Online Discriminative Kernel Density Estimator With Gaussian Kernels

Matej Kristan, *Member, IEEE,* Aleš Leonardis, *Member, IEEE,*

***Abstract*—We propose a new method for a supervised online estimation of probabilistic discriminative models for classification tasks. The method estimates the class distributions from a stream of data in form of Gaussian mixture models (GMM). The reconstructive updates of the distributions are based on the recently proposed online Kernel Density Estimator (oKDE). We maintain the number of components in the model low by compressing the GMMs from time to time. We propose a new cost function that measures loss of interclass discrimination during compression, thus guiding the compression towards simpler models that still retain discriminative properties. The resulting classifier thus independently updates the GMM of each class, but these GMMs interact during their compression through the proposed cost function. We call the proposed method the online discriminative Kernel Density Estimator (odKDE). We compare the odKDE to oKDE, batch state-of-the-art KDEs and batch/incremental support vector machines (SVM) on the publicly-available datasets. The odKDE achieves comparable classification performance to that of best batch KDEs and SVM, while allowing online adaptation from large datasets, and produces models of lower complexity than the oKDE.**

***Index Terms*—Online discriminative models , probability density estimation , Kernel density estimation , Gaussian mixture models.**

I. Introduction

Building discriminative models (classifiers) from streams of data is a central task of many applications in machine learning. In real-world environments, we may want to observe some process for an indefinite duration, while continually providing the best estimate of the model from the data observed so far. This generates the need for models that can be constructed in an *online* operation. In a strict online operation, we observe each data-point only once, update our model and then *discard* that data-point. Note that, to prevent an unbounded increase of the memory (and computational) requirements of the method, the data discarding is of crucial importance.

From a Bayesian perspective, we could construct an optimal classifier if we knew the exact underlying probability density functions (pdf) from which the data are sampled. In this respect, parametric *reconstructive* models based on the Gaussian mixture models, (GMM), (e.g., [1], [2], [3]) have been successfully applied in *batch* operation, i.e., in situations in which all the data is observed in advance. However, extension of the GMMs to online operation is not a straightforward task. Another drawback of the GMMs is that a bound on the number of components has to be specified in advance [1], [4], [3]. Improper choice of the number of components, may lead to models which fail to capture the complete structure of the underlying pdf and decrease the classifier's performance. In this paper we address the issue of online construction of a classifier through a Gaussian mixture model in which the complexity (the number of components) is automatically adjusted by considering the classifier's discrimination properties. The resulting method is called the online discriminative Kernel Density Estimator (odKDE).

A. Related work

An appealing property of the non-parametric methods, such as Parzen kernel density estimators (KDEs) [5], [6], is that they alleviate the problem of specifying the number of components in the GMM. They achieve this by treating each observation as a component in the mixture model and assuming all components have equal bandwidths (covariance matrices in case of Gaussian kernels). Indeed, several authors, e.g., [7], [8], [9], [10], [11], [12], have recently reported excellent performance of the KDE-based classifiers. The only free parameter in the KDEs is the bandwidth, and automatic estimation of this parameter is an active research area with many proposed solutions, e.g., [13], [14], [7], [15], [12].

While parameter-free estimation is attractive as such, the main drawback of the KDE-based methods is that model's complexity increases linearly with the observed data-points. To deal with this problem, several methods

M. Kristan is with the Faculty of Computer and Information Science and with the Faculty of Electrical Engineering, University of Ljubljana, Slovenia, e-mail: matej.kristan@fri.uni-lj.si.

A. Leonardis is with the School of Computer Science, University of Birmingham, United Kingdom and with the Faculty of Computer and Information Science, University of Ljubljana, Slovenia.

have been proposed to reduce the number of components (compress the model) either to a predefined value [16], [17], or to optimize some data-driven criteria [18], [19], [20], [21], [15]. Alternatively, Ozertem et. al. [22] have formulated the model compression as a clustering problem which can be solved through a mean-shift-based detection of modes in the estimated distribution. Recently, Rubio and Lobato [9] applied the non-stationary bandwidths from [7] to the compressed distribution, and reported improved performance.

Adapting batch methods to enable online operation is a nontrivial task. In contrast to the batch incremental models (e.g., [23], [24], [25]), who store and revisit all the data in multiple passes, the online models have to adapt from a (single) new data-point and then discard that data-point. The main difficulty therefore lies in maintaining sufficient information in the estimated models to generalize well to the yet unobserved data and adjusting their complexity as well as parameters without having access to all the observations simultaneously (future as well as past). There have been various attempts to extend the *reconstructive* GMMs to online operation, however, these either imply strong spatio-temporal constraints on the data [26], [27], assume constraints on the shape of the target distribution [28] or require tuning of parameters to a specific application [29]. Priebe and Marchette [30] proposed an online EM algorithm, that includes a heuristic for allocating new components. Kenji et. al. [31] proposed a similar approach to facilitate efficient online compression of data-streams by volume prototypes. Recently, we have proposed a non-parametric approach called the *online Kernel Density Estimator* (oKDE) [12]. The oKDE does not impose any of the above constraints but assumes only that the target pdf is sufficiently smooth and produces models with a high reconstructive performance. In [32] we have also considered a variant of the oKDE that allows adaptation from positive as well as negative examples. Unlike the related approaches, the oKDE [12] does not attempt to build a model of the target distribution directly, but rather maintains 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 a mixture of Gaussian and Dirac-delta functions. Each new data-point is added to the sample distribution as a Dirac-delta function and the sample distribution is compressed from time to time to keep its complexity low. The compression is implemented by hierarchical clustering, which approximates clusters of components with single Gaussians. The compression stops when the distance between the pdfs before and after compression becomes too great.

B. Our approach

A straight-forward extension of the oKDE [12] to online construction of classifiers would be to estimate the pdf of each class separately by the oKDE and construct a Bayesian classifer from the class pdfs. However, the oKDE is agnostic to the fact that we are estimating the pdfs for construction of a classifier, and completely ignores the class labels during online estimation of the pdfs. In fact, the oKDE compresses its class pdfs to simpler models by constraining the reconstruction loss. In practice this leads to overcomplex models for the task of classification. The high complexity has two undesired effects for the classification. First is the larger number of the model parameters that the method has to automatically estimate from the available data. Note that maintaining the number of components in the model sufficiently low, has a regularization effect, leading to smoother between-class boundaries which often result in improved classification performance [33]. The other effect is the redundancy of representation, which leads to redundant computations at classification. Namely, there is no need for having a detailed generative model in parts of the feature space that are irrelevant for the classification. From the classification standpoint, a good model can be as general as possible in the irrelevant parts of the feature space and detailed in the parts of the feature space that are close to the interclass boundary.

We propose a new KDE-based online approach to classifier estimation, which we call the online discriminative Kernel Density Estimator (odKDE). In our approach, we build class-wise pdfs in form of Gaussian mixture models by taking into account all the classes jointly. The online operation is composed of two main steps: (i) reconstructive update and (ii) discriminative compression. In the reconstructive update, the pdf of each class is updated independently from the others, similarly to the reconstructive updates in the oKDE [12]. However, in the discriminative compression, each class pdf is compressed by taking into account the pdfs of all remaining classes. We propose a new compression cost function that measures the changes of the resulting classifier's discrimination properties during the compression. This cost function allows each pdf to be compressed as much as possible as long as the compressed pdf does not induce a significant change in the classifiers posterior distribution (interclass boundary region). Our experiments show that the approach delivers low-complexity classifiers that can be estimated within the strict online estimation scenario.

