Local Adaptive and Incremental Gaussian Mixture for Online Density Estimation

Tianyu Qiu, Furao Shen(✉), and Jinxi Zhao

National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210093, China
tianyuqiu.nju@gmail.com, {frshen,jxzhao}@nju.edu.cn

Abstract. In this paper, we propose an incremental and local adaptive gaussian mixture for online density estimation (LAIM). Using a similarity threshold based criterion, the method is able to allocate components incrementally to accommodate novel data points without affecting previously learned components. A local adaptive learning strategy is presented for estimating density with complex structure in an online way. We also adopt a denoising scheme to make the algorithm more robust to noise. We compared the LAIM to the state-of-art methods for density estimation in both artificial and real data sets, the results show that our method outperforms the compared online counterpart and produces comparable results to the compared batch algorithms.

Keywords: Online density estimation · Gaussian mixture · Local adaptive · Incremental learning

1 Introduction

Let $X = (x_1, x_2, ...x_n)^T$ be a sample of size n from an unknown distribution F with density function f, the problem of density estimation is to construct a estimator $\hat{f}$ from X to approximate f as good as possible. Traditionally, the methods for density estimation are generally divided into two categories, parametric methods and non-parametric ones. Finite mixture models [1] have been used for constructing parametric probabilistic models successfully, but they suffer from choosing the appropriate number of mixture components and are sensitive to initialization. The traditional nonparametric kernel density estimator (Parazen Window) [2] is guaranteed to converge to the underlying density under practical assumptions without worrying about the magic number k (every datapoint is itself a component) [3]. However, it is not easy to choose an appropriate bandwidth parameter to achieve a good performance. A lot of researches have been made to find a data-driven criterion to search for a good value of bandwidth parameter [4], [5], [6]. In addition, the non-parametric methods and the bandwidth selection algorithms generally need to store all the training data in memory which is unfeasible in many cases.

There have been several attempts to address the problems of parametric and nonparametric methods. RSDE [7] reduces the computational cost of full sample

T. Cao et al. (Eds.): PAKDD 2015, Part I, LNAI 9077, pp. 418–428, 2015.
DOI: 10.1007/978-3-319-18038-0_33

KDE through a sparsity induced optimization process. [8] presents an adaptive kernel density estimator based on linear diffusion process and achieves satisfactory performance. In [9], the author proposes a greedy algorithm for learning a gaussian mixture model, which starts with a single component and adds components sequentially until a maximum number k. Due to the global and local search procedure, this algorithm need to keep all training data around and is not suitable for online learning. [10] uses a user defined likelihood based threshold parameter to add new gaussian components for the purposed of incremental learning of gaussian mixture models. However, its learning strategy involves all the components in the current model for every input sample. That makes the model converges slowly and tends to over-smooth the density in the context of online learning. SOMN [11] adopts Self-Organizing Map as its structure and proposes a learning algorithm that minimizes the Kullback-Leibler distance between the estimator and the objective density function, the learning process is limited within a small number of nodes around the input data to accelerate the convergence of nodes. The problem of SOMN is the same as SOM, that is, it is difficult to specify a network topology in advance. oKDE [12] combines the mixture model and the KDE to realize online multivariate density estimation, it maintains and updates a mixture model of the observed data from which the KDE can be calculated, compression and revitalization procedures are executed regularly to balance the accuracy and model complexity. The final estimator is defined as a convolution of the sample distribution by a kernel. This convolution strategy makes oKDE easy to over-smooth the underlying density.

To realize online density estimation that sensitive to local density structure, we propose an incremental and local adaptive gaussian mixture which estimates object density function in an online way by maximizing the sample likelihood locally around each mixture component. Unlike the SOMN, LAIM need not to specify the network structure in advance. Using a similarity threshold based criterion, the method is able to allocate components incrementally to accommodate novel data-points without destroying previously learned components. We also adopt a density based denoising algorithm that make the model more robust to noise.

