Computer Vision Winter Workshop 2008, Janez Perš (ed.)
Moravske Toplice, Slovenia, February 4–6
pages 25-32
Slovenian Pattern Recognition Society, Ljubljana, Slovenia

Incremental learning with Gaussian mixture models

Matej Kristan, Danijel Skočaj, and Aleš Leonardis

Faculty of Computer and Information Science, University of Ljubljana, Slovenia
matej.kristan@fri.uni-lj.si

Abstract *In this paper we propose a new incremental estimation of Gaussian mixture models which can be used for applications of online learning. Our approach allows for adding new samples incrementally as well as removing parts of the mixture by the process of unlearning. 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 an example of online estimation of a complex distribution, an example of unlearning, and with an interactive learning of basic visual concepts.*

1 Introduction

The process of learning in computer vision may be viewed as a task of generating models of visually perceived environment. Since most of the real-world environments are ever-changing and all the information cannot be available (nor processed) at once, the models should be acquired incrementally. Furthermore, models should allow for error-recovery in cases when erroneus information gets incorporated into them. The learning process should thus create, extend, update, delete, and modify models of real-world concepts in a continuous, life-long manner, while still keeping the representations of the environment compact and efficient.

Various models and methods for extracting these models have been proposed in different contexts and tasks. In this paper we discuss modelling data by probability density functions (pdf) based on Kernel Density Estimates (KDE). In particular, we focus on Gaussian mixture models (GMM), which are know to be a powerful tool in approximating various distributions which may be far from Gaussian [15].

In our previous work [12] we have proposed a framework for continuous learning of visual concepts. There, each basic visual concept, such as color or shape, was associated with one of the extracted features (e.g., hue value), using the model which was built from previously observed values of the same feature. These models were in the form of one-dimensional GMMs. In this paper we present the methodology for incremental building GMMs, that facilitates continuous, efficient and flexible learning as discussed above.

Our contributions are threefold. The first contribution is a new approach to incremental Gaussian mixture models which allows online estimation of the probability density function from the observed samples. 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 learned mixture models.

In an incremental setting, where all samples are not observed at once, traditional methods for density estimation based on Parzen estimators [15, 11, 16], expectation maximization (EM) algorithm [8] or variational estimation [9], to name a few, cannot be used directly. To that end, various incremental approaches have been proposed. Han et al. [5] present an incremental approach in which they detect only the modes of the distribution and approximate each mode by a single Gaussian. This approach fails to perform in cases when modes are non-Gaussian, e.g., skewed or uniform distributions. Recently, an online EM algorithm was proposed [1, 13], however, it cannot be applied to general situations, since it assumes a temporal coherence of the incoming data [1]. Szewczyk [14] applies a Dirichlet and Gamma density prior to assign new components to the mixture model in light of the incoming data. A drawback of this approach, however, is that the parameters of the prior need to be specified for a given problem.

In contrast to the above approaches, our approach does not require fine-tuning parameters for specific application, we do not assume specific forms of the target pdfs, temporal constraints are not assumed on the observed data and the proposed models allow for unlearning as well.

The remainder of the paper is structured as follows. In Section 2.1 we present the incremental 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 incremental approximation of complex distributions, unlearning, and apply the proposed methodology to the problem of learning simple concepts of object visual features. Conclusions are drawn in Section 4.

2 Estimation of Gaussian Mixture models

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

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

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

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

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

2.1 Incremental mixture model

Suppose that we have observed a set of n_t samples $\{x_i\}_{i=1:n_t}$ up to the current time-step t. The problem of modelling samples by a probability density function can be posed as a problem of kernel density estimation [15]. 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

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

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)

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

Applying a Taylor expansion, assuming a large sample-set and noting that the kernels in $\hat{p}_t(x; h_t)$ are Gaussians ([15], p.19), we can write the *asymptotic* MISE (AMISE) between $\hat{p}_t(x; h_t)$ and $p_t(x)$ as

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

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

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

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. [15]), however these require access to *all* observed samples, which is in contrast to the incremental learning where we wish to discard previous samples and retain only compact representations of them. Our setting is thus 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). Thus the incremental estimation of the mixture model boils down to estimating the *bandwidth* of the Gaussian kernel for the currently observed sample; in the following we propose an iterated plug-in rule to achieve that.