Our approach is outlined in Figure 1, which shows steps in adaptation of a three-class classifier. The pdf

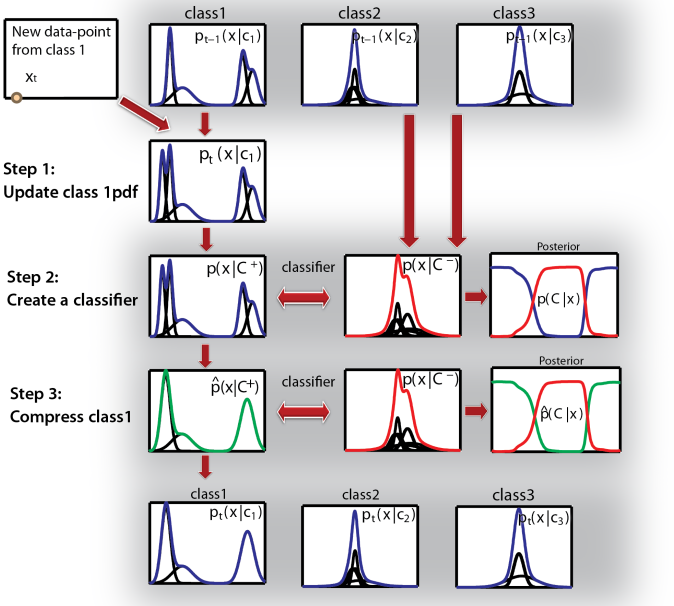

Fig. 1. Illustration of the main steps in the proposed online discriminative KDE. The example shows a three-class model in which the first class is updated by a new observation and compressed. While the distributions change significantly, the classifier's posterior does not.

of each class is modelled by a GMM. A data-point is observed for the first class and the corresponding four-component GMM is reconstructively updated into a new five-component GMM. The second and the third step illustrate the discriminative compression of the class 1. A two-class classifier is constructed in the second step. Class 1 represents the *positive example class* and the remaining classes are pooled together to form the *negative example class*. The positive example class is then compressed, such that the posterior distribution of the binary classifier does not change significantly. In Figure 1 we see that the first two and the last two components in the class 1 were compressed and the resulting GMM was simplified into a three-component model. This way get the updated and compressed discriminative model (bottom row in Figure 1).

The remainder of the paper is structured as follows. In Section II we define our model. In Section III we detail the oKDE-based reconstructive update and in Section IV we detail the compression algorithm. In Section V we define our new cost function that measures the discrimination loss and is designed to fit the compression algorithm. Section VII contains experimental study of the approach and we conclude the paper in Section VIII.

II. The Model Definition

Our goal is to estimate a classifier for K classes. In particular, we want to estimate the pdfs of all classes, and use these to create a Bayesian classifier. We can compactly write the classifier at time-step t as

$$\{p_t(\mathbf{x}|c_k), p_t(c_k)\}_{k=1:K} \tag{1}$$

where $p_t(\mathbf{x}|c_k)$ and $p_t(c_k)$ are the pdf and the prior for the class c_k, respectively. The prior can be estimated from the frequency of each class in the observed data. For online construction of the pdfs, we adapt the paradigm from the online Kernel Density Estimator (oKDE) [12] in that we continually estimate class-wise *sample distributions*. The sample distribution $q_t(\mathbf{x}|c_k)$ for the c_k-th class is modeled by a $N_t^{c_k}$ component mixture of Gaussians,

$$q_t(\mathbf{x}|c_k) = \sum_{i=1}^{N_t{}^{c_k}} w_i^{c_k} \phi_{\mathbf{\Sigma}_{\mathrm{si}}^{c_k}}(\mathbf{x} - \mu_i^{c_k}), \tag{2}$$

where

$$\phi_{\mathbf{\Sigma}}(\mathbf{x} - \mu) = (2\pi)^{-\frac{d}{2}} |\mathbf{\Sigma}|^{-\frac{1}{2}} e^{(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu))}, \tag{3}$$

is a Gaussian kernel centered at μ with covariance matrix $\mathbf{\Sigma}$, and d is the dimensionality of data-points. At time-step t, the sample distribution of the class c_k is thus defined by $N_t^{c_k}$ triplets of mixture weights, means and covariances, i.e., $\{w_i^{c_k}, \mu_i^{c_k}, \mathbf{\Sigma}_{\mathrm{si}}^{c_k}\}$. The Kernel Density Estimate of the pdf for the c_k-th class is then defined as a convolution of the sample distribution with a Gaussian kernel $\phi_{\mathbf{H}_t^{c_k}}(\mathbf{x})$,

$$p_t(\mathbf{x}|c_k) = \phi_{\mathbf{H}_t^{c_k}}(\mathbf{x}) * q_t(\mathbf{x}|c_k) = \sum_{i=1}^{N_t^{c_k}} w_i^{c_k} \phi_{\mathbf{\Sigma}_i^{c_k}}(\mathbf{x} - \mu_i^{c_k}), \tag{4}$$

where $\mathbf{\Sigma}_i^{c_k} = \mathbf{H}_t^{c_k} + \mathbf{\Sigma}_{\mathrm{si}}^{c_k}$, and $\mathbf{H}_t^{c_k}$ is the bandwidth (covariance matrix) of the convolution kernel estimated at time-step t. In the following we will omit the reference to classes, $(\cdot)^{c_k}$, in interest of clearer notations.

III. Online Reconstructive Update

In this section we will explain the updating of the pdf for a particular class. Assume that at time-step t we observe a data-point $\mathbf{x}_t$ for some class and want to update its corresponding pdf. If all the data-points are equally important (i.e., no temporal forgetting is involved), then the corresponding sample distribution is simply updated by augmenting it with a Dirac-delta function (a Gaussian with a zero covariance) centered at $\mathbf{x}_t$,

$$q_t(\mathbf{x}) = (1 - \frac{1}{N_t}) q_{t-1}(\mathbf{x}) + \frac{1}{N_t} \phi_{\mathbf{0}}(\mathbf{x} - \mathbf{x}_t), \tag{5}$$

where $N_t = N_{t-1}+1$ is the number of samples from the same class observed so far. The updated kernel density estimate is then defined by

$$p_t(\mathbf{x}) = \phi_{\mathbf{H}_t}(\mathbf{x}) * q_t(\mathbf{x}). \quad (6)$$

A crucial step in all KDE approaches is estimation of the kernel bandwidth $\mathbf{H}_t$ such that the distance between the KDE $p_t(\mathbf{x})$ and the (unknown) underlying distribution, that generated the data, is asymptotically minimized. While virtually all approaches to bandwidth estimation (e.g.,[34], [6], [13], [11], [15]) require access to all data-points, we have recently [12] proposed a solution that avoids this restriction and can use a Gaussian mixture model as defined in (2) instead. Following the derivations in [12], the bandwidth is calculated as

$$\mathbf{H}_t = \hat{\Sigma}_{\mathrm{smp}(t)}[d(4\pi)^{d/2}N_t\hat{R}]^{-\frac{2}{d+4}}, \quad (7)$$

where $\hat{\Sigma}_{\mathrm{smp}(t)}$ is the covariance of the observed samples from our class, and $\hat{R}$ is defined by

$$\begin{aligned}\hat{R} = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \phi_{\mathbf{A}_{ij}^{-1}}(\Delta_{ij}) \times \\ [2\mathrm{tr}(\hat{\Sigma}^2_{\mathrm{smp}(t)}\mathbf{A}_{ij}^2)[1-2m_{ij}] + \\ \mathrm{tr}^2(\hat{\Sigma}_{\mathrm{smp}(t)}\mathbf{A}_{ij})[1-m_{ij}]^2], \end{aligned} \quad (8)$$

where we have used the following definitions $\mathbf{A}_{ij} = (\mathbf{G} + \mathbf{\Sigma}_{\mathrm{si}} + \mathbf{\Sigma}_{\mathrm{sj}})^{-1}$, $\Delta_{ij} = \mu_i - \mu_j$, $m_{ij} = \Delta_{ij}^T\mathbf{A}_{ij}\Delta_{ij}$, $\mathbf{G} = \hat{\Sigma}_{\mathrm{smp}(t)}((d+2)N_t/4)^{-2/d+4}$.