2 Proposed Method

For density estimation, the LAIM is the same as traditional gaussian mixture model. Every gaussian component of LAIM could be summarized by three parameters: the mean vector μ, covariance matrix Σ and n the effective number of data-points it possesses. We introduces n here for the purpose of extending the maximum likelihood estimation for single gaussian to a local adaptive learning strategy(see Section 2.2).

The final density estimator of LAIM is

$$\hat{f}(x) = \sum_{i=1}^{K} w_i \phi(x|\mu_i, \Sigma_i), \tag{1}$$

where $w_i = \frac{n_i}{\sum_j n_j}$ is the mixing proportion for component i, μ_i and Σ_i are mean and covariance matrix of the ith component. $\phi(x|\mu, \Sigma)$ is the gaussian density function with mean μ and covariance matrix Σ.

The LAIM has three key steps:

1. **Component Allocation**. Construct a neighborhood set for input datapoint and decide whether it is necessary to insert a new gaussian component into the current model.
2. **Local Adaptive Learning**. Update the parameters of the components in the neighborhood set based on maximum likelihood principle.
3. **Denoising**. Eliminate the components induced by the noisy data.

We now give the details of each step.

2.1 Component Allocation

Suppose we have built a mixture model for a series of data $x_1, x_2, \ldots, x_{t-1} \in \mathbb{R}^n$

$$\hat{f}(x|\Theta) = \sum_{i=1}^{K} w_i \phi(x|\theta_i), \tag{2}$$

where $\phi(x|\theta_i)$ is the gaussian density function, w_i is the mixing proportion for component i and $\Theta = (\theta_i, \theta_2, \ldots, \theta_K)^T$ is the parameter matrix for the mixture model. When the new data point x_t is available, traditional EM-based algorithm would make a global adaption for all the components in the current model. This operation is guaranteed to increased the sample likelihood in the long run, however, it could also destroy the previously learned structures and trapped into local optimum if x_t and following inputs are somewhat novel to the current mixture model. It is also possible that x_t is just a noise that should not be learned.

Therefore, to fit the novel data without destroying the old model, we need to decide when to allocate a new component. If we could keep all the historical samples at hand, it is possible to make choice based on the sample likelihood or some model selection criterion. In the context of online learning, we must rely on the current learned model and make choices locally. To measure the novelty of the new coming data point x_t, we first evaluate its distance $D(x_t, i)$ to the components around it weighted by its covariance matrix

$$D(x_t, i) = \sqrt{(x_t - \mu_i)^T \Sigma_i^{-1} (x_t - \mu_i)}, \tag{3}$$

here $i = 1, 2, \ldots, K_t$, K_t is the number of gaussian components at time t. Then we construct the neighbourhood set S_t of x_t

$$S_t = \{i = 1, 2, \ldots, K_t | D(x_t, i) < T_i\}, \tag{4}$$

S_t contains the set of components in the current model that should most responsible for x_t, their parameters will be updated to fit x_t(see Section 2.2). T_i is the

similarity threshold for component i, it controls the tendency of the corresponding component to absorb an input sample nearby, therefore the smaller the T_i, the more local the algorithm will be. We know that $D^2(x_t, i) \sim \chi^2_n$, if the current mixture model is indeed well fitted, we could just let T_i be the value such that

$$\Pr\left[(x_t - \mu_i)^T \Sigma_i^{-1}(x_t - \mu_i) < T_i^2\right] = q \,, \tag{5}$$

where q is the confidence level, in practice we just set it with 0.9. However, we can't make the assumption that the previously learned model is reliable due to the context of online learning, the initialized components need enough samples to converge to a state of well fitted. Therefore, we let T_i be some constant α times T_i, here α is a user defined parameter. To let the new components converge fast, we usually set it with value ranged 1.5 to 2.0 at insertion and decreases to 1 through the training procedure.

