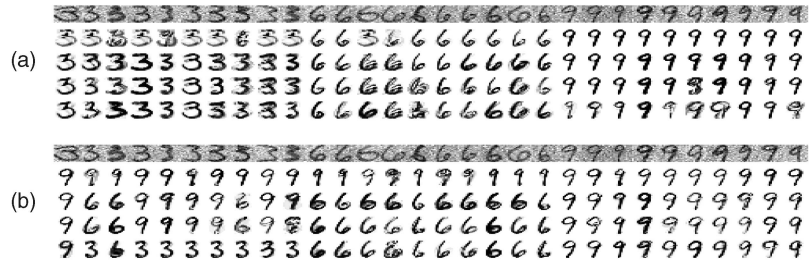

Fig. 19. Pattern denoising of USPS digits. (a) Kernel ECA. (b) Kernel PCA. In both cases, $\sigma = 3.0$, $k = 3, 8, 10, 15$ (top to bottom).

where $\mathbf{\Phi}^T\phi(\mathbf{x}_n) = \mathbf{k}_{\mathbf{x}_n} = [k_\sigma(\mathbf{x}_n, \mathbf{x}_1), \ldots, k_\sigma(\mathbf{x}_n, \mathbf{x}_N)]^T$. Hence, we obtain a formula for finding the euclidean distance as $\tilde{d}^2[P_{U_k}\phi(\mathbf{x}), \phi(\mathbf{x}_n)] = \mathbf{k}_\mathbf{x}^T\mathbf{M}\mathbf{K}\mathbf{M}\mathbf{k}_\mathbf{x} + K_{nn} - 2\mathbf{k}_\mathbf{x}^T\mathbf{M}\mathbf{k}_{\mathbf{x}_n}$.

We may translate the feature space euclidean distance $\tilde{d}^2[P_{U_k}\phi(\mathbf{x}), \phi(\mathbf{x}_n)]$ into an equivalent input space euclidean distance, which we may denote by d_n^2. Since we use a Gaussian kernel function, we have

$$\exp\left(-\frac{1}{2\sigma^2}d_n^2\right) = \frac{1}{2}\left[\|\phi(\mathbf{x})\|^2 - \tilde{d}^2\left[P_{U_k}\phi(\mathbf{x}), \phi(\mathbf{x}_n)\right] + \|\phi(\mathbf{x}_n)\|^2\right],$$

where $\|\phi(\mathbf{x})\|^2 = \|\phi(\mathbf{x}_n)\|^2 = K_{nn} = 1$. Hence, d_n^2 is given by

$$d_n^2 = -2\sigma^2 \log\frac{1}{2}\left[2 - \tilde{d}^2\left[P_{U_k}\phi(\mathbf{x}), \phi(\mathbf{x}_n)\right]\right]. \tag{32}$$

The Kwok and Tsang preimage method. Ideally, the preimage $\hat{\mathbf{x}}$ should obey the distance constraints d_i^2, $i = 1, \ldots, n$, which may be represented by a column vector $\mathbf{d}^2$. However, as pointed out by Kwok and Tsang [8] and others, in general there is no exact preimage in the input space, so a solution obeying these distance constraints may not exist. Hence, we must settle with an approximation. The neighbors are collected in the $(d \times n)$ matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$. These are centered at their centroid $\overline{\mathbf{x}}$ by a centering matrix $\mathbf{H}$. Assuming that the training patterns span a q-dimensional space, a singular value decomposition is performed $\mathbf{XH} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{U}\mathbf{Z}$, where $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n]$ is $(q \times n)$ and $\mathbf{d}_0^2 = [\|\mathbf{z}_1\|^2, \ldots, \|\mathbf{z}_n\|^2]^T$ represents the squared euclidean distance of each $\mathbf{x}_n \in \mathcal{D}_n$ to the origin. The euclidean distance between $\hat{\mathbf{x}}$ and $\mathbf{x}_n$ is required to resemble d_n^2 in a least-square sense. The pre-image is then obtained as $\hat{\mathbf{x}} = -\frac{1}{2}\mathbf{U}(\mathbf{Z}\mathbf{Z}^T)\mathbf{Z}(\mathbf{d}^2 - \mathbf{d}_0^2) + \overline{\mathbf{x}}$.

Note that the larger k is, the more the kernel ECA subspace U_k will resemble the kernel PCA subspace. Therefore, using kernel ECA in the denoising may be more beneficial as it is desirable to project onto a subspace of low dimensionality.

