Online Kernel Density Estimation For Interactive Learning[1]

M. Kristan[a,b,1], D. Skočaj[a], A. Leonardis[a]

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

Abstract

In this paper we propose a Gaussian-kernel-based online kernel density estimation which can be used for applications of online probability density estimation and online learning. Our approach generates a Gaussian mixture model of the observed data and allows online adaptation from positive examples as well as from the negative examples. The adaptation from the negative examples is realized by a novel concept of unlearning in mixture models. Low complexity of the mixtures is maintained through a novel compression algorithm. In contrast to the existing approaches, our approach does not require fine-tuning parameters for a specific application, we do not assume specific forms of the target distributions and temporal constraints are not assumed on the observed data. The strength of the proposed approach is demonstrated with examples of online estimation of complex distributions, an example of unlearning, and with an interactive learning of basic visual concepts.

Key words: Online learning, Kernel Density Estimation, Mixture models, Unlearning, Compression, Hellinger distance, Unscented transform
PACS: :

1. Introduction

The process of learning may be viewed as a task of generating models from a corpus of data. Interactive acquisition of such models is one of the fundamental tasks in many rapidly emerging research areas. One example is interactive search engines for browsing large-scale data bases [1], where a system has to chose an optimal strategy to build efficient models of the query while minimizing the required communication with the user. Interactive and online learning is also becoming important in cognitive computer vision and cognitive robotics (eg., [2, 3]), where the primary goal is to study and develop cognitive agents – systems which could continually learn and interact within natural environments. Since most of the real-world environments are ever-changing, and all the information which they provide cannot be available (nor processed) at once, an agent or a system interacting within such an environment has to fulfill some general requirements in the way it builds the models of its environment: (i) The learning algorithm should be able to create and update the models as new data arrives. (ii) The models should be updated without explicitly requiring access to the old data. (iii) The computational effort needed for a single update should not depend on the amount of data observed sofar. (iv) The models should be compact and should not grow significantly with the number of training instances. Furthermore, in real-life scenarios, an erroneus information will typically get incorporated into the models. In such situations, the models should allow for error-recovery without the need of complete relearning. This is especially important in the user-agent interaction settings, in which the user can provide not only positive but negative examples as well to improve the agents knowledge about its environment. Therefore, another requirement (v) is that the models should support the process of *unlearning*, i.e., they have to allow online adaptation using the *negative* examples as well.

[1] Accepted for publication at the Jounal of Image and Vision Computing
* Corresponding author.
URL: `http://vicos.fri.uni-lj.si/matejk` (M. Kristan)

The process of online learning should thus create, extend, update, delete, and modify models of the perceived data in a continuous manner, while still keeping the representations compact and efficient. Various models and methods for their extraction have been proposed in different contexts and tasks (eg., [4, 5, 6, 7, 8]). In this paper we explore modelling data by probability density functions (pdf) based on Kernel Density Estimates (KDE). In particular, we focus on Gaussian mixture models (GMM), which are known to be a powerful tool in approximating distributions even when their form is far from Gaussian [9]. We demonstrate the results of our approaches on examples of online approximation of probability density functions and on examples of interactive visual learning in cognitive agents.

1.1. Related work

Traditionally, methods for density estimation are based on Parzen estimators [9, 10, 11, 12], expectation maximization (EM) algorithm [13, 14, 15] or variational estimation [16, 17], to name a few. However, their extention to online estimation of mixture models is nontrivial, since they assume all the data is available in advance. Indeed, a major issue in an online estimation of mixture models is that we do not have an access to the previously observed data to re-estimate the model's parameters when the new data arrives. Instead, the model itself has to serve as a compact representation of the data. While the model has to generalize well the already observed data, at the same time it has to be complex enough to allow efficient adaptation to the new data. Some researchers therefore impose temporal constraints on the incoming data to allow online estimation of the mixture models. Song and Wang [18], for example, assume that data comes in blocks. They use an EM with model selection to estimate the mixture model for the block of data and use statistical tests to merge components with the model learnt from the previously observed data. Arandjelović and Cippola [19] proposed an online extension of EM with split and merge rules, which allows adding a single datum at a time, rather than blocks. They make a strong assumption, however, that the distances between consecutive data points are sufficiently small, which prohibits application of this approach in general situations. Deleclerq and Piater [20] assign a Gaussian with a predefined covariance to the newly observed data and merge it with the mixture model, which describes the previously observed data. To ensure that the resulting mixture model contains enough information to adapt to the new data, each component is modelled by another mixture model. Szewczyk [21] applies a Dirichlet and Gamma density priors to assign new components to the mixture model in light of the incoming data and then merges the components in the mixture which are sufficiently close. One drawback of this approach, however, is that the parameters of the prior need to be specified for a given problem. A conceptually different approach was proposed by Han et al. [22], which aims to detect only the modes of the distribution and approximate each mode by a single Gaussian. While the approach produces good models when the modes of the distribution are sufficiently Gaussian and well separated, it fails to properly estimate the distribution in cases when the modes are non-Gaussian, e.g., in skewed or uniform distributions.

1.2. Our Approach

In contrast to the above approaches, we propose in this paper a method for online estimation of mixture models which does not require fine-tuning the parameters for a specific application, we do not assume a very specific forms of the target pdfs, temporal constraints are not assumed on the observed data and the proposed methods allow for unlearning as well. Our contributions are threefold. The first contribution is a new approach to incremental Gaussian mixture models which allows online estimation of the probability density functions. This is achieved by deriving a novel method for online kernel density estimation. The second contribution is a method that enables unlearning parts of the learned mixture model, which allows for a more versatile learning. The third contribution is a method for maintaining a low complexity of the estimated mixture models.

The remainder of the paper is structured as follows. In Section 2.1 we present the online mixture model, the method for unlearning is introduced in Section 2.2 and the method for complexity reduction is proposed in Sections 2.3 and 2.4. In Section 3 we present experimental results from online approximation of complex distributions, model refinement by unlearning, and apply the proposed methodology to the problem of online interactive learning of simple visual concepts. Conclusions are drawn in Section 4.

2. Online Estimation of Mixture Models

Throughout this paper we will refer to a class of kernel density estimates based on Gaussian kernels, which are commonly known as the Gaussian mixture models. Formally, we define a one dimensional M-component Gaussian mixture model as

$$p_{mix}(x) = \sum_{j=1}^{M} w_j K_{h_j}(x - x_j), \tag{1}$$

where w_j is the weight of the j-th component and $K_\sigma(x-\mu)$ is a Gaussian kernel

$$K_\sigma(z) = (2\pi\sigma^2)^{-\frac{1}{2}} exp(-\frac{1}{2}z^2/\sigma^2), \tag{2}$$

centered at mean μ with standard deviation σ; note that σ is also known as *the bandwidth* of the Gaussian kernel.

2.1. Online kernel density estimation

Suppose that we have observed a set of n_t samples $\{x_i\}_{i=1:n_t}$ up to some time-step t. The problem of modelling samples by a probability density function can be posed as a problem of kernel density estimation [9]. In particular, if all of the samples are observed at once, then we seek a kernel density estimate with kernels placed at locations x_i with equal bandwidths h_t

$$\hat{p}_t(x; h_t) = \frac{1}{n_t} \sum_{i=1}^{n_t} K_{h_t}(x - x_i), \tag{3}$$

which is as close as possible to the underlying distribution that generated the samples. A classical measure used to define closeness of the estimator $\hat{p}_t(x; h_t)$ to the underlying distribution $p(x)$ is the *mean integrated squared error* (MISE)

$$\text{MISE} \quad = \quad \text{E}[\hat{p}_t(x; h_t) - p(x)]^2. \tag{4}$$

Applying a Taylor expansion, assuming a large sample-set and noting that the kernels in $\hat{p}_t(x; h_t)$ are Gaussians ([9], p.19), we can write the *asymptotic* MISE (AMISE) between $\hat{p}_t(x; h_t)$ and $p(x)$ as

$$\text{AMISE} = \frac{1}{2\sqrt{\pi}}(h_t n_t)^{-1} + \frac{1}{4} h_t^4 R(p''(x)), \tag{5}$$

where $p''(x)$ is the second derivative of $p(x)$ and $R(p''(x)) = \int p''(x)^2 dx$. Minimizing AMISE w.r.t. bandwidth h_t gives AMISE-optimal bandwidth

$$h_{t\text{AMISE}} = [\frac{1}{2\sqrt{\pi} R(p''(x)) n_t}]^{\frac{1}{5}}. \tag{6}$$