If the neighborhood set S_t is empty, we'll regard x_t as a novel data that deserves a new component to fit it. The initialization of the new component is as follows:

$$n_{new} = 1 \,, \mu_{new} = x_t \,, \Sigma_{new} = h^2 I \,. \tag{6}$$

where h is a user defined parameter, I is the identity matrix. h serves as a initial bandwidth for a new component, it should be relatively small compared to the actual standard deviation along each dimension in order to keep the locality sensitivity of LAIM.

2.2 Local Adaptive Learning

Once the neighborhood set S_t is determined for the input data at time t, we limit the learning process within S_t. The local learning strategy does not only accelerate the learning process, but also gives the LAIM the ability to fit new data without destroying the learned components far away from the current input. This property is essential for online learning due to the locality of the information, we could never have the global information about the whole training set, only the current model and current input are available. Doing things locally is a safe strategy so that we can handle the non-stationary input stream.

Starting from the incremental version of maximum likelihood estimation for a single gaussian density function [13]

$$\mu^{(n)} = \mu^{(n-1)} + \frac{1}{n}\left(x_n - \mu^{(n-1)}\right) \,, \tag{7}$$

$$\Sigma^{(n)} = \Sigma^{(n-1)} + \frac{n-1}{n^2}(x_n - \mu^{(n-1)})(x_n - \mu^{(n-1)})^T - \frac{1}{n}\Sigma^{(n-1)} \,. \tag{8}$$

It is easy to see the learning step here is $1/n$, where n is the current number of samples. We want the learning step of each component could be different according to the current model and the learning process within the neighborhood region could be accelerate further since the members of the set are supposed to generate

the current sample with high probability. Therefore, for every component $i \in S_t$ we make the following updates:

$$\begin{aligned} r_i^{(t)} &= \frac{\phi\left(x_t|\theta_i^{(t-1)}\right)}{\sum_j \phi\left(x_t|\theta_j^{(t-1)}\right)} \\ n_i^{(t)} &= n_i^{(t-1)} + r_i^{(t)} \\ \mu_i^{(t)} &= \mu_i^{(t-1)} + r_i^{(t)} \frac{1}{n_i^{(t)}}\left(x_t - \mu_i^{(t-1)}\right) \\ \Sigma_i^{(t)} &= \Sigma_i^{(t-1)} + \frac{n_i^{(t-1)}}{\left(n_i^{(t)}\right)^2}\left(x_t - \mu_i^{(t-1)}\right)\left(x_t - \mu_i^{(t-1)}\right)^T - \frac{1}{n_i^{(t)}}\Sigma_i^{(t-1)} . \end{aligned} \tag{9}$$

The quantity r_i evaluates the responsibility of component i to the current data x_t, it distributed the effective number of observed samples to each component in S_t respectively weighted by their responsibilies. We also use it to adapt the updating stepsize for each component $i \in S_t$. Notice that when multiple components exist in set S_t, this quantity would slow the process of the learning by shrinking the learning stepsize. In the case that S_t has only one element, the above updating rules degenerate to the original maximum likelihood estimation for single gaussian component naturally.

2.3 Denoising

For density estimation in practice, it is common that the training data are contaminated by noise. With the assumption that the noisy data are mostly distributed over the regions where the objective density function f has low probability density and their distribution is sparse enough so that the main structure of f could still be discovered, we adopt a denoising scheme based on the effective numbers of each components, which is used by some prototype based neural networks like [14], [15]. According to the insertion rule described by Section 2.1, those noisy data would lead to node insertion with high probability. However, the resulted components should not possess large effective numbers, i.e., their n is relatively small compared to the non-noise components. Let

$$M = \sum_{i=1}^{K} \frac{n_i}{K} \tag{10}$$

be the mean effective number of the current mixture model, where K is the current number of components. We eliminates those components whose effective number is lower than some constant $\beta \in [0, 1]$ times M after every λ input samples. Here β and λ are user defined parameters, large value of β and small values of λ should be set if the amount of noise is large.