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)

$$\hat{h}_t = c_{\text{scale}}[2\sqrt{\pi}R(\hat{p}_t''(x))n_t]^{-1/5}, \quad (7)$$

where c_{scale} is used to increase the bandwidth a little and thus avoid undersmoothing. Experimentally, we determined that the values of the scale parameter $c_{\text{scale}} \in [1, 1.5]$ in (7) give reasonable results and in all our experiments $c_{\text{scale}} = 1.3$ is used. 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

$$\hat{p}_t(x) = (1 - \frac{1}{n_t})\hat{p}_{t-1}(x) + \frac{1}{n_t}K_{\hat{h}_t}(x - x_t). \quad (8)$$

Next, the improved estimate $\hat{p}_t(x)$ from (8) is plugged back in the equation (7) to re-approximate $\hat{h}_t$ and then equations (7) and (8) 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 low complexity, the model is compressed from time to time into a model with a smaller number of components. A detailed description of the compression algorithm will be given in a later section. The procedure for incremental learning of mixture models is outlined in Algorithm 1.

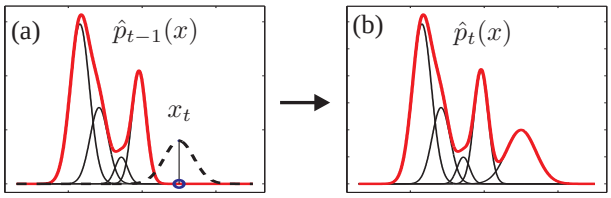

Figure 1: The left image shows a Gaussian mixture model $\hat{p}_{t-1}(x)$ from time-step $t-1$ (bold line) and the currently observed sample x_t (circle). A Gaussian kernel is centered on x_t (dashed line) and used to update $\hat{p}_{t-1}(x)$. The right image shows the current, updated, mixture $\hat{p}_t(x)$.

Algorithm 1 : Incremental density approximation

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 (7) using $\hat{p}_t(x)$.
3: Reestimate $\hat{p}_t(x)$ according to (8) 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.

2.2 Unlearning in mixture models

To increase the flexibility of incremental learning with mixture models we propose a method for *unlearning* parts of

the mixtures. Assume that by observing values of some feature x, we have constructed the following M_{ref}-component Gaussian mixture model

$$p_{ref}(x) = \sum_{i=1}^{M_{\text{ref}}} w_i K_{h_i}(x - x_i). \quad (9)$$

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

$$p_{neg}(x) = \sum_{j=1}^{M_{\text{neg}}} \eta_j K_{s_j}(x - y_j), \quad (10)$$

which represents a *negative example* to the $p_{ref}(x)$. For example, $p_{\text{ref}}(x)$ might model the hue values of a red color and $p_{\text{neg}}(x)$ could be a pdf obtained from the hue values of a yellow object. The question is how to incorporate $p_{\text{neg}}(x)$ into the reference $p_{\text{ref}}(x)$. One way to do that is to attenuate regions in $p_{\text{ref}}(x)$ that correspond to high probabilities in $p_{\text{neg}}(x)$ and leave those regions that correspond to low probabilities in $p_{\text{neg}}(x)$ unchanged. This can be achieved by constructing an *attenuation* function that 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. The $p_{\text{neg}}(x)$ can thus be incorporated into $p_{\text{ref}}(x)$ simply by multiplying $p_{\text{ref}}(x)$ with the attenuation function. We describe this procedure next.

The attenuation function is defined as

$$f_{att}(x) = 1 - C_{\text{opt}}^{-1} p_{\text{neg}}(x), \quad (11)$$

where the normalization constant C_{opt} makes sure 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 [2] which we initialize at the centers of the components of $p_{\text{neg}}(x)$ to detect the modes of $p_{\text{neg}}(x)$. The mode x_{opt} corresponding to the maximum value of $p_{neg}(x)$ is selected as the maximum of the distribution and the normalization is given as

$$C_{\text{opt}} = p_{\text{neg}}(x_{\text{opt}}). \quad (12)$$

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

$$\begin{aligned} p_{\text{att}}(x) &= C_{\text{norm}}^{-1} p_{\text{ref}}(x) f_{\text{att}}(x) \\ &= C_{\text{norm}}^{-1} [p_{\text{ref}}(x) - f_{\text{com}}(x)], \end{aligned} \quad (13)$$