Note that (6) cannot be calculated exactly since it depends on the second derivative of $p(x)$, and $p(x)$ is exactly the unknown distribution we are trying to approximate. Several approaches to approximating $R(p''(x))$ have been proposed in the literature (see e.g. [9]), however these require access to *all* the observed samples, which is in contrast to the online learning where we wish to discard previous samples and retain only their compact representations. Our setting is depicted in Figure 1: We start from a known pdf $\hat{p}_{t-1}(x)$ from the previous time-step and in the current time-step observe a sample x_t (Figure 1a). A Gaussian kernel corresponding to x_t is calculated and used to update $\hat{p}_{t-1}(x)$ to yield a new pdf $\hat{p}_t(x)$ (Figure 1b). Formally this means that the current estimate $\hat{p}_t(x)$ is obtained as

$$\hat{p}_t(x) = (1 - \frac{1}{n_t})\hat{p}_{t-1}(x) + \frac{1}{n_t} K_{h_t}(x - x_t), \tag{7}$$

and the online kernel density estimation boils down to estimating the *bandwidth* h_t of the Gaussian kernel for the currently observed sample x_t. We can rewrite (3) by separating the kernel corresponding to the currently observed sample x_t from the other kernels

$$\begin{aligned}\hat{p}_t(x;h_t) &= (1-\frac{1}{n_t})\sum_{i=1}^{n_t-1}\frac{1}{n_t-1}K_{h_t}(x-x_i)+\frac{1}{n_t}K_{h_t}(x-x_t)\\ &= (1-\frac{1}{n_t})\hat{p}_{t-1}(x,h_t)+\frac{1}{n_t}K_{h_t}(x-x_t).\end{aligned} \tag{8}$$

If we then assume that the KDE corresponding to the samples from the previous time-steps in (8) can be approximated by our estimate of the distribution from $(t-1)$, i.e., $\hat{p}_{t-1}(x,h^*)\approx\hat{p}_{t-1}(x)$ (where h^* is the optimal choice of h_t), then the optimal bandwidth h_t for $\hat{p}_t(x)$, (7) can be approximated by the bandwidth for $\hat{p}_t(x;h_t)$, (3). This means that, under the above assumptions, we can use (6) as a heuristic for estimating the optimal bandwidth of $\hat{p}_t(x)$. In the following we propose an *iterated plug-in rule* which uses this heuristic for online bandwidth estimation.

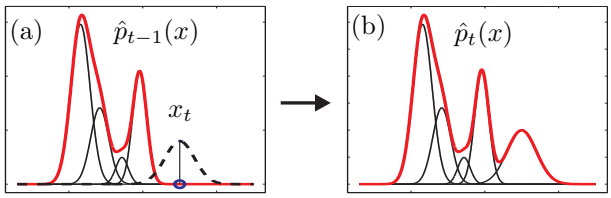

Figure 1: The left image shows a Gaussian mixture model $\hat{p}_{t-1}(x)$ from time-step $t-1$ (bold line) and the currently observed sample x_t (circle). A Gaussian kernel is centered on x_t (dashed line) and used to update $\hat{p}_{t-1}(x)$. The right image shows the current, updated, mixture $\hat{p}_t(x)$.

Let x_t be the currently observed sample and let $\hat{p}_{t-1}(x)$ be an approximation to the underlying distribution $p(x)$ from the previous time-step. The current estimate of the $p(x)$ is initialized using the distribution from the previous time-step $\hat{p}_t(x)\approx\hat{p}_{t-1}(x)$. The bandwidth $\hat{h}_t$ of the kernel $K_{\hat{h}_t}(x-x_t)$ corresponding to the current observed sample x_t is obtained by approximating the unknown distribution $p(x)\approx\hat{p}_t(x)$ and applying (6)

$$\hat{h}_t=[2\sqrt{\pi}R(\hat{p}_t''(x))n_t]^{-1/5}. \tag{9}$$

The resulting kernel $K_{\hat{h}_t}(x-x_t)$ is then combined with $\hat{p}_{t-1}(x)$ into an improved estimate of the unknown distribution

$$\hat{p}_t(x)=(1-\frac{1}{n_t})\hat{p}_{t-1}(x)+\frac{1}{n_t}K_{\hat{h}_t}(x-x_t). \tag{10}$$

Next, the improved estimate $\hat{p}_t(x)$ from (10) is plugged back into the equation (9) to re-approximate $\hat{h}_t$ and then equations (9) and (10) are iterated until convergence; usually, five iterations suffice.

Note that with each observed sample the number of components in the mixture model increases. Therefore, in order to maintain a low complexity, a compression algorithm is initiated whenever the number of components exceeds a value N_{comp}. As we will see in section 2.4 the compression (Algorithm 3) does not *force* removing components but merely *tries* to remove some. Therefore the threshold N_{comp} only determines the frequency at which the compression is called. To prevent unnecessary calls to compression, the threshold N_{comp} can be therefore set to some fixed large value or can be allowed to vary. In practice, we use a simple rule to adjust the N_{comp} online: If no components are removed in the compression step (Algorithm 1, step 5), then N_{comp} is increased, i.e., $N_{\text{comp}}\leftarrow c_{scale}N_{\text{comp}}$, otherwise, if the number of the remaining components falls below $c_{scale}^{-1}N_{\text{comp}}$, the threshold is decreased, i.e., $N_{\text{comp}}\leftarrow c_{scale}^{-1}N_{\text{comp}}$. In all the subsequent experiments, we use a scale factor $c_{scale}=1.5$. The procedure for online kernel density estimation is outlined in Algorithm 1.

Algorithm 1 : Online kernel density estimation

Input: $\hat{p}_{t-1}(x)$, x_t ... the initial density approximation and the new sample
Output: $\hat{p}_t(x)$ ... the new approximation of density

1: Initialize the current distribution $\hat{p}_t(x) \approx \hat{p}_{t-1}(x)$.
2: Estimate the bandwidth h_t of $K_{\hat{h}_t}(x - x_t)$ according to (9) using $\hat{p}_t(x)$.
3: Reestimate $\hat{p}_t(x)$ according to (10) using $K_{\hat{h}_t}(x - x_t)$.
4: Iterate steps 2 and 3 until convergence.
5: If the number of components in $\hat{p}_t(x)$ exceeds a threshold N_{comp}, compress $\hat{p}_t(x)$ using Algorithm 3.
6: If required, adjust the threshold N_{comp}.

2.2. Incorporating the negative examples

In the previous section we have proposed a method for online estimation of the mixture model from *all-positive* examples using an online kernel density estimation. As discussed in the introduction, another important aspect of the online learning is how to account for the *negative* examples. This feature is especially important in real-life scenarios since it allows learning models from a noisy data and then refining these models using additional negative examples – we call this the process of *unlearning*. Assume that, by observing values of some feature x, we have constructed the following M_{ref}-component Gaussian mixture model

$$p_{\text{ref}}(x) = \sum_{i=1}^{M_{\text{ref}}} w_i K_{h_i}(x - x_i). \tag{11}$$

Now assume that we obtain another pdf, a M_{neg}-component mixture

$$p_{\text{neg}}(x) = \sum_{j=1}^{M_{\text{neg}}} \eta_j K_{s_j}(x - y_j), \tag{12}$$

which represents a *negative example* to what we want to learn. For example, we might want to learn how the hue values of red objects are distributed. $p_{\text{ref}}(x)$ would then be a (possibly) noisy model of the red color which we have learnt sofar, and we wish to use a pdf $p_{\text{neg}}(x)$ estimated from the hue values of a yellow object to refine our model of the red color. The main issue in the unlearning is thus how to incorporate $p_{\text{neg}}(x)$ into the reference model $p_{\text{ref}}(x)$.

We can think about $p_{\text{neg}}(x)$ as of a pdf which specifies the likelihood by which particular values of x can be considered as a *negative* example to what we want to learn. A complement to $p_{\text{neg}}(x)$ therefore specifies another *positive* example, which effectively assigns low probability to those values of x which are very likely to be the negative example according to $p_{\text{neg}}(x)$. Thus, in summary, we propose to formulate the unlearning in the following two steps: First, we generate a so-called *attenuation* function $f_{\text{att}}(x)$, which presents a complement to $p_{\text{neg}}(x)$, and maps the feature values x into interval $[0, 1]$ by yielding 0 for the values of x where $p_{\text{neg}}(x)$ is maximal and 1 for the values where $p_{\text{neg}}(x)$ is zero. Then $p_{\text{neg}}(x)$ is incorporated into $p_{\text{ref}}(x)$ simply by multiplying $p_{\text{ref}}(x)$ with the attenuation function, yielding the attenuated mixture model $p_{att}(x)$. The analytical solution to this procedure is described next.

