Pattern Recognition 44 (2011) 2630–2642

Contents lists available at ScienceDirect

Pattern Recognition

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

Multivariate online kernel density estimation with Gaussian kernels

Matej Kristan [a,b,*], Aleš Leonardis [a], Danijel Skočaj [a]

[a] Faculty of Computer and Information Science, University of Ljubljana, Slovenia
[b] Faculty of Electrical Engineering, University of Ljubljana, Slovenia

ARTICLE INFO

Article history:
Received 13 December 2010
Received in revised form
7 March 2011
Accepted 17 March 2011
Available online 8 April 2011

Keywords:
Online models
Probability density estimation
Kernel density estimation
Gaussian mixture models

ABSTRACT

We propose a novel approach to online estimation of probability density functions, which is based on kernel density estimation (KDE). The method maintains and updates a non-parametric model of the observed data, from which the KDE can be calculated. We propose an online bandwidth estimation approach and a compression/revitalization scheme which maintains the KDE's complexity low. We compare the proposed online KDE to the state-of-the-art approaches on examples of estimating stationary and non-stationary distributions, and on examples of classification. The results show that the online KDE outperforms or achieves a comparable performance to the state-of-the-art and produces models with a significantly lower complexity while allowing online adaptation.

1. Introduction

Many tasks in machine learning and pattern recognition require building models from observing sequences of data. In some cases all the data may be available in advance, but processing all data in a batch becomes computationally infeasible for large datasets. Furthermore, in many real-world scenarios all the data may not available in advance, or we even want to observe some process for an indefinite duration, while continually providing the best estimate of the model from the data observed so far. We therefore require online construction of models.

Traditionally, parametric models based on the Gaussian mixture models (GMM) [1] have been applied successfully to model the data in terms of their probability density functions (pdf). They typically require specifying the number of components (or an upper bound) in advance [1,2] or implementing some data-driven criteria for selection of the appropriate number of components (e.g., [3]). Improper choice of the number of components, however, may lead to models which fail to capture the complete structure of the underlying pdf. Non-parametric methods such as Parzen kernel density estimators (KDE) [4–6] alleviate this problem by treating each observation as a component in the mixture model. There have been several studies on how to efficiently estimate the bandwidth of each component (e.g., [7–12]) and to incorporate the measurement noise into the estimated bandwidths, e.g., [13]. Several researchers have recognized the drawbacks of using same bandwidth for all components. Namely, it is beneficial to apply small bandwidth to densely populated regions of the feature space, while larger bandwidths may be appropriate for sparsely populated regions. As result, non-stationary bandwidth estimators have been proposed, e.g., [11,14,15]. One drawback of the standard KDEs is that their complexity (number of components) increases linearly with the number of the observed data. To remedy this increase, methods have been proposed to reduce the number of components (compress) either to a predefined value [16,17], or to optimize some data-driven criteria [18–20]. Recently, Rubio and Lobato [17] applied the non-stationary bandwidths from [15] to the compressed distribution, and reported improved performance.

There have been several attempts to address the online estimation in the context of merging the non-parametric quality of the kernel density estimators with the Gaussian mixture models in online applications. Typically, authors constrain their models by imposing various assumptions about the estimated distributions. Arandjelović et al. [21] proposed a scheme for online adaptation of the Gaussian mixture model which can be updated by observing as little as a single data-point at a time. However, a strong restriction is made that data are temporally coherent in feature space, which prevents its use in general applications. Priebe and Marchette [22] proposed an online EM algorithm, called active mixtures, which allows adaptation from a single observation at a time, assumes the data are randomly sampled from the underlying distribution, and includes a heuristic for allocating new components, which makes it less sensitive to data ordering. Kenji et al. [23] adapted this approach to compression of data-streams by volume prototypes. Song et al. [24] aimed to alleviate the restrictions on data orderings by processing data in large blocks.

Deleclerq and Piater [25] assume each data-point is a Gaussian with a predefined covariance. All data are stored in the model and

* Corresponding author at: Faculty of Computer and Information Science, University of Ljubljana, Slovenia.
E-mail address: matej.kristan@fri.uni-lj.si (M. Kristan).
URL: http://www.vicos.uni-lj.si (M. Kristan).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.03.019

a heuristic is used to determine when a subset of the data (Gaussians) can be replaced by a single component. Han et al. [26] proposed an online approach inspired by the kernel density estimation in which each new observation is added to the model as the Gaussian kernel with a predefined bandwidth. The model's complexity is maintained through the assumption that the underlying probability density function can be approximated sufficiently well by retaining only its modes. This approach deteriorates in situations when the assumed predefined bandwidths of kernels are too restrictive, and when the distribution is locally non-Gaussian (skewed or heavy tailed distribution).

A positive side of imposing assumptions on the estimated distribution is that we can better constrain the problem of estimation and design efficient algorithms for the task at hand. A downside is that once the assumptions are violated, the algorithms will likely break down and deteriorate in performance. In this paper we therefore aim at an algorithm, which would be applicable to multivariate cases, would be minimally constrained by the assumptions and therefore efficiently tackle the difficulties of online estimation.

1.1. Our approach

We propose a new online kernel density estimator which is grounded in the following two key ideas. The first key idea is that unlike the related approaches, we do not attempt to build a model of the target distribution directly, but rather maintain a non-parametric model of the data itself in a form of a *sample distribution*—this model can then be used to calculate the kernel density estimate of the target distribution. The sample distribution is constructed by online clustering of the data-points. The second key idea is that we treat each new observation as a distribution in the form of a Dirac-delta function and we model the *sample distribution* by the mixture of Gaussian and Dirac-delta functions. During online operation the sample distribution is updated by each new observation in essentially the following three steps (Fig. 1a): (1) In the step 1, we update the sample model with the observed data-point. (2) In the step 2, the updated model is used to recalculate the optimal bandwidth for the KDE. (3) In the step 3, the sample distribution is refined and compressed. This step is required because, without compression, the number of components in our model would increase linearly with the observed data. However, it turns out that a valid compression at one point in time might become invalid later, when new data-points arrive. The result of these invalid compressions is that the model misses the structure of the underlying distribution and produces significantly over-smoothed estimates.

To allow the recovery from the early compression, we keep for each component in the sample distribution another model of the data that generated that component. This detailed model is in the form of a mixture model with at most two components (Fig. 1b). The rationale behind constraining the detailed model to two components is that this is the simplest detailed model that allows detection of early over-compressions. After the compression and refinement step, the KDE can be calculated as a convolution of the (compressed) sample distribution with the optimal kernel calculated at step 2.

Our main contribution is the new multivariate online kernel density estimator (oKDE), which enables construction of a multivariate probability density estimate by observing only a single sample at a time and which can automatically balance between its complexity and generalization of the observed data-points. In contrast to the standard bandwidth estimators, which require access to all observed data, we derive a method which can use a mixture model (sample distribution) instead and can be applied to multivariate problems. To enable a controlled compression of the sample distribution, we propose a compression scheme which maintains low distance between the KDE before and after compression. To this end, we propose an approximation to the multivariate Hellinger distance on mixtures of Gaussians. Since over-compressions occur during online estimation, we propose a revitalization scheme, which detects over-compressed components and refines them, thus allowing efficient adaptation.

The remainder of the paper is structured as follows. In Section 2, we define our model. In Section 3, we derive a rule for automatic bandwidth selection. We propose the compression scheme in Section 4, where we also address the problem of over-compression. The online KDE (oKDE) algorithm is presented in Section 5. In Section 6, we analyze the influence of parameters, data order, and the reconstructive and discriminative properties of the oKDE. We compare the oKDE to existing online and batch state-of-the-art algorithms on examples of estimating distributions and on classification examples. We conclude the paper in Section 7.

2. The model definition

As stated in the introduction, we aim at maintaining a (compressed) model of the observed data-points in the form of a distribution model, and use this model to calculate the KDE when required. We therefore start with the definition of the distribution of the data-points. Each separate data-point can be presented in a distribution as a single Dirac-delta function, with its probability mass concentrated at that data-point. Noting that the Dirac-delta can be generally written as a Gaussian with zero covariance, we define the model of (potentially compressed) d-dimensional data as an N-component Gaussian mixture model

$$p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\Sigma_{si}}(\mathbf{x}-\mathbf{x}_i), \tag{1}$$

where

$$\phi_{\Sigma}(\mathbf{x}-\boldsymbol{\mu}) = (2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2} e^{(-1/2(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))} \tag{2}$$

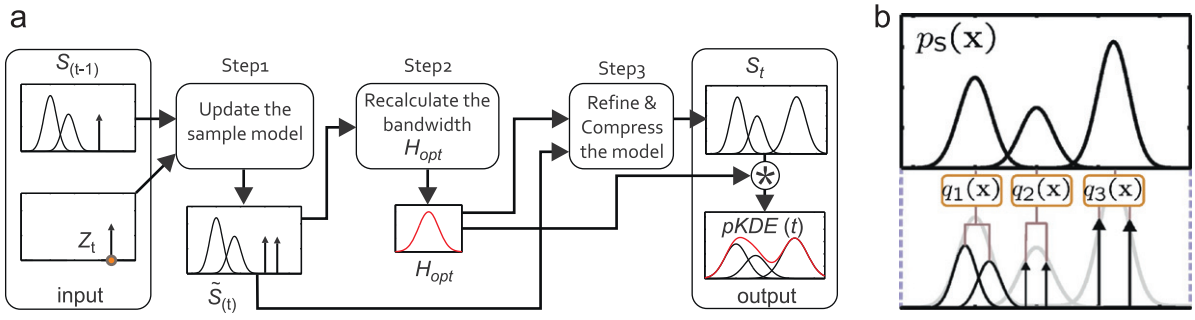

Fig. 1. A three-step summary of the online KDE iteration (a). The sample model $S_{(t-1)}$ is updated by a new observation $\mathbf{z}_t$ and compressed into a new sample model $S_{(t)}$. An illustration of the new sample model $S_{(t)}$ (sample distribution $p_s(\mathbf{x})$ along with its detailed model $\{q_i(\mathbf{x})\}_{i=1:4}$) is shown in (b).

is a Gaussian kernel centered at $\boldsymbol{\mu}$ with covariance matrix $\boldsymbol{\Sigma}$. We call $p_s(\mathbf{x})$ a *sample distribution* and a kernel density estimate (KDE) is defined as a convolution of $p_s(\mathbf{x})$ by a kernel with a covariance matrix (bandwidth) $\mathbf{H}$:

$$\hat{p}_{\mathrm{KDE}}(\mathbf{x}) = \phi_{\mathbf{H}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{H}+\boldsymbol{\Sigma}_{si}}(\mathbf{x}-\mathbf{x}_i). \tag{3}$$

To maintain a low complexity of the KDE during online operation, the sample distribution $p_s(\mathbf{x})$ is compressed from time to time by replacing clusters of components in the $p_s(\mathbf{x})$ by single Gaussian components. Details will be explained later in Section 4. As noted in the introduction, compressions at some point in time may later become invalid as new data arrive. To detect and recover from these early over-compressions, we keep an additional model of data for each component in the mixture model. We therefore define our *model of the observed samples* as

$$\mathbf{S}_{\mathrm{model}} = \{p_s(\mathbf{x}), \{q_i(\mathbf{x})\}_{i=1:N}\}, \tag{4}$$

where $p_s(\mathbf{x})$ is the *sample distribution* and $q_i(\mathbf{x})$ is a mixture model (with at most two components) for the i-th component in $p_s(\mathbf{x})$ (Fig. 1b). To obtain a KDE, we have to compute the optimal bandwidth from all the observed samples, which are now summarized in the sample model $p_s(\mathbf{x})$ (step 2 in Fig. 1a). In the following we propose a method for calculating this bandwidth.

