Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

Fast Parzen Window Density Estimator

Xiaoxia Wang and Peter Tiňo and Mark A. Fardal and Somak Raychaudhury and Arif Babul

***Abstract*— Parzen Windows (PW) is a popular non-parametric density estimation technique. In general the smoothing kernel is placed on all available data points, which makes the algorithm computationally expensive when large datasets are considered. Several approaches have been proposed in the past to reduce the computational cost of PW either by subsampling the dataset, or by imposing a sparsity in the density model. Typically the latter requires a rather involved and complex learning process. In this paper, we propose a new simple and efficient kernel-based method for non-parametric probability density function (pdf) estimation on large datasets. We cover the entire data space by a set of fixed radii hyper-balls with densities represented by full covariance Gaussians. The accuracy and efficiency of the new estimator is verified on both synthetic dataset and large datasets of astronomical simulations of the galaxy disruption process. Experiments demonstrate that the estimation accuracy of the new estimator is comparable to that of the previous approaches but with a significant speed-up. We also show that the pdf learnt by the new estimator could used to automatically find the most matching set in large scale astronomical simulations.**

I. Introduction

In many statistical learning problems, it is desirable to obtain an estimate of the underlying probability density function (pdf), given a set of observations. Finite mixture model is a general and powerful approach to the problem of probability density estimation. It uses relatively small number of mixture components to estimate arbitrary density functions. A mixture model provides a condensed representation of data samples in terms of sufficient statistics of each of the mixture components and their respective mixing weights. The parameters of these components (e.g. Gaussians) in the finite mixture model are usually learnt through an iterative optimization algorithm such as the likelihood maximizing Expectation-Maximization (EM) [15] (procedures which are known to be sensitive to initialization and prone to get trapped in local minima). Furthermore, when applied to large-scale datasets, the time spent on the optimization can be prohibitive [3].

The well known non-parametric Parzen Windows (PW) estimator can be viewed as a special case of the finite mixture model with mixture components located at all points of the data sample. Since all the components (Gaussian smoothing kernels) in this model have the same width, the parameter estimation is greatly simplified. Whereas the potentially large mixture size ensures reliable density estimates, the price to be paid is a heavy computational cost incured in the test phase.

To reduce the computational cost, several algorithms have been devised to either reduce the sample size, or to reduce the amount of components (kernels) in the original complex Parzen Windows model. The latter approaches, introduced recently, have proved successful in several applications [2], [4], [6]. The idea is to simplify a complex model (Parzen Windows in density estimation) while minimizing the distance between the new model and the target function. However, this process usually has complexity of $\mathcal{O}(N^2)$ or larger, where N is the size of the dataset. Compared with considering fewer points in the estimation, optimizing the simplified density model by using all of the available observations could avoid sacrificing useful information, but it also loses the simplicity of the non-parametric density model. On the other hand, the limitation of Parzen Windows, could also be noticed when the data points distribute along, or partially along, low dimensional structures [1]. In such cases, spherical smoothing kernels are not optimal.

In this paper, we propose a new algorithm to reduce the computational cost of PW but also to keep the simplicity of the nonparametric model by avoiding complex model construction procedures. The idea is to cover the entire data space by a set of hyper-balls of fixed radii. For each hyper-ball, the local density is captured by a full covariance Gaussian kernel. With carefully chosen radii, the density of the partitioned disk cells can be described by a single Gaussian kernel and the local data structure could be well preserved. Our model is formed by a mixture of such locally fitted Gaussians with appropriately set mixing weights.

The rest of paper is organized as follows. In Section 2 we review the algorithms and models proposed to deal with large-scale datasets for kernel density estimators. Section 3 describes our efficient kernel-based density estimator: fast Parzen Windows (FPW). In section 4, we report experimental results on synthetic data set, as well as on data sets obtained from galaxy disruption simulations. Finally, Section 5 brings concluding remarks.

II. Sparse Kernel Estimators

Consider a probability density estimator $\hat{f}(\mathbf{x}, \theta)$ (with parameters θ) fitted on a finite data sample $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathcal{R}^d$, $i = 1, 2, ..., N$. When no a priori information is available to guide the choice of the precise form of the density, nonparametric probability density estimation termed Parzen Windows (PW) [11] is particularly attractive:

$$\hat{f}(\mathbf{x}; h) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{K}(\mathbf{x}; \mathbf{x}_n, h), \qquad (1)$$

Email:{X.Wang.1, P.Tino}@cs.bham.ac.uk. School of Computer Science, University of Birmingham, UK. fardal@fcrao1.astro.umass.edu. Dept. of Astronomy, University of Massachusetts, USA. somak@star.sr.bham.ac.uk. School of Physics and Astronomy, University of Birmingham, UK. babul@uvic.ca. Department of Physics and Astronomy, University of Victoria, Canada.

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

where N is the total number of data points, $\mathcal{K}(.;\mathbf{x}_n,h)$ is a kernel function centered on $\mathbf{x}_n$ and having a scale parameter (width) h. For each component $\mathcal{K}(\mathbf{x};\mathbf{x}_n,h)$, the mixing coefficient $1/N$ can be thought to represent the probability $p(\mathbf{x}_n)$ of picking $\mathbf{x}_n$ (and hence the n-th mixture component of the PW estimator) from $\mathcal{D}$. Rewriting Eq.(1), we obtain

$$\hat{f}(\mathbf{x};h)=\sum_{n=1}^{N}p(\mathbf{x}_n)\mathcal{K}(\mathbf{x};\mathbf{x}_n,h). \quad (2)$$

The choice of $\mathcal{K}$ and the scale parameter $h>0$ determines the performance of $\hat{f}(.)$. However, as mentioned earlier, the main disadvantage of such an approach is the high computational cost when large data samples are involved.