The attenuation function is defined as

$$f_{att}(x) = 1 - C_{\text{opt}}^{-1} p_{\text{neg}}(x), \tag{13}$$

where the normalization constant C_{opt} guarantees that $C_{\text{opt}}^{-1} p_{\text{neg}}(x) \leq 1$ and thus all values of x are mapped into the interval $[0, 1]$. Note that C_{opt} corresponds to the maximum value in $p_{\text{neg}}(x)$ and cannot be trivially calculated, since the maximum may lay in-between the components of the mixture. For that reason we use a variable-bandwidth mean-shift algorithm [23], which we initialize at the centers of the components of $p_{\text{neg}}(x)$ to detect its modes. The mode x_{opt}, corresponding to the maximum value of $p_{neg}(x)$, is selected as the maximum of the distribution and the normalization C_{opt} is given as

$$C_{\text{opt}} = p_{\text{neg}}(x_{\text{opt}}). \tag{14}$$

The attenuated pdf $p_{\text{att}}(x)$ is obtained by multiplying the reference pdf with the attenuation function

$$p_{\text{att}}(x) = C_{\text{norm}}^{-1} p_{\text{ref}}(x) f_{\text{att}}(x) = C_{\text{norm}}^{-1}[p_{\text{ref}}(x) - f_{\text{com}}(x)], \tag{15}$$

where we have defined $f_{\text{com}}(x) = C_{\text{opt}}^{-1} p_{\text{ref}}(x) p_{\text{neg}}(x)$ and C_{norm} is a normalization constant such that $\int p_{\text{att}}(x)dx = 1$. Note that since the product of two Gaussians is another, scaled, Gaussian [24], we can rewrite $f_{\text{com}}(x)$ as

$$\begin{aligned} f_{\text{com}}(x) &= \sum_{i=1}^{M_{\text{ref}}} \sum_{j=1}^{M_{\text{neg}}} C_{\text{opt}} w_i \eta_j K_{h_i}(x - x_i) K_{s_j}(x - \mu_j) \\ &= \sum_{i=1}^{M_{\text{ref}}} \sum_{j=1}^{M_{\text{neg}}} z_{ij} \beta_{ij} K_{\sigma_{ij}}(x - \mu_{ij}), \end{aligned} \tag{16}$$

where $z_{ij} = \frac{\sigma_{ij}}{\sqrt{2\pi} h_i s_j} exp[\frac{1}{2}(\frac{\mu_{ij}^2}{\sigma_{ij}^2} - \frac{x_i^2}{h_i^2} - \frac{y_j^2}{s_j^2})]$, $\mu_{ij} = \sigma_{ij}^2(\frac{x_i}{h_i^2} + \frac{y_j}{s_j^2})$, $\beta_{ij} = C_{\text{opt}} w_i \eta_j$, $\sigma_{ij}^2 = (h_i^{-2} + s_j^{-2})^{-1}$. Now we can derive the normalization C_{norm} for the attenuated pdf $p_{att}(x)$ (15)

$$C_{\text{norm}} = (1 - \sum\nolimits_{i=1}^{M_{\text{ref}}} \sum\nolimits_{j=1}^{M_{\text{neg}}} \beta_{ij} z_{ij})^{-1}. \tag{17}$$

The proposed method for unlearning a mixture model is summarized in Algorithm 2.

Algorithm 2 : The algorithm for unlearning mixtures

Input: $p_{ref}(x)$, $p_{neg}(x)$ ... the reference mixture and the negative-example mixture.
Output: $p_{att}(x)$ ... the unlearned mixture.

1: Detect the location x_{opt} of the maximum mode in $p_{neg}(x)$ using the variable-bandwidth mean shift [23].
2: Scale $p_{neg}(x)$ with respect to the detected mode x_{opt} (14) and calculate the attenuation function $f_{att}(x)$ (13).
3: Multiply $f_{att}(x)$ with $p_{ref}(x)$ and normalize to obtain the attenuated mixture $p_{att}(x)$ (15,16,17).

Note that while $p_{\text{att}}(x)$ is indeed a proper pdf, it is not a proper mixture model, since some of the weights are negative. Furthermore, by introducing new attenuation functions, the number of components in (15) increases exponentially, which in practice makes subsequent calculations inefficient and slow. For that reason, after the attenuation, the resulting distribution needs to be *compressed*, i.e., we require an equivalent mixture with a smaller number of components. Furthermore, for algorithmic reasons (because of the way in which we define the compression in the subsequent sections) it is beneficial if the equivalent is a proper mixture with all-positive weights. In the following we propose a methodology for obtaining such equivalents.

2.3. Approximating mixtures with mixtures

Assume we are given a reference mixture

$$p(x) = \sum_{i=1}^{M} w_i K_{h_i}(x - x_i), \tag{18}$$

where all w_i are not necessarily positive, but they do sum to one. Our goal is then to approximate the reference (18) with a N-component mixture with all positive weights

$$\hat{p}(x|\theta) = \sum_{j=1}^{N} \gamma_j K_{\sigma_j}(x - \mu_j), \tag{19}$$

where $\theta = \{\gamma_j, \mu_j, \sigma_j\}_{j=1:N}$ denotes the parameters of the mixture, such that some difference criterion between $p(x)$ and $\hat{p}(x|\theta)$ is minimized. Since the reference mixture is known, the difference between the reference mixture and its approximation can be quantified by the integrated squared error (ISE)

$$\text{ISE}(\theta) = \int (p(\mathbf{x}) - \hat{p}(\mathbf{x}|\theta))^2 d\mathbf{x}. \tag{20}$$

The problem of finding an equivalent to $p(x)$ can thus be posed as seeking an optimal $\hat{\theta}$ while minimizing the ISE:

$$\hat{\theta} \quad = \quad \arg\min_{\theta} [\int \hat{p}^2(\mathbf{x}|\theta) d\mathbf{x} - 2E_{p(\mathbf{x})}\{\hat{p}(\mathbf{x}|\theta)\}], \tag{21}$$

where $E_{p(x)}\{\hat{p}(x|\theta)\}$ is the expectation with respect to the reference distribution $p(x)$ and where we have dropped $\int p^2(x)dx$ from the above equation since it does not depend on θ. Since we can calculate the derivatives of ISE, $\delta\text{ISE}(\theta)/\delta\theta$, analytically, efficient optimization schemes such as gradient descent or Levenberg-Marquardt can be used in optimizing (21). However, in practice, when M is large and $N \approx M$ it is likely that some components in $\hat{p}(x|\theta)$ will be redundant, which may result in a slow convergence of optimization. Moreover, in cases when $\hat{p}(x|\theta)$ is poorly initialized, optimization can get stuck in a local minimum. Therefore, a question remains how to determine the appropriate number of components in $\hat{p}(x|\theta)$. To address this issue, we note that component selection can be viewed as optimizing (21) with respect to the weights γ_i of $\hat{p}(x|\theta)$ (19). By driving some weights of (19) to zero we are effectively removing the corresponding components. A useful insight into such optimization is provided by the theory of the reduced-set-density estimation [25] and earlier results from the support estimation in the support vector machines [26]. In [25], Girolami and He proposed a reduced-set-density approximations of kernel density estimates. A central point of their approach was minimization of an ISE-based criterion, which is in spirit similar to our formulation in (20). In line with their observations, we now inspect how the two terms of the right-hand side of (21) affect the optimization of ISE w.r.t. the weights γ_i of $\hat{p}(x|\theta)$.

If the first term of the right-hand side of (21) is kept fixed, minimization is obtained by maximizing the second term $E_{p(x)}\{\hat{p}(x|\theta)\}$. By expanding this term we have

$$E_{p(x)}\{\hat{p}(x|\theta)\} = \sum_{j=1}^{N} \gamma_j \tilde{p}_j, \tag{22}$$

with $\tilde{p}_j = \int K_{\sigma_j}(\mathbf{x} - \mu_j)[\sum_{i=1}^{M} w_i K_{h_i}(x - \mathbf{x}_i)]d\mathbf{x}$.

Note that (22) is a convex combination of positive numbers $\tilde{p}_i$, which are expectations of components in $\hat{p}(x|\theta)$ under the reference $p(x)$. In this case, maximization would be achieved by assigning a weight one to the largest expectation $\tilde{p}_j$ and a zero weight to all others. Thus, if the reference distribution $p(x)$ has a dominant mode, then the component $K_{\sigma_j}(x - \mu_j)$ of the approximating distribution $\hat{p}(x|\theta)$ that agrees best with this mode will be assigned a weight one, while all other weights will be zero. We say that the term $E_{p(x)}\{\hat{p}(x|\theta)\}$ is *sparsity-inducing* in that it prefers those components of the approximating distribution $\hat{p}(x|\theta)$ which correspond to the high-probability regions in $p(x)$.

Now note that minimizing ISE (21) with $E_{p(x)}\{\hat{p}(x|\theta)\}$ kept fixed equals to minimizing the first term $\int \hat{p}^2(x|\theta)dx$. Expanding this term yields a weighted sum of expectations of pairs of components of $\hat{p}(x|\theta)$

$$\int \hat{p}^2(x|\theta)dx = \sum_{i=1}^{N}\sum_{j=1}^{N} \gamma_i \gamma_j c_{ij}, \tag{23}$$

where we have defined $c_{ij} = \int K_{\sigma_i}(x - \mu_i)K_{\sigma_j}(x - \mu_j)dx$.