IV. Compression

With the updates in (5), the model's reconstructive power increases, however, so does its complexity (i.e., the number of components). To maintain the complexity low, the sample distribution has to be compressed from time to time. In the interest of clearer notations, we will leave out the time indexes for now. The objective of the compression is to approximate the original N-component sample distribution

$$q(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x}-\mu_i) \quad (9)$$

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

$$\hat{q}(\mathbf{x}) = \sum_{m=1}^{M} \hat{w}_m \phi_{\hat{\mathbf{\Sigma}}_{sm}}(\mathbf{x}-\hat{\mu}_m), \quad (10)$$

such that the error induced by the compression does not increase significantly. Since a direct optimization (e.g., [32]) of the parameters in $\hat{q}(\mathbf{x})$ can be computationally prohibitive, and prone to slow convergence even for moderate number of dimensions, we resort to a clustering-based approach. The aim is therefore to identify clusters of components in $q(\mathbf{x})$, such that each cluster can be sufficiently well approximated by a single component in $\hat{q}(\mathbf{x})$. Let $\Xi(M) = \{\pi_m\}_{m=1:M}$ be a collection of disjoint sets of indexes, which cluster $q(\mathbf{x})$ into M sub-mixtures. The sub-mixture corresponding to the m-th cluster is defined as

$$q^{(m)}(\mathbf{x}) = \sum_{i\in\pi_m} w_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x}-\mu_i) \quad (11)$$

and is approximated by the m-th mixture component $\hat{w}_m\phi_{\hat{\mathbf{\Sigma}}_{sm}}(\mathbf{x}-\hat{\mu}_m)$ of $\hat{q}(\mathbf{x})$. The parameters of the m-th component are defined by matching the first two moments (mean and covariance) [35] of the sub-mixture:

$$\begin{aligned}\hat{w}_m = \textstyle\sum_{i\in\pi_m} w_i \ , \ \hat{\mu}_m = \hat{w}_m^{-1}\textstyle\sum_{i\in\pi_m} w_i\hat{\mu}_i \\ \hat{\Sigma}_{\mathrm{sm}} = \hat{w}_m^{-1}\textstyle\sum_{i\in\pi_m} w_i(\mathbf{\Sigma}_{\mathrm{si}} + \mu_i\mu_i^T) - \hat{\mu}_m\hat{\mu}_m^T. \end{aligned} \quad (12)$$

We therefore seek a clustering assignment $\Xi(M)$, such that the number of components in the resulting model is reduced, i.e., $M < N$, and that the clustering error remains sufficiently low:

$$\hat{M} = \arg\min_{M} E(\Xi(M)) \ , \ \text{s.t. } E(\Xi(\hat{M})) \le D_{\mathrm{th}}, \quad (13)$$

where $E(\Xi(\hat{M}))$ is the clustering error induced by clustering assignment $\Xi(\hat{M})$ with $\hat{M}$ clusters and D_{th} is a bound on that error.

A. Hierarchical compression

In principle, the global optimization of (13) 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.

In our implementation, we therefore first build a dendrogram among the centers of components in $q(\mathbf{x})$. This generates a binary tree in which each node represents one possible local clustering assignment π_m (Figure 2). Starting at the root node, we test if the distribution can be compressed into a single Gaussian without significantly increasing the error $E(\Xi(M))$. If that is not the case, we descend the tree, which effectively splits the distribution into a two-Gaussian approximation (one Gaussian per node). We then iteratively descend further down the tree, at each step along the node that maximally contributes to the clustering error $E(\Xi(M))$. We stop descending the tree once $E(\Xi(M))$ falls below a desired threshold. The corresponding $\hat{M}$ leafs of the tree represent the clustering assignments $\Xi(\hat{M}) = \{\pi_m\}_{m=1:\hat{M}}$. Once the clustering

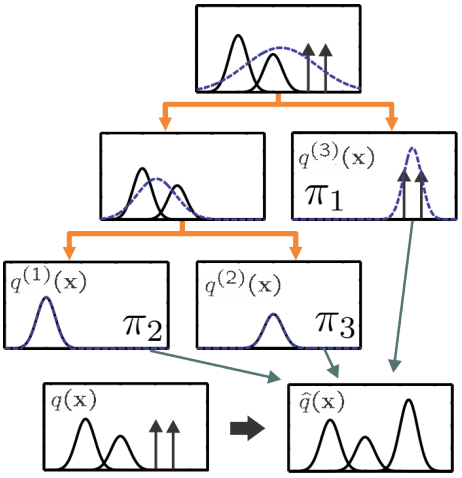

Fig. 2. Illustration of the hierarchical clustering. The components of the sample distribution $q(\mathbf{x})$ are hierarchically clustered to form a tree. Each of the resulting three leafs is approximated by a single Gaussian and together they form the compressed sample distribution $\hat{q}(\mathbf{x})$.

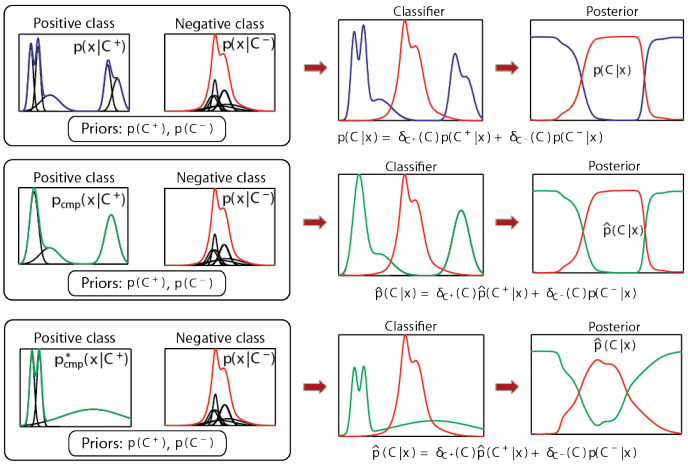

Fig. 3. First row shows a reference positive and negative class, along with the posterior $p(C|\mathbf{x})$. The second and the third rows show a valid and an invalid compression, respectively.

$\Xi(\hat{M})$ is found, the compressed sample distribution $\hat{q}(\mathbf{x})$ (10) is calculated using (11) and (12) and the corresponding compressed KDE $\hat{p}(\mathbf{x})$ can be calculated using (7) and (6). An illustration of the hierarchical clustering on a one-dimensional example is shown in Figure 2.

In order to efficiently implement the hierarchical compression, the cost of compression $E(\Xi(M))$ should be written as a sum over *local clustering errors*, which can be calculated for each cluster independently from the others. We propose such a cost function next.