3. Estimation of the bandwidth

If we retained (did not compress) all the observed samples in the sample model, then the sample distribution $p_s(\mathbf{x})$ would contain only components with zero covariances (i.e., $\boldsymbol{\Sigma}_{si} = \mathbf{0}$ for all i) and the KDE (3) would be defined as $\hat{p}_{\mathrm{KDE}}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{H}}(\mathbf{x}-\mathbf{x}_i)$. The goal of all KDE methods is to determine the kernel bandwidth $\mathbf{H}$ such that the distance between the $\hat{p}_{\mathrm{KDE}}(\mathbf{x})$ and the unknown pdf $p(\mathbf{x})$, that generated the data, is minimized. If the underlying distribution is known, a standard approach is to use the Kullback–Leibler divergence to measure the distance, however, in our case the $p(\mathbf{x})$ is unknown. In the KDE literature, a classical measure of closeness of the estimator $\hat{p}_{\mathrm{KDE}}(\mathbf{x})$ to the unknown underlying pdf is the *asymptotic mean integrated squared error* (AMISE), defined as ([11, pp. 95–98])

$$AMISE = (4\pi)^{-d/2} |\mathbf{H}|^{-1/2} N_\alpha^{-1} + \frac{1}{4} d^2 \int \mathrm{tr}^2\{\mathbf{H}\mathcal{G}_p(\mathbf{x})\}\, d\mathbf{x}, \tag{5}$$

where $\mathrm{tr}\{\cdot\}$ is the trace operator, $\mathcal{G}_p(\mathbf{x})$ is the Hessian of $p(\mathbf{x})$, and $N_\alpha = (\sum_{i=1}^{N} \alpha_i^2)^{-1}$. If we rewrite the bandwidth matrix in terms of scale β and structure $\mathbf{F}$, i.e., $\mathbf{H} = \beta^2\mathbf{F}$, and assume for now that $\mathbf{F}$ is known, then (5) is minimized at scale

$$\beta_{\mathrm{opt}} = [d(4\pi)^{d/2} N_\alpha R(p,\mathbf{F})]^{-1/(d+4)}, \tag{6}$$

where the term

$$R(p,\mathbf{F}) = \int \mathrm{tr}^2\{\mathbf{F}\mathcal{G}_p(\mathbf{x})\}\, d\mathbf{x} \tag{7}$$

is a functional of the second-order partial derivatives, $\mathcal{G}_p(\mathbf{x})$, of the unknown distribution $p(\mathbf{x})$. In principle, this functional could be estimated using the plug-in methods [11], however, these are usually numeric, iterative, assume we have access to *all the observed samples* and often suffer from numerical instabilities. In our case, we maintain only a (compressed) mixture model of the samples, and we require an approximation to the functional using this mixture model.

We first note that $R(p,\mathbf{F})$ can be written in terms of expectations of the derivatives $\psi_{\mathbf{r}} = \int p^{(\mathbf{r})}(\mathbf{x}) p(\mathbf{x})\, d\mathbf{x}$ (see, eg., [11]). We can then use the sample distribution $p_s(\mathbf{x})$ to obtain the following approximations:

$$p(\mathbf{x}) \approx p_s(\mathbf{x}), \quad p^{(\mathbf{r})}(\mathbf{x}) \approx p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x}), \tag{8}$$

where we approximate the derivative of $p(\mathbf{x})$, $p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x})$, through the following kernel density estimate

$$p_{\mathbf{G}}(\mathbf{x}) = \phi_{\mathbf{G}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi_{\Sigma_{gj}}(\mathbf{x}-\boldsymbol{\mu}_j). \tag{9}$$

The estimate $p_{\mathbf{G}}(\mathbf{x})$ plays a role of the so-called *pilot distribution* with covariance terms $\boldsymbol{\Sigma}_{gj} = \mathbf{G} + \boldsymbol{\Sigma}_{sj}$ and $\mathbf{G}$ is called the *pilot bandwidth*. Using the approximations in (8) we can approximate $R(p,\mathbf{F})$ by

$$\hat{R}(p,\mathbf{F},\mathbf{G}) = \int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_{\mathbf{G}}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\}. \tag{10}$$

Since $p_s(\mathbf{x})$ and $p_{\mathbf{G}}(\mathbf{x})$ are both the Gaussian mixture models, we can calculate the functional (10) using only matrix algebra:

$$\begin{aligned}\hat{R}(p,\mathbf{F},\mathbf{G}) = &\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j \phi_{\mathbf{A}_{ij}^{-1}}(\Delta_{ij}) \\ &\times [2\mathrm{tr}(\mathbf{F}^2\mathbf{A}_{ij}^2)[1-2m_{ij}] + \mathrm{tr}^2(\mathbf{F}\mathbf{A}_{ij})[1-m_{ij}]^2],\end{aligned} \tag{11}$$

where we have used the following definitions[1]:

$$\mathbf{A}_{ij} = (\boldsymbol{\Sigma}_{gi} + \boldsymbol{\Sigma}_{sj})^{-1}, \quad \Delta_{ij} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j, \quad m_{ij} = \Delta_{ij}^T \mathbf{A}_{ij} \Delta_{ij}. \tag{12}$$

Note that we still have to determine the pilot bandwidth $\mathbf{G}$ of $p_{\mathbf{G}}(\mathbf{x})$ and the structure $\mathbf{F}$ of the bandwidth matrix $\mathbf{H}$. We use the empirical covariance of the observed samples $\hat{\boldsymbol{\Sigma}}_{\mathrm{smp}}$ to approximate both.

We now resort to a practical assumption [11,27] that the *structure* of the bandwidth $\mathbf{H}$ can be reasonably well approximated by the structure of the covariance matrix of the observed samples, i.e., $\mathbf{F} = \hat{\boldsymbol{\Sigma}}_{\mathrm{smp}}$. We estimate the pilot bandwidth $\mathbf{G}$ by a multivariate normal-scale rule for the distribution's derivative ([11, p. 111]):

$$\mathbf{G} = \hat{\boldsymbol{\Sigma}}_{\mathrm{smp}} \left(\frac{4}{(d+2)N_\alpha}\right)^{2/(d+4)}. \tag{13}$$

4. Compression of the sample model

Having approximated the optimal bandwidth, the next step is to compress and refine the resulting model (step 3 in Fig. 1a). The objective of the compression is to approximate the original N-component sample distribution

$$p_s(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_{\boldsymbol{\Sigma}_{si}}(\mathbf{x}-\boldsymbol{\mu}_i) \tag{14}$$

by a M-component, $M < N$, equivalent $\hat{p}_s(\mathbf{x})$

$$\hat{p}_s(\mathbf{x}) = \sum_{j=1}^{M} \hat{w}_j \phi_{\hat{\boldsymbol{\Sigma}}_{sj}}(\mathbf{x}-\hat{\boldsymbol{\mu}}_j), \tag{15}$$

such that the resulting (compressed) KDE does not change significantly. Since a direct optimization (e.g., [28]) of the parameters in $\hat{p}_s(\mathbf{x})$ can be computationally prohibitive, and prone to slow convergence even for moderate number of dimensions, we resort to a clustering-based approach. The main idea is to identify clusters of components in $p_s(\mathbf{x})$, such that each cluster can be sufficiently well approximated by a single component in $\hat{p}_s(\mathbf{x})$. Let $\Xi(M) = \{\boldsymbol{\pi}_j\}_{j=1:M}$ be a collection of disjoint sets of indexes, which cluster $p_s(\mathbf{x})$ into M sub-mixtures. The sub-mixture corresponding to indexes $i \in \boldsymbol{\pi}_j$ is defined as

$$p_s(\mathbf{x}; \pi_j) = \sum_{i \in \pi_j} w_i \phi_{\boldsymbol{\Sigma}_{si}}(\mathbf{x}-\boldsymbol{\mu}_i), \tag{16}$$

[1] Derivation of (10) and (11) is rather laborious, and for convenience we have included the required derivations in the online supplemental material that is accessible from the authors homepage.

and is approximated by the j-th component $\hat{w}_j\phi_{\hat{\boldsymbol{\Sigma}}_{sj}}(\mathbf{x}-\hat{\boldsymbol{\mu}}_j)$ of $\hat{p}_s(\mathbf{x})$. The parameters of the j-th component are defined by matching the first two moments (mean and covariance) [29] of the sub-mixture:

$$\hat{w}_j=\sum_{i\in\pi(j)}w_i,\quad \hat{\boldsymbol{\mu}}_j=\hat{w}_j^{-1}\sum_{i\in\pi(j)}w_i\hat{\boldsymbol{\mu}}_i,$$

$$\hat{\Sigma}_j=\hat{w}_j^{-1}\sum_{i\in\pi(j)}w_i(\boldsymbol{\Sigma}_i+\boldsymbol{\mu}_i\boldsymbol{\mu}_i^T)-\hat{\boldsymbol{\mu}}_j\hat{\boldsymbol{\mu}}_j^T. \tag{17}$$

For better understanding, we illustrate in Fig. 2 an example in which components of a sample distribution $p_s(\mathbf{x})$ are clustered to form another (compressed) sample distribution $\hat{p}_s(\mathbf{x})$ with a smaller number of components. We also show the KDEs corresponding to the original and the compressed KDE. While the number of components in the sample distribution is reduced, the resulting KDE does not change significantly.

As indicated in Fig. 2 the compression seeks to identify the clustering assignment $\Xi(M)$, along with the minimal number of clusters M, such that the error induced by each cluster is sufficiently low, i.e., it does not exceed a prescribed threshold D_{th},

$$\hat{M}=\underset{M}{\operatorname{argmin}}E(\Xi(M))\quad \text{s.t. } E(\Xi(\hat{M}))\le D_{\text{th}}, \tag{18}$$

where we define $E(\Xi(M))$ as the largest local clustering error $\hat{E}(p_s(x;\pi_j),\mathbf{H}_{\text{opt}})$ under the clustering assignment $\Xi(M)$,

$$E(\Xi(M))=\max_{\pi_j\in\Xi(M)}\hat{E}(p_s(x;\pi_j),\mathbf{H}_{\text{opt}}). \tag{19}$$

The local clustering error $\hat{E}(p_s(x;\pi_j),\mathbf{H}_{\text{opt}})$ tells us the error induced under the KDE with bandwidth $\mathbf{H}_{\text{opt}}$, if the cluster $p_s(x;\pi_j)$ is approximated by a single Gaussian. We define this error next.

