Computer Vision Winter Workshop 2010, Libor Špaček, Vojtěch Franc (eds.)
Nové Hrady, Czech Republic, February 3–5
Czech Pattern Recognition Society

Multivariate Online Kernel Density Estimation

Matej Kristan[1,2] and Aleš Leonardis[1]

[1]Faculty of Computer and Information Science, University of Ljubljana
[2]Faculty of Electrical Engineering, University of Ljubljana
mailto:matej.kristan@fri.uni-lj.si

Abstract *We propose an approach for online kernel density estimation (KDE) which enables building probability density functions from data by observing only a single data-point at a time. The method maintains a non-parametric model of the data itself and uses this model to calculate the corresponding KDE. We propose an new automatic bandwidth selection rule, which can be computed directly from the non-parametric model of the data. Low complexity of the model is maintained through a novel compression and refinement scheme. We compare the online KDE to some state-of-the-art batch KDEs on examples of estimating distributions and on an example of classification. The results show that the online KDE generally achieves comparable performance to the batch approaches, while producing models with lower complexity and allowing online updating using only a single observation at a time.*

1 Introduction

Many tasks in machine learning and pattern recognition require building models from observing sequences of data. In real-world environments all the data may not available in advance, or we even want to observe the process for an indefinite duration, while continually providing the best estimate of the model from the data observed so far. These tasks require online construction of the models.

A popular approach to generating models from data is to model the probability density function (pdf) associated with the observed data. Traditionally, parametric models based on Gaussian mixture models (GMM) [9] have been applied with some success in estimation of the pdf when all data are observed in advance. The parametric mixture models typically require specifying the number of components in the mixture and may not capture the complete structure of the underlying pdf. Non-parametric methods such as Parzen [21, 11, 28] kernel density estimators (KDE), with Gaussian kernels, alleviate this problem by treating each observation as a component in the mixture model and assuming all components have equal covariances (bandwidths). The problem of KDE is then how to automatically set this bandwidth. While most of the literature on the bandwidth selection have dealt with one-dimensional problems, recently Murillo and Rodriguez [20] have proposed an efficient method for calculating the multivariate bandwidths. One drawback of the KDEs is that their complexity (number of components) increases linearly with the number of the observed data. To remedy this, methods have been proposed to reduce the number of components either to a predefined value [11, 28] or by optimizing a data-based [10] and MDL-based [18] cost functions.

There have been several attempts to merge the non-parametric quality of the kernel density estimators with the Gaussian mixture models in online applications. Arandjelović et.al. [1] proposed a scheme for online adaptation of the Gaussian mixture model which can be updated by one sample at a time. However, a strong restriction is made that data is temporally coherent in feature space, which prevents its use in general applications. Priebe and Marchette proposed an online EM algorithm called active mixtures [23] which is less sensitive to the data order, allows adaptation from single observation at a time, and includes a heuristic for allocating new components. Song et. al. [24] aim to alleviate the temporal restrictions by processing data in large blocks. Bischof and Leonardis [4] use a radial-basis-function (RBF) based network and apply an MDL-based procedure for basis function allocation and deletion – they assume a predefined initial rbf size. Deleclerq and Piater [7] assume each data is a Gaussian with a predefined covariance. All data are stored in the model and a heuristic is used to determine when a subset of the data can be replaced by a single component. Han et. al. [14] proposed an online approach inspired by the kernel density estimation in which each new observation is added to the model as a Gaussian kernel with a predefined bandwidth. The model's complexity is maintained low by retaining only the modes of the distribution, which they approximate by Gaussians. This approach deteriorates in situations when the assumed predefined bandwidths of kernels are too restrictive, and when the distribution is locally non-Gaussian (e.g., in skewed or uniform distribution). Recently, Kristan et.al [17] have proposed an online kernel density estimator, which uses the least assumptions in comparison to the related methods, but is restricted to one-dimensional problems.

1.1 Our approach

We propose a, new, multivariate online version of the kernel density estimator, which enables adaptation from only a single observation at a time. Our approach 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

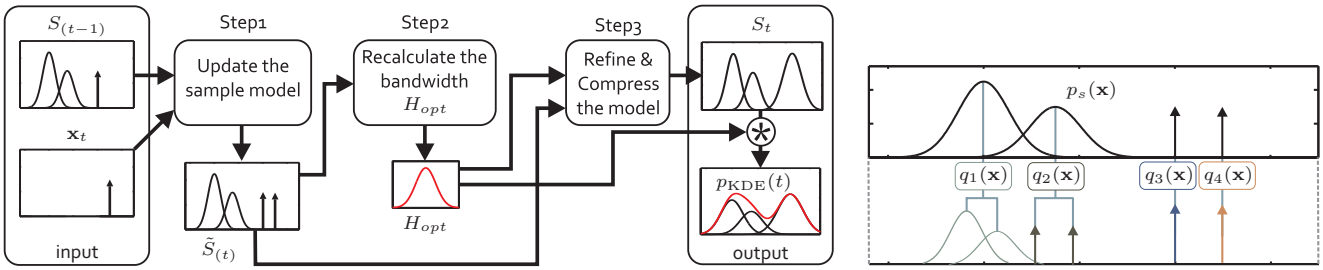


Figure 1: A three-step summary of the online KDE iteration (a) and an illustration of the sample model $\tilde{S}_{(t)}$ (sample distribution $p_s(\mathbf{x})$ along with its detailed model $\{q_i(\mathbf{x})\}_{i=1:4}$) (b).