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 [3], we can rewrite $f_{\text{com}}(x)$ as

$$\begin{aligned} f_{\text{com}}(x) &= \sum_{i=1}^{M_{\text{ref}}} \sum_{j=1}^{M_{\text{neg}}} C_{\text{opt}} w_i \eta_j K_{h_i}(x - x_i) K_{s_j}(x - \mu_j) \\ &= \sum_{i=1}^{M_{\text{ref}}} \sum_{j=1}^{M_{\text{neg}}} z_{ij} \beta_{ij} K_{\sigma_{ij}}(x - \mu_{ij}), \end{aligned} \quad (14)$$

where $\beta_{ij} = C_{\text{opt}} w_i \eta_j$, $\sigma_{ij}^2 = (h_i^{-2} + s_j^{-2})^{-1}$, $\mu_{ij} = \sigma_{ij}^2 (\frac{x_i}{h_i^2} + \frac{y_j}{s_j^2})$, $z_{ij} = \frac{\sigma_{ij}}{\sqrt{2\pi} h_i s_j} exp[\frac{1}{2}(\frac{\mu_{ij}^2}{\sigma_{ij}^2} - \frac{x_i^2}{h_i^2} - \frac{y_j^2}{s_j^2})]$.

From above, we can now derive the normalization C_{norm} for the attenuated pdf $p_{att}(x)$ (13)

$$C_{\text{norm}} = (1 - \sum_{i=1}^{M_{\text{ref}}} \sum_{j=1}^{M_{\text{neg}}} \beta_{ij} z_{ij})^{-1}. \quad (15)$$

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 $p_{ref}(x)$ using the variable-bandwidth mean shift [2].

2: Scale $p_{neg}(x)$ with respect to the detected mode x_{opt} (12) and calculate the attenuation function $f_{att}(x)$ (11).

3: Multiply $f_{att}(x)$ with $p_{ref}(x)$ and normalize to obtain the attenuated mixture $p_{att}(x)$ (13,14,15).

Note that while $p_{\text{att}}(x)$ is indeed a proper pdf, it is not a proper mixture, since some of the weights are negative. Furthermore, by introducing new attenuation functions, the number of components in (13) increases exponentially, which in practice makes subsequent calculations inefficient and slow. For that reason, after attenuation, the resulting distribution needs to be *compressed*, i.e., we require an equivalent mixture with a smaller number of components. Furthermore, 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

$$p(x) = \sum_{i=1}^{M} w_i K_{h_i}(x - x_i), \quad (16)$$

where all w_i are not necessarily positive, but they do sum to one. Our goal is then to approximate the reference (16) with a N-component mixture with all positive weights

$$\hat{p}(x|\theta) = \sum_{j=1}^{N} \gamma_j K_{\sigma_j}(x - \mu_j), \quad (17)$$

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 approximate and the reference mixture can be quantified by the integrated squared error (ISE)

$$\text{ISE}(\theta) = \int (p(\mathbf{x}) - \hat{p}(\mathbf{x}|\theta))^2 d\mathbf{x}. \quad (18)$$

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

$$\hat{\theta} = \arg\min_{\theta} [\int \hat{p}^2(\mathbf{x}|\theta) d\mathbf{x} - 2E_{p(\mathbf{x})}\{\hat{p}(\mathbf{x}|\theta)\}], \quad (19)$$

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 (19). 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 (19) with respect to the weights γ_i of $\hat{p}(x|\theta)$ (17). By driving some weights of (17) to zero we are effectively removing the corresponding components. A useful insight into such optimization is provided by the theory of reduced-set-density estimation [4] and earlier results from support estimation in support vector machines [10]. In [4], Girolami and He proposed a reduced-set-density approximations of kernel density estimates. A central point of their approach was minimization of ISE-based criterion, which was in spirit similar to our formulation in (18). In line with their observations, we now inspect how the two terms of the right-hand side of (19) 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 (19) is kept fixed, minimization is obtained by maximizing the second term $E_{p(x)}\{\hat{p}(x|\theta)\}$. By expanding this term we have

$$E_{p(x)}\{\hat{p}(x|\theta)\} = \sum_{j=1}^{N} \gamma_j \tilde{p}_j, \tag{20}$$

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 (20) 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. This means that the term $E_{p(x)}\{\hat{p}(x|\theta)\}$ is *sparsity-inducing* in that it prefers those components of the approximating distribution $\hat{p}(x|\theta)$ that correspond to the high-probability regions in $p(x)$.