4.1. The local clustering error

Let $\mathbf{H}_{\text{opt}}$ be the current estimated bandwidth, and let $p_1(x)=p_s(\mathbf{x};\pi_j)$ be a cluster, a sub-mixture of the sample distribution defined by (16), which we want to approximate with a single Gaussian $p_0(x)$ according to (17). We define the local clustering error as the distance

$$\hat{E}(p_1(\mathbf{x}),\mathbf{H}_{\text{opt}})=D(p_{1KDE}(\mathbf{x}),p_{0KDE}(\mathbf{x})), \tag{20}$$

between the corresponding *KDEs*

$$p_{1\text{KDE}}(\mathbf{x})=p_1(\mathbf{x})*\phi_{\mathbf{H}_{\text{opt}}}(\mathbf{x}),$$

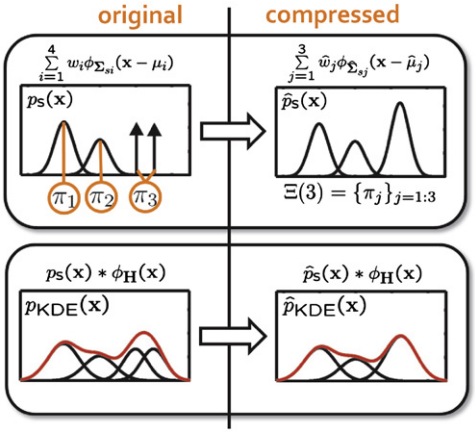

Fig. 2. The images illustrate a compression of a four-component sample distribution $p_s(\mathbf{x})$ into a three-component counterpart $\hat{p}_s(\mathbf{x})$ using the clustering assignment $\Xi(3)=\{\pi_j\}_{j=1:3}$. The left and right columns show the sample distribution (upper row) and the corresponding KDE (lower row) before and after compression, respectively.

$$p_{0\text{KDE}}(\mathbf{x})=p_0(\mathbf{x})*\phi_{\mathbf{H}_{\text{opt}}}(\mathbf{x}).$$

In particular, we can quantify the distance between distributions using the Hellinger distance [30], which is defined as

$$D^2(p_{1\text{KDE}}(\mathbf{x}),p_{0\text{KDE}}(\mathbf{x}))\triangleq\frac{1}{2}\int(p_{1\text{KDE}}(\mathbf{x})^{1/2}-p_{0\text{KDE}}(\mathbf{x})^{1/2})^2\,d\mathbf{x}. \tag{21}$$

Note that while the Hellinger distance is a proper metric between distributions and is bounded to interval [0,1] (see, e.g., [30]), it cannot be calculated analytically for the mixture models. We therefore calculate its approximation using the *unscented transform* [31] (see Appendix A).

4.2. Compression by hierarchical error minimization

In principle, the global optimization of (18) would require evaluation of all possible cluster assignments $\Xi(M)$ for the number of clusters M ranging from one to N, which becomes quickly computationally prohibitive. A significant reduction in complexity of the search can be obtained by a *hierarchical* approach to cluster discovery. Similar approaches have been previously successfully applied for a controlled data compression with the Gaussian mixture models to a predefined number of clusters [16,32].

In our implementation, the hierarchical clustering proceeds as follows. We start by splitting the entire sample distribution $p_s(\mathbf{x})$ into two sub-mixtures using Goldberger's [16] K-means algorithm for mixture models[2] with $K=2$. Each sub-mixture is approximated by a single Gaussian and the sub-mixture which yields the largest local error $\hat{E}(p_s(\mathbf{x};\pi_j),H_{\text{opt}})$ is further splitted into two sub-mixtures. This process is recursively continued until the largest local error is sufficiently small and the condition $E(\Xi(M))\le D_{\text{th}}$ in (18) fulfilled. This approach generates a binary tree with $\hat{M}$ leafs among the components of the sample distribution $p_s(\mathbf{x})$, in which the leafs of the tree represent the clustering assignments $\Xi(\hat{M})=\{\boldsymbol{\pi}_j\}_{j=1:M}$. Once the clustering $\Xi(\hat{M})$ is found, the compressed sample distribution $\hat{p}_s(\mathbf{x})$ (15) is calculated using (16) and (17).

Recall that we keep track of a detailed model for each component in the sample distribution (see, e.g., Fig. 1b). The detailed model $\hat{q}_j(\mathbf{x})$ of the j-th component in the compressed model $\hat{p}_s(\mathbf{x})$ is calculated as follows. If the set π_j contains only a single index, i.e., $\pi_j=\{i\}$, then the j-th component of the compressed sample distribution is equal to the i-th component in the original sample distribution and therefore the detailed model remains unchanged, i.e., $\hat{q}_j(\mathbf{x})=q_i(\mathbf{x})$. On the other hand, if π_j contains multiple indexes, then the detailed models corresponding to these indexes are first concatenated into a single *extended* mixture model

$$\hat{q}_{j\text{ext}}(x)=\sum_{i\in\pi_j}q_i(\mathbf{x}). \tag{22}$$

Then the required two-component detailed model $\hat{q}_j(\mathbf{x})$ is generated by splitting $\hat{q}_{j\text{ext}}(\mathbf{x})$ into two sub-mixtures again using Goldberger's K-means and each sub-mixture is approximated by a single Gaussian using (17). Note that the detailed model is constrained to at most two components, since this is the least complex model which enables detection of the early over-compressions as discussed next.

4.3. Revitalizing the sample distribution

The compression identifies and compresses those clusters of components whose compression does not introduce a significant error into the KDE with the bandwidth $\mathbf{H}_{\text{opt}}$ estimated at the time of compression. However, during online operation, new samples

[2] Note that to avoid the singularities associated with the components in the sample distribution with zero covariance, the K-means algorithm for the Gaussian mixtures is carried out on the corresponding KDE.

arrive, the sample distribution and $\mathbf{H}_{\text{opt}}$ change, and so does the estimated KDE. Therefore, a compression which may be valid for a KDE at some point in time, may become invalid later on.

The over-compression can be detected through inspection of the *detailed model* of each component in the sample distribution $p_s(\mathbf{x})$. The local clustering error $\hat{E}(q_i(\mathbf{x}),\mathbf{H}_{\text{opt}})$ (19) of each component $w_i\phi_{\mathbf{\Sigma}_{si}}(\mathbf{x})$ in the sample distribution can be evaluated against its detailed model $q_i(\mathbf{x})$ to verify whether the global clustering error from (18) exceeds the threshold D_{th}. Those components in $p_s(\mathbf{x})$ for which $\hat{E}(q_i(\mathbf{x}),\mathbf{H}_{\text{opt}}) > D_{\text{th}}$ are removed from the sample distribution and replaced by the two components of their detailed model. A detailed model is then created for each of the new components. For example, let $w_i\phi_{\Sigma_i}(\mathbf{x}-\boldsymbol{\mu}_i)$ be one of the new components. If the determinant of $\mathbf{\Sigma}_i$ is zero, then this component corresponds to a single data-point and therefore its detailed model is just the component itself. However, in case the determinant is nonzero, it means that the component has been generated through clustering of several detailed models in the previous compression steps. Its detailed model is then initialized by splitting $\phi_{\mathbf{\Sigma}_i}(\mathbf{x}-\boldsymbol{\mu}_i)$ along its principal axis into a two-component mixture, whose first two moments match those of the original component. More precisely, let $\mathbf{UDU}^{\mathrm{T}} = \mathbf{\Sigma}_i$ be a singular value decomposition of $\mathbf{\Sigma}_i$ with eigenvalues and eigenvectors ordered by the descending eigenvalues. Then the new detailed mixture model is defined as

$$q_i(\mathbf{x}) = \sum_{k=1}^{2} \alpha_k \phi_{\mathbf{\Sigma}_k}(\mathbf{x}-\boldsymbol{\mu}_k),\ \mu_1 = \mathbf{FM}+\boldsymbol{\mu}_i,\quad \mu_2 = \mathbf{FM}-\boldsymbol{\mu}_i,$$
$$\mathbf{\Sigma}_k = \mathbf{FCF}^{\mathrm{T}},\quad \alpha_k = \tfrac{1}{2}w_i, \tag{23}$$

where $\mathbf{C} = diag([3/4,\mathbf{0}_{1\times(d-1)}])$, $\mathbf{M} = [0.5,\mathbf{0}_{1\times(d-1)}]^{\mathrm{T}}$, $\mathbf{F} = \mathbf{U}\sqrt{\mathbf{D}}$ and $\mathbf{0}_{1\times(d-1)}$ are all-zeros row vector of length $(d-1)$. The entire compression procedure along with the revitalization routine is summarized in Algorithm 1.

Algorithm 1. Compression of the sample model.

Input:

$\hat{\mathbf{S}}_{\text{model}} = \{\tilde{p}_s(\mathbf{x}),\{\tilde{q}_i(\mathbf{x})\}_{i=1:\tilde{M}}\}\ldots$ the $\tilde{M}$-component sample model.

$\mathbf{H}_{\text{opt}}\ldots$ the current optimal bandwidth.

$D_{\text{th}}\ldots$ the maximal allowed local compression error.

Output:

$\hat{\mathbf{S}}_{\text{model}} = \{\hat{p}_s(\mathbf{x}),\{\hat{q}_j(\mathbf{x})\}_{j=1:M}\},\ldots$ the compressed M-component sample model.

Procedure:

1: Revitalize each i-th component in $\tilde{p}_s(\mathbf{x})$ for which $\hat{E}(\tilde{q}_i(\mathbf{x}),\mathbf{H}_{\text{opt}}) > D_{\text{th}}$ according to Section 4.3 and replace the sample model with the N-component revitalized model: $\mathbf{S}_{\text{model}}xlarr;\{p_s(\mathbf{x}),\{q_i(\mathbf{x})\}_{i=1:N}\}$.

2: Initialize the cluster set:
$\Xi(M) = \{\pi_1\}$, $\pi_1 = \{1,\ldots,N\}$, $M=1$

3: **while** $D_{\text{th}} < \max_{\pi_j\in\Xi(M)}\hat{E}(p_s(\mathbf{x};\pi_j))$ **do**

4: Select the cluster with the maximum local error:
$\pi_j = \text{argmax}_{\pi_j\in\Xi(M)}\hat{E}(p_s(\mathbf{x};\pi_j))$

5: Split the sub-mixture $p_s(\mathbf{x};\pi_j)$ into two sets using the Goldberger's K-means: $\pi_j \longrightarrow \{\pi_{j1},\pi_{j2}\}$.

6: Update the cluster set:
$M \leftarrow M+1$, $\Xi(M) \leftarrow \{\{\Xi(M)\backslash\pi_j\},\pi_{j1},\pi_{j2}\}$.

7: **end while**

8: Regroup the components of $p_s(\mathbf{x})$ according to clustering $\Xi(M)$ and construct the compressed sample model $\hat{p}_s(\mathbf{x})$.

9: For each j-th component in $\hat{p}_s(\mathbf{x})$ create its detailed model $\hat{q}_j(\mathbf{x})$ from the reference detailed models $\{q_i(\mathbf{x})\}_{i=1:N}$ according to the clustering $\Xi(M)$.

5. Online kernel density estimation

In this section, we describe an iteration of the online kernel density estimation, whose steps were outlined in the introduction (Fig. 1a). Let us denote the model of the samples observed up to time-step $(t-1)$ as

$$\mathbf{S}_{\text{model}(t-1)} = \{p_{s(t-1)}(\mathbf{x}),\{q_{i(t-1)}(\mathbf{x})\}_{i=1:M_{t-1}}\}, \tag{24}$$

where $p_{s(t-1)}$ is a M_{t-1}-component sample distribution,

$$p_{s(t-1)}(\mathbf{x}) = \sum_{i=1}^{M_{t-1}} \alpha_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x}-\boldsymbol{\mu}_i). \tag{25}$$

Let N_{t-1} denote the *effective number* of observations[3] up to time-step $(t-1)$, let $N_{\alpha(t-1)}$ be the value of the parameter for bandwidth calculation (N_α in Eq. (6)) and let f be a forgetting factor.[4]

At time-step t we observe a sample $\mathbf{x}_t$ and re-estimate the sample model $\mathbf{S}_{\text{model}(t)} = \{p_{s(t)}(\mathbf{x}),\{q_{i(t)}(\mathbf{x})\}_{i=1:M_t}\}$ (and hence the KDE) in the following steps.