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 second key idea is that we treat each new observation as a distribution in a form of a Dirac-delta function and we model the *sample distribution* by a 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 (Figure 1a): In the first step, we update the sample model with the observed data-point. In the second step, the updated model is used to recalculate the optimal bandwidth for the KDE – here, the main issue is how to calculate the bandwidth without having access to the previously observed individual samples. In the third step, 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. We therefore require a refinement algorithm to detect such events and to recover from them. The main issues here are: (i) how to devise an optimization which would efficiently compress the sample model, (ii) how to determine the extent of the allowed compression and (iii) how to recover from early compressions. To allow the recovery from early compression, we keep for every component another model of the data that generated that component in a form of a mixture model with at most two components (Figure 1b). 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 (see Figure 2).

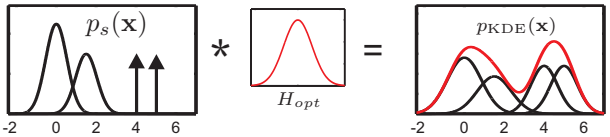


Figure 2: Calculation of the KDE $p_{\mathrm{KDE}}(\mathbf{x})$ from the sample distribution $p_s(\mathbf{x})$ (Dirac-deltas are depicted by upward arrows).

Our main contribution is the approach for multivariate online kernel density estimation (oKDE), which enables construction of a multivariate density estimate by observing only a single sample at a time and automatically adjusts its complexity. 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 estimates the distance between the KDE calculated before and after the compression. To this end, we propose an approximation to the multivariate Hellinger distance. We also propose a scheme to detect and recover from early over-compressions.

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. The compression and model refinement scheme is described in Section 4. The online KDE algorithm is presented in Section 5 and in Section 6 we compare it to some existing online and batch state-of-the-art KDE algorithms on examples of estimating distributions and on a classification example. We conclude the paper in Section 7.

2 The model definition

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 a 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

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

where

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

is a Gaussian kernel centered at μ with covariance matrix $\mathbf{\Sigma}$. 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) $\mathbf{H}$ (see Figure 2):

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

To maintain a low complexity of the KDE during online operation, the sample distribution $p_s(\mathbf{x})$ is compressed

from time to time. 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

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

where $p_{\text{s}}(\mathbf{x})$ is the *sample distribution* and $q_i(\mathbf{x})$ is a mixture model (with at most two components) for the i-th component in $p_{\text{s}}(\mathbf{x})$ (Figure 1b). To obtain a KDE, we need to compute the optimal bandwidth from all the observed samples, which are now summarized in the sample model $p_s(\mathbf{x})$. 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, $\mathbf{\Sigma}_{\text{s}i} = \mathbf{0}$ for all i) and the KDE (3) would be defined as $\hat{p}_{\text{KDE}}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i)$. The goal of all KDE methods is to determine the kernel bandwidth $\mathbf{H}$ such that the distance between the $\hat{p}_{\text{KDE}}(\mathbf{x})$ and the unknown pdf $p(\mathbf{x})$, that generated the data, is minimized. A classical measure used to define the closeness of the estimator $\hat{p}_{\text{KDE}}(\mathbf{x})$ to the underlying pdf is the *asymptotic mean integrated squared error* (AMISE), defined as ([27], pp.95-98),

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

where $\text{tr}\{\cdot\}$ is the trace operator, $\mathcal{G}_p(\mathbf{x})$ is a Hessian of $p(\mathbf{x})$, and $N_\alpha = (\sum_{i=1}^{N} \alpha_i^2)^{-1}$. If we rewrite the bandwidth matrix into $\mathbf{H} = h^2\mathbf{F}$ for $|\mathbf{F}| = 1$, and assume for now that $\mathbf{F}$ is known, then (5) is minimized at h_{opt}

$$h_{\text{opt}} = [d(4\pi)^{\frac{d}{2}} N_\alpha R(p, \mathbf{F})]^{-\frac{1}{d+4}}, \quad (6)$$

where the term

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

is a functional of the second-order partial derivatives of the unknown distribution $p(\mathbf{x})$. In principle, this functional could be estimated using the plug-in methods [27], however, these are usually numeric, iterative, and assume we have access to *all the observed samples*. 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 (see, eg., [27]) that $R(p, \mathbf{F})$ can be written in terms of expectations of the derivatives $\psi_{\mathbf{r}} = \int p^{(\mathbf{r})}(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$. We can then use the sample distribution $p_{\text{s}}(\mathbf{x})$ to obtain the following approximations

$$p(\mathbf{x}) \approx p_{\text{s}}(\mathbf{x}) \; ; \; p^{(\mathbf{r})}(\mathbf{x}) \approx p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x}), \quad (8)$$

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

$$p_{\mathbf{G}}(\mathbf{x}) = \phi_{\mathbf{G}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi_{\Sigma_{\text{g}j}}(\mathbf{x} - \mu_j). \quad (9)$$

The estimate $p_{\mathbf{G}}(\mathbf{x})$ plays a role of the so-called *pilot distribution* with covariance terms $\mathbf{\Sigma}_{\text{g}j} = \mathbf{G} + \mathbf{\Sigma}_{sj}$ and $\mathbf{G}$ is called the *pilot bandwidth*. Using the approximations in (8) and the Fact C.2.3. on p.181 in [27], we can approximate $R(p, \mathbf{F})$ by