Denoising experiment. A clean training data set is available, consisting of the USPS digits 3, 6, and 9. Since we have three classes, we initially project onto a three-dimensional subspace in order to remove the noise. We use $\sigma = 3.0$ and a Gaussian kernel. Actually, for $\sigma < 5.1$, kernel ECA and kernel PCA never coincide for this data set, so the two methods project onto different subspaces. The top panel of Figs. 19a and 19b shows the noisy test digits (additive Gaussian noise). The second panel of each figure shows the kernel ECA and kernel PCA denoising results for $k = 3$, respectively. Note that for kernel PCA, the 9 class totally dominates the other two classes, so the denoising of, say, a digit 3 outputs a digit 9. This is not the case for kernel ECA, which performs better, although not perfectly, as can be seen by visual inspection. This shows that kernel ECA enhances the denoising in this particular case, compared to kernel PCA. The remaining panels show the denoising results for $k = 8$, 10, and 15, respectively. For $k = 8$ and 10, kernel ECA performs very well compared to kernel PCA, even though kernel PCA is improved compared to for $k = 3$. For $k = 15$ and higher, the dimension of the denoising subspace is sufficiently large to produce basically equal results for the two methods.

6 Conclusions

Kernel entropy component analysis has been proposed as a new spectral data transformation method. To the best of our knowledge, this is the only spectral method founded on information theory, since it is directly related to the Renyi entropy of the input space data set via a kernel-based Renyi entropy estimator that is expressed in terms of projections onto kernel feature space principal axes. The kernel ECA transformation is based on the most entropy preserving such axes. These do not, in general, correspond to the top eigenvalues of the kernel matrix. This is another key difference between kernel ECA and other methods, which are exclusively based on the top or bottom eigenvalues and eigenvectors of the relevant data matrixes. There is a close relationship between kernel ECA and uncentered kernel PCA. In the "ideal" case, these two methods have been shown to be equivalent. However, in general, kernel ECA may produce strikingly different data transformations than kernel PCA. Kernel ECA typically produces a data set with a distinct angular structure, reflecting the cluster structure of the data. A discussion relating to this structure has been provided, and classification results on IDA data sets have corroborated experimentally that kernel ECA seems to select eigenvectors that reveal cluster structure.

A new spectral clustering algorithm has been developed taking advantage of the kernel ECA angular structure. The algorithm is based on a Cauchy-Schwarz divergence measure between pdfs, which is equivalent to a measure of the cosine of the angle between kernel feature space mean vectors. Through a series of experiments, positive clustering results have been reported, comparable to the state-of-the-art Laplacian matrix-based method proposed in [12].

Finally, kernel ECA has been shown to represent an useful alternative to kernel PCA for pattern denoising

especially when projecting onto small subspaces, using the pre-image-based method introduced in [8].

Several challenges remain with respect to future research. We have in this exposition assumed a psd Parzen window, thus, obeying Mercer's conditions. This enables an analysis in terms of Mercer kernel spaces. However, the Renyi entropy estimator may be expressed in terms of the spectral properties of a kernel matrix even though the kernel used is indefinite, such as the Epanechnikov kernel. Kernel ECA based on indefinite kernels is a subject which needs further attention. Moreover, our analysis indicates that individual eigenvectors seem to be tuned in a sense to individual clusters. Analyzing further the relationship among the Renyi entropy, the eigenvectors, and the cluster structure is a task for future work. Furthermore, we note that the use of kernel ECA may be explored in any pattern analysis algorithm previously based on kernel PCA.

ACKNOWLEDGMENTS

The author wishes to acknowledge T. Eltoft, M. Girolami, and D. Erdogmus for their contributions at a very early stage of this work [20], and O. Storås for contributing in the last stage [21]. K.-R. Müller and M.L. Braun are thanked for useful discussions while hosting the author for 2008/2009 at TU Berlin. Finally, the associate editor O. Chapelle and the three anonymous reviewers are thanked for their helpful comments.