Now note that minimizing ISE (19) 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 yields a weighted sum of expectations of pairs of components of $\hat{p}(x|\theta)$

$$\int \hat{p}^2(x|\theta)dx = \sum_{i=1}^{N}\sum_{j=1}^{N} \gamma_i \gamma_j c_{ij}, \tag{21}$$

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 large weights to the components that do not overlap and low weights to those components that do overlap. Thus we can say that the term (21) 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 (19) 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 those configurations where the selected components are far apart.

Using (20,21) we can rewrite minimization of ISE (19) with respect to the weights γ_i into a classical quadratic program

$$\arg\min_{\gamma}\{\frac{1}{2}\gamma^T\mathbf{C}\gamma - \gamma^T\mathbf{P}\}\ ;\ \gamma^T\mathbf{1} = 1, \gamma_j > 0, \forall j, \tag{22}$$

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} (21) and a vector of $\mathbf{P} = [\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_N]$ with elements $\tilde{p}_j$ (20).

Note that since all components in the reference and the approximating distributions (16,17) are Gaussians, $\mathbf{C}$ and $\mathbf{P}$ in (22) can be evaluated analytically. There is a number of optimization techniques available for solving the quadratic program (22). In our approach we use a variant of a Sequential Minimization Optimization (SMO) scheme [10], which was previously used by Girolami and He [4] for optimization of a form similar to (22).

2.4 Compression algorithm

Using the results from previous section, we propose an iterative compression algorithm, which is similar in spirit to [7], 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 the 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 (18).

We propose an algorithm for compression which proceeds in 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 reduce the error between the reduced $\hat{p}(x|\theta)$ and the reference $p(x)$. This is achieved by optimizing (19) w.r.t. all parameters θ in $\hat{p}(x|\theta)$ using a Levenberg-Marquardt 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 $\hat{p}(x|\theta)$. We can

Algorithm 3 : Compression algorithm.

1. *Initialization*: construct $\hat{p}(x|\theta)$ from $p(x)$, such that all components have positive weights (see text).
2. *Reduction*:
   - Inflate $\hat{p}(x|\theta)$ by increasing the covariances σ_j^2 by some $\alpha > 1$, i.e., $\sigma_j^2 \leftarrow \alpha\sigma_j^2$.
   - Optimize (22) between the inflated $\hat{p}(x|\theta)$ and $p(x)$ w.r.t. γ using a sequential minimal optimization.
   - Remove those components from $\hat{p}(x|\theta)$ for which $\gamma_i = 0$; allow at most five optimization steps.
3. *Organization*: Optimize ISE between $\hat{p}(x|\theta)$ and $p(x)$ using a Levenberg-Marquardt optimization w.r.t. θ.
4. If no components were removed during the reduction procedure, then optimize θ using a Levenberg-Marquardt optimization until convergence and terminate the algorithm, otherwise go back to step (2).

influence the number of the removed components at each step by introducing the *inflation* parameter α, which increases the variances of components before the *reduction* step (see Algorithm 3). For a large α many components of $\hat{p}(x|\theta)$ will overlap significantly and thus many components will be removed. However, by choosing a very large α, e.g. $\alpha > 5$, we also risk that too many components may be removed and the resulting components will inefficiently approximate $p(x)$. In our experience, selecting $\alpha < 5$ results in a reasonable subset of components removed at each stage. In all subsequent experiments, we use $\alpha = 2$. 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 inflation and will be removed in the *reduction* step. Thus in practice we allow at most five Levenberg-Marquardt iterations at each organization step. Only after no more components are removed we let the Levenberg-Marquardt optimization to run until convergence.