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

Since $p_{\text{s}}(\mathbf{x})$ and $p_{\mathbf{G}}(\mathbf{x})$ are Gaussian mixture models, (10) can be calculated using only matrix algebra:

$$\hat{R}(p, \mathbf{F}, \mathbf{G}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \phi_{\mathbf{\Sigma}_{\text{g}i} + \mathbf{\Sigma}_{\text{s}j}}(\mu_{\text{g}i} - \mu_{\text{s}j}) \times [2\text{tr}(\mathbf{A}_{ij}\mathbf{B}_{ij}) + \text{tr}^2(\mathbf{F}\mathbf{C}_{ij})], \quad (11)$$

where for each pair (i, j) we have used the following definitions

$$\begin{aligned} \mathbf{A}_{ij} &= (\Sigma_{\text{g}i} + \Sigma_{\text{s}j})^{-1}, \\ \mathbf{B}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I} - 2(\mu_{\text{g}i} - \mu_{\text{s}j})(\mu_{\text{g}i} - \mu_{\text{s}j})^T \mathbf{A}_{ij}\}, \\ \mathbf{C}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I} - (\mu_{\text{g}i} - \mu_{\text{s}j})(\mu_{\text{g}i} - \mu_{\text{s}j})^T \mathbf{A}_{ij}\}. \end{aligned} \quad (12)$$

Derivation of (11,12) is rather laborious, and is based on the M. P. Wand's [26] study of an integral similar to (10). In the interest of space we removed the derivation here.

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 $\hat{\mathbf{\Sigma}}_{\text{smp}}$ to approximate both. First we resort to a practical assumption [27, 8] that the *structure* of the bandwidth $\mathbf{H}$ can be reasonably well approximated by the structure of the covariance matrix of the observed samples, and thus,

$$\mathbf{F} = \hat{\mathbf{\Sigma}}_{\text{smp}} |\hat{\mathbf{\Sigma}}_{\text{smp}}|^{-1/d}. \quad (13)$$

We estimate the pilot bandwidth $\mathbf{G}$ by a normal-scale rule [27]. The normal-scale provides a bandwidth that is optimal in AMISE sense if the unknown distribution $p(\mathbf{x})$ is in fact normal. While this assumption is too restrictive to directly estimate $\mathbf{H}$, it is admissable in practice for estimation of the bandwidths for the derivatives (see, eg. [27] page 71). The pilot bandwidth using the multivariate normal-scale rule for the derivative ([27], page 111) is given by

$$\mathbf{G} = \hat{\mathbf{\Sigma}}_{\text{smp}} \left(\frac{4}{(d+2)N_\alpha}\right)^{\frac{2}{d+4}}. \quad (14)$$

4 Compression of the sample model

The objective of the compression is to approximate the original N-component sample distribution

$$p_{\text{s}}(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x} - \mu_i) \quad (15)$$

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

$$\hat{p}_{\text{s}}(\mathbf{x}) = \sum_{j=1}^{M} \hat{w}_j \phi_{\hat{\mathbf{\Sigma}}_{sj}}(\mathbf{x} - \hat{\mu}_j), \quad (16)$$

such that the resulting KDE does not change significantly. Since a direct optimization (e.g., [17]) of the parameters in $\hat{p}_{\mathrm{s}}(\mathbf{x})$ can be computationally prohibitive, and prone to slow convergence even for moderate dimensions, we resort to a clustering-based approach. The main idea is to identify clusters of components in $p_{\mathrm{s}}(\mathbf{x})$, such that each cluster can be sufficiently well approximated by a single component in $\hat{p}_{\mathrm{s}}(\mathbf{x})$. Let $\Xi(M) = \{\pi_j\}_{j=1:M}$ be a collection of disjoint sets of indexes, which cluster $p_{\mathrm{s}}(\mathbf{x})$ into M sub-mixtures. The sub-mixture corresponding to indexes $i \in \pi_j$ is defined as

$$p_{\mathrm{s}}(\mathbf{x}; \pi_j) = \sum_{i \in \pi_j} w_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x} - \mu_i) \tag{17}$$

and is approximated by the j-th component $\hat{w}_j \phi_{\hat{\mathbf{\Sigma}}_{sj}}(\mathbf{x} - \hat{\mu}_j)$ of $\hat{p}_{\mathrm{s}}(\mathbf{x})$. The parameters of the $j-th$ component are defined by matching the first two moments (mean and variance) [3] of the sub-mixture:

$$\begin{aligned} \hat{w}_j &= \textstyle\sum_{i \in \pi(j)} w_i \ , \ \hat{\mu}_j = \hat{w}_j^{-1} \sum_{i \in \pi(j)} w_i \hat{\mu}_i \\ \hat{\Sigma}_j &= \hat{w}_j \textstyle\sum_{i \in \pi(j)} w_i (\mathbf{\Sigma}_i + \mu_i \mu_i^T) - \hat{\mu}_j \hat{\mu}_j^T . \end{aligned} \tag{18}$$

We seek to identify the clustering assignment $\Xi(M)$, along with M, such that the error of any cluster approximation *under the current KDE* does not exceed a prescribed value D_{th}. In other words, we want to minimize the following composite error function

$$\begin{aligned} E(\Xi(M)) = \max_{\pi_j \in \Xi(M)} \hat{E}(p_s(x; \pi_j), H_{opt}) \\ E(\Xi(M)) \leq D_{th}, \end{aligned} \tag{19}$$

where $\hat{E}(p_s(x; \pi_j), H_{opt})$ denotes the local error induced by the j-th cluster in the resulting KDE. We define this error next.