V. Discriminative Compression Cost

Note that the compression error is always evaluated on the KDEs calculated from the (compressed) sample distributions. In the following we will assume that we want to compress the c_k-th class pdf $p(\mathbf{x}|c_k)$ into $\hat{p}(\mathbf{x}|c_k)$, while constraining the induced errors in the classifier. In particular, we want to keep the classification properties of the class models unchanged as much as possible. First we have to rewrite this model into a *classification model*. We consider the class c_k as a *positive example* class C^+, described by a mixture model $p(C^+|\mathbf{x}) \propto p(\mathbf{x}|c_k)p(c_k)$. Then we collect all the remaining classes to form a single *negative example* class C^-, $p(C^-|\mathbf{x}) \propto \sum_{j\setminus i} p(\mathbf{x}|c_j)p(c_j)$. The posterior over the resulting two-class model is then defined as

$$p(C|\mathbf{x}) = \delta_{C^+}(C)p(C^+|\mathbf{x}) + \delta_{C^-}(C)p(C^-|\mathbf{x}), \quad (14)$$

where $\delta_{C^*}(C)$ is a Dirac-delta function centered at C^*. The compressed counterpart of the posterior (14), is obtained by setting $\hat{p}(C^+|\mathbf{x}) \propto \hat{p}(\mathbf{x}|c_k)p(c_k)$:

$$\hat{p}(C|\mathbf{x}) = \delta_{C^+}(C)\hat{p}(C^+|\mathbf{x}) + \delta_{C^-}(C)p(C^-|\mathbf{x}). \quad (15)$$

From the classification point of view we can say that $p(\mathbf{x}|c_k)$ can be compressed into $\hat{p}(\mathbf{x}|c_k)$ as long as the posterior pdf over C before compression remains approximately unchanged after the compression. This idea is illustrated in Figure 3. The first row shows the positive and negative mixture model and the corresponding posterior of C over the feature space. The second the and the third row show a valid and invalid compression, respectively. The first compression is valid, since the posterior remains approximately unchanged. On the other hand, the posterior in the second compression changes significantly, which means that this compression changes classification properties of our classifier. In line with these observations we can conclude, that we require a distance measure between the posterior distribution before and after compression. We present such a measure next.

A. Distance between two classifiers

We define the distance between the posterior $p(C|\mathbf{x})$ and its compression $\hat{p}(C|\mathbf{x})$, at some value of $\mathbf{x}$, using the Hellinger distance [36],

$$D^2(p, \hat{p}|\mathbf{x}) \triangleq \frac{1}{2} \sum_{C\in[C^+,C^-]} (p(C|\mathbf{x})^{\frac{1}{2}} - \hat{p}(C|\mathbf{x})^{\frac{1}{2}})^2. \quad (16)$$

Note that (16) is just a point-wise distance evaluated at a particular value of $\mathbf{x}$. We therefore have to integrate it over the relevant values of $\mathbf{x}$, which gives the *expected*

Hellinger distance

$$\hat{D}^2(p,\hat{p}) = \int D^2(p,\hat{p}|\mathbf{x})p_0(\mathbf{x})d\mathbf{x}, \quad (17)$$

where the expectation is calculated over some *importance* distribution $p_0(\mathbf{x})$ that determines the regions of the feature space in which the compression effects are relevant for our classifier. We write the importance distribution as a mixture of the reference and the compressed distribution, i.e., $p_0(\mathbf{x}) = 0.5p(\mathbf{x}|c_k) + 0.5\hat{p}(\mathbf{x}|c_k)$. For a particular clustering $\Xi(M) = \{\pi_m\}_{m=1:M}$, we can rewrite $p_0(\mathbf{x})$ in terms of clustered components

$$p_0(\mathbf{x}) = \sum_{m=1}^{M} p_0^{(m)}(\mathbf{x}), \quad (18)$$

where we have defined $p_0^{(m)}(\mathbf{x}) = 0.5p^{(m)}(\mathbf{x}|c_k) + 0.5\hat{p}^{(m)}(\mathbf{x}|c_k)$, and

$$p^{(m)}(\mathbf{x}) = \sum_{i\in\pi_m} w_i\phi_{\mathbf{\Sigma}_i}(\mathbf{x}-\mu_i),$$
$$\hat{p}^{(m)}(\mathbf{x}) = \hat{w}_m\phi_{\hat{\mathbf{\Sigma}}_m}(\mathbf{x}-\hat{\mu}_m). \quad (19)$$

With these definitions, we can rewrite (17) into

$$D_{\mathrm{e}}^2(p,\hat{p}) = \sum_{m=1}^{M}\int D^2(p,\hat{p}|\mathbf{x})p_0^{(m)}(\mathbf{x})d\mathbf{x}. \quad (20)$$

The integrals in (20) are expectations over $p_0^{(m)}(\mathbf{x})$ and can be approximated by

$$\int \hat{D}^2(p,\hat{p})p_0^{(m)}(\mathbf{x})d\mathbf{x} \approx \int \hat{D}^2(p^{(m)},\hat{p}^{(m)})p_0^{(m)}(\mathbf{x})d\mathbf{x}. \quad (21)$$

Note that the approximation (21) is valid only when the relation $\hat{D}^2(p,\hat{p}) \approx D^2(p^{(m)},\hat{p}^{(m)}|\mathbf{x})$ approximately holds for those values of $\mathbf{x}$ for which the value of the pdf $p_0^{(m)}(\mathbf{x})$ is significant. This in not true for the general forms of $p_0^{(m)}(\mathbf{x})$, $p^{(m)}$ and $\hat{p}^{(m)}$. However, our compression scheme compresses the pdfs by verifying a sequence of clusterings. As a result the compressed pdfs have components that tend to form clusters. At the same time, the hierarchical compression evaluates the compression error at exactly these clusters and since the $p_0^{(m)}(\mathbf{x})$ captures a single potential cluster by design, the approximation (21) is valid for our application. With (21) we can therefore approximate (20) as

$$D_{\mathrm{e}}^2(p,\hat{p}) \approx \sum_{m=1}^{M}\int D^2(p^{(m)},\hat{p}^{(m)}|\mathbf{x})p_0^{(m)}(\mathbf{x})d\mathbf{x}, \quad (22)$$

which means that the compression cost function $D_{\mathrm{e}}^2(p,\hat{p})$ decomposes into a sum of independent contributions from separate clusters and can be as such directly applied in the hierarchical clustering algorithm from Section IV-A. One remaining issue is that since $p_0^{(m)}(\mathbf{x})$ are Gaussian mixture models, the integrals in (22) do not have an analytical solution. We overcome this issue by a simplified version of the unscented transform [37]. We simplify the mixture model $p_0^{(m)}(\mathbf{x})$ by setting the covariances of its components to zero, effectively approximating it by a mixture of weighted Dirac-delta functions. Each m-th integral in (22) then reduces into the following weighted sums

$$D_m^2 = \sum_{j\in\pi_m} w_j^{(m)} D^2(p,\hat{p}|\mathbf{x}_j^{(m)}), \quad (23)$$

and we can write our discrimination cost function as the expected Hellinger distance, written in terms of the local clustering errors, as

$$E(\Xi(M)) = (\sum_{m=1}^{M} D_m^2)^{1/2}. \quad (24)$$

Note that since the Hellinger distance is a metric constrained to interval between zero and one, small values, i.e., $E(\Xi(M)) \approx 0$, mean that the classification properties do not change during compression, while $E(\Xi(M)) = 1$ implies a maximal change. Since the metric is constrained, it is fairly easy to specify a threshold that generally applies for various input data. This is a practical advantage over some other unconstrained distance measures such as the Kullback Liebler divergence.