3 Experimental Results

Three sets of experiments were conducted to evaluate the proposed methods for incremental learning of mixture models. The first two experiments were designed to demonstrate incremental learning of complex mixtures and to demonstrate how unlearning can be used for more versatile learning. The third experiment shows the results of the application of the proposed algorithms for learning of basic visual properties.

3.1 Incremental learning of complex distributions

The aim of the first experiment was to demonstrate the incremental bandwidth selection method proposed in Section 2. We generated 1000 samples from a 1D mixture of two Gaussians 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 Algorithm 1; the mixture compression was executed whenever the number of components in the mixture exceeded $N_{\text{comp}} = 20$ components. At each time-step two other KDE approximations were also built for reference: an optimal and a suboptimal. These were *batch approximations*, and were thus built by processing *all* samples observed up to the given time-step simultaneously. The optimal bandwidth was estimated using the solve-the-equation plug-in method [6], which is currently theoretically and empirically one of the most successful bandwidth selection methods. For the suboptimal bandwidth selection we have chosen Silverman's rule-of-thumb ([15], page 60), which was also used to initialize our algorithm from the first two samples.

The results are shown in Figure 2. The final KDEs, after observing all 1000 samples, estimated by the proposed *incremental* method and the two reference *batch* methods, are shown in Figure 2(a). It is clear that Silverman produced an oversmoothed approximation to the ground-truth distribution. The incrementally constructed KDE and the batch-estimated KDE using the solve-the-equation plug-in both visually agree well with the ground-truth. This can be further quantitatively verified from Figure 2(b) where we show how the integrated squared error (ISE) between the three approximations and the ground-truth was changing as new samples were observed. While initially errors are high for all three approximations they decrease as new samples arrive. As expected, the error of the KDE calculated using Silverman's rule remains high even after all 1000 samples have been observed. On the other hand, the error decreases faster for the KDE constructed by the proposed method and comes close to the error of the optimally selected bandwidth with increasing numbers of samples. Note that the optimal bandwidth was calculated using *all* samples observed up to a given step. On the other hand, the proposed incremental method produced similar bandwidths using only a low-dimensional *representation* of the observed samples (Figure 2c).

3.2 Unlearning: A toy example

To demonstrate how the unlearning from Algorithm 2 can be used in interactive learning, we consider a toy-example of training a system to learn the concept of a *red color*. Assume that we present the system with an object to which we refer as a *red* fork (Figure 3a). The system tries to learn the concept of the *red color* by sampling hue values of pixels corresponding to the fork (green dots in Figure 3a) and constructs a mixture model $p_{\text{red}}(x)$ from the sampled values (Figure 3b). 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 3a) are shown in (Figure 3h). Note that high beliefs are assigned to the color of the handle and even higher to the color of the head. Thus the system wrongly believes that the red as well

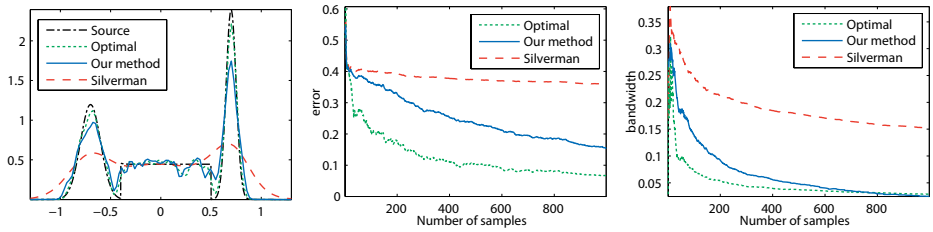

Figure 2: Illustration of the incremental mixture models: (a) final estimated distributions, (b) ISE with respect to the source distribution, (c) estimated bandwidth.