4.1 The local clustering error

Let $\mathbf{H}_{\mathrm{opt}}$ be the bandwidth estimated from the current sample distribution $p_{\mathrm{s}}(\mathbf{x})$, let $p_1(x) = p_{\mathrm{s}}(\mathbf{x}; \pi_j)$ be a sub-mixture (17) of $p_{\mathrm{s}}(\mathbf{x})$ and let $p_0(x)$ be the single-component approximation (18) of that sub-mixture. We define the local clustering error as the distance

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

between the corresponding *KDEs*

$$\begin{aligned} p_{1\mathrm{KDE}}(\mathbf{x}) &= p_1(\mathbf{x}) * \phi_{\mathbf{H}_{\mathrm{opt}}}(\mathbf{x}) \\ p_{0\mathrm{KDE}}(\mathbf{x}) &= p_0(\mathbf{x}) * \phi_{\mathbf{H}_{\mathrm{opt}}}(\mathbf{x}). \end{aligned}$$

In particular, we can quantify the distance between distributions using the Hellinger distance [22], which is defined as

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

Note that, while the Hellinger distance is a proper metric between distributions and is bounded to interval $[0, 1]$ (see [22]) it cannot be calculated analytically for the mixture models. We therefore calculate its approximation using the *unscented transform* [16]. For convenience, the derivation is given in the Appendix A.

4.2 Compression by hierarchical error minimization

In principle, the global minimization of $E(\Xi(M))$ (19) would require evaluation of all possible cluster assignments for the number of clusters 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 speeding up the EM algorithm [19], online visual category discovery [12] and controlled data compression with Gaussian mixture models [11].

In our implementation, the hierarchical clustering proceeds as follows. We start by splitting the entire sample distribution $p_{\mathrm{s}}(\mathbf{x})$ into two sub-mixtures using the Goldberger's [11] K-means algorithm for mixture models[1] with $K = 2$. Each sub-mixture is approximated by a single Gaussian and the sub-mixture which yields the largest local error $\hat{E}(p_{\mathrm{s}}(\mathbf{x}; \pi_j), H_{\mathrm{opt}})$ is further splitted into two sub-mixtures. This process is recursively continued until the largest local error is sufficiently small and the condition $E(\Xi(M)) \leq D_{th}$ in (19) fulfilled. This approach generates a binary tree with M leafs among the components of the sample distribution $p_{\mathrm{s}}(\mathbf{x})$, in which the leafs of the tree represent the clustering assignments $\Xi(M) = \{\pi_j\}_{j=1:M}$. Once the clustering $\Xi(M)$ is found, the compressed sample distribution $\hat{p}_{\mathrm{s}}(\mathbf{x})$ (16) is calculated using (17) and (18).

Recall that we keep track of a detailed model for each component in the sample distribution. The detailed model $\hat{q}_j(\mathbf{x})$ of the j-th component in the compressed model $\hat{p}_{\mathrm{s}}(\mathbf{x})$ 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

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

Then the required two-component detailed model $\hat{q}_j(\mathbf{x})$ is generated by splitting $\hat{q}_{j\mathrm{ext}}(\mathbf{x})$ into two sub-mixtures again using the Goldberger's K-means and each sub-mixture is approximated by a single Gaussian using (18).

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 bandwith $\mathbf{H}_{\mathrm{opt}}$ estimated at the time of compression. However, during online operation, new samples arrive, the sample distribution and $\mathbf{H}_{\mathrm{opt}}$ 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. Such an event can be detected through inspection of the *detailed model* of each component

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

in the sample distribution $p_{\rm s}(\mathbf{x})$. Particularly, the composite error (19) of each component in the sample distribution model can be evaluated against its detailed model to verify whether the compression error is stil low enough under the current KDE, i.e.,

$$E_{\rm detailed} = \max_{i=1:N} \hat{E}(q_i(\mathbf{x}), \mathbf{H}_{\rm opt}) \; ; \; E_{\rm detailed} \leq D_{th}. \quad (23)$$

Those components in $p_{\rm s}(\mathbf{x})$ for which $\hat{E}(q_i(\mathbf{x}), \mathbf{H}_{\rm opt}) > D_{th}$ 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 $w_i \phi_{\Sigma_i}(\mathbf{x} - \mu_i)$ be one of the new components. If the determinant of $\mathbf{\Sigma}_i$ is zero, then this component corresponds to a single datum 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 $\phi_{\mathbf{\Sigma}_i}(\mathbf{x} - \mu_i)$ along its principal axis into a two-component mixture, whose first two moments match those of the original component. More precisely, let $\mathbf{U}\mathbf{D}\mathbf{U}^{\rm T} = \mathbf{\Sigma}_i$ 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

$$q_i(\mathbf{x}) = \sum_{k=1}^{2} \alpha_k \phi_{\mathbf{\Sigma}_k}(\mathbf{x} - \mu_k), \quad (24)$$
$$\mu_1 = \mathbf{F}\mathbf{M} + \mu_i \; ; \; \mu_2 = \mathbf{F}\mathbf{M} - \mu_i,$$
$$\mathbf{\Sigma}_k = \mathbf{F}\mathbf{C}\mathbf{F}^{\rm T} \; ; \; \alpha_k = \frac{1}{2} w_i,$$

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

5 Online Kernel Density Estimation

Let us denote the model of the samples observed up to time-step $(t-1)$ as

$$\mathbf{S}_{{\rm model}(t-1)} = \{p_{{\rm s}(t-1)}(\mathbf{x}), \{q_{i(t-1)}(\mathbf{x})\}_{i=1:M_{t-1}}\}, \quad (25)$$

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

$$p_{{\rm s}(t-1)}(\mathbf{x}) = \sum\nolimits_{i=1}^{M_{t-1}} \alpha_i \phi_{\mathbf{\Sigma}_{{\rm s}i}}(\mathbf{x} - \mu_i). \quad (26)$$

Let N_{t-1} denote the number of all observed samples up to time-step $(t-1)$. At time-step t we observe a sample $\mathbf{x}_t$ and reestimate the sample model $\mathbf{S}_{{\rm model}(t)} = \{p_{{\rm s}(t)}(\mathbf{x}), \{q_{i(t)}(\mathbf{x})\}_{i=1:M_t}\}$ (and hence the KDE) in the following steps.