Step 1: *Update the sample model.* The effective number of observed samples is augmented using the forgetting factor, $N_t = N_{t-1}f+1$ and the weight $w_0 = N_t^{-1}$ of the new sample is computed. The sample distribution is updated by the new observation[5] as

$$\tilde{p}_{s(t)}(\mathbf{x}) = (1-w_0)p_{s(t-1)}(\mathbf{x}) + w_0\phi_{\mathbf{0}}(\mathbf{x}-\mathbf{x}_t). \tag{26}$$

The detailed model $\tilde{q}_{\tilde{M}_t}(\mathbf{x}) = \phi_{\mathbf{0}}(\mathbf{x}-\mathbf{x}_t)$ corresponding to $\mathbf{x}_t$ is added to the existing set of detailed models

$$\{\tilde{q}_{i(t)}(\mathbf{x})\}_{i=1:\tilde{M}_t} = \{\{q_i(\mathbf{x})\}_{i=1:M_{t-1}},\tilde{q}_{\tilde{M}_t}(\mathbf{x})\}. \tag{27}$$

Thus yielding an updated sample model

$$\tilde{\mathbf{S}}_{\text{model}(t)} = \{\tilde{p}_{s(t)}(\mathbf{x}),\{\tilde{q}_{i(t)}(\mathbf{x})\}_{i=1:\tilde{M}_t}\}. \tag{28}$$

Step 2: *Re-estimate the bandwidth.* The empirical covariance of the observed samples $\hat{\mathbf{\Sigma}}_{\text{smp}}$ is calculated by approximating $\tilde{p}_{s(t)}(\mathbf{x})$ by a single Gaussian using the moment matching (17) and the parameter for bandwidth calculation is updated as $N_{\alpha t} = (N_{\alpha(t-1)}^{-1}(1-w_0)^2 + w_0^2)^{-1}$. The new optimal bandwidth is then approximated according to Section 3 as

$$\mathbf{H}_t = \mathbf{F}[d(4\pi)^{d/2}N_{\alpha t}\hat{R}(p,\mathbf{F},\mathbf{G})]^{-2/(d+4)} \tag{29}$$

with $\mathbf{F} = \hat{\mathbf{\Sigma}}_{\text{smp}}$, $\mathbf{G} = \hat{\mathbf{\Sigma}}_{\text{smp}}(4/(2+d)N_{\alpha t})^{2/(d+4)}$, and with the functional $\hat{R}(p,\mathbf{F},\mathbf{G})$ calculated according to (11).

Step 3: *Refine and compress the model.* After the current bandwidth $\mathbf{H}_t$ has been calculated, the sample model $\tilde{\mathbf{S}}_{\text{model}(t)}$ is refined and compressed, using Algorithm 1, into

$$\mathbf{S}_{\text{model}(t)} = \{p_{s(t)}(\mathbf{x}),\{q_{i(t)}(\mathbf{x})\}_{i=1:M_t}\}. \tag{30}$$

In our implementation, the compression is called after some threshold on number of components M_{thc} has been exceeded. Note that this threshold does not determine the number of components in the final model, but rather influences the *frequency* at which the compression is called. To avoid too frequent calls to compression, the threshold is also allowed to vary during the online operation using a simple hysteresis rule: If the number of components M_t still exceeds M_{thc} after the compression, then the threshold increases $M_{\text{thc}} \leftarrow 1.5M_{\text{thc}}$, otherwise, if $M_t < \frac{1}{2}M_{\text{thc}}$, then it decreases $M_{\text{thc}} \leftarrow 0.6M_{\text{thc}}$.

Recalculate the KDE: After the three steps of the online update have finished, the sample distribution is a M_t-component mixture

[3] Note that if there is no forgetting involved then all the data-points are equally important, regardless of the order in which they arrive. In this case the effective number of observations is just the number of all observed samples.

[4] When estimating stationary distribution, this factor is 1 and less than one when estimating a non-stationary distribution.

[5] Note that $\hat{(\cdot)}$ denotes the updated model before the compression.

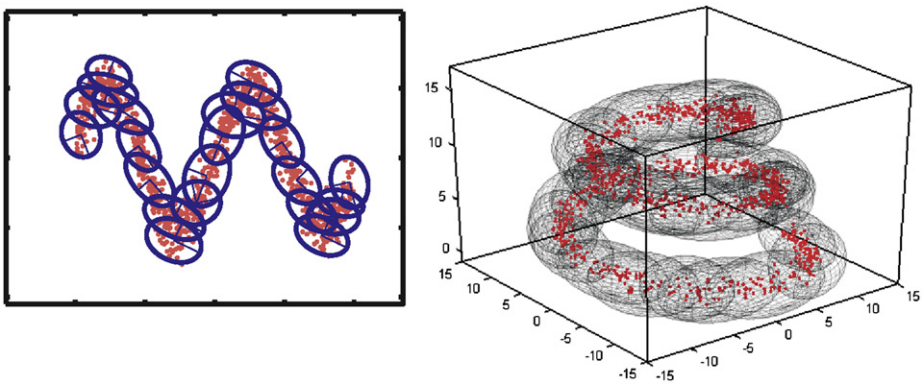

Fig. 3. The 1000 sampled data-points along with the estimated distribution using $\text{oKDE}_{0.02}$ for the sinusoidal (left) and spiral (right) distribution, respectively.

model

$$p_{s(t)}(\mathbf{x}) = \sum_{i=1}^{M_t} \alpha_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x}-\boldsymbol{\mu}_t), \tag{31}$$

and the current KDE $p_{\text{KDE}t}(\mathbf{x})$ is calculated from the sample distribution according to (3).

6. Experimental study

First, we have compared the oKDE's performance to the related online and batch methods in density estimation on artificial datasets (Section 6.1) and real datasets (Section 6.2). Then we have analyzed the oKDE's performance on publicly available classification problems (Section 6.3). Finally, in Sections 6.4 and 6.5 we have analyzed the effects of compression and the revitalization scheme in the oKDE. All experiments were performed on a standard 2 GHz CPU, 2 GB RAM PC in Matlab.

6.1. Density estimation on artificial datasets

This experiment was divided into two parts. In the first part we analyzed estimation of two stationary distributions and in the second part we analyzed estimation of a non-stationary distribution. We have compared the performance of the oKDE with an online method called the adaptive mixtures [22][6] (AM) and with three state-of-the-art batch KDE methods: Hall et al. [8] plug-in Murillo et al. [12] cross-validation (CV) and Girolami et al. [18] reduced-set-density estimator (RSDE).

The first stationary distribution was a two-dimensional sinusoidal distribution defined by

$$\mathbf{x} = [a, \sin(3a)+w]^T,\ a = 4(t-1/2), \quad w \sim \phi_{\sigma_w}(\cdot) \tag{32}$$

with $\sigma_w = 0.2^2$. The second distribution was a three-dimensional spiral distribution defined by

$$\mathbf{x} = \left[(13-\tfrac{1}{2}t)\cos(t), -(13-\tfrac{1}{2}t)\sin(t), t\right]^T + \mathbf{w},\ \mathbf{w} \sim \phi_{\mathbf{\Sigma}_\mathbf{w}}(\cdot), \quad t \sim \mathcal{U}(0,14), \tag{33}$$

where $\mathbf{\Sigma}_\mathbf{w} = \text{diag}\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\}$, and $\mathcal{U}(1,14)$ is a uniform distribution constrained to interval [0,14]. Both distributions are visualized in Fig. 3. A set of 10 thousand test samples was generated from the distribution—the first 10 samples were used for initialization and the rest were used one at a time with the oKDE and AM to approximate the underlying distribution. The reconstructive performance of the models was evaluated by the average negative log-likelihood of additionally sampled 20 000 observations. This experiment was repeated 10 times. In the following we will use notation $\text{oKDE}_{D_{\text{th}}}$, where $(\cdot)_{D_{\text{th}}}$ denotes the used compression threshold value D_{th}. An example of the estimated distributions with $\text{oKDE}_{0.02}$ after observing a 1000 samples is shown in Fig. 3. The results are summarized in Table 1.

Among the batch approaches, the CV outperformed the other two batch methods in accuracy. While the advantage of the batch methods is that they optimize their parameters by having access to all the data-points, they become increasingly slow with increasing the number of data-points and can also run into computer's memory constraints. Indeed this was the case for the particular implementations of the batch RSDE and Hall, which prohibited estimation for very large sets of samples. This is indicated in Table 1 by the symbol "/". For smaller number of samples, the batch CV outperformed the online methods in accuracy, however, at a cost of severely increased model complexity. For example, after observing 1000 data-points, the complexity of CV model was 1000 components, while the complexity of the $oKDE_{0.01}$ was less than 5% of that. For increasing the number of samples over (approximately 6000), the oKDE started to outperform the CV also in terms of accuracy, while maintaining the number of components low. While the number of components from the 6000 to the 10 000 sample increased by 4000 in CV model, this increase was less than 10 for the $\text{oKDE}_{0.01}$. All online methods on average produced models with a smaller number of components than the batch RSDE. In all experiments, the $\text{oKDE}_{0.01}$ and $\text{oKDE}_{0.02}$ consistently outperformed the online AM model in accuracy and on average in complexity.

In the second part of the experiment we applied the $\text{oKDE}_{0.02}$ to approximate a non-stationary distribution, which was a mixture of two distributions,

$$p_0(\mathbf{x},t) = w(t)p_1(\mathbf{x}) + (1-w(t))p_2(\mathbf{x}), \tag{34}$$

whose mixing weight $w(t)$ was changing with time-steps t. The first distribution, $p_1(\mathbf{x})$, was a heavily skewed distribution (Fig. 4a), while the second, $p_2(\mathbf{x})$, was a mixture of a uniform and a skewed distribution (Fig. 4c). The weight was set to $w(t)=1$ for the first 1000 samples and it gradually decreased to zero for the next 7000 samples at rate $w(t) = w(t-1)0.995$. Thus $p_0(\mathbf{x},t)$ transited from pure $p_1(\mathbf{x})$ to pure $p_2(\mathbf{x})$. Figs. 4(a–c) show the distribution at time-steps $t=1$, 1800 and 8000, respectively.

Since the distribution was non-stationary, the forgetting factor in $\text{oKDE}_{0.02}$ was set to $f=0.999$. Thus the effective sample size converges to $N_t=1000$. The $\text{oKDE}_{0.02}$ and AM were initialized from the first three samples and the rest were added one at a

[6] The *adaptive mixtures* (AM) approach [22] is essentially an online EM algorithm for the Gaussian mixture models with an automatic component-allocation heuristic.

Table 1
Average negative log-likelihood $-\mathcal{L}$ (along with $\pm$one standard deviation) w.r.t. the number of observed samples. We also show the model complexity (number of components) in the parentheses. The symbol "/" indicates that the estimator could not be calculated due to memory limitations.