The expectations among non-overlapping components will yield low values of c_{ij}, while expectations among overlapping components will yield high values of c_{ij}. In this case, ISE would be minimized by

assigning small weights to the overlapping components and large weights to those which do not overlap. Thus we can say that the term (23) is *sparsity-inducing* in that it prefers selection of those components that are far apart.

From the above discussion, we can see that optimizing ISE (21) between $p(x)$ and $\hat{p}(x|\theta)$ will yield a subset of components in $\hat{p}(x|\theta)$ by selecting components in high-probability regions of $p(x)$, while preferring configurations in which the selected components are far apart.

Using (22,23) we can rewrite the minimization of ISE (21) with respect to the weights γ_i into a classical quadratic program

$$\arg\min_{\gamma}\{\frac{1}{2}\gamma^T\mathbf{C}\gamma - \gamma^T\mathbf{P}\}\ ;\ \gamma^T\mathbf{1} = 1, \gamma_j > 0, \forall j, \tag{24}$$

where $\mathbf{1}$ is a column vector of ones, and where we have defined the vector of weights $\gamma^T = [\gamma_1, \gamma_2, \ldots, \gamma_N]$, a symmetric $N \times N$ matrix $\mathbf{C}$ with elements c_{ij} (23) and a vector of $\mathbf{P} = [\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_N]$ with elements $\tilde{p}_j$ (22).

Note that since all components in the reference and the approximating distributions (18,19) are Gaussians, $\mathbf{C}$ and $\mathbf{P}$ in (24) can be evaluated analytically. There is a number of optimization techniques available for solving the quadratic program (24). In our approach we use a variant of a Sequential Minimal Optimization (SMO) scheme [26], which was previously used by Girolami and He [25] for a similar optimization.

2.4. Compression algorithm

Using the results from the previous section, we propose an iterative compression algorithm, which is similar in spirit to [27, 28], for finding a reduced equivalent to a reference Gaussian mixture model $p(x)$. We start from an approximation $\hat{p}(x|\theta)$ which is equal to the reference mixture in cases when mixture $p(x)$ does not contain any negative weights. When compressing the *unlearned* mixture, we initially increase the number of the components in the approximation by splitting each component with a negative weight into two components and then make their weights positive. This in practice makes the approximation adapt quicker to the reference distribution in regions where the reference distribution contains positive as well as negative components. After the approximation has been initialized, the components are gradually removed from $\hat{p}(x|\theta)$ while minimizing the ISE criterion (20) between $\hat{p}(x|\theta)$ and $p(x)$. The compression algorithm proceeds in the following two steps: *reduction* and *organization*. At the *reduction* step, a subset of components from $\hat{p}(x|\theta)$ is removed using SMO. In the next step, the *organization* step, we further reduce the error between the reduced $\hat{p}(x|\theta)$ and the reference $p(x)$. This is achieved by optimizing (21) w.r.t. all parameters θ in $\hat{p}(x|\theta)$ using a Levenberg-Marquardt (LM) optimization with a constraint that all weights in $\hat{p}(x|\theta)$ are positive. These two steps are iterated until convergence. The procedure is outlined in Algorithm 3.

At each step of the iterative procedure described in Algorithm 3, a subset of components from $\hat{p}(x|\theta)$ is removed, thus gradually reducing the complexity of $\hat{p}(x|\theta)$. The number of components removed at each step can be controlled by inflating the variances of $\hat{p}(x|\theta)$ by some inflation parameter $\alpha > 1$ before applying the SMO. For a large α, many components of $\hat{p}(x|\theta)$ will overlap significantly and thus many components will be removed. As a result, the final pdf $\hat{p}(x|\theta)$ will be a smoothed equivalent of the reference pdf $p(x)$. However, removing too many components will increase the error in the approximation of $p(x)$, and therefore the inflation parameter α has to be set such that the inflated pdf is always *close enough* to the reference pdf $p(x)$. In our implementation, this is achieved by adjusting the α such that the Hellinger distance [29] between $p(x)$ and the inflated $\hat{p}(x|\theta)$, $H(p(x), \hat{p}(x|\theta))$, is always smaller than some predefined distance H_{dist}. Note that while the Hellinger distance is a proper metric between probability density functions and is constrained to the interval $[0, 1]$, it cannot be calculated analytically in a closed-form for mixture models. We therefore calculate its approximation using the *unscented transform* [30]. For convenience, we derive the *unscented* approximation of the Hellinger distance for mixture models in the Appendix A.

Once the selection step removes a set of components, the remaining set is optimized in order to minimize the ISE between $\hat{p}(x|\theta)$ and the reference $p(x)$ (*organization*). Note that we do not have to optimize $\hat{p}(x|\theta)$ until convergence in this step. We only need to reduce the error between $\hat{p}(x|\theta)$ and $p(x)$ to the extent that a set of components in $\hat{p}(x|\theta)$ will overlap after the inflation and will be removed in the *reduction* step.

Algorithm 3 : Compression algorithm

Input: $p_{ref}(x)$, H_{dist} ... the reference mixture and the maximum allowed Hellinger distance between $p_{ref}(x)$ and compressed counterpart.

Output: $\hat{p}(x|\theta)$ ... the compressed equivalent.

1: *Initialization*: construct $\hat{p}(x|\theta)$ from $p(x)$, such that all components have positive weights (see text).

2: *Reduction*:

- Inflate $\hat{p}(x|\theta)$ into $\hat{p}_\alpha(x|\theta)$ by increasing the variances $\sigma_j^2 \leftarrow \alpha\sigma_j^2$ such that $H(p_{ref}(x), \hat{p}_\alpha(x|\theta)) = 0.7H_{dist}$.
- Optimize (24) between the inflated $\hat{p}(x|\theta)$ and $p(x)$ w.r.t. γ using a SMO.
- Remove those components from $\hat{p}(x|\theta)$ for which $\gamma_i = 0$.

3: *Organization*: Optimize ISE between $\hat{p}(x|\theta)$ and $p(x)$ using a LM optimization w.r.t. θ.

4: If the distance between $p_{ref}(x)$ and $\hat{p}(x|\theta)$ is small enough, i.e., $H(p_{ref}(x), \hat{p}(x|\theta)) < H_{dist}$, then accept $\hat{p}(x|\theta)$ as a potential compressed equivalent.

5: If at least one component was removed during the reduction procedure, and $\hat{p}(x|\theta))$ was accepted at step (4), then go to to step (2), otherwise optimize θ until convergence and end compression.

Thus in practice we use five Levenberg-Marquardt iterations at each organization step. Only after no more components are removed, we optimize $\hat{p}(x|\theta)$ until convergence.

3. Experimental Results

Three sets of experiments were conducted to evaluate the proposed methods for online learning and approximation of probability density functions. The first two experiments were designed to demonstrate online approximation of complex mixtures and to illustrate the concept of unlearning. In the third experiment we show how the proposed algorithms can be applied for interactive learning of basic visual properties.

3.1. Online approximation of complex distributions

The aim of the first experiment was to demonstrate the performance of the online kernel density estimation proposed in Section 2.1 – we will refer to this method as an *online* KDE. A set of 1000 samples was generated from a reference pdf $p_{ref}(x)$, which was a 1D mixture of a Gaussian and a uniform distribution (Figure 2a). These samples were then used one at a time to incrementally build the approximation to the original distribution using the Algorithm 1. At each time-step three other models were also built for reference. The first two were *batch* KDEs, and were built by processing *all* samples observed up to the given time-step simultaneously. In the first KDE model, we refer to it as the *optimal* batch KDE, the bandwidths of the kernels were estimated via solve-the-equation plug-in method [31], which is currently theoretically and empirically one of the most successful bandwidth-selection methods. In the second KDE model, we call this model a *suboptimal* batch KDE, the bandwidths were estimated using the Silverman's rule-of-thumb ([9], page 60), which is a common choice of practicians. The third reference model was a Gaussian Mixture Model, which was built by applying the online Expectation Maximization (EM) coupled with model resampling and the Bayes Information Criterion [32] was used to select the number of components in the model. Since the online EM-based model requires processing data in partial batches, this model was updated using 100 samples at a time, and a maximum of 50 components was allowed in the model selection. Our online KDE was initialized using the solve-the-equation plug-in method from the first ten samples and the threshold N_{comps} in the Algorithm 1 was initialized to $N_{comps} = 10$. The online KDE was updated using only a single sample at a time.

To quantify how the approximations evolve with each new sample, three different distance measures were calculated between the obtained approximations and the reference model: the integrated squared error

(ISE); a Hellinger distance[2]; and the log-likelihood of another set of 1000 randomly drawn samples from the reference $p_{ref}(x)$. The distances were averaged over thirty repetitions of the experiment. Along with the distances, we have also recorded the average number of components in the approximations and for the KDE-based methods also the average bandwidth assigned to the new kernel after observing each new sample.