REFERENCES

[1] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification.* John Wiley & Sons, 2001.
[2] S. Theodoridis and K. Koutroumbas, *Pattern Recognition.* Academic Press, 1999.
[3] I.T. Jolliffe, *Principal Component Analysis.* Springer Verlag, 1986.
[4] H. Hotelling, "Analysis of a Complex of Statistical Variables into Principal Components," *J. Educational Psychology,* vol. 24, pp. 417-441, 1933.
[5] B. Schölkopf, A.J. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation,* vol. 10, pp. 1299-1319, 1998.
[6] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, "Spectral Relaxation for K-means Clustering," *Advances in Neural Information Processing Systems, 14,* pp. 1057-1064, MIT Press, 2002.
[7] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proc. Berkeley Symp. Math. Statistics and Probability,* pp. 281-297. 1967.
[8] J.T. Kwok and I.W. Tsang, "The Pre Image Problem in Kernel Methods," *IEEE Trans. Neural Networks,* vol. 15, no. 6, pp. 1517-1525, 2004.
[9] S. Mika, B. Schölkopf, A. Smola, K.R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and Denoising in Feature Space," *Advances in Neural Information Processing Systems, 11,* pp. 536-542, MIT Press, 1999.
[10] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A.J. Smola, "Input Space versus Feature Space in Kernel-Based Methods," *IEEE Trans. Neural Networks,* vol. 10, no. 5, pp. 1299-1319, 1999.
[11] M.L. Braun, J.M. Buhmann, and K.-R. Müller, "On Relevant Dimensions in Kernel Feature Spaces," *J. Machine Learning Research,* vol. 9, pp. 1875-1908, 2008.
[12] A.Y. Ng, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Advances in Neural Information Processing Systems, 14,* pp. 849-856, MIT Press, 2002.
[13] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Computation,* vol. 15, pp. 1373-1396, 2003.
[14] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.
[15] S. Roweis and L. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science,* vol. 290, pp. 2323-2326, 2000.
[16] J. Tenenbaum, V. de Silva, and J.C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science,* vol. 290, pp. 2319-2323, 2000.
[17] K.Q. Weinberger and L.K. Saul, "Unsupervised Learning of Image Manifolds by Semidefinite Programming," *Int'l J. Computer Vision,* vol. 70, no. 1, pp. 77-90, 2006.
[18] L.K. Saul, K.Q. Weinberger, J.H. Ham, F. Sha, and D.D. Lee, "Spectral Methods for Dimensionality Reduction," *Semisupervised Learning,* O. Chapelle, B. Schölkopf, and A. Zien, eds., chapter 1, MIT Press, 2005.
[19] C.J.C. Burges, "Geometric Methods for Feature Extraction and Dimensional Reduction," *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Researchers and Practitioners,* O. Maimon and L. Rokach, eds., chapter 4, Kluwer Academic Publishers, 2005.
[20] R. Jenssen, T. Eltoft, M. Girolami, and D. Erdogmus, "Kernel Maximum Entropy Data Transformation and an Enhanced Spectral Clustering Algorithm," *Advances in Neural Information Processing Systems 19,* pp. 633-640, MIT Press, 2007.
[21] R. Jenssen and O.K. Storås, "Kernel ECA Pre-Images for Pattern Denoising," *Proc. Scandinavian Conf. Image Analysis,* June 2009.
[22] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis.* Cambridge Univ. Press, 2004.
[23] J. Mercer, "Functions of Positive and Negative Type and Their Connection with the Theory of Integral Equations," *Philosophical Trans. Royal Soc. London,* vol. A, pp. 415-446, 1909.
[24] K.R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An Introduction to Kernel-Based Learning Algorithms," *IEEE Trans. Neural Networks,* vol. 12, no. 2, pp. 181-201, Mar. 2001.
[25] C.K.I. Williams, "On a Connection between Kernel PCA and Metric Multidimensional Scaling," *Machine Learning,* vol. 46, pp. 11-19, 2002.
[26] A. Renyi, "On Measures of Entropy and Information," *Selected Papers of Alfred Renyi,* vol. 2, pp. 565-580, Akademiai Kiado, 1976.
[27] E. Parzen, "On the Estimation of a Probability Density Function and the Mode," *The Annals of Math. Statistics,* vol. 32, pp. 1065-1076, 1962.
[28] B.W. Silverman, *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, 1986.
[29] M. Girolami, "Orthogonal Series Density Estimation and the Kernel Eigenvalue Problem," *Neural Computation,* vol. 14, no. 3, pp. 669-688, 2002.
[30] R. Murphy and D. Ada, "UCI Repository of Machine Learning Databases," technical report, Dept. of Computer Science, Univ. of California, Irvine, 1994.
[31] R. Jenssen and T. Eltoft, "A New Information Theoretic Analysis of Sum-of-Squared-Error Kernel Clustering," *Neurocomputing,* vol. 72, nos. 1-3, pp. 23-31, 2008.

Robert Jenssen received the degree of Dr Scient (PhD) in electrical engineering in 2005 from the University of Tromsø, Norway, where he is currently an associate professor in the Department of Physics and Technology. He was a visiting guest researcher at the University of Florida for the academic year 2002/2003 and March/April 2004 and at the Technical University of Berlin for the academic year 2008/2009. In his research, he has focused on developing an information-theoretic approach to machine learning based on Renyi entropy. This approach has strong connections to Mercer kernel methods and spectral clustering and dimensionality reduction methods. He received the Honorable Mention for the 2003 *Pattern Recognition* journal Best Paper Award, the 2005 IEEE ICASSP Outstanding Student Paper Award, and the 2007 University of Tromsø Young Investigator Award. He is on the IEEE Signal Processing Society Machine Learning for Signal Processing Technical Committee. He is a member of the IEEE and the IEEE Computer Society.