The computational complexity reduction strategies to kernel density estimation are developed in two ways. One group of such approaches represents the whole dataset by a condensed dataset maintaining the statistical properties of the original one. Another recently introduced group of approaches approximates complex models containing a large number of components with simpler ones having less components. Parameters of the simpler models are estimated in a complex optimization procedure. In the following, we shall briefly review some of those approaches.

A. Data Reduction Methods

Data reduction methods could be used simply as a preprocessing step to machine learning algorithms (such as classification or density estimation) to deal with the large sample size problem. The simplest way is to randomly draw the desired number of samples from the original data set. However, we could expect that the performance of machine learning algorithms with this subset is random as well. Data condensation of more generic nature is performed by classical vector quantization methods using a set of codebook vectors which minimize the quantization error [12]. Recently, [10] introduced a density-based multiscale data condensation method that uses discs of adaptive radii for both density estimation and sample pruning.

More sophisticated approaches explicitly sample more data points from densely populated areas of the data space, while fewer data points are drawn from the sparse areas. Alternatively, the original data set is represented by a set of reference vectors and their associated weights. For example, [7] prebinned data samples (from low dimensional data space), and the kernel density estimator employed the bin centers as the "sample" points which are each weighted by the normalized bin-counts. A multivariate form of the binned kernel density estimator was analyzed in [8].

B. Model Simplifying Methods

Rather than shrinking the size of the dataset, computational savings could also be obtained by reducing the number of components (model complexity). The idea is to employ an optimization process to minimize a distance between the simplified model and the complex one. [2] presented a probability density estimator which employs a small percentage of the data sample. Kernel weighting coefficients associated with each data sample were estimated in a sparsity enforcing optimization routine. Since many of the weights are set to values close to 0 in this optimization process, the algorithm automatically removes most of the redundant data points. In [4], mixture components of a large mixture model are grouped into clusters. In the simplified model, each cluster is represented by a single component. The parameters are found by minimizing the upper bound on the approximation error (in the L_2 sense) between the original and the simplified models. An earlier work along those lines has been done by Goldberger and Roweis [5]. There the distance between the original and the simplified model is measured by a local Kullback-Leibler (KL) divergence. [6] uses forward constrained regression to construct a sparse kernel density estimator. Components of the kernel density estimator are selected from the existing components in Parzen Windows by minimizing the mean square error.

Although these algorithms claim they provide superior density estimators for similar levels of data reduction [2], the relative simplicity of the nonparametric density estimator is lost in their nonlinear optimization processes. Also, the target function of the optimization processes is set to be the PW estimator. Past work [1] demonstrated that PW may not be optimal for datasets living in a high-dimensional space but aligned along a low dimensional structure. Compared with the original PW, manifold Parzen windows, replacing the spherical kernels by kernels fitted to local distributions, achieved superior performance [1].

III. Fast Parzen Windows

The proposed efficient kernel density estimator involves segmenting the whole data set into hyper-balls with a fix radii, associating each cell with a kernel and a mixture weight, and updating the kernels to fit the local distribution.

A. Partition the Dataset

We partition (in a hard or soft manner) the data space with hyper-discs of fixed radii r and let the data distribution be approximated by local densities fitted within the hyper-discs. We use the following algorithm to position the hyperdiscs:

1) Set $\mathcal{S}=\emptyset$, $\mathcal{T}=\emptyset$, $j=0$.
2) While $\mathcal{D}\setminus\mathcal{T}\neq\emptyset$, repeat:
   - $j\leftarrow j+1$
   - Randomly select a data point $\mathbf{x}_q\in\mathcal{D}\setminus\mathcal{T}$ and add it to the partition set $\mathcal{S}$, as well as to the set $\mathcal{T}$ ($\mathcal{T}\leftarrow\mathcal{T}\cup\{\mathbf{x}\}$).
   - For every data point $\mathbf{x}\in\mathcal{D}\setminus\mathcal{T}$
     - Compute the (Euclidean) distance $D(\mathbf{x},\mathbf{x}_q)$
     - if $D(\mathbf{x},\mathbf{x}_q)\leq r$, where r is a predefined threshold, add $\mathbf{x}$ to the partition $\mathcal{S}_j$ centered at $\mathbf{s}_j=\mathbf{x}_q$ and add $\mathbf{x}$ to $\mathcal{T}$.

We thus obtain a set of partition centers $\mathcal{S}=\{\mathbf{s}_1,\ldots,\mathbf{s}_M\}$ ($M\leq N$) representing the partitions $\{\mathcal{S}_1,\ldots,\mathcal{S}_M\}$ of the data set $\mathcal{D}$. Note that, unlike in vector quantization, the

partition elements $\mathcal{S}_i$ cover the same volume but are populated according to the local density of data points. Compared with vector quantization based downsampling algorithms, relatively more data samples are chosen from sparse regions. Therefore, the local distribution in sparse regions described by these representative points can be more accurate.