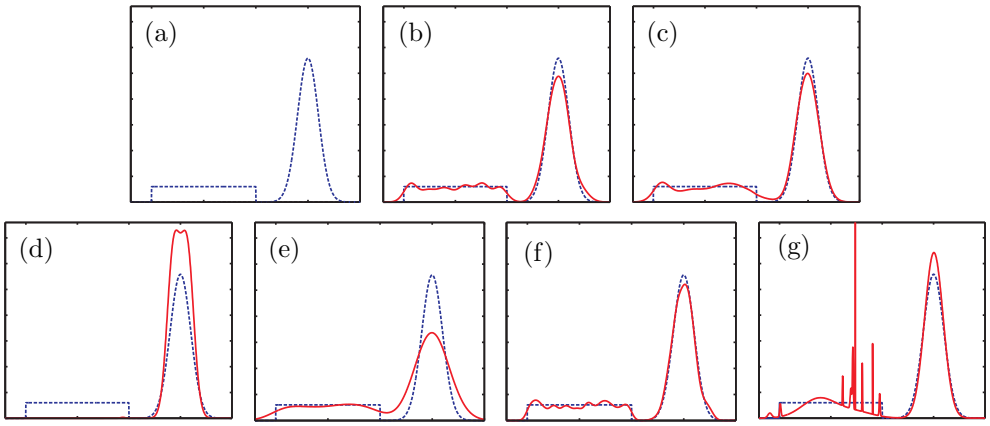

Figure 2: A reference mixture model (a) used in the online density approximation experiment. Approximations by online KDE with parameter H_{dist} set to 0.05, 0.1 and 0.3 are shown in (b), (c) and (d), respectively, the approximations obtained by suboptimal batch KDE and the optimal batch KDE are shown in (e) and (f), respectively and the approximation obtained by the online EM is shown in (g). In (b,c,d,e,f,g), the reference is drawn in dashed (blue) line and the approximations are shown by a full (red) line.

Figure 2 shows the approximations of the reference pdf after observing all 1000 samples. The approximations from the proposed online KDE with the parameter H_{dist} set to 0.05, 0.1 and 0.3 are shown in Figures 2(b,c,d). Recall that the parameter H_{dist} defines the maximal allowed error in the approximation during the compression steps. We see that for small values of H_{dist}, the approximations agree well with the reference pdf (Figure 2b,c). However, when H_{dist} was large, i.e., $H_{dist} = 0.3$, the reduction step in the compression eventually removed the components which corresponded to the low-weight uniform distribution and only retained the dominant mode (Figures 2d). In terms of the bandwidth selection, we can see from the Figure 3(a), that the proposed online KDE with parameters $H_{dist} \in \{0.05, 0.1\}$ produced similar bandwidths as the optimal batch KDE, while the bandwidths of the suboptimal batch KDE were consistently over-estimated, which resulted in an over-smoothed approximation of the reference pdf (Figure 2e). Note that the graphs of the estimated bandwidths in Figure 3(a) corresponding to the online KDEs with parameters $H_{dist} \in \{0.05, 0.1\}$ are virtually equal. This result is consistent with the evolution of the approximation errors in Figure 4 where we compare the online KDE with the batch KDE: while initially the errors were high for all approximations, they decreased with increasing number of samples. As expected, the error of the batch KDE calculated using Silverman's rule remained high even after all the 1000 samples have been observed. On the other hand, the errors decreased faster for the online KDEs and came close to the errors of the optimal batch KDE with increasing number of samples. This result is consistent across all three measures of the approximation error in Figure 4. Note that the bandwidths in the optimal batch KDE were calculated using *all* samples observed up to a given step. In contrast, the proposed online KDEs produced similarly small errors using only a low-dimensional *representations* of the observed samples. This is seen in Figure 3(b) where we show the evolution of the average number of components in the approximations. In the batch KDEs (optimal and suboptimal), the complexity was increasing linearly with the observed number of samples. Thus, after observing 1000 samples, the batch KDEs were composed of 1000 kernels. On the other hand, the final approximations obtained by the proposed online KDEs with parameters $H_{dist} = 0.1$

[2]The Hellinger distances between the reference distribution and its approximations were calculated by a Monte Carlo integration

and $H_{dist} = 0.05$ contained only 12 and 20 kernels, respectively. While the complexity of the models produced by the online KDE with $H_{dist} = 0.05$ was comparable to that of the models produced by the online EM (Figure 3b), both online KDEs consistently produced smaller errors than the online EM (Figure 5). Note that, due to the resampling step, the limited sample size and the lack of regularization in the model selection of the online EM algorithm, the resulting models sometimes contained components with nearly singular variance (see, for example, Figure 2g). The consequence of this was an increased effective number of components in the EM models. Note that, due to the resampling step, the limited sample size and the lack of regularization in the model selection of the online EM algorithm, the resulting models sometimes contained components with nearly singular variance (see, for example, Fig. 2g). The consequence of this was an increased effective number of components in the EM models.

To provide an insight into the computational complexity of the online KDE, we have calculated the average times required for the model update after observing the 1000th sample in our experiment. The tests were performed with non-optimized code in Matlab [33] on a personal computer with a 2.6GHZ CPU. The times are given in Table 1. The online EM required the longest computational time, which is largely due to the model selection stages, while the suboptimal batch KDE required the shortest of computational time. The online KDEs required on average a smaller amount of processing time per update step than the optimal batch KDE. The reason si that the online KDE models have reached a bound on their complexity fairly early (after observing approximately 300 samples – see Figure 3) and therefore the processing time after observing the 1000th sample was approximately the same as if only 300 samples have been observed.

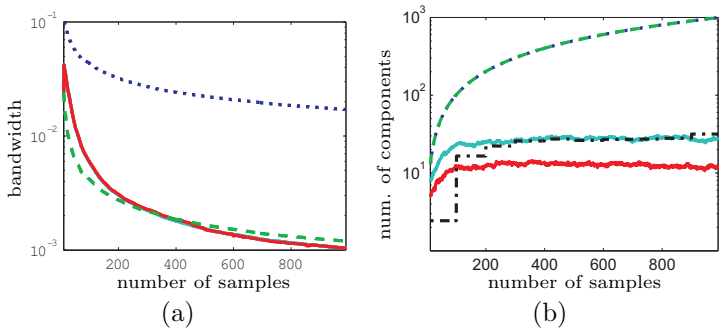

Figure 3: The graphs show how the bandwidths (a) and the number of components (b) in the KDE approximations change with increasing number of samples. The results for online KDE with H_{dist} 0.05 and 0.1 are depicted by bright (cyan) and dark (red) full lines. The results for the batch KDE with the optimal and suboptimal bandwidth selection are depicted by dashed (green) and dotted (blue) lines, respectively.

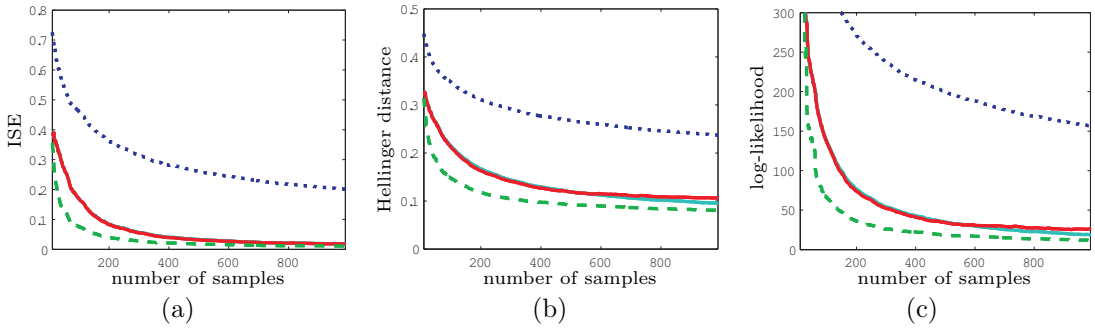

Figure 4: Three error measures of the KDE approximations w.r.t. the number of samples: ISE (a), Hellinger distance (b) and the log-likelihood (c). The results for online KDE with H_{dist} 0.05 and 0.1 are depicted by bright (cyan) and dark (red) full lines. The results for the batch KDE with the optimal and suboptimal bandwidth selection are depicted by dashed (green) and dotted (blue) lines, respectively.