VI. Online Discriminative KDE

In this section we summarize the main steps of the online discriminative Kernel Density Estimator (odKDE) and discuss its implementation. Our goal is to continually update a K-class classifier from a stream of labelled data-points. Each class is modelled by a probability density function, which is continually updated. Our model at time-step $t-1$ is therefore a collection of K *sample distributions* along with their priors:

$$\{q_{t-1}(\mathbf{x}|c_k), p_{t-1}(c_k)\}_{k=1:K}. \quad (25)$$

At time-step t we observe a set of labelled observations $\{\mathbf{z}_i, c_i\}_{i=1:I}$ and update the model through a reconstructive update (Section III), followed by a discriminative compression (Section IV) which uses the error function defined in (24). As a result we get the updated model $\{q_t(\mathbf{x}|c_k), p_t(c_k)\}_{k=1:K}$. Note that the compression step is a crucial difference between the odKDE and the oKDE from [12]. The oKDE compresses the distributions independently and measures compression cost via the difference between the pdf of each class before and after compression. In contrast, the odKDE considers all

distributions jointly and allows compressions that do not induce significant changes of the joint *posterior* distribution. In the classification phase, a new observation $\mathbf{z}$ is classified into a class $\hat{c}$ by applying the Bayesian rule

$$\hat{c} = \arg\max_{c_k} p_t(\mathbf{z}|c_k)p_t(c_k), \tag{26}$$

where the class likelihoods are calculated by the KDE (4), i.e., by the convolution of the sample distributions with their respective kernels,

$$p_t(\mathbf{x}|c_k) = \phi_{\mathbf{H}_t^{(c_k)}}(\mathbf{x}) * q_t(\mathbf{x}|c_k). \tag{27}$$

Note that the classification (26) is carried out as the maximum a posteriori estimation of the most likely class. Such estimation can be hampered in cases when the class priors $p_t(c_k)$ are not correctly estimated. From a Bayesian point of view, with increasing the number of samples, these priors should converge to true priors and hence the Bayes-optimal classifier (26). However, in situations where the number of the observed samples does not reflect the true prior distributions on the classes, the classifier may become imbalanced. This can be remedied by setting the priors of all classes to a uniform distribution.

As in [32], [12], we do not invoke the compression after each update in our implementation. The compression is rather called after some threshold on number of components M_{th} in the sample distribution has been exceeded. Note that this threshold does not determine the number of components in the final model, but 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 [12]: If the number of components N_t still exceeds M_{th} after the compression, then the threshold increases $M_{\text{th}} \leftarrow 1.5M_{\text{th}}$, otherwise, if $N_t < \frac{1}{2}M_{\text{th}}$, then it decreases $M_{\text{th}} \leftarrow 0.6M_{\text{th}}$.

Recall from Section IV that compression clusters the sample distribution of a particular class into clusters, such that the discrimination cost for that class does not exceed a prescribed threshold D_{th}. In practice we wish to constrain the maximum discrimination cost $D_{\max}$ over all classes together. We therefore set the threshold for each class as the overall maximum discrimination cost divided by the number of classes: $D_{\text{th}} = D_{\max}/K$. The online discriminative Kernel Density Estimator[1] is summarized in Algorithm 1.

[1]A reference Matlab code is available from the authors' homepage http://www.vicos.si/People/Matejk.

Algorithm 1 : The online discriminative Kernel Density Estimator

Require:

$\{q_{t-1}(\mathbf{x}|c_k), p_{t-1}(c_k)\}_{k=1:K}$ ... the input models.

$\{\mathbf{z}_i, c_i\}_{i=1:I}$ ... labelled observations.

Ensure:

$\{\hat{q}_t(\mathbf{x}|c_k), p_t(c_k)\}_{i=1:K}$ ... the output models.

Procedure:

Step 1 – Reconstructive update:

for $i = 1 : I$ **do**

  Update the prior $p_t(c_i)$.

  Update the sample distribution of class c_i, $q_{t-1}(\mathbf{x}|c_i)$, with the data-point $\mathbf{z}_i$ into $q_t(\mathbf{x}|c_i)$ according to (5).

  Calculate the new bandwidth for class c_i, $\mathbf{H}_t^{(c_i)}$, from $q_t(\mathbf{x}|c_i)$ according to (7).

end for

Calculate the KDEs for all $k = 1 : K$ classes from the respective sample distributions using (4), i.e., $p_t(\mathbf{x}|c_k) = \Phi_{\mathbf{H}_t^{(c_k)}}(\mathbf{x}) * q_t(\mathbf{x}|c_k)$.

Step 2 – Discriminative compression:

for $k = 1 : K$ **do**

  If the number of components in $q_t(\mathbf{x}|c_k)$ is lower than $M_{\text{th}}^{(c_k)}$ then $\hat{q}_t(\mathbf{x}|c_k) \leftarrow q_t(\mathbf{x}|c_k)$ and skip the next step.

  Compress $q_t(\mathbf{x}|c_k)$ into $\hat{q}_t(\mathbf{x}|c_k)$ by hierarchical merging of components (Section IV), such that the compression cost does not exceed a predefined threshold, i.e., $E(\Xi(\hat{M})) \leq D_{\text{th}}$, and update the threshold on the number of components $M_{\text{th}}^{(c_k)}$ in c_k-th class.

end for

VII. Experimental study

Two sets of experiments were performed to demonstrate properties of the odKDE. The first set of experiments was conducted on a simulated data and the second set of experiments was conducted on the publicly-available datasets. Note that the only parameter in the odKDE is the compression threshold that determines the maximum allowed discriminative cost at a single compression step. In all our experiments we have used a constant compression threshold $D_{\max} = 0.12$ (Section VI). All experiments were run on a 2.7GHz laptop with 3GB RAM.

A. Illustrative experiment with synthetic data

Recall that the odKDE simplifies its model as much as possible as long as the simplifications do not significantly change the classifier's properties. In contrast,

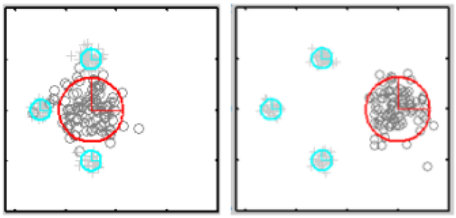

Fig. 4. The reference models along with the sampled data points from the first variant of the experiment with simulated data (left) and from the second variant of the experiment (right). Dark (red) and bright (cyan) lines denote the models of class 1 and class 2, respectively.

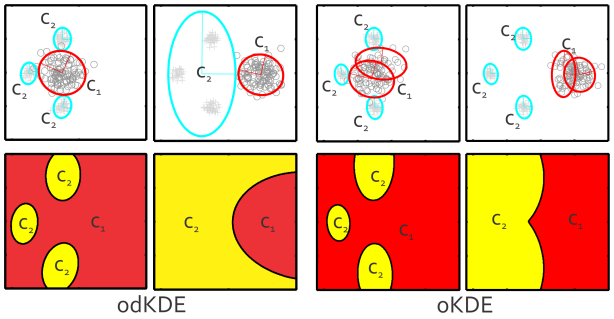

Fig. 5. The models estimated by the odKDE (first two columns) and the models estimated by the oKDE (second two columns). Upper row shows the estimated models along with the observed data-points. Dark (red) and bright (cyan) lines denote class 1 class 2, respectively. The lower row shows the classification decision boundaries for each model. Dark (red) color corresponds to class 1, while the bright (yellow) color corresponds to class 2.

the oKDE [12] allows only simplifications that do not change the class pdfs significantly. To illustrate this point, we have created an experiment with simulated data from two classes. The first class model was a three-component GMM, with components arranged in a triangle, while the second class model was a single Gaussian (Figure 4). Two sets of 2D samples were drawn from two models and used to estimate the two-class classifier. The first 10 samples from each set were used for initialization and additional 100 were used for updating with one sample at a time. Additional 2000 samples were drawn from the generative models and were classified using the estimated models. The portion of correctly classified samples (classification score) was used as a quantitative measure of performance. For reference, these samples were also classified by the original models. We have conducted these experiments for odKDE and for oKDE in two variants. In the first variant, the Gaussian from the second class was positioned between the Gaussians of the first class. In the second variant, the Gaussian from the second class was positioned away from the Gaussians of the first class (Figure 4).