The sets $\mathcal{S}_1, \ldots, \mathcal{S}_M$, cover the data sample and we estimate the measures $\kappa(\mathcal{S}_i)$, $i = 1, 2, ..., M$, assigned to them by the original (unknown) generating distribution in two ways:

- **Hard version:**
$$\kappa(\mathcal{S}_i) \approx P(\mathcal{S}_i) = \frac{|\mathcal{S}_i|}{N},$$
where $|\mathcal{S}_i|$ denotes the size of $\mathcal{S}_i$.
- **Soft version:** Points can be assigned to more than one partition with different weights.
$$\kappa(\mathcal{S}_i) \approx Q(\mathcal{S}_i) = \frac{\sum_{n=1}^{N} \mathcal{K}(\mathbf{x}_n; \mathbf{s}_i, R)}{\sum_{j=1}^{M}\sum_{n=1}^{N} \mathcal{K}(\mathbf{x}_n; \mathbf{s}_j, R)}$$
where $R > 0$ is the kernel scale parameter. In our experiments, R was set to be equal to the hyperdisc radius r.

B. Fast Parzen Windows

Some kernel-based estimators use diagonal Gaussians as mixture components. If the true density that we would like to model is indeed 'close to' a lower dimensional manifold embedded in the higher dimensional data space, spherical Gaussians will spread their density mass equally along all input space directions, thus giving too much probability mass to irrelevant regions of the data space. We employ local data-adaptive kernels (Gaussians) [1]. Our density estimator reads:

$$\hat{f}_s(\mathbf{x}; h) = \sum_{j=1}^{M} P_j \; \mathcal{N}(\mathbf{x}; \mathbf{m}_j, C_j) \tag{3}$$

where either $P_j = P(\mathcal{S}_j)$, or $P_j = Q(\mathcal{S}_j)$, and $\mathcal{N}(\mathbf{x}; \mathbf{s}_j, C_j)$ is a multivariate Gaussian density with mean vector $\mathbf{m}_j$ and covariance matrix C_j:

$$\mathcal{N}(\mathbf{x}; \mathbf{m}_j, C_j) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m}_j)^T C_j^{-1}(\mathbf{x}-\mathbf{m}_j)}}{\sqrt{(2\pi)^d |C_j|}}. \tag{4}$$

Here $|C_j|$ denotes the determinant of C_j.

In the hard partition version (FPW-H) we determine the mean and the covariance matrix of each Gaussian kernel using

$$\begin{aligned} \mathbf{m}_j &= \frac{1}{|\mathcal{S}_j|} \sum_{\mathbf{x}_i \in \mathcal{S}_j} \mathbf{x}_i \\ C_j &= \frac{1}{|\mathcal{S}_j|} \sum_{\mathbf{x}_i \in \mathcal{S}_j} (\mathbf{x}_i - \mathbf{m}_j)(\mathbf{x}_i - \mathbf{m}_j)^T. \end{aligned} \tag{5}$$

The soft partition of $\mathcal{S}$ associates an influence weight $p(\mathbf{x}_i|\mathbf{s}_j)$ in the partition $\mathbf{s}_j$ to any point $\mathbf{x}_i$ by the neighborhood kernel:

$$p(\mathbf{x}_i|\mathbf{s}_j) = \frac{\mathcal{K}(\mathbf{x}_i; \mathbf{s}_j, R)}{\sum_{n=1}^{N} \mathcal{K}(\mathbf{x}_n; \mathbf{s}_j, R)}. \tag{6}$$

Then the weighted means and covariance matrices of the soft FPW version (FPW-S) are computed as follows:

$$\mathbf{m}_j = \sum_{i=1}^{N} p(\mathbf{x}_i|\mathbf{s}_j)\mathbf{x}_i \tag{7}$$

$$C_j = \sum_{i=1}^{N} p(\mathbf{x}_i|\mathbf{s}_j)(\mathbf{x}_i - \mathbf{m}_j)(\mathbf{x}_i - \mathbf{m}_j)^T. \tag{8}$$

Even though the sums in eqs.(6)-(8) run through all data points, in practice we need to consider only a small fraction of points with kernel values or responsibilities $p(\mathbf{x}_i|\mathbf{s}_j)$ larger than some small predefined threshold value (in our experiments 0.00001).

Note that calculating the **inverse** of the covariance matrices may be complicated as C_j's may be ill-conditioned. A common way to deal with this problem is to add a small isotropic (spherical) Gaussian noise of variance γ^2 in all directions, which is done by simply adding γ^2 to the diagonal of the covariance matrix: $C_j = C_j + \gamma^2 I$, where I is an identity matrix. In our experiments $\gamma^2 = 0.000001$.

Since the amount of kernels has been reduced to the number M of the partitions, fitting each kernel to its local distribution could improve the performance of the density estimator without increasing the time and memory costs significantly.

C. Complexity analysis

The most time consuming operation is partitioning the dataset. The algorithm has complexity of $\mathcal{O}(NM)$, where N is the sample size, M is the number of partitions. We could estimate the partition number M by dividing the data volume by the hyper-ball volume r^d, where d is the dimension of the data space. By adopting k-d tree [16] in the partitioning operation, the complexity can be reduced to $\mathcal{O}(N \log(M))$. The following experiments demonstrate a significant advantage of our approach over the alternatives with respect to the running time.