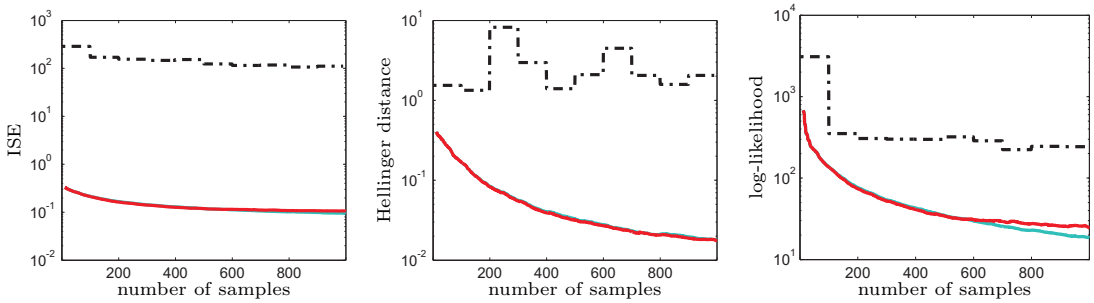

Figure 5: Three error measures of the online approximations w.r.t. the number of samples: ISE (a), Hellinger distance (b) and the log-likelihood (c). The results for online KDE with H_{dist} 0.05 and 0.1 are depicted by bright (cyan) and dark (red) full lines, while the results for the online EM are depicted by the black dash-dotted lines.

Table 1: Average time spent per model adaptation (t_{spent}) after observing 1000 samples for the suboptimal batch KDE (KDE_{sil}), the optimal batch KDE ($\text{KDE}_{\text{plugin}}$), the online KDE with $H_{\text{dist}} = 0.1$ ($\text{OKDE}_{0.1}$), the online KDE with $H_{\text{dist}} = 0.05$ ($\text{OKDE}_{0.05}$) and the online Expectation Maximization with model selection *oEM*. (Tested on a PC with a 2.6GHz cpu)

method	KDE_{sil}	$\text{KDE}_{\text{plugin}}$	$\text{OKDE}_{0.1}$	$\text{KDE}_{0.05}$	oEM
t_{spent}	0.04s	2s	0.4s	1.6s	13s

From the above results we can conclude that, when H_{dist} parameter is set to a low value,e.g. $\{0.05, 0.1\}$, the online KDE produces approximations with smaller errors than those obtained by a suboptimal batch KDE and the online EM, while the errors are comparable to those of the batch optimal KDE . In terms of the model's complexity, the online KDE produces models with a much lower number of components than the batch methods. While the complexity of the models is comparable to the models produced by the online EM, the online KDEs produce significantly lower errors. Therefore, with increasing the number of samples, the online KDEs converge in terms of estimation accuracy to the models produced by the batch state-of-the-art KDEs, and do so using significantly less components in their models. The reason is that the optimization (in compression as well as bandwidth selection) is not merely local in time, but is in fact distributed through time. Each new observation improves the models, bringing them closer to the optimum in terms of accuracy, while maintaining a low complexity. In the online KDE, the errors of approximation using $H_{dist} = 0.05$ and $H_{dist} = 0.1$ were virtually equal for number of samples lower than 600. As the number of samples grew, the approximation error was decreasing faster for $H_{dist} = 0.05$, however, at a cost of an increased complexity. To balance between the model complexity and approximation error, we choose $H_{dist} = 0.1$ for all our subsequent experiments. For additional examples of online estimation of probability density functions using the online KDE, see http://vicos.fri.uni-lj.si/data/matejk/ivcj08/SupplementalMaterial.htm.

3.2. Unlearning: A toy example

To demonstrate how the unlearning from the Algorithm 2 can be used in interactive learning, we consider a toy-example of training a cognitive agent to learn the concept of a *red color*. Assume that we present the agent with an object to which we refer as a "red fork" (Figure 6a). The agent tries to learn the concept of the *red color* by sampling hue values of pixels corresponding to the fork (green dots in Figure 6a) and constructs a mixture model $p_{\text{red}}(x)$ from the sampled values (Figure 6b). Note that two modes arise in $p_{\text{red}}(x)$ – one for the hue values of the red handle and one for the hue values of the yellow head. The model $p_{\text{red}}(x)$ can now be used to calculate a belief of whether the color of a given pixel is red or not. The beliefs of all pixels in the fork image (Figure 6a) are shown in (Figure 6h). Note that high beliefs are assigned to the color of the handle and even higher to the color of the head. Thus the agent would wrongly believe that the red as well as the yellow hues make up the concept of the *red color*. To rectify this we present a yellow ball (Figure 6c) and say that its color is *yellow* and *NOT red*. As before, hue values are sampled from the ball and a mixture model $p_{\text{yell}}(x)$ is constructed (Figure 6d). An attenuation function $f_{\text{att}}(x)$

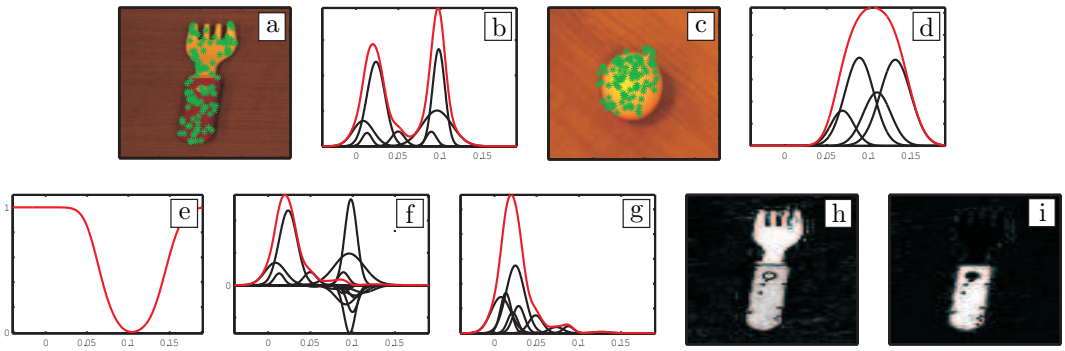

Figure 6: Hue values are sampled from (a) to initialize the mixture model $p_{\text{red}}(x)$ (b). The mixture $p_{\text{yell}}(x)$ corresponding to the sampled hue values of a yellow ball (c) is shown in (d). The attenuation function $f_{\text{att}}(x)$ is shown in (e) and the final $p_{\text{red}}(x)$ before and after compression is shown in (f) and (g), respectively. The belief images corresponding to $p_{\text{red}}(x)$ before and after unlearning, (b) and (g), are shown in (h) and (i), respectively. White colors correspond to high beliefs, while dark colors correspond to low beliefs. The mixtures in (b,d,f,g) are show in bright (red) lines, while their components are depicted in black lines.

(Figure 6e) is calculated from $p_{\text{yell}}(x)$ and used to unlearn the corresponding parts of $p_{red}(x)$; the resulting mixture is shown in (Figure 6f). After compression, we obtain the corrected model of the *red color* concept $\hat{p}_{\text{red}}(x)$ (Figure 6g). Note that the mode corresponding to the yellow color has been attenuated, which is also verified in the belief image (Figure 6i) where we have used $\hat{p}_{\text{red}}(x)$ to calculate the beliefs of hue values in (Figure 6a). The belief image shows that now only the colors of pixels on the fork's handle are believed to correspond to the concept of the *red color*.

3.3. Interactive learning of basic visual concepts

To further demonstrate the strength of the proposed algorithms for online learning we have embedded them into a system for continuous online learning of basic visual concepts [34]. The system operates by learning associations between six object properties (four colors and two shapes) and six low-level visual features (median hue value, eccentricity of the segmented region, etc.). Once an association (a concept) is created, a detailed model of each concept (properties-feature association) is constructed. As a testbed we have used a set of everyday objects (see Figure7a for examples).

In the process of learning, a tutor presented one object at a time to the system and provided its description – the concept labels. The system created associations between features and concept labels such that, for example, the concepts of colors were associated with the hue feature. The associated visual features were modelled by KDEs using the algorithms proposed in this paper. Therefore, the concept of the green color, for example, was modelled by a KDE over the hue values of the observed green objects.

In this experiment, a set of 300 images of everyday objects was randomly split in two sets of 150 images. One set was used for training and the other for testing. The images entered the learning system one by one and at every step the quality of the current models was evaluated by trying to recognize the visual properties of all testing images. The *accuracy* of recognition was defined as the ratio between the number of correctly recognized concepts and the number of all the concepts from the test set. The experiment was repeated 50 times with different training and test sets and the results were averaged.

The evolution of the accuracy of the concept recognition is shown in Figure 7(b). It is evident that the overall accuracy increases by adding new samples. The growth of the accuracy is very rapid at the beginning, when new models of newly introduced concepts are being added, and remains positive even after all models are formed, which is due to refinement of the corresponding representations (KDEs). Note that this refinement does not come from increasing the number of components in the KDEs. This can be seen in Figure 7(c), where we show how the number of components evolved on average with new observations. Initially, the number of components is rapidly increasing, and after a while it remains approximately constant.

To demonstrate the unlearning algorithm, we repeated the experiment, but now every 10-th training sample was labelled incorrectly in the first half of the incremental learning process. As a result, the underlying

KDE representations were corrupted by the incorrect feature values and the recognition accuracy degraded (red line in Figure 7b). However, by applying unlearning to the corrupted representations, they were successfully corrected, which resulted in a significant improvement of the recognition accuracy (blue line). The recognition results after unlearning were very similar to those obtained in an error-free learning process (compare the blue and the green line in Figure 7b).