Step 1: Update the sample model. The number of observed samples is augmented, $N_t = N_{t-1} + 1$. Assuming all observations are equally probable, the sample distribution is updated by the new observation[2] as

$$\tilde{p}_{s(t)}(\mathbf{x}) = \frac{N_{t-1}}{N_t} p_{s(t-1)}(\mathbf{x}) + \frac{1}{N_t} \phi_{\mathbf{0}}(\mathbf{x} - \mathbf{x}_t). \quad (27)$$

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

Algorithm 1 : Compression of the sample model

Input:

$\tilde{\mathbf{S}}_{\rm model} = \{p_s(\mathbf{x}), \{\tilde{q}_i(\mathbf{x})\}_{i=1:\tilde{M}}\}$ ... the $\tilde{M}$-component sample model.

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

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

Output:

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

Procedure:

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

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

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

4: Select the cluster with the maximum local error: $\pi_j = \arg\max_{\pi_j \in \Xi(M)} \hat{E}(p_{\rm s}(\mathbf{x}; \pi_j))$

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

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

7: **end while**

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

9: For each j-th component in $\hat{p}_{\rm s}(\mathbf{x})$ 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 $\Xi(M)$.

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

$$\{\tilde{q}_{i(t)}(\mathbf{x})\}_{i=1:\tilde{M}_t} = \{\{q_i(\mathbf{x})\}_{i=1:M_{t-1}}, \tilde{q}_{\tilde{M}_t}(\mathbf{x})\}, \quad (28)$$

Thus yielding an updated sample model

$$\tilde{\mathbf{S}}_{{\rm model}(t)} = \{\tilde{p}_{{\rm s}(t)}(\mathbf{x}), \{\tilde{q}_{i(t)}(\mathbf{x})\}_{i=1:\tilde{M}_t}\}. \quad (29)$$

Step 2: Reestimate the bandwidth. The empirical covariance of the observed samples $\hat{\mathbf{\Sigma}}_{\rm smp}$ is calculated by approximating $\tilde{p}_{{\rm s}(t)}(\mathbf{x})$ by a single Gaussian using the moment matching (18). The new optimal bandwidth is then calculated (Section 3) as

$$\mathbf{H}_t = [d(4\pi)^{d/2} N_t \hat{R}(p, \mathbf{F}, \mathbf{G})]^{\frac{-1}{d+4}} \mathbf{F},$$
$$\mathbf{F} = \hat{\mathbf{\Sigma}}_{\rm smp} |\hat{\mathbf{\Sigma}}_{\rm smp}|^{-1/d},$$
$$\mathbf{G} = \hat{\mathbf{\Sigma}}_{\rm smp} \left(\frac{4}{(2+d)N_t}\right)^{\frac{2}{d+4}}, \quad (30)$$

and $\hat{R}(p, \mathbf{F}, \mathbf{G})$ is defined in (11).

Step 3: Refine and compress the model. After the current optimal bandwidth $\mathbf{H}_t$ has been calculated, the sample model $\tilde{\mathbf{S}}_{{\rm model}(t)}$ is refined and compressed by minimizing the composite error (19), using the Algorithm 1, into

$$\mathbf{S}_{{\rm model}(t)} = \{p_{{\rm s}(t)}(\mathbf{x}), \{q_{i(t)}(\mathbf{x})\}_{i=1:M_t}\}. \quad (31)$$

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 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 $M_{\text{t}} < \frac{1}{2}M_{\text{thc}}$, then it decreases $M_{\text{thc}} \leftarrow 0.6M_{\text{thc}}$.

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

$$p_{\text{s}(t)} = \sum_{i=1}^{M_t} \alpha_i \phi_{\mathbf{\Sigma}_{\text{s}i}}(\mathbf{x} - \mu_t), \tag{32}$$

and the current KDE is calculated according to (3):

$$\begin{aligned} p_{\text{KDE}t}(\mathbf{x}) &= p_{\text{s}(t)}(\mathbf{x}) * \phi_{\mathbf{H}_t}(\mathbf{x}) \\ &= \sum_{i=1}^{M_t} \alpha_i \phi_{\mathbf{\Sigma}_{\text{s}i}+\mathbf{H}_t}(\mathbf{x} - \mu_i). \end{aligned} \tag{33}$$

Since oKDE may be initialized using smaller number of samples than the sample dimensionality, the updates may suffer from singular covariances. To avoid these, the oKDE and the new data-point are first projected into a subspace using PCA, the updates are carried out there, and then the oKDE is backprojected into the original space.