IV. Experiments

In this section, we verify our approach on both synthetic dataset and real large scale astronomical simulation datasets. The experiments demonstrate that, compared with other recently proposed methods for sparse kernel density estimation as well as the classical Parzen Windows and Gaussian Mixture Model, our Fast Parzen Windows can lead to similar or better levels of performance in terms of the accuracy and sparseness of representation, but at much less computational cost. Furthermore, the advantages of our simple, yet effective method for density estimation are highlighted in experiments on large datasets of galaxy disruption simulations.

A. Data aligned along a lower dimensional manifold

We use a sample (300 points) of 2D data aligned along a 1D structure. We obtain a 300-point training set and a

10000-point test set from the following distribution of two dimensional points used in [1]:

$$x_1 = 0.04t\sin(t) + \epsilon_{x_1},$$
$$x_2 = 0.04t\cos(t) + \epsilon_{x_2}$$

where $t \sim \mathcal{U}(3, 15)$, $\epsilon_{x_1} \sim \mathcal{N}(0, 0.01)$, $\epsilon_{x_2} \sim \mathcal{N}(0, 0.01)$, $\mathcal{U}(a, b)$ is the uniform distribution over the interval (a, b) and $\mathcal{N}(0, \delta)$ is the zero-mean Gaussian distribution with standard deviation δ.

We compare the performance of Parzen Windows (PW), Simplified Mixture Model (SMM) [4], Reduced Set Density Estimator (RSDE) [2], Manifold Parzen Windows [1], Gaussian Mixture Model (GMM) (We use GMM^k to denote the model initialized with K-means as opposed to GMM initialized randomly from the data) and our approaches. We keep the same parameter setting for Manifold Parzen Windows as in [1]. Figure 1 shows the training set, as well as the densities estimated by these algorithms. Note the excessive 'bumpiness' and holes in figures 1 (a)-(c) caused by the sparseness of the training set. As expected, the Manifold PW density in figure 1(d) is better aligned with the underlying manifold. Our approach was not only able to learn the underlying manifold-aligned density well, but also to represented it in a sparse manner (see figures 2(a),(b) and table II).

Quantitative comparison of the models is based on the average (across 30 randomized initializations) log likelihood (ALL) on the test dataset and the model sparseness (number of mixture components M, shown in table II). The performance measures along with the parameters used (estimated via cross-validation) are listed in table I. For each algorithm A with randomized initialization, when FPW-H and FPW-S show significantly better performance than A (as detected by t-test at significance level 0.01), we put h and s in the *Signif* column.

TABLE I
COMPARISON RESULTS ON THE 2D ARTIFICIAL DATA

Alg	Parameters	ALL (ave+std)	Signif
PW	δ=0.0173	1.3417+0	–
SMM	m=20	0.8573+0.3356	h,s
SMM	m=50	1.2126+0.0693	h,s
SMM	m=100	1.3075+0.0187	h,s
RSED	δ=0.0190	1.0515+0	–
GMM^k	m=50	1.4575+ 0.0243	s
GMM	m=70	1.4609+ 0.0271	s
Man PW	d=1, k=11, δ=0.09	1.4660 +0	–
FPW-H	r=0.141, γ=0.0001	**1.3737+0.0246**	s
FPW-S	r=0.051, γ=0.0001	**1.5045+0.0090**	

The results reveal that our approach can preserve the low dimensional manifold structure while keeping the model complexity low.

To investigate the time consumption of these approaches, we perform our experiments on desk top computer having Pentium(R)4, 3.06GHz CPU and 1GB RAM, and all the algorithms are implemented in Matlab. Table II shows the running time consumed on this 300-sample dataset and the model complexity (by components number, denoted by M). We report the time consumed in the form $t_1 + t_2$, where t_1 is the time used for building the model and t_2 is the time of deploying the model.

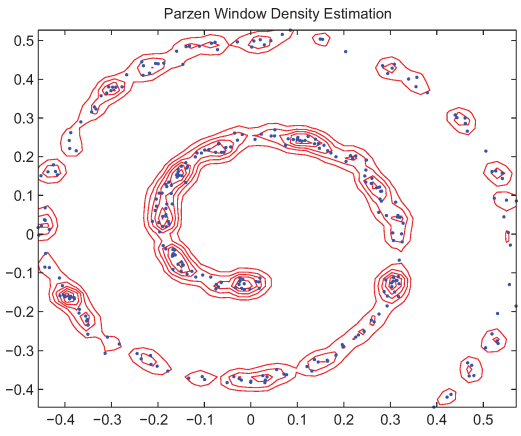
TABLE II
TIME CONSUMPTION AND THE MODEL COMPLEXITY

Methods	M	CPU Time (sec)
PW	300	0+1.229
SMM	20	16.751 + 0.117
SMM	50	20.319 + 0.215
SMM	100	33.319 + 0.686
RSDE	112	5.299 + 0.546
GMM^k	50	9.162+0.346
GMM	70	12.078+0.463
Manifold PW	300	0.155+1.312
FPW-H	21	**0.156+0.102**
FPW-S	57	**0.580+0.231**