In both variants of the experiment, the odKDE as well as the oKDE have achieved maximal classification performance, which was equal to the classification on the original models (98% and 100% for the first and second variant, respectively). The estimated models are shown in Figure 5. Note that there is a substantial difference in the way the observed data affects the structure and complexity of the classifiers constructed by the odKDE and the oKDE. In both variants of the experiment, the oKDE has constructed a classifier of same complexity (i.e., five components). The reason is that the difference in the two variants was merely the position of the second class Gaussian, and since the oKDE updates a model of a single class independently of the others it is invariant to such changes. In contrast, the odKDE considers the models of all classes jointly during the compression. As a result, the model complexity varied between the two variants. In the first variant, the odKDE estimated a model with three components, which is the minimal number of components that does not hamper the classification performance. In the second variant, approximating all three components from the first class by a single Gaussian does not change the classifier's performance and that was the complexity estimated by the odKDE. Note that the decision boundaries of the classifiers estimated by the odKDE and oKDE do differ (see Figure 5). However, from a classification point of view, these boundaries are equivalent, since they similarly classify *the relevant part* of the feature space (i.e., the part which is populated by the two classes).

B. Experiments with real-world datasets

To evaluate the odKDE's performance on real data sets, we have compared it to several related methods. These were the online reconstructive KDE, oKDE [12], and three state-of-the-art batch KDEs: the cross-validation (CV) KDE [10], the reduced-set density estimator [19] (RSDE) initialized by the CV, and the Hall KDE [38] (Hall). For the baseline classification, we have applied a batch multiclass support vector machine (SVM) with an RBF kernel [39], an incremental SVM [23], [40] (iSVM) and an online SVM from [25], [41] (oSVM). The methods were compared on a set of public classification problems [42] (Table I). All datasets were pre-scaled such that the standard deviation along each of the features was one. The data in each dataset were randomly reordered, 75% were used for training and the rest for testing. For all data-sets we have generated twelve such random partitionings.

The odKDE, oKDE, iSVM and oSVM were initialized for each class using the first 10 samples and the rest were added one at a time. The parameter for the batch SVM

TABLE I
PROPERTIES OF THE DATA-SETS 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
Wine	178	13	3
Breast cancer (BCW)	285	30	2
Pima	768	8	2
Yeast	1484	8	10
WineRed	1599	11	6
Steel plates (Plates)	1941	27	7
Image segmentation (Seg)	2310	18	7
WineWhite	4898	11	7
Letter	20000	16	26
Skin	307699	3	2
CovType	543441	10	7

kernel was determined separately in each experiment via cross validation on the training dataset. Since oSVM and iSVM were used in an online scenario, the kernel parameters were set in advance. Note that the iSVM and oSVM are indeed incremental in that they allow adding one sample at a time, and reestimate their parameters at each time-step. However, they still store all the observed samples for optimization of their parameters, which leads to slow computation and significant memory usage. Because of this and some numerical stability issues that we encountered, we have used a linear kernel in the iSVM. The oSVM guaranties a bound on the number of support vectors for the kernels that map into a finite feature space. We have therefore chosen a second-order polynomial kernel for the oSVM.

Table III summarizes the classification scores of the models after observing all the samples and shows the average number of components per class in the models, averaged over all repetitions of the experiment. The number of components refers to the number of Gaussians in the KDEs, while in case of SVMs this number refers to the number of support vectors. Note that we were unable to perform the experiments for the larger datasets in case of CV, RSDE, Hall as well as for iSVM and oSVM in some cases due to lack of computer memory. This is a good demonstration of the strict requirement of the online methods. Namely, in order to cope with large sequences of data, the algorithms should not store all the datapoints and should carry out the adaptations only with their compressed models. These constraints are only met by the odKDE and oKDE, and are hence the only two methods that were able to process the largest of datasets – *Skin* (more than 300.000 datapoints) and *CovType* (more that 500.000 datapoints). In terms of classification accuracy, the SVM-based methods have performed very well and sometimes slightly better than the odKDE. The reason is that SVM is a batch method and as such it revisits all the data-points during the optimization of its parameters (i) in terms of cross-validation for setting the kernel parameters, (ii) as well as in the optimization of the support vector weights (this also holds for the incremental SVMs). In contrast, a strict online approach observes a data-point only once, updates the compressed model and discards the data. Therefore, it is reasonable to assume that the batch methods will perform best in terms of classification (however, not in terms of model complexity) in situations that allow revisiting the datapoints. For the same reason, however, the batch approaches are not suitable for situations in which the data arrives in streams. Incremental versions of the SVM alleviate this issue a bit by adapting the SVM weights when the new datapoint arrives without repeating the entire computations from the previous time-steps. However, when the number of datapoints significantly increased, even these methods eventually failed, since they stored all the observed datapoints to perform the required updates. On the other hand, in most cases, the odKDE delivered comparable classification performance to the best batch method per dataset, while the model complexity was significantly lower and was also able to handle extremely large datasets.

For better orientation we also provide the times required by online methods for a single-sample update per class, averaged over the last twenty samples in Table II. Caution has to be taken in the interpretation of these results, since most of the implementations are research code and were not optimized for speed.

TABLE II
AVERAGE TIMES (IN SECONDS) PER SINGLE-SAMPLE UPDATES OVER THE LAST 20 UPDATES. SEE TEXT FOR DETAILS.

dataset	odKDE	oKDE [12]	iSVM [23]	oSVM [25]
Iris	0.004	0.038	0.001	0.001
Pima	0.007	0.034	0.012	0.001
Wine	0.005	0.036	0.003	0.002
WineRed	0.098	0.076	0.052	0.002
WineWhit	0.525	0.039	0.996	0.007
Letter	0.268	0.087	0.220	0.117
BCW	0.008	0.044	0.004	0.004
Seg	0.021	0.074	0.013	0.008
Plates	0.134	0.262	0.056	0.014
Yeast	0.395	0.038	0.049	0.003
Skin	0.004	0.051	/	0.093
CovType	0.849	0.074	/	/

TABLE III
AVERAGE CLASSIFICATION RESULTS [%] ALONG WITH THE NUMBER OF COMPONENTS WRITTEN IN PARENTHESIS. THE SYMBOL / INDICATES THAT THE CLASSIFIER COULD NOT BE ESTIMATED DUE TO MEMORY LIMITATIONS.

dataset	odKDE	oKDE [12]	CV [10]	RSDE [19]	Hall [38]	SVM [39]	iSVM [23]	oSVM [25]
Iris	97(3)	97(26)	96(38)	96(11)	97(38)	96(16)	97(10)	99(76)
Pima	74(14)	72(46)	72(288)	65(48)	74(288)	78(163)	69(36)	76(576)
Wine	96(3)	94(43)	92(44)	91(37)	96(44)	96(24)	95(25)	93(89)
WineRed	64(43)	64(42)	64(200)	44(31)	66(200)	63(173)	30(65)	58(400)
WineWhit	60(87)	55(37)	62(525)	25(41)	62(525)	60(473)	28(66)	54(1049)
Letter	96(24)	96(60)	96(577)	53(24)	95(577)	96(311)	84(136)	86(1154)
BCW	96(4)	94(100)	96(214)	94(188)	91(214)	97(52)	84(330)	97(427)
Seg	94(9)	93(44)	94(248)	78(23)	94(248)	94(83)	95(100)	95(496)
Plates	70(34)	72(61)	72(208)	53(38)	65(208)	76(146)	67(267)	73(416)
Yeast	53(42)	49(29)	49(111)	36(41)	25(111)	60(91)	54(36)	60(222)
Skin	100(3)	99(6)	/	/	/	100(169)	/	98(6)
CovType	71(123)	66(13)	/	/	/	/	/	/

Table IV shows the classification results only for the odKDE and the oKDE, where we also denoted by the asterisk symbol $()^*$ cases when the difference in classification performance was statistically significant. On average we observe a 1:6 reduction in the model's complexity at improved or comparable performance for the odKDE. In the case of Wine dataset, the complexity of the odKDE is only 7% of the oKDEs complexity at improved classification performance. In BCW dataset the performance is improved for odKDE, while the complexity reduces from 100 components (oKDE) to merely 4. Only for the *WineWhit*, *Yeast* and *CoverType* did the odKDE produce model whose complexity was larger than those produced by the oKDE. Note, however, that in those cases the classification performance of the odKDE significantly surpassed that of the oKDE. The reason for this difference is exactly in the way the odKDE compresses, simplifies, its models. Since the oKDE's compression is agnostic of the classifier that it is estimating, the odKDE considers all the classes jointly and simplifies the models as long as the classifier's classification properties do not change. This constraint prevented the odKDE to oversimplify the models, since further simplifications would change the classifier. The result was an improved performance compared to the oKDE.