Batch	Results for the two-dimensional sinus distribution (Fig. 3a)				
	50 samples	1000 samples	6000 samples	8000 samples	10 000 samples
CV	$\mathbf{1.7}\pm\mathbf{0.1}(50\pm0)$	$\mathbf{1.3}\pm\mathbf{0.0}(1\times10^3\pm0)$	$1.3\pm0.0(6\times10^3\pm0)$	$\mathbf{1.3}\pm\mathbf{0.0}(8\times10^3\pm0)$	$\mathbf{1.4}\pm\mathbf{0.0}(10^4\pm0)$
Hall	$2.4\pm0.0(50\pm0)$	$2.0\pm0.0(1\times10^3\pm0)$	$1.8\pm0.0(6\times10^3\pm0)$	$1.8\pm0.0(8\times10^3\pm0)$	/
RSDE	$2.0\pm0.2(23\pm4)$	$1.3\pm0.0(380\pm11)$	$\mathbf{1.3}\pm\mathbf{0.0}(2.2\times10^3\pm47)$	/	/
Online	50 samples	1000 samples	6000 samples	8000 samples	10 000 samples
AM	$2.2\pm0.1(11\pm3)$	$1.7\pm0.1(22\pm4)$	$1.5\pm0.1(38\pm6)$	$1.5\pm0.1(41\pm6)$	$1.5\pm0.1(43\pm6)$
$oKDE_{0.01}$	$\mathbf{2.0}\pm\mathbf{0.07}(16\pm2)$	$\mathbf{1.5}\pm\mathbf{0.0}(34\pm2)$	$\mathbf{1.3}\pm\mathbf{0.0}(48\pm3)$	$\mathbf{1.3}\pm\mathbf{0.0}(51\pm3)$	$\mathbf{1.3}\pm\mathbf{0.0}(54\pm3)$
$oKDE_{0.02}$	$2.0\pm0.1(12\pm2)$	$1.5\pm0.0(21\pm2)$	$1.5\pm0.0(28\pm2)$	$1.4\pm0.0(29\pm3)$	$1.4\pm0.0(30\pm2)$
$oKDE_{0.04}$	$2.0\pm0.1(8\pm1)$	$1.7\pm0.0(11\pm2)$	$1.6\pm0.0(13\pm2)$	$1.6\pm0.0(14\pm2)$	$1.6\pm0.0(14\pm2)$
$oKDE_{0.05}$	$2.0\pm0.1(6\pm1)$	$1.7\pm0.0(9\pm1)$	$1.7\pm0.0(10\pm1)$	$1.6\pm0.0(11\pm1)$	$1.6\pm0.0(11\pm2)$
Batch	Results for the three-dimensional spiral distribution (Fig. 3b)				
	50 samples	1000 samples	6000 samples	8000 samples	10 000 samples
CV	$\mathbf{8.1}\pm\mathbf{0.3}(50\pm0)$	$\mathbf{6.6}\pm\mathbf{0.0}(10^3\pm0)$	$\mathbf{6.5}\pm\mathbf{0.0}(6\times10^3\pm0)$	$\mathbf{6.5}\pm\mathbf{0.0}(8\times10^3\pm0)$	$\mathbf{6.5}\pm\mathbf{0.0}(10^4\pm0)$
Hall	$8.1\pm0.2(50\pm0)$	$6.7\pm0.0(10^3\pm0)$	$6.7\pm0.0(6\times10^3\pm0)$	/	/
RSDE	$8.6\pm0.7(30\pm8)$	$6.7\pm0.0(516\pm83)$	$6.6\pm0.0(2.6\times10^3\pm17)$	/	
Online	50 samples	1000 samples	6000 samples	8000 samples	10 000 samples
AM	$8.6\pm0.16(18\pm3)$	$6.9\pm0.1(42\pm4)$	$6.6\pm0.1(64\pm6)$	$6.6\pm0.1(68\pm6)$	$6.6\pm0.1(72\pm6)$
$oKDE_{0.01}$	$\mathbf{8.0}\pm\mathbf{0.2}(24\pm2)$	$\mathbf{6.8}\pm\mathbf{0.0}(46\pm2)$	$\mathbf{6.5}\pm\mathbf{0.0}(51\pm2)$	$\mathbf{6.5}\pm\mathbf{0.0}(52\pm2)$	$\mathbf{6.5}\pm\mathbf{0.0}(52\pm1)$
$oKDE_{0.02}$	$8.0\pm0.3(19\pm2)$	$6.8\pm0.0(29\pm1)$	$6.5\pm0.0(32\pm1)$	$6.5\pm0.0(33\pm1)$	$6.5\pm0.0(33\pm1)$
$oKDE_{0.04}$	$8.1\pm0.3(14\pm1)$	$6.8\pm0.0(20\pm1)$	$6.7\pm0.0(23\pm2)$	$6.7\pm0.0(24\pm1)$	$6.6\pm0.0(24\pm1)$
$oKDE_{0.05}$	$8.1\pm0.3(13\pm1)$	$6.9\pm0.0(18\pm1)$	$6.8\pm0.0(21\pm1)$	$6.7\pm0.0(21\pm1)$	$6.7\pm0.0(21\pm1)$

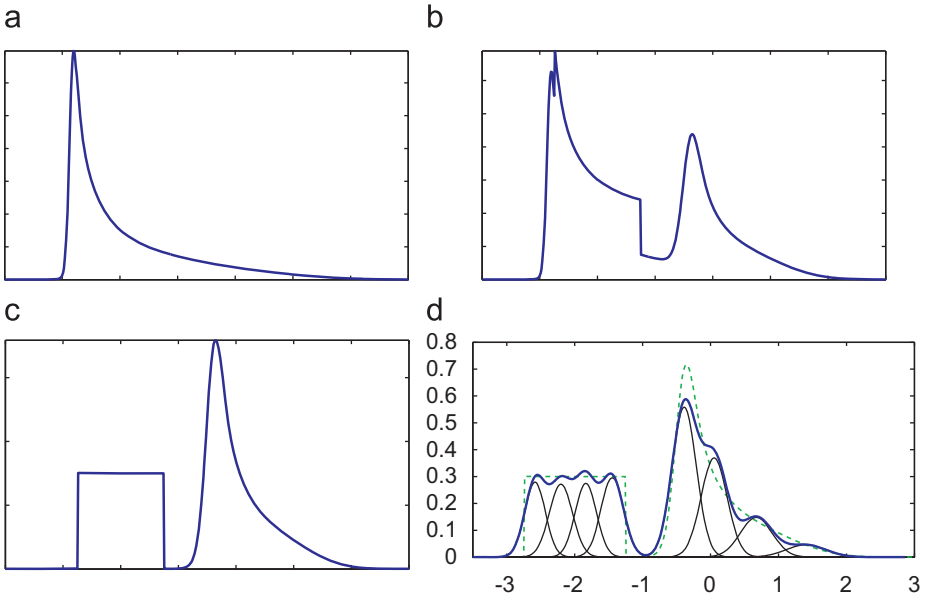

Fig. 4. The phases of the non-stationary distribution at $t=1$ (a), $t=1800$ (b) and $t=8000$ (c), and the estimated distribution with oKDE after observing the 8000th sample (d). The components of the oKDE model in (d) are depicted by solid thin lines and the oKDE is shown in solid thick line, while the reference distribution is depicted by a dashed green line. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

time. The quality of estimation at time-step t was measured by the L_1 distance between the current estimate and $p_0(\mathbf{x},t)$. The performance of the $oKDE_{0.02}$ was compared to AM and the two sliding-window batch methods, CV and Hall batch KDEs, which have been computed using the last 1000 observed samples. The forgetting factor in the AM was set as in the $oKDE_{0.02}$. Fig. 5 summarizes the results.

Both batch methods outperformed the AM model in accuracy, however, they produced models of significantly greater complexity. On average, the $oKDE_{0.02}$ outperformed both, CV and Hall, batch KDEs by maintaining lower error and using a three orders of magnitude smaller number of components. The approximation error of models produced by the $oKDE_{0.02}$ was lower for 1000 samples, became slightly greater than that of the batch KDEs for 2000 and 3000 samples, and then became again lower. We have noticed that in some (rare) cases, the CV produced an under-smoothed estimate of the distribution which temporarily increased the L_1 error. On the other hand, this behavior has not been observed for the oKDE, AM and Hall's method. In all experiments, the $oKDE_{0.02}$ outperformed AM.

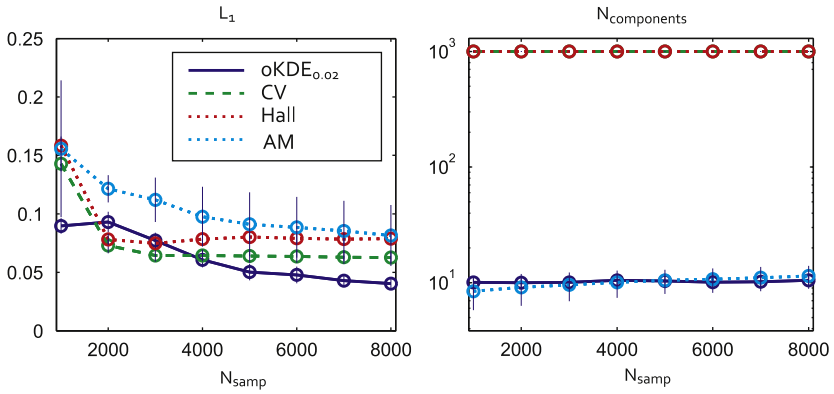

Fig. 5. The L_1 estimation error (left) and the number of components (right) w.r.t the time-step, along with one-standard-deviation bars. The results are shown for the oKDE (full line), CV (dashed), Hall (dotted dark) and AM (dotted bright).

Table 2
Properties of the datasets used in the experiment with real-life data. The number of samples in each dataset, the dimensionality and the number of classes are denoted by N_S, N_D and N_C, respectively.

Dataset	N_S	N_D	N_C
Iris	150	4	3
Pima	768	8	2
Wine	178	13	3
WineRed	1599	11	6
WineWhite	4898	11	7
Letter	20000	16	26
Breast cancer (cancer)	285	30	2
Image segmentation (seg)	2310	18	7
Steel plates (plates)	1941	27	7
Yeast	1484	8	10

6.2. Density estimation on real datasets

We have repeated the density estimation experiment on several real-life datasets from the UCI machine learning repository [33] which differed in the length, data dimensionality as well as in the number of classes. The general properties of the datasets are summarized in Table 2. For the density estimation experiment, we estimated the density for each class separately. The data in each dataset were randomly reordered, 75% of the data were used for training and the rest for testing. For each of the datasets we have generated 12 such random partitionings. The oKDE and the AM were initialized from the first 10 samples and the rest were added one at a time. The compression threshold in the oKDE was set to $D_{th}=0.1$. To measure the estimation quality, we have computed the average negative log-likelihood of the test data, while the Bayes information criterion (BIC) was used to measure the tradeoff between the model's complexity and its ability to explain the input data.

The results of the experiments after observing all the data-points are summarized in Table 3. Among the online methods, the oKDE models obtained better trade-off between their complexity and the ability to explain the input data. This is indicated by the fact that the oKDE s obtained on an average a lower BIC than AM. From Table 3 we can also see that in nearly all examples the oKDE produced models with lower complexity than the AM, while achieving a lower average negative log-likelihood. Compared to the batch methods, the CV and Hall achieved the lowest average negative log-likelihood, while the oKDE outperformed them in BIC measure. The RSDE was comparable to oKDE in BIC measure, but oKDE nearly always outperformed the RSDE by achieving a lower average negative log-likelihood.

An important aspect of online methods, apart from the estimation quality, is the resulting model's complexity and the time required to perform an update when a new data-point arrives. Note that unlike the batch methods, the oKDE does not store all the observed data-points, but maintains their compact representation in the form of a sample distribution. To enable recovery from early over-compression, each component in the sample distribution stores its two-component representation. Therefore, the model's complexity and storage requirements are proportional to the number of components in the model, in particular, the storage requirements are twice the model's complexity. The complexity of the model directly affects the oKDE's update speed. Most of the time, the computational complexity is dominated by the functional (11) in the bandwidth calculation, which scales as $O((N^2-N)/2+N)$, where N is the number of components in the model (complexity). From time to time, the model is compressed and then the complexity is dominated by the evaluation of the clustering error (21), which again scales in number of components, and the compression threshold. The required storage size and the complexity are visualized for all datasets in the first two columns of Fig. 6. We see that the complexity and storage requirements of oKDE's models were on average lower than AM's models. The oKDE's models were also significantly less complex than the models produced by batch methods except for the RSDE. For reference, we also provide the measured times required for a single update in the last column of Fig. 6, but these have to be interpreted with caution. Namely, all the code was written in Matlab, except from Hall's estimator and parts of our bandwidth estimator that were written in C. Our implementation of the oKDE is a non-optimized research code and thus heavily redundant, but we believe that some measurements of the speed will be nevertheless interesting for practical considerations. To measure the times required for a single update, we have therefore calculated the time required for a single update, averaged over the last 20 updates in the experiments. We see that the AM allowed the fastest updates. The oKDE was faster than the batch methods in all cases except for the dataset Iris and Wine, where Hall's method was slightly faster. Averaged over the datasets, the oKDE required 0.07 s for updating a distribution by a new observation.