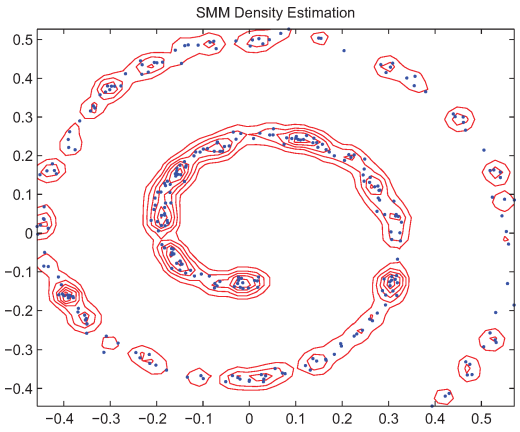
B. Experiments on galaxy disruption simulations

Recent exciting discoveries of low-dimensional structures such as shells and streams of stars in the vicinity of large galaxies led the astronomers to investigate the possibility that such structures are in fact remnants of disrupted smaller satellite galaxies [14],[13]. A simulation model was designed to simulate the process of satellite galaxy disruption in the vicinity of a large galaxy [14]. It is a particle model, with particles representing stars. Each particle has six dimensions, three describing the spatial position, the other three describing the velocity along the spatial coordinates. The particle evolution is governed by the laws of physics. One particular stream of research is concerned with simulating the disruption of satellite galaxy (M32) with the large galaxy (M31). To track the evolution process starting from a particular initial condition, the astronomers collect the simulation datasets at successive evolution stages. Hence, the disruption process is captured in a series of simulation datasets.

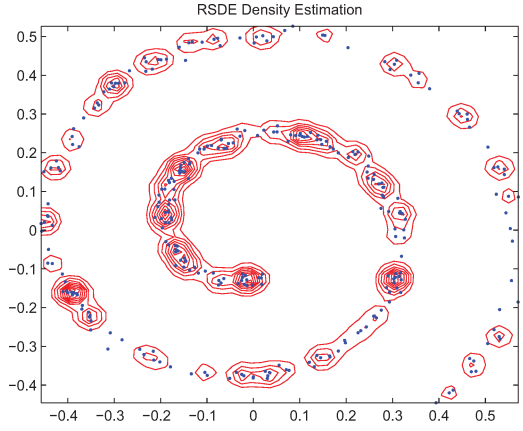
The ultimate goal is to identify the most plausible set of initial conditions leading to the distribution of stars currently observed by the astronomers. Of course, particle simulations cannot be compared with the real observations on a point by point bases, but the observed density of stars can be compared with that of the simulated particles. The first step is to build good density models on the simulated data. The densities will be then projected into the observation space (typically 2 spatial coordinates + the line of sight velocity) and the likelihood given the real observations will be calcu-