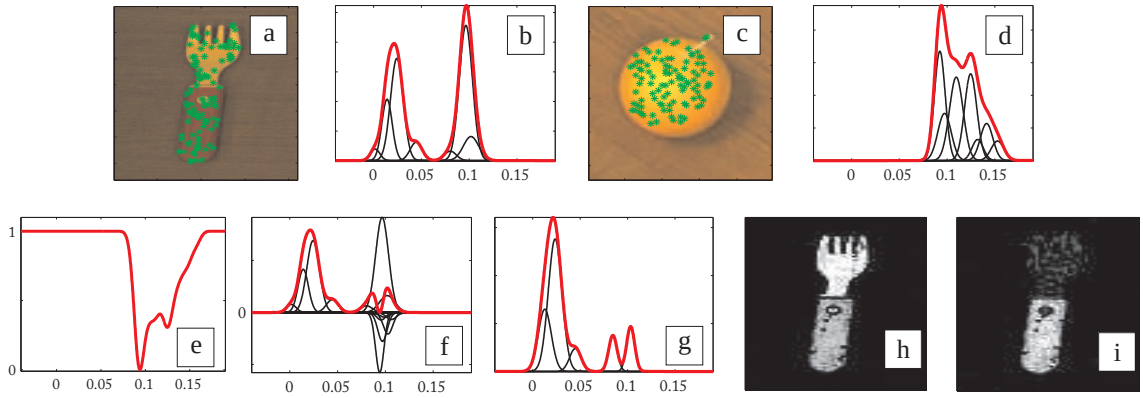

Figure 3: The mixture model $p_{\mathrm{red}}(x)$ of hue values sampled from an image of fork (a) is shown in (b). The mixture $p_{\mathrm{yell}}(x)$ corresponding to the hue values of a yellow ball (c) is shown in (d). The attenuation function $f_{\mathrm{att}}(x)$ created from $p_{\mathrm{yell}}(x)$ (d) is shown in (e), the unlearned distribution of (b) is shown in (f) and the final model $\hat{p}_{\mathrm{red}}(x)$ after compression is shown in (g). The belief images corresponding to the mixture models (b) and (g) are shown in (h) and (i), respectively. White colors correspond to high beliefs, while dark colors correspond to low beliefs. In mixture models (b,d,f,g), the separate components are drawn in thin lines, while the resulting pdfs are shown in thick lines.

as yellow hues make up the concept of the *red color*. To rectify this we show to the system a yellow ball (Figure 3c) and say that its color is *yellow* and *NOT red*. As before, hue values are sampled from the ball and a mixture model $p_{\mathrm{yell}}(x)$ is constructed (Figure 3d). An attenuation function $f_{\mathrm{att}}(x)$ (Figure 3e) is calculated from $p_{\mathrm{yell}}(x)$ and used to unlearn the corresponding parts of $p_{red}(x)$; the resulting mixture is shown in (Figure 3f). After compression, we obtain the corrected model of the *red color* concept $\hat{p}_{\mathrm{red}}(x)$ (Figure 3g). Note that the mode corresponding to the yellow color has been attenuated, which is also verified in the belief image (Figure 3i) where we have used $\hat{p}_{\mathrm{red}}(x)$ to calculate the beliefs of hue values in (Figure 3a). 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 incremental learning, unlearning and compression, we have embedded them into a system for continuous learning of basic visual concepts, which we have developed in our previous work [12].

The goal was to learn 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.). As a testbed we have used a set of 125 everyday objects (for examples, see Figure 4a). A tutor presented one object at a time to the system and provided its description, i.e., the concept labels. The system created associations between features and concepts such that, for example, the colors were associated with the hue feature, etc. The associated visual features were modelled by Gaussian mixture models using the algorithms proposed in this paper. Therefore, the concept of the green color, for instance, was modelled with a mixture model over the hue values of the observed green objects.

Figures 4(d-f) depict the evolution of the individual models over time. At the beginning (Figure 4d), the models were very simple and rather weak, since they were obtained considering only a few training examples. However, by observing new training samples, they improved and were able to adapt to the variability of the individual visual concepts. Figure 4(f) shows the models after all 120 training samples have been observed.