6.3. Online estimation of classifiers

To analyze the discriminative properties of the oKDE, we have repeated the experiment from the previous section with the online and batch methods to construct classifiers from the publicly available datasets described in Table 2. For the baseline classification, we have applied a multiclass SVM with a RBF kernel [34]. The classification performance of the KDE-based methods

Table 3
The average negative log-likelihood $-\mathcal{L}$, BIC and the number of components in the models. For each dataset, the two best-performing methods are in bold and the asterisk ()* sign denotes that the difference between the best and second best method is statistically significant.

Dataset	oKDE$_{0.1}$	AM	CV	RSDE	Hall
	$-\mathcal{L}$				
Iris	8.4 ± 3.7	**2.4 ± 0.4**	2.7 ± 0.9	2.5 ± 0.9	**2.0 ± 0.7**
Pima	30.1 ± 0.3	30.1 ± 0.5	**29.5 ± 0.5**	38.4 ± 11.3	**29.1 ± 0.1***
Wine	23.8 ± 3.9	12.3 ± 0.8	**11.6 ± 1.5**	12.3 ± 1.9	**9.6 ± 0.4***
WRed	−24.2 ± 1.1	**−27.4 ± 0.2**	−27.2 ± 1.0	−12.3 ± 4.3	**−27.4 ± 0.7**
WWhit	13.3 ± 0.3	14.5 ± 0.4	**11.6 ± 0.4***	91.3 ± 44.6	**11.9 ± 0.1**
Letter	**8.4 ± 0.1**	11.4 ± 4.7	**6.4 ± 0.3***	142.2 ± 13.0	10.2 ± 0.1
BCW	27.5 ± 6.9	97.9 ± 23.1	**8.9 ± 3.1***	18.1 ± 5.4	**11.1 ± 2.2**
Seg	−16.6 ± 1.8	−12.1 ± 3.8	**−25.3 ± 2.3***	46.9 ± 35.9	**−23.0 ± 1.7**
Plates	−3.5 ± 2.0	44.9 ± 11.6	**−11.2 ± 1.5***	54.1 ± 18.9	**−9.5 ± 0.8**
Yeast	11.2 ± 1.1	**3.9 ± 1.6**	**3.0 ± 2.1**	62.8 ± 30.7	6.0 ± 0.5
	BIC				
Iris	5595 ± 37	**4377 ± 70**	8183 ± 72	**1766 ± 58***	8242 ± 49
Pima	**57 280 ± 250***	138 444 ± 224	194 657 ± 144	**65 092 ± 13 944**	196 005 ± 44
Wine	28 395 ± 46	**25 854 ± 167**	30 582 ± 72	**21 225 ± 299***	31 255 ± 87
WRed	**53 195 ± 466**	**−23 710 ± 199***	579 462 ± 341	74 706 ± 9205	586 600 ± 309
WWhit	**260 724 ± 769***	1 560 692 ± 1327	2 391 884 ± 654	**817 970 ± 312 008**	2 417 345 ± 531
Letter	**2 353 134 ± 1849***	14 228 704 ± 41 592	22 016 590 ± 7341	**5 130 425 ± 387 481**	22 240 178 ± 994
BCW	**615 967 ± 770***	1 267 175 ± 6253	1 269 862 ± 1611	**814 150 ± 2477**	1 287 566 ± 862
Seg	**331 204 ± 1150**	1 776 111 ± 7393	2 341 933 ± 3878	**342 581 ± 104 979**	2 357 185 ± 1208
Plates	**1 069 658 ± 3291**	4 239 189 ± 13 418	4 216 830 ± 11 345	**600 487 ± 156 551***	4247 392 ± 1533
Yeast	**101 473 ± 290***	194 856 ± 3530	348 154 ± 4585	**156 825 ± 16 827**	357 542 ± 996
	Number of components per model				
Iris	27 ± 4	16 ± 3	38 ± 0	11 ± 8	38 ± 0
Pima	42 ± 8	165 ± 47	288 ± 89	48 ± 27	288 ± 89
Wine	43 ± 6	32 ± 6	44 ± 7	37 ± 10	44 ± 7
WRed	42 ± 24	13 ± 4	200 ± 215	31 ± 32	200 ± 215
WWhit	37 ± 26	343 ± 365	525 ± 596	41 ± 40	525 ± 596
Letter	60 ± 10	321 ± 120	577 ± 17	24 ± 22	577 ± 17
BCW	102 ± 9	214 ± 56	214 ± 56	186 ± 57	214 ± 56
Seg	43 ± 10	181 ± 24	248 ± 0	23 ± 10	248 ± 0
Plates	60 ± 17	206 ± 155	208 ± 156	23 ± 16	208 ± 156
Yeast	29 ± 13	53 ± 51	111 ± 124	42 ± 61	111 ± 124

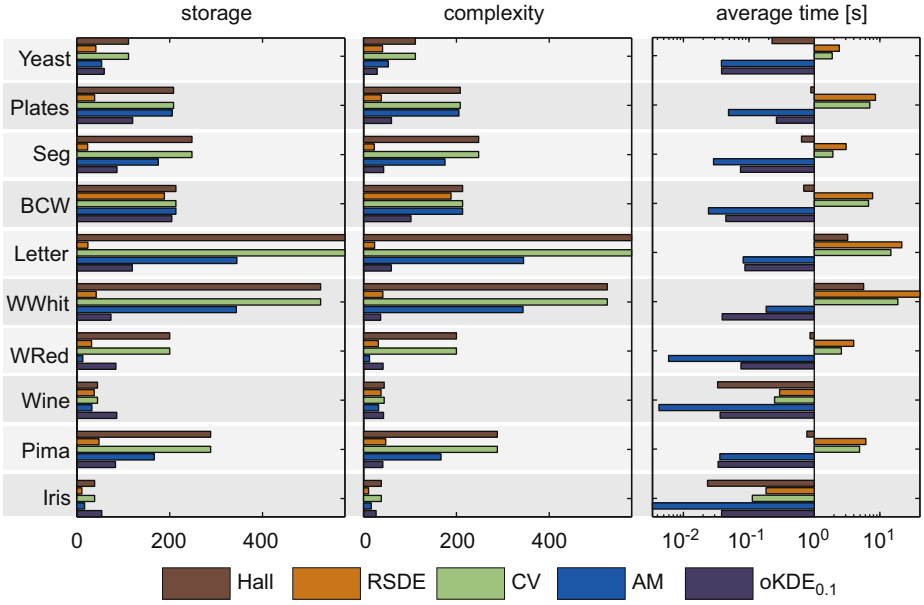

Fig. 6. The estimated time required for a single-class update from a single sample, the average storage requirement and the model's average complexity.

and the AM was tested using a simple Bayesian criterion

$$\hat{y} = \underset{l}{\operatorname{argmax}} p(\mathbf{x}|c_l)p(c_l). \tag{35}$$

The parameter for the SVM kernel was determined separately in each experiment via cross-validation on the training dataset.

The results are shown in Table 4. We can see that the oKDE achieved a better classification than the AM for all datasets except for the Iris, for which the performance was matched. In additional analysis we have verified that in all datasets except for the Iris, the improved performance was also statistically significant. The batch methods, SVM, CV and Hall produced on average best classification.

Table 4
Average classification results along with ± one standard deviation. With each classification performance we also show the model's complexity in parentheses. For each dataset, the two best methods are in bold and the asterisk ()* sign denotes that the difference between the best and second best method is statistically significant.

Dataset	$oKDE_{0.1}$	AM	CV	RSDE	Hall	SVM
Iris	**97 ± 3(27 ± 4)**	97 ± 3(16 ± 3)	96 ± 3(38 ± 0)	96 ± 2(11 ± 8)	**97 ± 3(38 ± 0)**	96 ± 4(16 ± 6)
Pima	72 ± 2(42 ± 8)	69 ± 3(165 ± 47)	72 ± 2(288 ± 89)	65 ± 3(48 ± 27)	**74 ± 2(288 ± 89)**	**78 ± 3(163 ± 4)***
Wine	94 ± 3(43 ± 6)	91 ± 4(32 ± 6)	92 ± 6(44 ± 7)	91 ± 5(37 ± 10)	**96 ± 3(44 ± 7)**	**96 ± 3(24 ± 8)**
WRed	**64 ± 2(42 ± 24)**	57 ± 3(13 ± 4)	64 ± 1(200 ± 215)	44 ± 4(31 ± 32)	**66 ± 2(200 ± 215)***	63 ± 3(173 ± 179)
WWhit	55 ± 1(37 ± 26)	53 ± 2(343 ± 365)	**62 ± 1(525 ± 596)**	25 ± 6(41 ± 40)	**62 ± 2(525 ± 596)**	60 ± 2(473 ± 523)
Letter	96 ± 0(60 ± 10)	91 ± 3(321 ± 120)	**96 ± 0(577 ± 17)**	53 ± 2(24 ± 22)	95 ± 0(577 ± 17)	**96 ± 0(311 ± 60)***
BCW	93 ± 2(102 ± 9)	90 ± 3(214 ± 56)	**96 ± 2(214 ± 56)**	94 ± 2(186 ± 57)	91 ± 2(214 ± 56)	**97 ± 2(52 ± 7)***
Seg	93 ± 1(43 ± 10)	90 ± 2(181 ± 24)	**94 ± 1(248 ± 0)**	79 ± 3(23 ± 10)	94 ± 1(248 ± 0)	**94 ± 1(83 ± 48)**
Plates	**72 ± 2(60 ± 17)**	68 ± 2(206 ± 155)	71 ± 2(208 ± 156)	56 ± 4(23 ± 16)	65 ± 1(208 ± 156)	**76 ± 2(146 ± 137)***
Yeast	**51 ± 2(29 ± 13)**	48 ± 5(53 ± 51)	49 ± 4(111 ± 124)	35 ± 10(42 ± 61)	25 ± 2(111 ± 124)	**60 ± 1(91 ± 105)***

Table 5
The average negative log-likelihood ($-\mathcal{L}$) and the number of components in the model (N_{cmp}) for $oKDE_{D_{th}}$ and AM w.r.t. to the three data orders: random, center-to-outermost and outermost-to-center, and averaged over the different orders.

Method	[Mean ± standard deviation]							
	Random order		Sorted outward		Sorted inward		Averaged	
	$-\mathcal{L}$	N_{cmp}	$-\mathcal{L}$	N_{cmp}	$-\mathcal{L}$	N_{cmp}	$-\mathcal{L}$	N_{cmp}
AM	5.45 ± 0.04	43.3 ± 7.25	5.46 ± 0.01	216 ± 12.5	5.41 ± 0.01	115 ± 6.3	5.44 ± 0.04	125 ± 71.6
$oKDE_{0.01}$	**5.39 ± 0.01**	37.7 ± 3.22	**5.39 ± 0.01**	48.9 ± 2.6	**5.39 ± 0.01**	36.8 ± 2.8	**5.39 ± 0.01**	41.1 ± 6.2
$oKDE_{0.02}$	5.39 ± 0.01	19.6 ± 1.43	5.39 ± 0.01	20.3 ± 2.48	5.41 ± 0.01	17.8 ± 1.38	5.4 ± 0.01	19.2 ± 2.1
$oKDE_{0.04}$	5.48 ± 0.03	11.8 ± 1.44	5.63 ± 0.09	8.97 ± 1.5	5.76 ± 0.06	7.5 ± 1.5	5.63 ± 0.13	9.43 ± 2.3
$oKDE_{0.05}$	5.55 ± 0.06	10 ± 1.76	5.83 ± 0.07	5.03 ± 1.4	5.84 ± 0.04	5.2 ± 1.03	5.74 ± 0.15	6.74 ± 2.71

The oKDE outperformed batch RSDE, and produced a comparable classification to the SVM, CV and Hall's KDE. An important observation is that the oKDE produced comparable performance to the batch methods, eventhough the oKDE was constructed by observing only a single sample at a time. In contrast, the SVM and the batch KDEs optimized their structure by having access to all the samples. Note also that the oKDE's classification performance was comparable to SVM, eventhough the oKDE is in its nature reconstructive, while the SVM optimizes its structure to maximize discrimination. Note also that the complexity of the models learnt by batch KDEs is generally larger than that of the oKDE. For example, for the *letter* dataset, the oKDE required one ninth as many components to achieve a comparable performance to the CV and Hall's KDE. While, compared to the oKDE, the RSDE retained only half as many components for the *letter* dataset, the RSDE's classification performance was also significantly lower.