2.4 Complete Algorithm

As a summary, we give the complete algorithm of LAIM here.

Algorithm 1.. Local Adaptive and Incremental Gaussian Mixture

1: Initialize a component for the first sample according to (6).
2: Input new sample $x \in \mathbb{R}^n$.
3: Determine the neighborhood set S for x according to (4).
4: If $S = \emptyset$, initialize a new component for x according to (6), then goto step 2.
5: If $S \neq \emptyset$, update the parameters of components in S according to (9).
6: If the number of input presented so far is a multiple of the parameter λ, make the denoising operation as Section 2.3.
7: Go to step 2 if there is new sample available, otherwise the algorithm terminates and return the trained mixture model.

3 Experiments

3.1 Artificial Data-Sets

The artificial density functions used here are the same to those in [8]. We first adopt the common used bimodal density to verify the effectiveness of our method

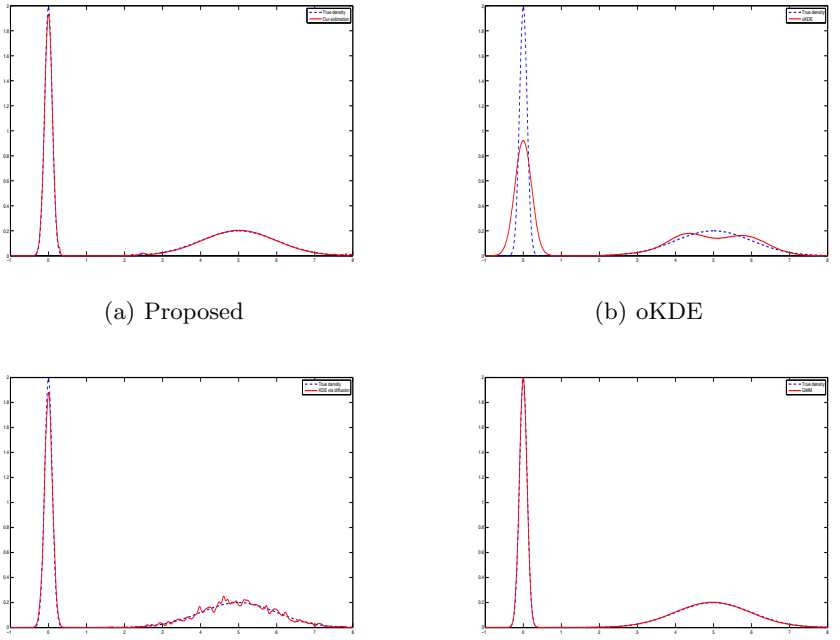
$$\frac{1}{2}N\left(0,(0.1)^2\right)+\frac{1}{2}N\left(5,1\right)\ . \tag{11}$$

We compared the proposed method with oKDE[12], a batch kernel density estimator(kernel density estimation via diffusion, KDE-d for simple)[8] and gaussian mixture models with 2 components (the optimal choice) trained by batch EM algorithm. The parameters are set as follows: oKDE($D_{th} = 0.1$), LAIM($h = 0.5, \alpha = 1.5, \beta = \lambda = \inf$). 3000 samples are drawn from (11) as training set, the resulted estimators are shown in fig. 1. We can see from fig. 1(a) and fig. 1(d) that our method and batch EM reconstruct the underlying density function almost perfectly. That is reasonable since the assumption of mixture of gaussian is perfect for (11), but LAIM doesn't need to specify the number of components due to the incremental nature of the algorithm. From fig. 1(b), we can see that oKDE fits the right hand gaussian pretty well but over-smoothes the left hand gaussian component, that's mainly because its estimation is based on a global convolution operation that lacks of locality sensitivity. fig. 1(c) also shows some under-smoothness on the right hand gaussian. This result shows that LAIM achieves the comparable performance to the batch EM algorithm in this simple bimodal situation.