TABLE IV
AVERAGE CLASSIFICATION RESULTS ALONG WITH $\pm$ ONE STANDARD DEVIATION. THE AVERAGE NUMBER OF COMPONENTS PER MODEL $\pm$ ONE STANDARD DEVIATION ARE GIVEN IN PARENTHESIS. THE $()^*$ DENOTES STATISTICALLY SIGNIFICANT IMPROVEMENT OF CLASSIFICATION.

dataset	odKDE	oKDE [12]
Iris	$97 \pm 3(3 \pm 1)$	$97 \pm 3(27 \pm 2)$
Pima	$74 \pm 2(14 \pm 6)^*$	$72 \pm 3(44 \pm 5)$
Wine	$96 \pm 3(3 \pm 1)^*$	$94 \pm 3(43 \pm 1)$
WineRed	$64 \pm 2(43 \pm 3)$	$64 \pm 1(42 \pm 2)$
WineWhit	$60 \pm 2(87 \pm 5)^*$	$55 \pm 1(37 \pm 2)$
Letter	$96 \pm 0(24 \pm 2)$	$96 \pm 0(60 \pm 1)$
BCW	$96 \pm 2(4 \pm 4)^*$	$94 \pm 3(100 \pm 6)$
Seg	$94 \pm 1(9 \pm 1)^*$	$93 \pm 1(44 \pm 3)$
Plates	$70 \pm 3(34 \pm 3)$	$72 \pm 1(61 \pm 3)^*$
Yeast	$53 \pm 2(42 \pm 3)^*$	$49 \pm 3(29 \pm 1)$
Skin	$100 \pm 0(3 \pm 0)^*$	$99 \pm 0(6 \pm 0)$
CovType	$71 \pm 4(123 \pm 24)^*$	$66 \pm 5(13 \pm 1)$

Note that the odKDE jointly estimates the appropriate number of components as well as the component parameter from the observed data. Figure 6 shows the variation of the classification score and the number of components with respect to the number of observations for the dataset *Plates* and *Letter*. We can see from the Table IV that the oKDE outperformed the odKDE only for he *Plates* dataset in classification by two percent at statistically significant difference. A reasonable explanation for this result would be that the odKDE did not properly estimate the model's complexity up to the last observed datapoint, which resulted in reduced classification performance in comparison to the oKDE. Looking at the results for the *Plates* dataset in Figure 6 it is clear that the odKDE's model complexity was gradually increasing up to the last datapoint with the number of observations. At the same time, we can verify that its classification performance was also increasing. It is therefore reasonable to assume that the model's complexity would increase with further observations, and thereby increase the classification performance. Note that, during the online adaptations, the number of components may stop increasing, but the components themselves are further adapted to the data. In practice, we have observed a general trend that the number of components initially significantly increases in

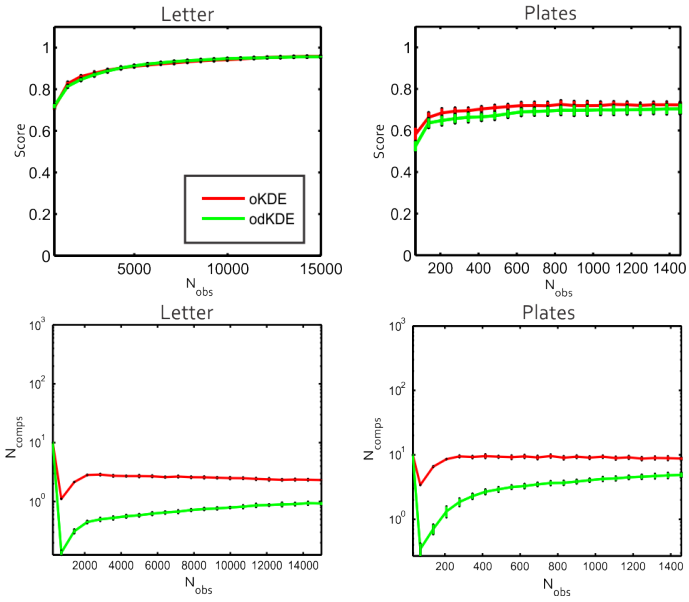

Fig. 6. The classification results (Score) w.r.t. the number of observations (N_{obs}) in the upper row and the number of components per class (N_{cmp}) w.r.t. the number of observations in the second row.

odKDE (as does in oKDE), but then stabilizes for larger number of samples. For example, in the case of the *Letter* dataset, the rate of increase in complexity was reduced early on after observing 115 samples per class (i.e., after 3000th observation), while the classification performance further increased through the model refinement (see Figure 6).

From the experimental results we therefore conclude, that the odKDE operates well in online setup since it produces compressed models with good classification performance. At the same time it maintains sufficient reconstructive information to allow online refinements of the models from new observations. We also conclude that the advantage of the odKDE over the oKDE comes from the new compression cost that constraints the changes of the classifier's properties during compression.

VIII. Discussion and Conclusion

We have proposed an online approach to estimation of discriminative models. The approach is based on an online Kernel Density Estimator that reconstructively updates its representation of the observed data. In order to prevent the complexity of the model from linearly increasing with the observed data points, the model is compressed from time to time into a simpler model. By defining a cost function that penalizes changes in the posterior distribution of the classifier, the resulting models are simplified while retaining their classification properties. Experiments demonstrate that the proposed odKDE produces comparable classification performance to, or outperforms, the batch state-of-the-art and the purely reconstructive oKDE. At the same time, it produces models whose complexity is significantly lower than those estimated by the batch methods, while retaining the online property of the oKDE, namely, that it allows adaptation from as little as a single sample at a time.

There are many venues of further research and extensions of the odKDE that we intend to pursue. Recently, Li et. al. [43] have shown how to incorporate measurement noise in the KDE's to improve classification performance. The odKDE already affords a principled way of accounting for the measurement noise. Recall that each new observation is added to our sample distribution as a Dirac-delta function. One way to account for the noise would be to add a Gaussian with covariance equal to the measurement noise instead. Hoti and Holmström [8] have shown that KDEs can be also efficiently applied to modelling distributions in extremely high-dimensional feature space. One venue of research would be to apply results of this paper to their work on high-dimensional distribution estimation. Another solution to deal with the extremely high-dimensional feature spaces would be to apply the manifold constraints from [7] to our bandwidth estimation. Since the odKDE is based on a simple reconstructive update rule (5), it can be easily adapted to address building classifiers from temporally nonstationary distributions – the distributions that change in time. The situations in which the input distribution changes in time is known as the concept drift [31]. Although this was not the focus of the present paper, the odKDE could be adapted to the concept drift by introducing an exponential forgetting in the updates (5) (similarly to [31], [12]). The result of such an adaptation would be an adaptive online classifier. We also note that the proposed model generates a Gaussian mixture model with a general structure of covariance matrices. However, if the covariances were constrained to be diagonal, all the computations in the odKDE could be rewritten as dot products. Such a constraint would then allow application of the "kernel trick" used in many kernel machines to allow coping with extremely high-dimensional data. We expect these will be the topics of further research.

Acknowledgment

This research has been supported in part by ARRS research programs P2-0214, P2-0094, and research projects J2-4284, J2-3607 and J2-2221.