6.4. Influence of the compression parameter D_{th} and data order

The only free parameter in the oKDE is the compression parameter D_{th}, which quantifies the local approximation error during compression (and revitalization) in terms of the unscented Hellinger distance. The aim of this experiment was therefore to illustrate how the different values of this parameter affect the oKDE's performance. The experiments involved approximating a spiral-shaped two-dimensional stationary distribution defined as

$$\mathbf{x} = [(1+\theta)\cos(\theta),(1+\theta)\sin(\theta)]^{\mathrm{T}} + \mathbf{w}, \quad \mathbf{w} \sim \phi_{\boldsymbol{\Sigma}_{\mathbf{w}}}(\cdot), \quad \theta \sim \mathcal{U}(0,10), \tag{36}$$

where $\boldsymbol{\Sigma}_{\mathbf{w}} = \mathrm{diag}\{0.9^2, 0.9^2\}$ and $\mathcal{U}(0,10)$ is a uniform distribution on interval [0,10]. A set of 1000 samples was generated from this distribution—the first 10 samples were used for initialization and the rest were used one at a time to update the oKDE. After all 1000 samples have been observed, the reconstructive performance of the KDE model was evaluated as the average negative log-likelihood of additionally drawn 20 000 samples.

The performance of the oKDE with various compression values was compared with the AM. The performance results are shown under the "*random order*" label in Table 5. We see that the oKDE with the smallest compression threshold produced the most accurate models with 37 components. By increasing the compression threshold, the number of components decreased, while the approximation error increased. The oKDE outperformed the AM in accuracy for D_{th} values smaller than 0.03. Note that the AM-estimated models contained on average 45 components, while for example, the oKDE with $D_{\text{th}} = 0.02$ produced more accurate models which contained on average 20 components. We show typical estimated models for AM and oKDE in the first row of Fig. 7.

6.4.1. Sorted data

To analyze the performance of the oKDE w.r.t. the different data orderings, we have performed two variants of the previous experiment in which we enforced a predefined order to the observed samples. The order was enforced by deterministically selecting the values of the position parameter θ in (36) along the spiral at equal distances from the interval [0,10].

In the first variant, the position values θ were organized in an ascending order, thus yielding an *outward* ordering of data-points from the spiral's center, while the second variation used a descending order of position values, which yielded an *inward* ordering. In both orderings, the early samples indicated a small scale of the estimated distribution, and the entire scale became evident slowly at later time-steps. In the outward ordering the scale became apparent only slowly, since a large number of samples are concentrated at the center of the spiral. The results for the outward and inward ordering are given in the second and third columns of Table 5. With increasing D_{th} the oKDE produced models with lower number of components at a cost of larger

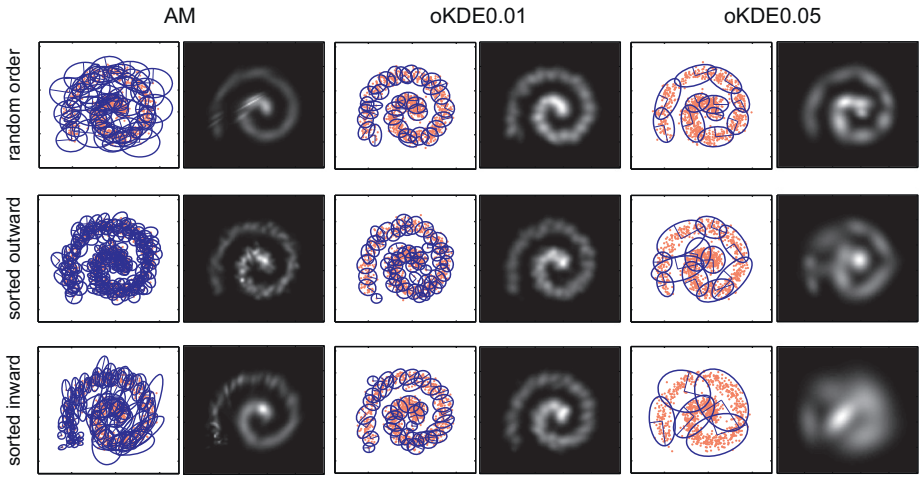

Fig. 7. Mixture models of the spiral distribution for different orderings of data: random (first row), sorted outward (second row) and sorted inward (third row). Each model is shown as a decomposed mixture model, and as an image of its distribution.

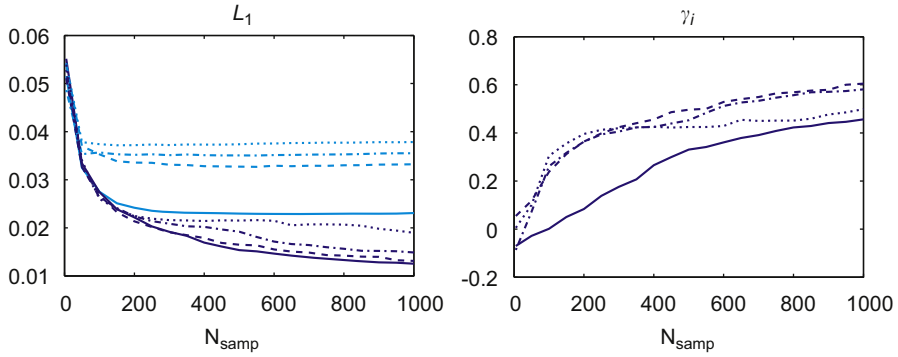

Fig. 8. The left plot shows the L_1 distance errors for the oKDE with the revitalization scheme (dark blue), and without the revitalization scheme (bright cyan), w.r.t. the number of samples N_{samp}. The right graph shows the improvements in terms of error reduction (improvement factor γ_i). The results for $oKDE_{0.01}$, $oKDE_{0.02}$, $oKDE_{0.04}$ and $oKDE_{0.05}$ are depicted by solid, dashed, dash-dotted and dotted lines, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

reconstruction error. With respect to the values of D_{th}, the number of components remained comparable with the random sampling. On the other hand, the AM produced models whose complexity was significantly larger. This can be attributed directly to the missing scale information in the early samples, which initially caused allocation of a larger number of components in the AM model.

With increasing the compression threshold D_{th}, the degradation of the models in oKDE was faster for inward than outward ordering. The reason is that greater D_{th} allows greater loss of information about the structure of the distribution during online estimation. In the absence of the structure information the models deteriorate. To estimate how the oKDE performs regardless of the data order, the results over different orders were averaged and are shown in the last column of Table 5. We see that on average the oKDE with $D_{th} < 0.03$ outperforms the AM by producing models with smaller errors and smaller number of components. With increasing the compression values, the number of components further decreases, while the errors increase.

6.5. *Influence of the revitalization scheme*

To analyze the benefit of the revitalization scheme from Section 4.3, we generated 1000 samples from a heavily skewed one-dimensional reference distribution (see Fig. 4a), and used one sample at a time with the oKDE to approximate this distribution. One experiment was performed with the revitalization scheme and one without it. We have calculated the improvement factors γ_i w.r.t. the number of samples as $\gamma_i = (\hat{\varepsilon}_i - \varepsilon_i)/\hat{\varepsilon}_i$, where $\hat{\varepsilon}_i$ is the L_1 distance between the reference distribution and the model without revitalization, ε_i is the L_1 distance between the reference distribution and the model with revitalization. The index i represents the observed sample. Fig. 8 (right column) shows these results for the different values of the compression threshold D_{th}. We can see that the improvement of using the revitalization scheme increases with the number of samples regardless of the compression threshold D_{th}. For example, after observing 1000 samples, the improvement for all tested values D_{th} was between 45% and 65%.

7. Conclusion

We have proposed an approach for the kernel density estimation which can be applied in online operation. The central point of the proposed scheme is that it does not store all the observed samples, but maintains only their compressed model and uses this model to compute the kernel density estimate of the underlying distribution. During online operation, the low complexity is automatically maintained by a new compression/revitalization scheme. The approach was analyzed using examples of online estimation of stationary as well as non-stationary distributions and on classification examples. In all experiments, the oKDE was

able to produce comparable or better results to the state-of-the-art online and batch approaches, while producing models whose complexity was significantly lower.

The only parameter in the oKDE is the compression threshold that specifies the allowed loss of reconstructive properties during compression. Experiments showed that a low value of D_{th} produces models with larger number of components and influences the reconstruction properties of the estimated model. If the task at hand involves estimation of probability density function for reconstruction or compression of the input stream, then choosing a low value may be advantageous. On the other hand, larger values imply greater smoothing and therefore regularization of the distribution. Note that for the classification tasks, loss of reconstructive information does not necessarily mean loss of discriminative properties. In fact, we have observed in the experiments that choosing a value $D_{\text{th}}=0.1$, yielded a very good recognition performance at a small number of components. At the same time, the models retained enough reconstructive information to sufficiently adapt to the new data. For online construction of classifiers we therefore propose setting $D_{\text{th}}=0.1$. Alternatively, the threshold D_{th} might be adapted during the online operation of the model. For example, one could keep in memory a small set of past observed values and validate the estimated model at a few compression thresholds, to select the best-performing threshold.

Due to the nature of the compression algorithm, different components in the oKDE have different covariances and thus the result is equivalent to a KDE with a non-stationary bandwidth. Nevertheless, related research [14,15,17] shows that further adjusting the non-stationary bandwidths by taking into account the local structure of data (e.g., nearest neighbors) can significantly improve the density estimation. The reason is that the regions of feature space with few samples require more intensive smoothing than the more densely populated regions. Another reason is that the data within a neighborhood of a component better determine the local structure (the local manifolds) of the density function. We believe that applying the methodology from [15,17] to oKDE would be beneficial and it is likely to improve the performance in density estimation.

Note that the update procedure in the oKDE makes it a reconstructive estimator, since the compression algorithm penalizes errors in the reconstruction. We can think about the compression algorithm itself as an approximate optimization, which seeks a minimum of the reconstructive cost function. We believe that replacing this cost function with some other criterion would yield different properties of the online KDE, without modification of the optimization algorithm. Indeed, we have already explored a possibility of replacing this cost function with a criterion that, instead of reconstruction errors, it penalizes discriminative errors in [35] and obtained encouraging preliminary results. We believe that this venue of research will lead to online probabilistic discriminative models based on the kernel density estimation, which will be based on the theory presented here. These are the topics of our ongoing research.

Acknowledgment

This research has been supported in part by: RP P2-0214 and P2-0094 (RS), ARRS project "Learning a large number of visual object categories for content-based retrieval in image and video databases", and EU FP7-ICT-215181-IP project CogX.