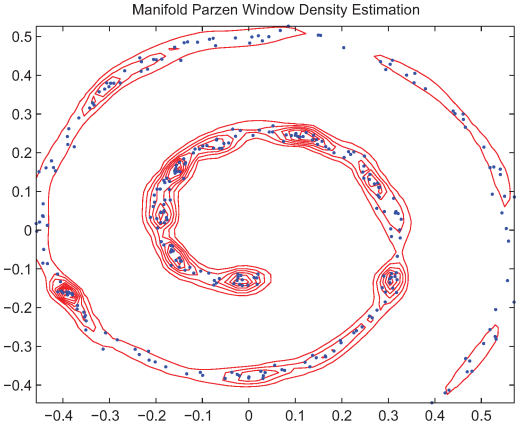
(a) Density estimated by PW



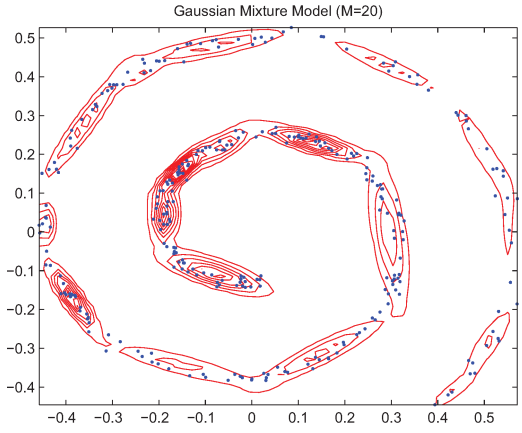
(b) Density estimated by SMM



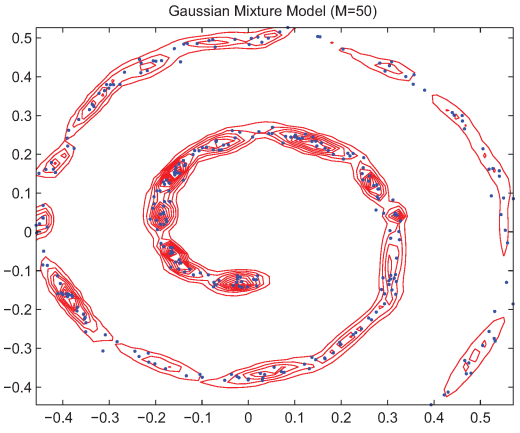
(c) Density estimated by RSDE



(d) Density estimated by Manifold PW

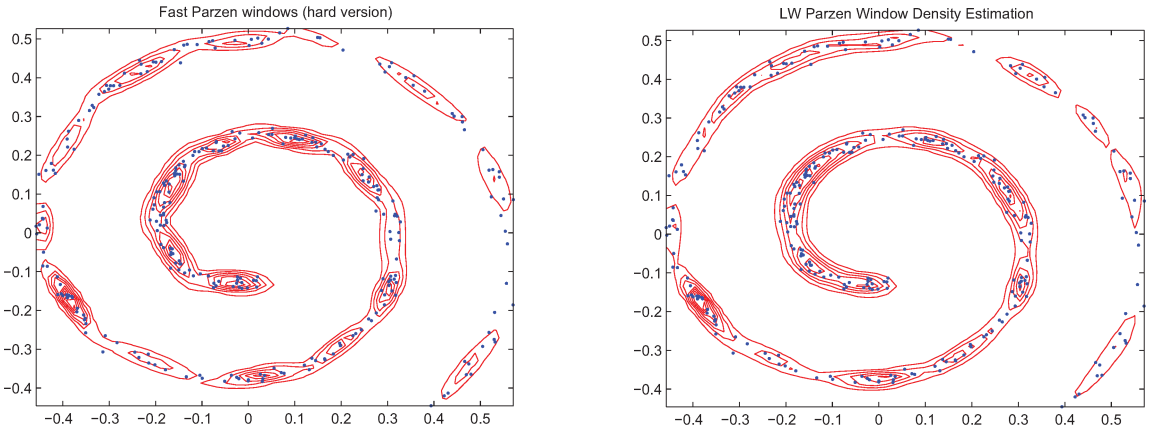


(e) Density estimated by GMM (M=20)



(f) Density estimated by GMM (M=50)

Fig. 1. Densities estimated on 2 dimensional dataset aligned along a 1 dimensional structure



(a) Density estimated by FPW-H

(b) Density estimated by FPW-S

Fig. 2. Densities estimated on 2 dimensional dataset aligned along a 1 dimensional structure

lated. This is a work in progress, here we are only concerned with models build in the full 6-dimensional phase space. The number of particles representing the satellite galaxy being disrupted is typically large (in our case $\approx$ 30,000) and we found that using RSDE was computationally infeasible. Encouraged by the results in the previous experiments, we employ the PW, GMM and our methodology as the principal density estimators.

We have 22 simulation sets representing 22 stages of galaxy disruption in a single simulation run. Each set has $32,768$ data points.

In the first set of experiments we used 10-fold cross-validation in each simulation data set to measure the quality of the estimated density models. The average log-likelihoods (ALL) on each individual set estimated by PW, our approach (FPW) and SMM are plotted in figure 3 (using hyper parameters estimated by 10-fold cross validation within the training folds). The X-axes indicates the simulation set index (numbered from 0 to 21), the Y-axes shows the ALLs estimated by the models. Our FPW method shows a superior

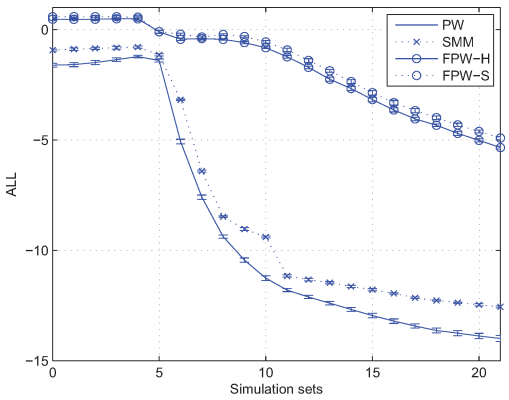


Fig. 3. Average log-likelihood on simulation sets

performance relative to both PW and SMM estimators.

RSDE involves calculations with matrices of size $N \times N$. As mentioned earlier, running the algorithm on $N = 32,768$ points proved to be infeasible. To be able to investigate the performance of RSDE and compare it to our approach, for each of the 22 stages, we estimated the density from a downsampled simulation set (10% of the original set) and tested on the hold-out data (90% of the original set). Analogously, we demonstrate the performance of GMMs on the downsampled simulation sets due to computational demanding E-M parameter fitting. The process was repeated 10 times. The results for RSDE, GMM and FPW are shown in figure 4. In figure 5, we report the time consumed on

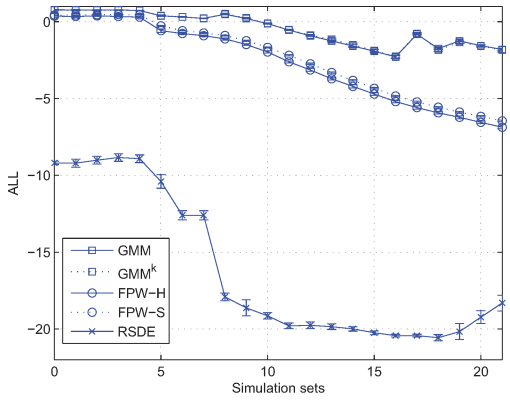


Fig. 4. Average log-likelihood on simulation sets of models trained on reduced size datasets

building the density model by using these approaches on the 22 stages of galaxy disruption. The X-axes indicates the simulation sets, and the Y-axis to show the time used (seconds). Note algorithm RSDE, GMM and GMM^k are estimated on the downsampled simulation set (10% of the original set). A clear advantage of our approaches on running times are illustrated.