Figures 7(d-g) show an example of the evolution of the individual models over time. At the beginning, the models were very simple and rather weak, since they were obtained considering only a few training samples (Figure 7d). However, as new samples were observed, they adapted to the variability of the individual concepts. Figure 7(g) shows the models after all 150 training samples have been observed. The efficiency of the compression algorithm can be seen by inspecting columns in Figures 7(e) and (f) – most obvious compression results are seen in the second and the last column. The compressed models resemble the original ones at a very high degree. Figure 7(h) shows an example of updating the final models from Figure 7(g) by incorrectly labelled training samples. The proposed unlearning algorithm was then applied to these models by unlearning the incorrectly presented images and concepts. Figure 7(i), which depicts the obtained models, shows that the error recovery was successful and that the information that was incorrectly added into the models was successfully removed.

4. Conclusion

A new approach to online estimation of Gaussian mixture models for interactive learning was proposed. The approach consists of three main contributions. The first contribution is a new approach to incremental Gaussian mixture models which allows online estimation of probability density function from the observed samples. This approach was derived by an online extension of a batch kernel density estimation (KDE). The second contribution is a method for unlearning parts of the learned mixture model, which allows for a more versatile learning. The third contribution is a method for maintaining a low complexity of the learned mixture models, a compression, which is based on iterative removal of the mixture components and minimization of the L_2 distance between the original mixture and its approximation. Results of the experiments have shown that, in an online estimation of the probability density functions, a crucial part is to allow the models to have some redundant complexity, so that they can efficiently adapt to the future data. We have seen that if the errors introduced by the model compressions are kept sufficiently low, the adaptation will be successful even after observing a large number of samples. In our approach, these errors were quantified in terms of the Hellinger distance and we have proposed a numerical approximation for its evaluation between two mixture models.

The performance of the online kernel density estimation (OKDE) was first demonstrated with an example of online estimation of a complex distribution from individual samples. In terms of the error between the approximations and the reference distribution, the OKDE outperformed an online EM-based mixture model, a widely-used batch KDE and produced results which were comparable to a batch state-of-the-art KDE. In contrast to the batch KDEs, where the model complexity increased with the number of samples, the OKDE maintained a low complexity even after a large number of samples have been observed. We have applied the proposed methodologies for online learning/unlearning to a tutor-supervised interactive learning of basic visual concepts in a cognitive agent. In this experiment the tutor presented the agent with an object and provided labels (concepts) associated with the objects visual properties. The agent was gradually building representations of the visual concepts using the OKDE and was able to achieve a high accuracy of recognition when the tutor provided error-free labels. When the labels contained errors, the recognition decreased, since false examples got incorporated into the models. However, using the unlearning strategy, the tutor was able to easily improve the models such that the accuracy of the recognition increased back to the level of learning with error-free labels.

The methodologies proposed in this paper were designed for online estimation of models for stationary distributions (i.e., distributions which do not change in time), and have been demonstrated to allow an efficient approach to online learning in cognitive agents. In our future work we will extend this approach and apply it to other online learning problems which consider nonstationary distributions as well. Another important aspect to be further researched is the apparent nonstationarity of the distribution which can come

form a particular order in which the data arrive. We expect that this will have important effects on the convergence properties of the method. We have noticed in our experiments that, while the online KDEs converge to those of the batch state-of-the-art, initialization plays an important role in the rate of convergence. If the initial data points are sampled randomly from the target distribution they convey the information of the distribution's scale, the bandwidths are initialized properly and the models converge fairly fast to the optimum estimates. However, in a pathological situation the initial data might be sampled only from a small part of the distribution. Then the initial scale would have been severely underestimated, resulting in underestimated initial bandwidths and under-smoothed distribution. During the online approximation many additional samples would have been required to remedy these effects. Another pathological situation would be the case when the bandwidths are initially severely overestimated, producing an overestimated distribution. In that situation, it is likely that the models would be initially over-compressed. Indeed, compressions which are valid at some point in time can turn out to be invalid only after many additional samples arrive. Again, more samples would be required to remedy the effects of these early compressions. The reasons for such behavior in the pathological situations is that the optimizations involved in compression and bandwidth selection produce only a locally optimal solutions. While these situations typically do not come to effect when the samples arrive randomly from a stationary distribution, we expect them to be significant when considering nonstationary distributions and ordered data. A further consideration is deriving a faster method for compressing the mixture models. Indeed, we have found that, while the bandwidth selection consumes only a fraction of the method's processing time, the majority of the processing time is spent for compression of the mixture models. In our future work, we will also consider derivation of a faster compression method which would provide a comparable estimation accuracy.

Acknowledgement

This research has been supported in part by: Research program P2-0214 (RS), EU FP6-004250-IP project CoSy and EU FP7-ICT215181-IP project CogX.

References

[1] A. Holub, P. Perona., M. Burl, Entropy-based active learning for object recognition, in: Workshop on Online Learning for Classification, in conjunction with Conf. Comp. Vis. Pattern Recognition, 2008, pp. 1–8.
[2] E. F. project, CoSy: Cognitive systems for cognitive assistants, http://www.cognitivesystems.org (2004-2008).
[3] E. F. project, CogX: Cognitive systems that self-understand and self-extend, http://cogx.eu (2008-2012).
[4] E. Ardizzone, A. Chella, M. Frixione, S. Gaglio, Integrating subsymbolic and symbolic processing in artificial vision, Journal of Intelligent Systems 1(4) (1992) 273–308.
[5] A. M. Arsenio, Developmental learning on a humanoid robot, in: IEEE International Joint Conference On Neural Networks, 2004, pp. 3167–3172.
[6] S. Kirstein, Wersing, E. H., Körner, Rapid online learning of objects in a biologically motivated recognition architecture, in: 27th DAGM, 2005, pp. 301–308.
[7] Online learning for classification workshop, in conjunction with IEEE Computer Society Conference on Computer Vision and Pattern Recognition (June 2007).
[8] Online learning for classification workshop, in conjunction with IEEE Computer Society Conference on Computer Vision and Pattern Recognition (June 2008).
[9] M. P. Wand, M. C. Jones, Kernel Smoothing, Chapman & Hall/CRC, 1995.
[10] D. W. Scott, W. F. Szewczyk, From kernels to mixtures, Technometrics 43 (3) (2001) 323–335.
[11] J. Goldberger, S. Roweis, Hierarchical clustering of a mixture model, in: Neural Inf. Proc. Systems, 2005, pp. 505–512.
[12] K. Zhang, J. T. Kwok, Simplifying mixture models through function approximation, in: Neural Inf. Proc. Systems, 2006.
[13] G. J. Mc Lachlan, T. Krishan, The EM algorithm and extensions, Wiley, 1997.
[14] M. A. F. Figueiredo, A. K. Jain, Unsupervised learning of finite mixture models, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 381–396.
[15] Z. Živkovič, F. van der Heijden, Recursive unsupervised learning of finite mixture models 26 (5) (2004) 651 – 656.
[16] A. Corduneanu, C. M. Bishop, Artificial Intelligence and Statistics, Morgan Kaufmann, Los Altos, CA, 2001, Ch. Variational Bayesian model selection for mixture distributions, pp. 27–34.
[17] C. A. McGrory, D. M. Titterington, Variational approximations in Bayesian model selection for finite mixture distributions, Comput. Stat. Data Analysis 51 (11) (2007) 5352–5367.
[18] M. Song, H. Wang, Highly efficient incremental estimation of gaussian mixture models for online data stream clustering, in: SPIE: Intelligent Computing: Theory and Applications, 2005, pp. 174–183.