6 Experimental study

6.1 Estimation of probability density functions

A significant difference between the online and batch algorithms is that the batch algorithms have access to all data, while the online algorithms discard the data and retain only their models. These models thus have to retain enough information to be able to successfully update when the new observations arrive. We have therefore compared the performance of the oKDE with several batch approaches. The first three were batch state-of-the-art KDE methods: Hall's et. al. [13] plug-in (implementation [15]), Murillo's et. al. [20] cross validation and Girolami's et. al. [10] reduced-set-density estimator. The fourth method was the batch EM algorithm with integrated regularization and model selection [9]. We have compared the oKDE also to the *adaptive mixtures* (AM) [23], which is essentially an online EM algorithm for Gaussian mixture models with automatic component-allocation mechanism. The parameters of the adaptive mixtures were set as in [6].

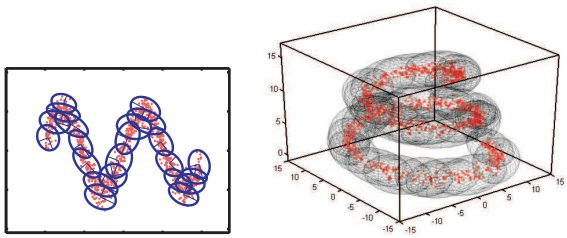
The experiment involved estimation of two separate distributions. The first was a 2D sinusoidal distribution defined by

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

with $\sigma_w = 0.2^2$ (Figure 3a). The second distribution was a 3D spiral distribution defined by the following model

$$\begin{aligned} \mathbf{x} = [(13 - \frac{1}{2}t)\cos(t), -(13 - \frac{1}{2}t)\sin(t), t]^T + \mathbf{w} \\ \mathbf{w} \sim \phi_{\mathbf{\Sigma}_\mathbf{w}}(\cdot) \; ; \; t \sim \mathcal{U}(0, 14), \end{aligned}$$

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 $[0, 14]$ (Figure 3b). A set of 1000 samples was generated from the model distribution. Batch models were estimated from all 1000 samples. In the online approaches, the first ten samples were used for initialization and the rest were added one at a time. The predictive performance of the models was evaluated by the average log-likelihood of additionally drawn 50,000 observations. The experiment was repeated 20 times. The performance of the batch an online methods after observing 50, 100, 400 and 1000 samples is shown in Table 1 and Table 2. Among the batch approaches, the CV performed on average best for 50 samples, but with increasing number of samples, the batch EM algorithm was superior. Among the online approaches, the oKDE consistently outperformed the AM in terms of the log-likelihood even for the values $D_{\text{th}} = 0.05$. Note also, that it also required less components to achieve better performance. The oKDE also consistently outperformed the Hall's batch KDE in terms of the log-likelihood while requiring significantly smaller number of components – e.g., after observing a thousand samples the oKDE required only rougly 15 percent of number of components compared to the Hall's batch KDE.



(a) (b)

Figure 3: Samples from the sinusoidal (a) and the spiral distribution (b) along with the models obtained by the $\text{oKDE}_{0.02}$.

6.2 Construction of an online classifier

We have compared the classification performance of the oKDE with three batch KDEs: the CV batch KDE [20], the reduced-set density estimator [10] initialized by the CV and the Hall's batch KDE [13]. For the baseline classification, we have applied a multiclass SVM with an rbf kernel [5]. We have used $D_{\text{th}} = 0.05$ in the oKDE. The methods were compared on a set of public classification problems [2] (Table 3). The classification performance of the KDE-based methods was tested using a simple Bayesian criterion

$$\hat{y} = \arg\max_l p_{\text{KDE}}(\mathbf{x}|c_l). \tag{34}$$

The parameter for the SVM kernel was determined separately in each experiment via cross validation on the training data set. The classification experiment was conducted via four-fold cross validation and the results are shown in Table 3. On average, the SVM and CV method produced best classification. The oKDE outperformed RSDE and slightly Hall's KDE batch method, and produced a comparable classification to the SVM and the CV. An important observation is that the oKDE outperformed, or produced comparable performance, to the batch methods, even though the oKDE was

Table 1: The average negative log-likelihood ($\mathcal{L}$) and the number of components in the model (N_{cmp}) w.r.t. the number of observed samples for the experiment with the sinusoidal distribution.

	50 samples		100 samples		400 samples		1000 samples	
Batch methods	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$
CV	**1.70±0.10**	50±0.0	**1.52±0.06**	100±0.0	1.37±0.02	400±0.0	1.37±0.01	1000±0.0
Hall	2.39±0.04	50±0.0	2.28±0.04	100±0.0	2.09±0.02	400±0.0	1.98±0.01	1000±0.0
RSDE	1.88±0.13	24±4.8	1.63±0.07	43±5.9	1.38±0.02	135±11.0	1.32±0.02	298±38.5
EM	1.90±0.13	6±1.0	1.67±0.11	8±1.3	**1.34±0.03**	10±2.3	**1.26±0.01**	11±1.4
Online methods	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$
AM	2.17±0.10	10±2.7	2.02±0.09	12±2.9	1.83±0.10	17±3.1	1.74±0.11	21±4.0
$\mathrm{oKDE}_{0.01}$	1.95±0.07	16±1.3	**1.80±0.04**	20±1.5	**1.56±0.02**	27±2.3	**1.44±0.01**	33±2.1
$\mathrm{oKDE}_{0.02}$	**1.94±0.06**	12±1.4	1.80±0.04	14±1.2	1.57±0.02	18±1.7	1.48±0.01	21±2.0
$\mathrm{oKDE}_{0.04}$	2.00±0.08	8±1.0	1.87±0.04	9±1.0	1.72±0.04	10±1.5	1.67±0.04	11±1.2
$\mathrm{oKDE}_{0.05}$	2.01±0.06	7±1.1	1.91±0.05	7±1.2	1.78±0.04	8±1.3	1.73±0.03	9±0.8

Table 2: The average negative log-likelihood $-\mathcal{L}$ and the number of components in the model (N_{cmp}) w.r.t. the number of observed samples for the experiment with the 3D spiral.