Then we use the density function "claw", which has more complex structure. The parameters are set as follows: oKDE($D_{th} = 0.1$), LAIM($h = 0.5, \alpha = 1.5, \beta = \lambda = \inf$), the number of components for GMM is identical to LAIM, which is 10 in this case.

$$\frac{1}{2}N\left(0,(0.1)^2\right)+\sum_{k=0}^{4}\frac{1}{10}N\left(k/2-1,(0.1)^2\right). \tag{12}$$

the results shown in fig. 2 suggest that our method could approximate the density function in each local region hence gives a reliable estimation. Batch EM



(a) Proposed (b) oKDE

(c) KDE-d (d) GMM with 2 components

Fig. 1. Blue dashed line is the density function (11), red solid line is the coresponding estimator. The estimator of proposed method and GMM with 2 components is almost perfect. oKDE over-smoothed the left hand gaussian while KDE-d under-smoothes the right hand gaussian.

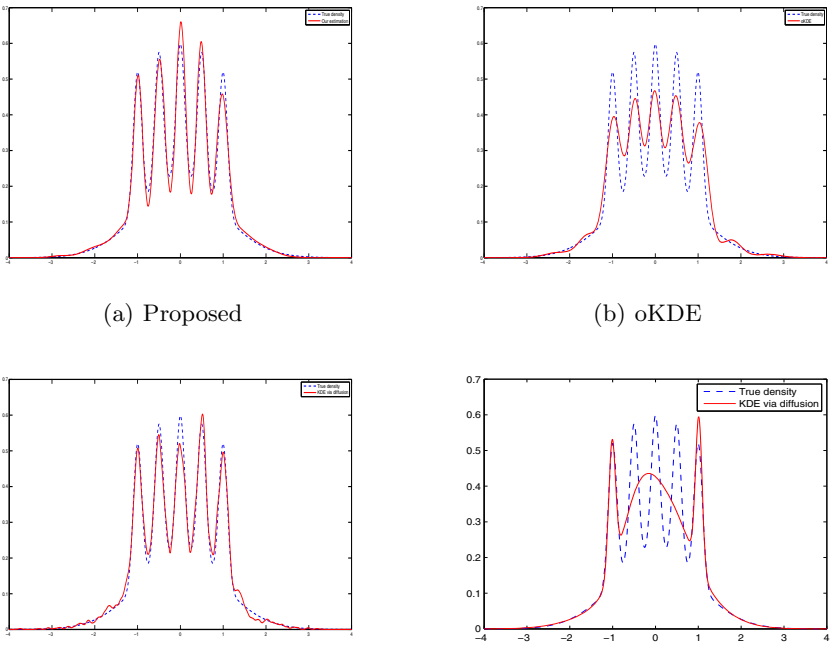
algorithm fails to capture the whole structure of (12). Fig .2(b) is also a result of over-smoothing the sample distribution constructed by oKDE.

To quantify the approximation performance of our method, we did the numerical experiments on five artificial data sets (including (10), (11)). The criterion for the comparison is the numerical approximation to the following ratio,

$$Ratio = \frac{||\hat{f} - f||^2}{||\hat{g} - f||^2} \tag{13}$$

which was adopted by KDE-d [8]. Here $\hat{g}$ is the estimator with which we want to compare, $\hat{f}$ is our estimator and f is the underlying density function. (11) is the integrated squared error of the diffusion estimator to the integrated squared error of the alternative estimator. The results are shown in table .1. When compared to online method oKDE, our approach has lower integrated squared error, which means the corresponding estimator is more accurate. The proposed method outperformes the batch method KDE-d in case 1 and 3, that is

reasonable since the density in those two cases are well-separated gaussians that are more suitable for the mixture models. In case 2 where the density function contains complex local structure, LAIM achieves comparable result to batch algorithm KDE-d.