In the second set of experiments, we investigate how

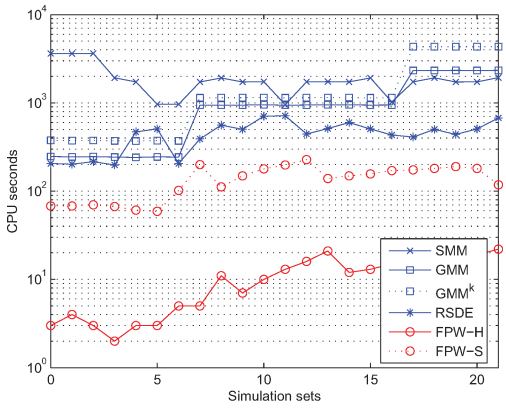


Fig. 5. Time consumed on running these experiments

reliably can a stage in the galaxy disruption process be detected based on "observations" not used in the model building process. We run a rolling window of size 3 over the series of 22 simulation data sets. We thus obtain a set of 20 simulation set triplets (f, g, h), starting with simulation sets (f, g, h) at stages (0,1,2) and ending with the triplet (f, g, h) containing simulation set for stages (20,21,22). For each triplet of consecutive simulation sets (f, g, h), we estimate 3 models, one on 90% of data from f, one on 90% of data from g, and one on 90% of data from h. (We use 10% here for GMM and GMM^k) All three models are then tested on the 10% hold-out set from g. In this way we can determine how well can the "true" source density g be distinguished from the densities at the nearby stages f and h. This process is repeated 10 times. The densities corresponding to the nearby stages of galaxy disruption can be quite similar and obtaining an accurate density model is essential for further investigations by the astronomers. As an illustration, we present in figure 6 two sets of 3-dimensional projections of the simulation data for the triplet $(f, g, h) = (20, 21, 22)$. The first projection is onto the spatial coordinates, the second is onto the leading 3 eigenvectors found by the Principal Component Analysis of the merged f, g and h sets.

The models constructed on the set g have the highest hold-out ALL in each of the 20 triplets $((f, g, h)$. To qualify the confidence margin with which the true source is detected, we calculate likelihood ratios (LR) :

$$LR_{fg} = \frac{e^{f_{all}}}{e^{g_{all}}} \quad (9)$$

$$LR_{hg} = \frac{e^{h_{all}}}{e^{g_{all}}} \quad (10)$$

where f_{all}, g_{all} and h_{all} denote the ALL on the testing data from g obtained by models built on the sets f, g and h, respectively. Figures 7 and 8 show the likelihood ratios LR_{fg} and LR_{hg}, respectively, for the PW and FPW methods. The likelihood rations of the two FPW versions are almost the same and in both cases the variations in LR due to different experimental runs are negligible. The FPW methods outperform the classical PW estimation and show performance levels very similar to those of GMM, but with much less computational effort (as illustrated in figure 5). The number of components in FPW-H and FPW-S varied from 150 to 850 and 700 to 4000, respectively. Note that the number of components in PW estimates was $N \approx 30,000$.

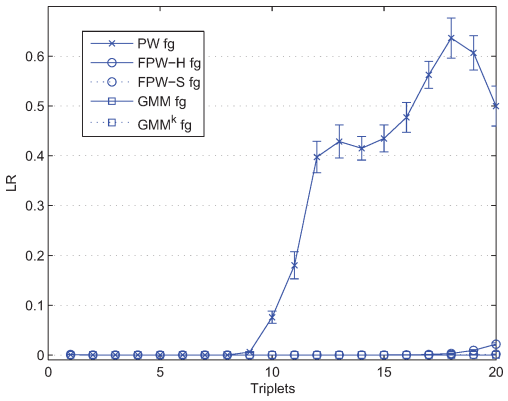


Fig. 7. Likelihood Ratios for finding the best matching source.

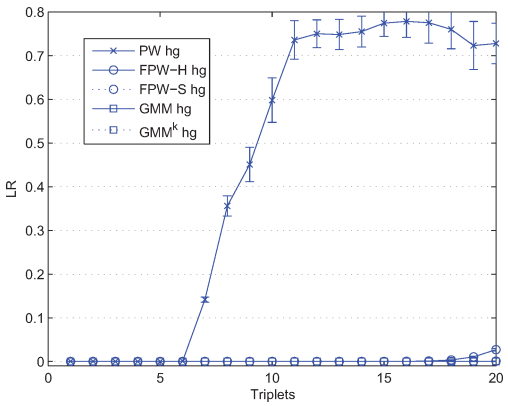


Fig. 8. Likelihood Ratios for finding the best matching source.

V. Conclusion

We presented an efficient probability density estimator for large-scale datasets, termed fast Parzen Windows (FPW). This algorithm reduces the computational cost by first partitioning the data space into a set of fixed radii hyper-balls, and then letting a Gaussian with full covariance matrix to represent the probability density in each hyper-ball. In this way, global density in the data space is approximated by a mixture of Gaussians. Since all densities are estimated locally, no global optimization is required in FPW.