	50 samples		100 samples		400 samples		1000 samples	
Batch methods	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$
CV	**8.05±0.45**	50±0.0	**7.39±0.29**	100±0.0	**6.76±0.02**	400±0.0	**6.62±0.01**	1000±0.0
Hall	8.12±0.33	50±0.0	7.61±0.12	100±0.0	7.15±0.02	400±0.0	6.95±0.01	1000±0.0
RSDE	8.83±0.77	27±6.6	7.87±0.47	57±8.0	6.88±0.06	155±15.0	6.65±0.02	286±6.8
EM	10.36±1.37	6±0.8	9.55±1.07	11±0.9	7.05±0.11	21±1.4	6.58±0.02	22±2.0
Online methods	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$
AM	8.68±0.17	17±2.9	8.12±0.16	21±3.5	7.29±0.12	31±4.2	6.94±0.10	42±5.1
$\mathrm{oKDE}_{0.01}$	**8.08±0.39**	24±1.8	**7.52±0.10**	32±1.6	**6.97±0.02**	43±3.0	**6.75±0.01**	46±2.3
$\mathrm{oKDE}_{0.02}$	8.09±0.40	19±1.6	7.52±0.08	23±1.8	6.99±0.02	27±1.8	6.77±0.01	28±1.3
$\mathrm{oKDE}_{0.04}$	8.09±0.32	14±1.0	7.59±0.08	16±1.4	7.06±0.02	18±0.9	6.84±0.02	20±1.0
$\mathrm{oKDE}_{0.05}$	8.11±0.27	13±1.2	7.60±0.09	14±1.3	7.07±0.03	17±0.8	6.89±0.03	18±1.1

constructed by observing only a single sample at a time. In contrast, the SVM and batch KDEs optimized their structure by having access to all the samples. A further thing to note is that, with the exception of the Pima dataset, the oKDE's classification performance quite closely matched that of an SVM, eventhough the oKDE is in its nature reconstructive, while the SVM optimizes its structure to maximize discrimination.

7 Conclusion

We have proposed an approach for a kernel density estimation which can be applied in online operation (oKDE). The central point of the proposed scheme is that it maintains a compressed model of the observed samples and uses this model to compute the kernel density estimate of the underlying distribution. The oKDE automatically maintains its complexity thorough the process of compression and refinement. Experiments have shown that, in terms of probability density estimation, the oKDE produces comparable results to the state-of-the-art batch approaches and outperforms the online EM algorithm. In the experiment with classification problems the oKDE outperformed some batch KDEs and produced comparable classification to the state-of-the art batch KDE and the support vector machine. While the proposed oKDE is a contribution to the literature on kernel density estimation as such, parts of our approach can contribute to solutions of some other problems as well. The proposed unscented Hellinger distance may be used, for example, as a general metric in applications where one needs to compare mixtures of Gaussians (e.g., [11]). Recently, an approximate probability density estimator was proposed for visual tracking in [14]. The estimator is based on KDE, however, the kernel bandwidth is user predefined. Our bandwidth selection rule can be directly applied to that estimator to provide means of automatic bandwidth selection. In our future work we will apply the oKDE to estimation of non-stationary distributions and consider extensions to enhance its discrimination performance, which, we expect, will lead to online KDE-based probabilistic discriminative models.

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 [25], we define an *importance* distribution $p_0(\mathbf{x}) = \gamma(p_1(\mathbf{x}) + p_2(\mathbf{x}))$, which contains the support of both, $p_1(\mathbf{x})$ as well as $p_2(\mathbf{x})$, 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 $p_0(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_{\mathbf{\Sigma}_i}(\mathbf{x} - \mathbf{x}_i)$, and we rewrite (21) into

$$D^2(p_1, p_2) = \frac{1}{2} \int g(\mathbf{x}) p_0(\mathbf{x}) d\mathbf{x} =$$

Table 3: Average classification results (Recog.) and average number of components per class ($\mathrm{N_{cmp}}$) in the models. 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.

				$oKDE_{0.05}$		CV		RSDE		Hall		SVM	
dataset	N_S	N_D	N_C	Recog.	$\mathrm{N_{cmp}}$	Recog.	$\mathrm{N_{cmp}}$	Recog.	$\mathrm{N_{cmp}}$	Recog.	$\mathrm{N_{cmp}}$	Recog.	$\mathrm{N_{cmp}}$
Iris	150	4	3	97%	31	96%	38	97%	9	97%	38	96%	16
Pima	768	8	2	70%	162	72%	288	63%	15	67%	288	78%	162
Wine	178	13	3	99%	45	92%	45	83%	12	99%	45	98%	22
Letter	20000	16	26	95%	222	96%	613	55%	26	95%	613	96%	320

$$\frac{1}{2}\sum_{i=1}^{N} w_i \int g(\mathbf{x})\phi_{\Sigma_i}(\mathbf{x}-\mathbf{x}_i)d\mathbf{x}, \tag{35}$$

where we have defined $g(\mathbf{x}) = \frac{(\sqrt{p_1(\mathbf{x})}-\sqrt{p_2(\mathbf{x})})^2}{p_0(\mathbf{x})}$. Note that the integrals in (35) are simply expectations over a non-linearly transformed Gaussian random variable $\mathbf{X}$, and therefore admit to the unscented transform. According to [16] we then have

$$D^2(p_1,p_2) \approx \frac{1}{2}\sum_{i=1}^{N} w_i \sum_{j=0}^{2d+1} g({}^{(j)}\mathcal{X}_i)\,{}^{(j)}\mathcal{W}_i, \tag{36}$$