(a) Proposed (b) oKDE

(c) KDE-d (d) GMM with 10 components

Fig. 2. Blue dashed line is the density function (12), red solid line is the corresponding estimator. The proposed method and KDE via diffusion gave the satisfactory estimation results. oKDE over-smoothes the peaks of the density. Batch EM algorithm didn't capture the whole density structure.

3.2 Real Data-Sets

We compared our method with oKDE, RSDE and KDE with bandwidth selected by cross validation(CV) on the real datasets obtained from the UCI Machine learning Repository[16]. Five data-sets are used here for testing, Iris, Pima, Wine, WineRed and WineWhite. For the density estimation, we estimated the density for each class separately. The data were randomly reordered, 75% of the data in each class were used for training, and the rest for testing. We conduct the same experiment twelve times and recorded the mean and standard deviation

Table 1. Ratio of approximated integrated squared error

Case	Target density	Ratio(oKDE)	Ratio(KDE-d)
1	$\frac{1}{2}N\left(0,(1/10)^2\right)+\frac{1}{2}N\left(5,1\right)$	0.04	0.38
2	$\frac{1}{2}N\left(0,(1/10)^2\right)+\sum_{k=0}^{4}\frac{1}{10}N\left(k/2-1,(1/10)^2\right)$	0.67	1.05
3	$\frac{1}{2}N\left(-2,(1/4)^2\right)+\frac{1}{2}N\left(2,(1/4)^2\right)$	0.26	0.76
4	$\frac{2}{3}N\left(0,1\right)+\frac{1}{3}N\left(0,(1/10)^2\right)$	0.90	1.42
5	$\frac{3}{4}N\left(0,1\right)+\frac{1}{4}N\left(3/2,(1/3)^2\right)$	0.91	1.78

Table 2. Negative log likelihood on real datasets

Dataset	Proposed	oKDE	RSDE(batch)	CV(batch)
Iris	**0.3(±0.2)**	2.1(±0.5)	2.5(±0.9)	2.7(±0.9)
Pima	**28.8(±0.4)**	32.3(±0.3)	38.4(±11.3)	29.5(±0.5)
Wine	23.5(±3.1)	26.4(±3.4)	12.3(±1.9)	**11.6(±1.5)**
Winered	13.9(±2.7)	18.4(±3.5)	-12.3(±4.9)	**-27.2(±1.0)**
Winewhite	**8.7(±0.3)**	11.4(±0.3)	91.3(±44.6)	11.6(±0.4)

Table 3. Average classification rate

Dataset	Proposed	oKDE	RSDE(batch)	CV(batch)	SVM(batch)
Iris	**97(±3)**	**97(±3)**	96(±4)	96(±3)	96(±2)
Pima	73(±3)	72(±2)	65(±3)	72(±2)	**78(±3)**
Wine	89(±3)	94(±3)	91(±5)	92(±6)	**96(±3)**
Winered	**65(±2)**	64(±2)	44(±4)	64(±1)	63(±3)
Winewhite	**63(±2)**	55(±1)	25(±6)	62(±1)	60(±2)

as the result. The oKDE were initialized by the first 10 samples and parameter D_{th} was set to 0.1. Two parameters of LAIM are set as follows: $\alpha = 2.2$, $\beta = 0.05$, $\lambda = 0.1N$, where N is the total number of training samples. h is set according to the the scale of the data because it will affect the the complexity of the model: Iris(0.01), Pima(15), Wine(15), Winered(15), Winewhite(5). To measure the quality of estimation, we have computed the average negative likelihood(NLL) per test point, lower NLL generally suggests more accurate estimation.

The results of the experiments after observing all the data-points are summarized in Table 2. Compared to the online method oKDE, the proposed method achieves better results on all the data-sets. LAIM also ourperforms the batch methods on Iris, Pima and WineWhite.