[19] O. Arandjelovic, R. Cipolla, Incremental learning of temporally-coherent gaussian mixture models, in: British Machine Vision Conference, 2005, pp. 759–768.
[20] A. Declercq, 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, 2008, pp. 605–611.
[21] W. F. Szewczyk, Time-evolving adaptive mixtures, Tech. rep., National Security Agency (2005).
[22] B. Han, D. Comaniciu, Y. Zhu, L. S. Davis, Sequential kernel density approximation and its application to real-time visual tracking, IEEE Trans. Pattern Anal. Mach. Intell. 30 (7) (2008) 1186–1197.
[23] D. Comaniciu, V. Ramesh, P. Meer, The variable bandwidth mean shift and data-driven scale selection, in: Proc. Int. Conf. Computer Vision, Vol. 1, 2001, pp. 438 – 445.
[24] M. J. F. Gales, S. S. Airey, Product of gaussians for speech recognition, Computer Speech & Language 20 (1) (2004) 22–40.
[25] M. Girolami, C. He, Probability density estimation from optimally condensed data samples., IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (2003) 1253–1264.
[26] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, R. C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comp. 13 (7) (2001) 1443–1471.
[27] A. Leonardis, H. Bischof, An efficient mdl-based construction of rbf networks, Neural Networks 11 (5) (1998) 963 – 973.
[28] H. Bischof, A. Leonardis, View-based object representations using rbf networks, "IVC" 19 (2001) 619–629.
[29] D. E. Pollard, A user's guide to measure theoretic probability, Cambridge University Press, 2002.
[30] S. Julier, J. Uhlmann, A general method for approximating nonlinear transformations of probability distributions, Tech. rep., Department of Engineering Science, University of Oxford (1996).
[31] M. C. Jones, J. S. Marron, S. J. Sheather, A brief survey of bandwidth selection for density estimation, J. Amer. Stat. Assoc. 91 (433) (1996) 401–407.
[32] K. P. Burnham, D. R. Anderson, Multimodel inference: Understanding aic and bic in model selection, Sociological Methods and Research 33 (2) (2004) 261–304.
[33] Matlab - the language of technical computing (2009).
URL `http://www.mathworks.com/products/matlab/`
[34] D. Skočaj, M. Kristan, A. Leonardis, Continuous learning of simple visual concepts using Incremental Kernel Density Estimation, in: International Conference on Computer Vision Theory and Applications, 2008.
[35] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical recipes in C (2nd ed.): the art of scientific computing, Cambridge University Press, 1992.
[36] E. Veach, L. J. a. Guibas, Optimally combining sampling techniques for monte carlo rendering, in: Computer graphics and interactive techniques, 1995, pp. 419 – 428.

A. The unscented Hellinger distance

In this appendix we derive a numerical approximation to the Hellinger distance between two one-dimensional Gaussian mixture models $p_1(x)$ and $p_2(x)$ via the *unscented transform*. The unscented transform is a special case of a numerical integration technique called a Gaussian quadrature [35], which uses a small set of carefully placed samples to evaluate nonlinear transformations of Gaussian variables (see, e.g. [30]). The squared Hellinger distance [29] between $p_1(x)$ and $p_2(x)$, is defined as

$$H^2(p_1,p_2)\stackrel{\Delta}{=}\frac{1}{2}\int(p_1(x)^{\frac{1}{2}}-p_2(x)^{\frac{1}{2}})^2dx. \tag{25}$$

Similarly to a Monte Carlo integration [36] we define an *importance* distribution $p_0(x)=\frac{1}{2}p_1(x)+\frac{1}{2}p_2(x)$, which contains the support of both, $p_1(x)$ as well as $p_2(x)$. In our case, $p_0(x)$ is a Gaussian mixture model of a form $p_0(x)=\sum_{i=1}^{N}w_iK_{h_i}(x-x_i)$, and we rewrite (25) into

$$\begin{aligned}H^2(p_1,p_2) &= \frac{1}{2}\int p_0(x)\frac{(p_1(x)^{\frac{1}{2}}-p_2(x)^{\frac{1}{2}})^2}{p_0(x)}dx\\ &= \frac{1}{2}\sum_{i=1}^{N}w_i\int g(x)K_{h_i}(x-x_i)dx,\end{aligned} \tag{26}$$

where we have defined $g(x)=\frac{(p_1(x)^{\frac{1}{2}}-p_2(x)^{\frac{1}{2}})^2}{p_0(x)}$. Note that the integrals in (26) are simply expectations over a nonlinearly transformed Gaussian random variable x, and therefore admit to the unscented transform.

The unscented squared Hellinger distance is thus defined as

$$H^2(p_1, p_2) \approx \frac{1}{2} \sum_{i=1}^{N} w_i \sum_{j=0}^{2} g({}^{(j)}\mathcal{X}_i) {}^{(j)}\mathcal{W}_i, \tag{27}$$

where $\{{}^{(j)}\mathcal{X}_i, {}^{(j)}\mathcal{W}_i\}_{j=0,1,2}$ are triplets of weighted sigma-points corresponding to the i-th Gaussian $K_{h_i}(x - x_i)$, and are defined as

$$\begin{aligned} {}^{(j)}\mathcal{X}_i &= x_i + (-1)^j \sqrt{1+\kappa} h_i, \\ {}^{(j)}\mathcal{W}_i &= \begin{cases} \frac{\kappa}{1+\kappa} & ; \quad j = 0 \\ \frac{1}{2(1+\kappa)} & ; \quad otherwise \end{cases} . \end{aligned} \tag{28}$$

In line with the discussion on the properties of the unscented transform in [30], we set the parameter κ to $\kappa = 2$.

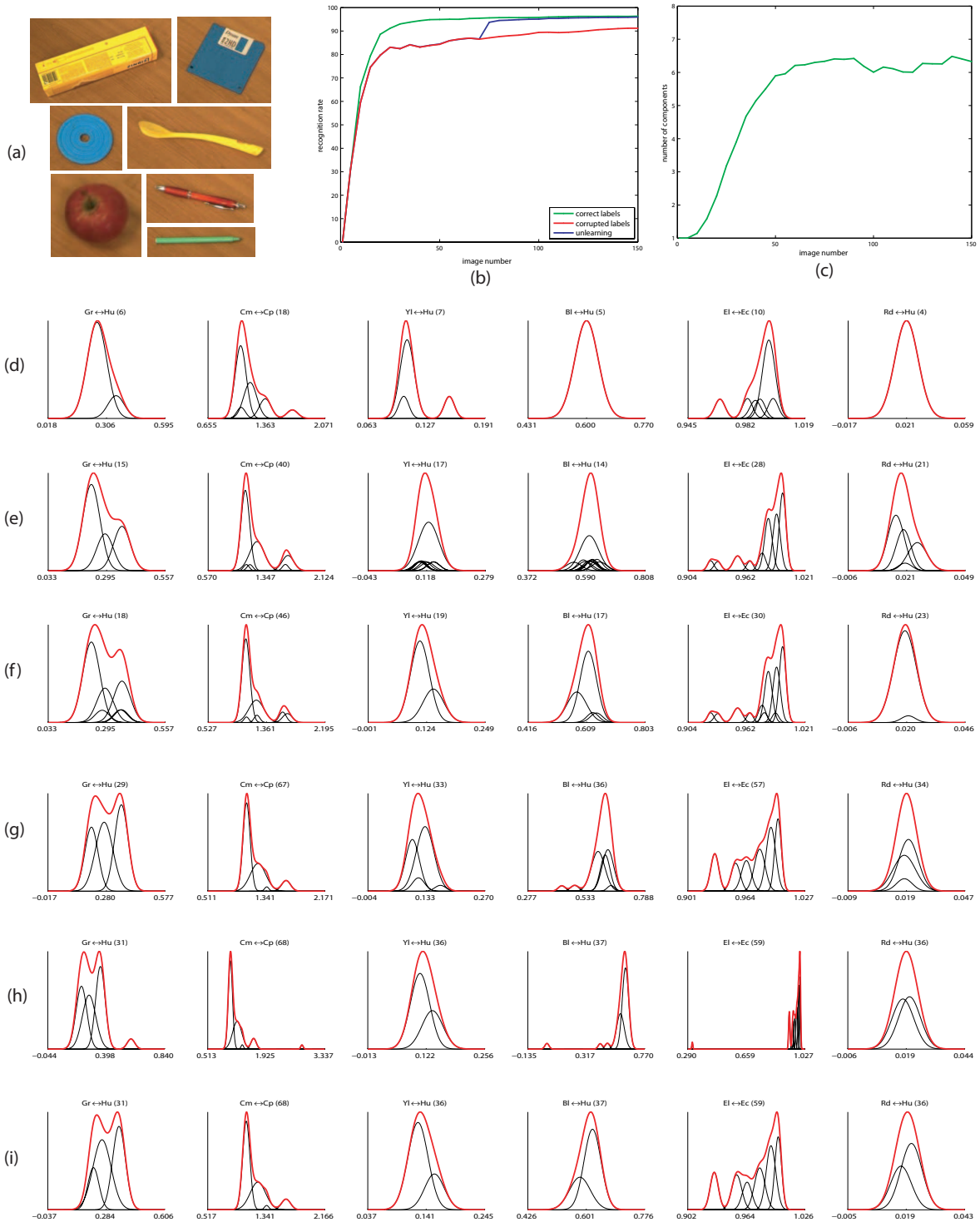

Figure 7: Learning of basic object properties. Seven everyday objects from the database (a). The evolution of the recognition accuracy through time for online learning on correct data (green line), on partially incorrect data (red line), and after unlearning (blue line) (b). The average number of components through time (c). An example of the evolution of the KDE models representing six basic object properties ('green', 'compact', 'yellow', 'blue', 'elongated', 'red') through time after observing 30 (d), 80 (e), 90 (f), and 150 (g) training images. Final models updated with incorrect labels (h). Models after unlearning (i).