After each training sample was observed and added to the models, the updated models were evaluated by trying to recognize the colors and shapes of the objects in 300 test images (which have not been observed during the training). Figure 4(b) shows the evolution of the overall accuracy of the recognition of concepts. 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, but still grows even after all models are formed due to refinement of the corresponding representations. Note that after observing 120 samples, the accuracy of recognition using the models

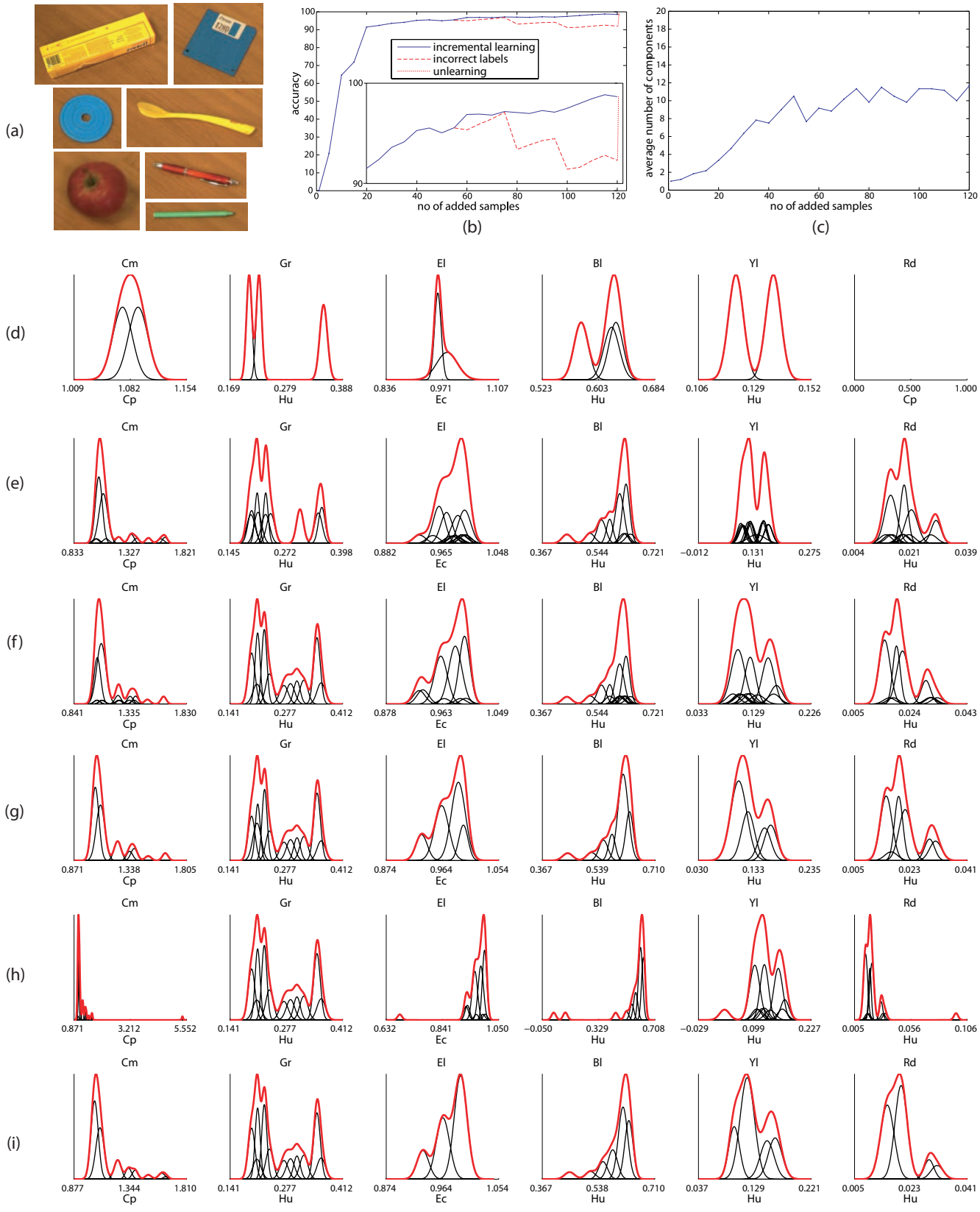

Figure 4: Learning of basic object properties. Seven everyday objects from the database are shown in (a). Results of incremental learning are shown in (b) and (c). Image (b) shows the evolution of accuracy through time for incremental learning on correct data (solid line), on partially incorrect data (dashed line), and after unlearning (dotted line). For better visualization, we also show magnified graphs of the evolution of accuracy in (b) from 20 to 120 added samples. Image (c) shows the evolution of the average number of components through time. The KDE models representing six basic object properties ('compact', 'green', 'elongated', 'blue', 'yellow', 'red') are shown in (d), (e) and (f) afer adding 15, 85, 120 training samples, respectively. Image (g) shows these models after compression. The final models learned with incorrect labels are shown in (h) and (i) shows these models after unlearning.

in Figure 4(f) was 98.6% (Figure 4b, solid line).