Appendix A. The unscented Hellinger distance

The unscented transform is a special case of a Gaussian quadrature, which, similarly to Monte Carlo integration, relies on evaluating integrals using carefully placed points, called *the sigma points*, over the support of the integral. Therefore, as in Monte Carlo integration [36], we define an *importance* distribution $p_0(\mathbf{x})=\gamma(p_1(\mathbf{x})+p_2(\mathbf{x}))$, which contains the support of both, $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$, with γ set such that $\int p_0(\mathbf{x})\,d\mathbf{x}=1$. In our case, $p_0(\mathbf{x})$ is a Gaussian mixture model of a form $p_0(\mathbf{x})=\sum_{i=1}^{N} w_i\phi_{\boldsymbol{\Sigma}_i}(\mathbf{x}-\mathbf{x}_i)$, and we rewrite the Hellinger distance (21) into

$$D^2(p_1,p_2)=\frac{1}{2}\int g(\mathbf{x})p_0(\mathbf{x})\,d\mathbf{x}=\frac{1}{2}\sum_{i=1}^{N} w_i\int g(\mathbf{x})\phi_{\Sigma_i}(\mathbf{x}-\mathbf{x}_i)\,d\mathbf{x}, \tag{37}$$

where we have defined $g(\mathbf{x})=(\sqrt{p_1(\mathbf{x})}-\sqrt{p_2(\mathbf{x})})^2/p_0(\mathbf{x})$. Note that the integrals in (37) are simply expectations over a nonlinearly transformed Gaussian random variable $\mathbf{X}$, and therefore admit to the unscented transform. According to [31] we then have

$$D^2(p_1,p_2)\approx\frac{1}{2}\sum_{i=1}^{N} w_i\sum_{j=0}^{2d+1} g({}^{(j)}\mathcal{X}_i)\,{}^{(j)}\mathcal{W}_i, \tag{38}$$

where $\{{}^{(j)}\mathcal{X}_i,{}^{(j)}\mathcal{W}_i\}_{j=0:d}$ are weighted sets of sigma points corresponding to the i-th Gaussian $\phi_{\boldsymbol{\Sigma}_i}(\mathbf{x}-\mathbf{x}_i)$, and are defined as

$$\begin{aligned}&{}^{(0)}\mathcal{X}_i=\mathbf{x}_i,\quad {}^{(0)}\mathcal{W}_i=\frac{\kappa}{1+\kappa},\quad {}^{(j)}\mathcal{X}_i=\mathbf{x}_i+s_j\sqrt{1+\kappa}(\sqrt{d\boldsymbol{\Sigma}_i})_j,\\ &{}^{(j)}\mathcal{W}_i=\frac{\kappa}{2(1+\kappa)},\quad s_j=\begin{cases}1, & j\le d,\\ -1 & \text{otherwise}\end{cases}\end{aligned} \tag{39}$$

with $\kappa=\max([0,m-d])$, and $(\sqrt{\boldsymbol{\Sigma}_i})_j$ is the j-th column of the matrix square root of $\boldsymbol{\Sigma}_i$. Concretely, let $\mathbf{UDU}^T$ be a singular value decomposition of covariance matrix $\boldsymbol{\Sigma}$, such that $\mathbf{U}=\{U_1,\ldots,U_d\}$ and $\mathbf{D}=\text{diag}\{\lambda_1,\ldots,\lambda_d\}$, then $(\sqrt{\Sigma})_k=\sqrt{\lambda_k}U_k$. In line with the discussion on the properties of the unscented transform in [31], we set the parameter m to $m=3$.

Appendix B. Supplementary data

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.patcog.2011.03.019.

References

[1] G. McLachlan, D. Peel, Finite Mixture Models, Wiley-Interscience, 2000.
[2] Z. Živkovič, F. van der Heijden, Recursive unsupervised learning of finite mixture models, IEEE Trans. Pattern Anal. Mach. Intell. 26 (5) (2004) 651–656.
[3] M.A.F. Figueiredo, A.K. Jain, Unsupervised learning of finite mixture models, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 381–396.
[4] E. Parzen, On estimation of a probability density function and mode, Ann. Math. Stat. 33 (1962) 1065–1076.
[5] F. Hoti, L. Holmström, A semiparametric density estimation approach to pattern classification, Pattern Recognition 37 (3) (2004) 409–419.
[6] G. Fu, F. Shih, H. Wang, A kernel-based parametric method for conditional density estimation, Pattern Recognition 44 (2011) 284–294.
[7] B.W. Silverman, Density Estimation, Chapman and Hall, 1986.
[8] P. Hall, S.J. Sheater, M.C. Jones, J.S. Marron, On optimal data-based bandwidth selection in kernel density estimation, Biometrika 78 (2) (1991) 263–269.
[9] Y. Hamamoto, Y. Fujimoto, S. Tomita, On the estimation of a covariance matrix in designing Parzen classifiers, Pattern Recognition 29 (1996) 1751–1759.
[10] D. Chaudhuri, B. Chaudhuri, C. Murthy, A data driven procedure for density estimation with some applications, Pattern Recognition 29 (10) (1996) 1719–1736.
[11] M.P. Wand, M.C. Jones, Kernel Smoothing, Chapman & Hall/CRC, 1995.
[12] J.M.L. Murillo, A.A. Rodriguez, Algorithms for Gaussian bandwidth selection in kernel density estimators, in: Neural Information Processing Systems, 2008.
[13] Y. Li, D. de Ridder, R. Duin, M. Reinders, Integration of prior knowledge of measurement noise in kernel density classification, Pattern Recognition 41 (1) (2008) 320–330.
[14] J. Cwik, J. Koronacki, A combined adaptive-mixtures/plug-in estimator of multivariate probability densities, Computational Statistics and Data Analysis 26 (1998) 199–218.
[15] P. Vincent, Y. Bengio, Manifold Parzen windows, Adv. Neural Inf. Process. Syst. (2003) 849–856.

[16] J. Goldberger, S. Roweis, Hierarchical clustering of a mixture model, in: Neural Information Processing Systems, 2005, pp. 505–512.
[17] E. López-Rubio, J. Ortiz-de Lazcano-Lobato, Soft clustering for nonparametric probability density function estimation, Pattern Recognition Lett. 29 (16) (2008) 2085–2091.
[18] M. Girolami, C. He, Probability density estimation from optimally condensed data samples, IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (2003) 1253–1264.
[19] Z. Deng, F. Chung, S. Wang, FRSDE: fast reduced set density estimator using minimal enclosing ball approximation, Pattern Recognition 41 (2008) 1363–1372.
[20] A. Leonardis, H. Bischof, An efficient mdl-based construction of rbf networks, Neural Networks 11 (5) (1998) 963–973.
[21] O. Arandjelovic, R. Cipolla, Incremental learning of temporally-coherent Gaussian mixture models, in: British Machine Vision Conference, 2005, pp. 759–768.
[22] C.E. Priebe, D.J. Marchette, Adaptive mixture density estimation, Pattern Recognition 26 (1993) 771–785.
[23] K. Tabata, M. Sato, M. Kudo, Data compression by volume prototypes for streaming data, Pattern Recognition 43 (2010) 3162–3176.
[24] M. Song, H. Wang, Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering, in: SPIE, 2005, pp. 174–183.
[25] A. Declercq, J.H. Piater, Online learning of Gaussian mixture models—a two-level approach, in: VISAPP, 2008, pp. 605–611.
[26] B. Han, D. Comaniciu, Y. Zhu, L.S. Davis, Sequential kernel density approximation and its application to real-time visual tracking, IEEE Trans. Pattern Anal. Mach. Intell. 30 (7) (2008) 1186–1197.
[27] T. Duong, M.L. Hazelton, Plug-in bandwidth matrices for bivariate kernel density estimation, Nonparametric Stat. 15 (1) (2003) 17–30.
[28] M. Kristan, D. Skočaj, A. Leonardis, Online kernel density estimation for interactive learning, Image Vision Comput. 28 (7) (2010) 1106–1116.
[29] Y. Bar-Shalom, X.R. Li, T. Kirubarajan, Estimation with Applications to Tracking and Navigation, John Wiley & Sons, Inc., 2001, pp. 438–440 (Chapter 11).
[30] D.E. Pollard, A User's Guide to Measure Theoretic Probability, Cambridge University Press, 2002.
[31] S. Julier, J. Uhlmann, A general method for approximating nonlinear transformations of probability distributions, Technical Report, Department of Engineering Science, University of Oxford, 1996.
[32] A.T. Ihler, Inference in sensor networks: graphical models and particle methods, Ph.D. Thesis, Massachusetts Institute of Technology, 2005.
[33] A. Asuncion, D. Newman, UCI machine learning repository, 2007.
[34] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, 2001.
[35] M. Kristan, A. Leonardis, Online discriminative kernel density estimation, in: International Conference on Pattern Recognition, 2010, pp. 581–584.
[36] E. Veach, L.J.A. Guibas, Optimally combining sampling techniques for Monte Carlo rendering, in: Computer Graphics and Interactive Techniques, 1995, pp. 419–428.

Matej Kristan received the Dipl.ing., M.Sc., and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2003, 2005, and 2008, respectively.

He is currently an Assistant Professor at the Machine Vision Laboratory, Faculty of Electrical Engineering, University of Ljubljana and a Researcher with the Visual Cognitive Systems Laboratory, Faculty of Computer and Information Science, University of Ljubljana. His research interests include probabilistic methods for computer vision and pattern recognition with focus on tracking, probabilistic dynamic models, online learning and mobile robotics. He has received several awards for his research in the field of computer vision and pattern recognition.

Aleš Leonardis received the Dipl.ing. and M.Sc. degrees in electrical engineering and the Ph.D. degree in computer science from the Faculty of Electrical Engineering and Computer Science, University of Ljubljana, Ljubljana, Slovenia, in 1985, 1988, and 1993, respectively.

From 1988 to 1991, he was a Visiting Researcher at the General Robotics and Active Sensory Perception Laboratory, University of Pennsylvania, Philadelphia. From 1995 to 1997, he was a Postdoctoral Associate with the Pattern Recognition and Image Processing Group, Vienna University of Technology, Vienna, Austria. He was also a Visiting Researcher and a Visiting Professor at the Swiss Federal Institute of Technology ETH, Zurich, and at the Technische Fakultaet der Friedrich-Alexander-Universitaet, Erlangen, respectively. He is a Full Professor and the Head of the Visual Cognitive Systems Laboratory, Faculty of Computer and Information Science, University of Ljubljana. He is also an Adjunct Professor with the Faculty of Computer Science, Graz University of Technology, Graz, Austria. He is an author or coauthor of more than 160 papers published in journals and conferences and he coauthored the book Segmentation and Recovery of Superquadrics (Kluwer, 2000). His research interests include robust and adaptive methods for computer vision, object and scene recognition and categorization, statistical visual learning, 3-D object modeling, and biologically motivated visions. He is an Editorial Board Member of Pattern Recognition, an Editor of the Springer Book Series Computational Imaging and Vision, and an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He has served on the program committees of major computer vision and pattern recognition conferences. He was also a Program Cochair of the European Conference on Computer Vision 2006. He has received several awards. In 2002, he coauthored a paper, "Multiple Eigenspaces," which won the 29th Annual Pattern Recognition Society award. In 2004, he was awarded a prestigious national award for scientific achievements. He is a fellow of the International Association for Pattern Recognition and a member of the IEEE Computer Society.

Danijel Skočaj received the Ph.D. degree in computer and information science from the Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia, in 2003.

He is currently an Assistant Professor at the Faculty of Computer and Information Science, University of Ljubljana. His research interests include robust methods for subspace analysis, cognitive systems, and learning methods for cognitive robots.