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

$$ {}^{(0)}\mathcal{X}_i = \mathbf{x}_i \;;\; {}^{(0)}\mathcal{W}_i = \frac{\kappa}{1+\kappa} $$

$$ {}^{(j)}\mathcal{X}_i = \mathbf{x}_i + s_j\sqrt{1+\kappa}(\sqrt{d\mathbf{\Sigma}_i})_j \tag{37}$$

$${}^{(j)}\mathcal{W}_i = \frac{\kappa}{2(1+\kappa)} \;;\; s_j = \begin{cases} 1 & ; \quad j \le d \\ -1 & ; \quad otherwise \end{cases} \tag{38}$$

with $\kappa = \max([0, m-d])$, and $(\sqrt{\mathbf{\Sigma}_i})_j$ is the j-th column of the matrix square root of $\mathbf{\Sigma}_i$. Concretely, let $\mathbf{UDU}^T$ be a singular value decomposition of covariance matrix $\mathbf{\Sigma}$, such that $\mathbf{U} = \{U_1, \ldots, U_d\}$ and $\mathbf{D} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_d\}$, then $(\sqrt{\Sigma})_k = \sqrt{\lambda_k}U_k$. In line with the discussion on the properties of the unscented transform in [16], we set the parameter m to $m = 3$.

Acknowledgement

This research has been supported in part by: Research program P2-0214 (RS), Research program P2-0095 (RS), TP-MIR SOVZO (Ministry of Defense of Republic of Slovenia), and EU FP7-ICTŰ215181-IP project CogX.

References

[1] O. Arandjelovic and R. Cipolla. Incremental learning of temporally-coherent gaussian mixture models. In *British Machine Vision Conference*, pages 759–768, 2005.

[2] A. Asuncion and D.J. Newman. UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.

[3] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*, chapter 11, pages 438–440. John Wiley & Sons, Inc., 2001.

[4] H. Bischof and A. Leonardis. View-based object representations using rbf networks. *"IVC"*, 19:619–629, 2001.

[5] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[6] J. Cwik and J. Koronacki. A combined adaptive-mixtures/plug-in estimator of multivariate probability densities. *Computational Statistics and Data Analysis*, 26:199Ű218, 1998.

[7] A. Declercq and J. H. Piater. Online learning of gaussian mixture models - a two-level approach. In *Intl.l Conf. Comp. Vis., Imaging and Comp. Graph. Theory and Applications*, pages 605–611, 2008.

[8] T. Duong and M. L. Hazelton. Plug-in bandwidth matrices for bivariate kernel density estimation. *Nonparametric Statistics*, 15(1):17Ű30, 2003.

[9] M. A. F. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):381–396, 2002.

[10] M. Girolami and C. He. Probability density estimation from optimally condensed data samples. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1253–1264, 2003.

[11] J. Goldberger and S. Roweis. Hierarchical clustering of a mixture model. In *Neural Inf. Proc. Systems*, pages 505–512, 2005.

[12] R. Gomes, M. Welling, and P. Perona. Incremental learning of nonparametric Bayesian mixture models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[13] P. Hall, S. J. Sheater, M. C. Jones, and J. S. Marron. On optimal data-based bandwidth selection in kernel density estimation. *Biometrika*, 78(2):263–269, 1991.

[14] B. Han, D. Comaniciu, Y. Zhu, and L. S. Davis. Sequential kernel density approximation and its application to real-time visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(7):1186–1197, 2008.

[15] A. Ihler and M. Mandel. Kernel density estimation toolbox for MATLAB. http://www.ics.uci.edu/ ihler/code/, 2007.

[16] S. Julier and J. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, Department of Engineering Science, University of Oxford, 1996.

[17] M. Kristan, D. Skočaj, and A. Leonardis. Online kernel density estimation for interactive learning. *Image and Vision Computing*, page submitted, 2009.

[18] A. Leonardis and H. Bischof. An efficient mdl-based construction of rbf networks. *Neural Networks*, 11(5):963 – 973, 1998.

[19] A. W. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In *Advances in Neural Information Processing Systems*, page 543Ű549, 1999.

[20] J. M. L. Murillo and A. A. Rodriguez. Algorithms for gaussian bandwidth selection in kernel density estimators. In *Neural Inf. Proc. Systems*, 2008.

[21] E. Parzen. On estimation of a probability density function and mode. *Annals of Math. Statistics*, 33:1065–1076, 1962.

[22] D. E. Pollard. *A user's guide to measure theoretic probability*. Cambridge University Press, 2002.

[23] C. E. Priebe and D. J. Marchette. Adaptive mixture density estimation. *Patt. Recogn.*, 26:771–785, 1993.

[24] M. Song and H. Wang. Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In *SPIE: Intelligent Computing: Theory and Applications*, pages 174–183, 2005.

[25] E. Veach and L. J. and Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Computer graphics and interactive techniques*, pages 419 – 428, 1995.

[26] M. P. Wand. Error analysis for general multivariate kernel estimators. *Nonparametric Statistics*, 2:1–15, 1992.

[27] M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman & Hall/CRC, 1995.

[28] K. Zhang and J. T. Kwok. Simplifying mixture models through function approximation. In *Neural Inf. Proc. Systems*, 2006.