Figure 4(c) shows the average number of Gaussian components averaged over all mixture models. Observe that after a while this number does not grow any more, thus the model size remains limited. The models do not improve by increasing their complexity, but rather due to refinement of the underlying representations. The low complexity of the models was maintained by the proposed compression algorithm, which compressed each mixture model in which the number of components exceeds 15. For an example of compression compare the sixth column of Figs. 4(e) and (f). To further demonstrate the compression, all final mixture models from Figure 4(f) were compressed. Resulting models are shown in Figure 4(g) and resemble the original ones at a very high degree of accuracy. Note that the accuracy of recognition before and after compression did not change and remained at a 98.6% level.

To demonstrate the unlearning algorithm, we have repeated the learning experiment, but this time we labelled incorrectly every 20-th training sample in the second half of the incremental learning process. As a result, the mixture models were corrupted and now accounted also for the incorrect feature values. The resulting models after observing 120 training samples are shown in Figure 4(h); the accuracy of recognition using these models was 92.3% (Figure 4b, dashed line). The unlearning algorithm was then applied to these models by unlearning the incorrectly presented images and concepts. Figure 4(i) depicts the *unlearned* models. Note that the information that was incorrectly added into the models was successfully removed and the models obtained were very similar to those obtained in an error-free learning process. The accuracy of recognition using the unlearned models increased from 92.3% back to 98.6% (Figure 4(b), dotted line).

4 Conclusion

A new approach to incremental estimation of Gaussian mixture models was proposed, which can be applied to problems of online learning. Our contributions are threefold. The first contribution is a new approach to incremental Gaussian mixture models which allows online estimation of the probability density function from the observed samples. This approach was derived by incrementalizing a batch kernel density estimation. The second contribution is a method for unlearning parts of the learned mixture model, which allows for a more versatile learning. This is achieved by deriving an attenuation function which in effect attenuates parts of the mixture model. The third contribution is a method for maintaining a low complexity of the learned mixture models, which is based on iterative removal of mixture components and minimization of the L_2 distance between the approximation and the original mixture.

The strength of the proposed methods was demonstrated with an example of online estimation of a complex distribution, an example of unlearning, and with an interactive learning of basic visual concepts. The presented approach deals with one-dimensional distributions and the results are very promising. In our future work we will extend this approach and apply it to multi-dimensional online learning problems.

Acknowledgement

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

References

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

[2] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *Proc. Int. Conf. Computer Vision*, volume 1, pages 438 – 445, 2001.

[3] M. J. F. Gales and S. S. Airey. Product of gaussians for speech recognition. *Computer Speech & Language*, 20(1):22–40, 2004.

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

[5] B. Han, D. Comaniciu, and L. Davis. Sequential density approximation through mode propagation: Applications to background modeling. In *Asian Conf. Computer Vision*, 2004.

[6] M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *J. Amer. Stat. Assoc.*, 91(433):401–407, March 1996.

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

[8] G. J. Mc Lachlan and T. Krishan. *The EM algorithm and extensions*. Wiley, 1997.

[9] C. A. McGrory and D. M. Titterington. Variational approximations in Bayesian model selection for finite mixture distributions. *Comput. Stat. Data Analysis*, 51(11):5352–5367, 2007.

[10] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comp.*, 13(7):1443–1471, 2001.

[11] D. W. Scott and W. F. Szewczyk. From kernels to mixtures. *Technometrics*, 43(3):323–335, August 2001.

[12] D. Skočaj, M. Kristan, and A. Leonardis. Continuous learning of simple visual concepts using Incremental Kernel Density Estimation. In *International Conference on Computer Vision Theory and Applications*, 2008.

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

[14] W. F. Szewczyk. Time-evolving adaptive mixtures. Technical report, National Security Agency, 2005.

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

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