References

[1] G. McLachlan and D. Peel, *Finite mixture models*. Wiley-Interscience, 2000.

[2] M. A. F. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 381–396, 2002.

[3] N. Nasios and A. Bors, "Variational learning for gaussian mixture models," *IEEE Trans. Systems, Man and Cybernetics, B*, vol. 36, no. 4, pp. 849–862, 2006.

[4] Z. Živkovič and F. van der Heijden, "Recursive unsupervised learning of finite mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 651 – 656, May 2004.

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

[6] D. Chaudhuri, B. Chaudhuri, and C. Murthy, "A data driven procedure for density estimation with some applications," *Patt. Recogn.*, vol. 29, no. 10, pp. 1719–1736, 1996.

[7] P. Vincent and Y. Bengio, "Manifold parzen windows," in *Advances in Neural Information Processing Systems*, 2003, pp. 849–856.

[8] F. Hoti and L. Holmström, "A semiparametric density estimation approach to pattern classification," *Patt. Recogn.*, vol. 37, no. 3, pp. 409–419, 2004.

[9] E. López-Rubio and J. Ortiz-de Lazcano-Lobato, "Soft clustering for nonparametric probability density function estimation," *Pattern Recognition Letters*, vol. 29, no. 16, pp. 2085–2091, 2008.

[10] J. M. Leiva-Murillo and A. Arts-Rodrguez, "Algorithms for maximum-likelihood bandwidth selection in kernel density estimators," *Pattern Recognition Letters*, vol. 33, no. 13, pp. 1717 – 1724, 2012.

[11] A. Bors and N. Nasios, "Kernel bandwidth estimation for nonparametric modeling," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 6, pp. 1543–1555, 2009.

[12] M. Kristan, A. Leonardis, and D. Skočaj, "Multivariate Online Kernel Density Estimation with Gaussian Kernels," *Pattern Recognition*, vol. 44, no. 10–11, pp. 2630–2642, 2011.

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

[14] J. Cwik and J. Koronacki, "A combined adaptive-mixtures/plug-in estimator of multivariate probability densities," *Computational Statistics and Data Analysis*, vol. 26, pp. 199–218, 1998.

[15] S. Chen, X. Hong, and C. Harris, "Probability density estimation with tunable kernels using orthogonal forward regression," *IEEE Trans. Systems, Man and Cybernetics, B*, vol. 40, no. 4, pp. 1101–1114, 2010.

[16] J. Goldberger and S. Roweis, "Hierarchical clustering of a mixture model," in *Neural Inf. Proc. Systems*, 2005, pp. 505–512.

[17] K. Zhang and J. Kwok, "Simplifying mixture models through function approximation," *IEEE Trans. Neural Networks*, vol. 21, no. 4, pp. 644 – 658, 2010.

[18] A. Leonardis and H. Bischof, "An efficient MDL-based construction of RBF networks," *Neural Networks*, vol. 11, no. 5, pp. 963 – 973, 1998.

[19] M. Girolami and C. He, "Probability density estimation from optimally condensed data samples." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1253–1264, 2003.

[20] S. Chen, X. Hong, and C. Harris, "Sparse kernel density construction using orthogonal forward regression with leave-one-out test score and local regularization," *IEEE Trans. Systems, Man and Cybernetics, B*, vol. 34, no. 4, pp. 1708–1717, 2004.

[21] Z. Deng, F. Chung, and S. Wang, "FRSDE: Fast reduced set density estimator using minimal enclosing ball approximation," *Patt. Recogn.*, vol. 41, no. 4, pp. 1363–1372, 2008.

[22] U. Ozertem, D. Erdogmus, and R. Jenssen, "Mean shift spectral clustering," *Patt. Recogn.*, vol. 41, no. 6, pp. 1924–1938, 2008.

[23] C. Diehl and G. Cauwenberghs, "Svm incremental learning, adaptation and optimization," in *Int. Joint Conf. on Neural Networks*, 2003, pp. 2685–2690.

[24] D. Mansjur and B. Juang, "Incremental learning of mixture models for simultaneous estimation of class distribution and inter-class decision boundaries," in *Proc. Int. Conf. Pattern Recognition*, 2008, pp. 1–4.

[25] F. Orabona, C. Castellini, B. Caputo, L. Jie, and G. Sandini, "On-line independent support vector machines," *Patt. Recogn.*, vol. 43, no. 4, pp. 1402–1412, 2010.

[26] O. Arandjelovic and R. Cipolla, "Incremental learning of temporally-coherent Gaussian mixture models," in *British Machine Vision Conference*, 2005, pp. 759–768.

[27] M. Song and H. Wang, "Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering," in *SPIE*, 2005, pp. 174–183.

[28] 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.*, vol. 30, no. 7, pp. 1186–1197, 2008.

[29] W. F. Szewczyk, "Time-evolving adaptive mixtures," National Security Agency, Tech. Rep., 2005.

[30] C. E. Priebe and D. J. Marchette, "Adaptive mixture density estimation," *Patt. Recogn.*, vol. 26, pp. 771–785, 1993.

[31] K. Tabata, M. Sato, and M. Kudo, "Data compression by volume prototypes for streaming data," *Patt. Recogn.*, vol. 43, pp. 3162–3176, 2010.

[32] M. Kristan, D. Skočaj, and A. Leonardis, "Online kernel density estimation for interactive learning," *Image and Vision Computing*, vol. 28, no. 7, pp. 1106–1116, 2010.

[33] N. Cristianini and T. J. Shawe, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[34] Y. Hamamoto, Y. Fujimoto, and S. Tomita, "On the estimation of a covariance matrix in designing Parzen classifiers," *Patt. Recogn.*, vol. 29, pp. 1751–1759, 1996.

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

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

[37] S. Julier and J. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Department of Engineering Science, University of Oxford, Tech. Rep., 1996.

[38] P. Hall, S. J. Sheater, M. C. Jones, and J. S. Marron, "On optimal data-based bandwidth selection in kernel density estimation," *Biometrika*, vol. 78, no. 2, pp. 263–269, 1991.

[39] C. C. Chang and C. J. Lin, *LIBSVM: A library for support vector machines*, 2001.

[40] C. P. Diehl and M. Spler, "mcpincsvm - incremental svm learning with multiclass support and probabilistic output," http://www-ti.informatik.uni-tuebingen.de/ spueler/mcpIncSVM/, 2011.

[41] F. Orabona, "Dogma - discriminative online (good?) matlab algorithms," http://dogma.sourceforge.net/ , 2011.

[42] A. Asuncion and D. Newman, "UCI machine learning repository," http://archive.ics.uci.edu/ml/, 2007.

[43] Y. Li, D. de Ridder, R. Duin, and M. Reinders, "Integration of prior knowledge of measurement noise in kernel density classification," *Patt. Recogn.*, vol. 41, no. 1, pp. 320–330, 2008.

Matej Kristan received a Ph.D. degree in electrical engineering from the Faculty of Electrical Engineering, University of Ljubljana in 2008. He is currently an Assistant Professor at the Faculty of Computer and Information Science and also an Assistant Professor at the Faculty of Electrical Engineering, both at the University of Ljubljana. His research interests include probabilistic methods for computer vision and pattern recognition with focus on visual tracking, probabilistic dynamic models, online learning, object detection and vision for mobile robotics.

Aleš Leonardis is Professor of Robotics at the University of Birmingham and co-Director of the Centre for Computational Neuroscience and Cognitive Robotics. He is also full professor at the Faculty of Computer and Information Science, University of Ljubljana and adjunct professor at the Faculty of Computer Science, Graz University of Technology. His research interests include robust and adaptive methods for computer vision, object and scene recognition and categorization, statistical visual learning, 3D object modeling, and biologically motivated vision. He is a fellow of the IAPR and a member of the IEEE and the IEEE Computer Society.