We have also tested the classification ability of the proposed method on the previous five data-sets. Although density estimator is not generally the most accurate classifier, the classification results based on simple Bayesian criterion

$$\hat{y} = \arg\max_{k} p(x|c_k)p(c_k) \tag{14}$$

still reflects the quality of density estimation. We have chosen a multiclass SVM with RBF kernel[17] as the baseline classifer and compares our method with oKDE, RSDE, and KDE with cross validation. The results of classification are summarized in Table 3. From the table, we can see that the proposed method outperformes the online counterparts in most data-sets except for Wine and produces comparable results to the batch methods. Noticed that in the context of online learning, we don't store any historical data but the current input and the learned model, therefore, the time complexity and space complexity of LAIM are much smaller than the batch methods.

4 Conclusion

In this paper, the incremental and local adaptive gaussian mixture for online density estimation(LAIM) is proposed. With the similarity threshold, the method could allocate components incrementally while training without specifying the number of gaussian components in advance. We proposes a local learning algorithm for updating the parameters of mixture model based on maximum likelihood principle, this locality sensitivity enables the our model to discover the local density structure of the data samples in the context of online learning. A denoising scheme is used to eliminate the components initialized by noise. Experiments show that it outperforms the compared online density estimators and produces comparable results to the compared batch methods while keeping a lower model complexity.

Acknowledgments. The authors are very grateful to Youlu Xing for many useful discussions and programming assistance. We also acknowledge Tao Zhu and Haoran Xu for helpful comments. This work is supported in part by the National Science Foundation of China under Grant Nos. (61375064, 61373001) and Jiangsu NSF grant (BK20131279).

References

1. McLachlan, G., Peel, D.: Finite mixture models. Wiley-Interscience (2000)
2. Parzen, E.: On estimation of a probability density function and mode. Annals of Math. Statistics **33**, 1065–1076 (1962)
3. Wand, M.P., Jones, M.C.: Kernel Smoothing. Chapman Hall/CRC (1995)
4. Hall, P., Sheater, S.J., Jones, M.C., Marron, J.S.: On optimal data-based bandwidth selection in kernel density estimation. Biometrika **78**(2), 263–269 (1991)
5. Silverman, B.W.: Density Estimation. Chapman and Hall, London (1986)

6. Hall, P.: Large sample optimality of least squares cross-validation in density estimation. The Annals of Statistics, 1156–1174 (1983)
7. Girolami, M., He, C.: Probability density estimation from optimally condensed data samples. IEEE Trans. Pattern Anal. Mach. Intell. **25**(10), 1253–1264 (2003)
8. Botev, Z.I., Grotowski, J.F., Kroese, D.P.: Kernel density estimation via diffusion. The Annals of Statistics **38**(5), 2916–2957 (2010)
9. Vlassis, N., Likas, A.: A greedy EM algorithm for Gaussian mixture learning. Neural processing letters **15**(1), 77–87 (2002)
10. Engel, P.M., Heinen, M.R.: Incremental learning of multivariate gaussian mixture models. In: da Rocha Costa, A.C., Vicari, R.M., Tonidandel, F. (eds.) SBIA 2010. LNCS, vol. 6404, pp. 82–91. Springer, Heidelberg (2010)
11. Yin, H., Allinson, N.M.: Self-organizing mixture networks for probability density estimation. IEEE Transactions on Neural Networks **12**(2), 405–411 (2001)
12. Kristan, M., Leonardis, A., Skoaj, D.: Multivariate online kernel density estimation with Gaussian kernels. Pattern Recognition **44**(10), 2630–2642 (2011)
13. Bishop, C.M.: Pattern recognition and machine learning, vol. 1. Springer, New York (2006)
14. Ouyang, Q., Shen, F., Zhao, J.: A local distribution net for data clustering. In: Anthony, P., Ishizuka, M., Lukose, D. (eds.) PRICAI 2012. LNCS, vol. 7458, pp. 411–422. Springer, Heidelberg (2012)
15. Furao, S., Ogura, T., Hasegawa, O.: An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Networks **20**(8), 893–903 (2007)
16. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
17. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) **2**(3), 27 (2011)