We presented two versions of FPW: FPW-H uses a hard-partitioning of the data sample, while FPW-S partitions the data sample in a soft manner. The former version is simpler and computationally more efficient than the latter one. Compared with the soft version, the hard version typically leads to only small performance degradation. Compared with

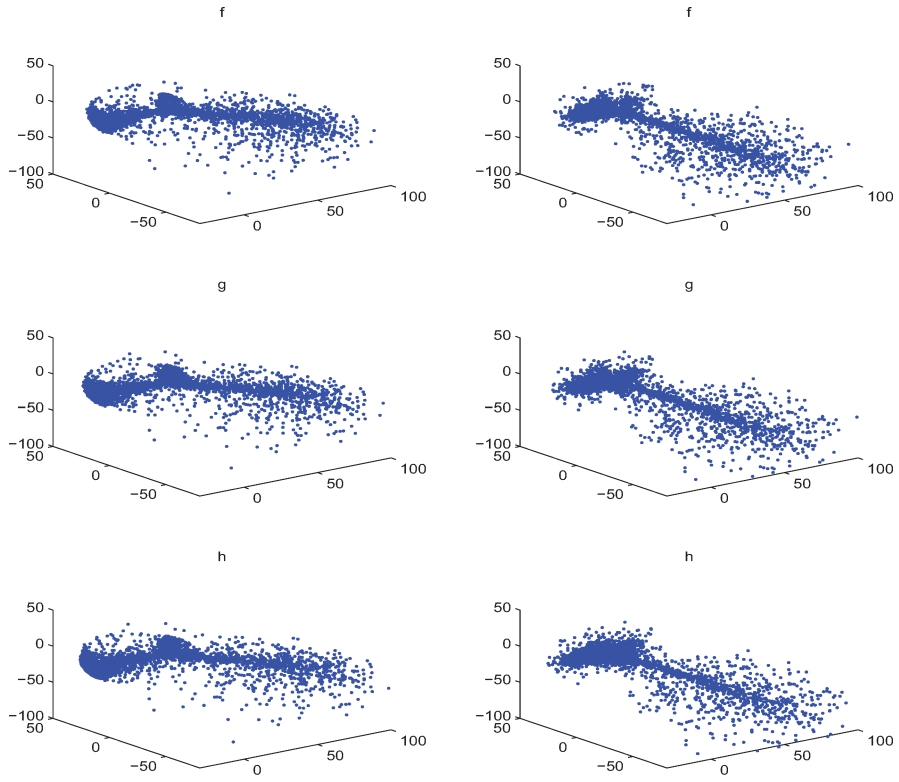


Fig. 6. 3-dimensional projections of the simulation data for the triplet $(f, g, h) = (20, 21, 22)$. The first projection (1st column) is onto the 3 spatial coordinates, the second (2nd column) is onto the leading 3 eigenvectors found by the Principal Component Analysis of the merged f, g and h sets.

other sparse kernel based density estimation methods, FPW methods often showed comparable or superior performance at a much lower computational cost.

In the future work, the proposed FPW methods will be employed to identify the most plausible astronomical simulation dataset among a large group of simulation datasets, given the real observed galaxy distribution. We also plan to put our density estimation framework on a stronger theoretical footing.

REFERENCES

[1] P. Vincent and Y. Bengio, *Manifold Parzen windows*, Advances in Neural Information Processing Systems 15, (2003), pp. 825–832.

[2] M. Girolami and C. He, *Probability Density Estimation from Optimally Condensed Data Samples*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 25 (2003), pp. 1253–1264.

[3] K. Popat and R. W. Picard, *Cluster-based probability model and its application to image and texture processing* , IEEE Transactions on Image Processing, 6 (1997), pp. 268-284.

[4] K. Zhang and J. T. Kwok, *Simplifying Mixture Models through Function Approximation*, Advances in Neural Information Processing Systems 15, Englewood Cliffs, NJ, (2006), pp. 825–832.

[5] J. Goldberger and S. Roweis, *Hierarchical clustering of a mixture model*, Advances in Neural Information Processing Systems 17, (2005), pp. 505–512.

[6] X. Hong, S. Chen and C. J. Harris, *A forward-constrained regression algorithm for sparse kernel density estimation*, IEEE Transactions on Neural Networks, 19 (2008), pp. 193-198.

[7] D. W. Scott and S. J. Sheather, *Kernel density estimation with binned data*, Comm. Statistics–Theory and Methods, 14 (1985), pp. 1353–1359.

[8] L. Holmström , *The accuracy and the computational complexity of a multivariate binned kernel density estimator*, J. Multivar. Anal.,72 (2000), pp. 264–309.

[9] J. Byeungwoo and D. A. Landgrebe, *Fast Parzen density estimation using clustering-based branch and bound*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 16 (1994), pp. 950 – 954.

[10] P. Mitra, C. A. Murthy and S. K. Pal, *Density-Based Multiscale Data Condensation*, IEEE Trans. Pattern Anal. Mach. Intell., 24 (2002), pp. 734–747.

[11] E. Parzen, *On Estimation of a Probability Density Function and Mode*, The Annals of Mathematical Statistics, 33 (1962), pp. 1065-1076.

[12] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Springer, 1991.

[13] M. A. Fardal and P. Guhathakurta and A. Babul and A. W. McConnachie, *Investigating the Andromeda stream - III. A young shell system in M31*, Monthly Notices of the Royal Astronomical Society, 380 Sept (2007), pp. 15-32.

[14] M. A. Fardal and A. Babul and J. J. Geehan and P. Guhathakurta, *Investigating the Andromeda stream - II. Orbital fits and properties of the progenitor*, Monthly Notices of the Royal Astronomical Society, 366 Mar (2006), pp. 1012-1028.

[15] A. Dempster and N. Laird and D. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, Series B 39(1) (1977), pp. 1–38.

[16] J. M. Kubica and J. Masiero and A. Moore and R. Jedicke and A. J. Connolly*Variable KD-Tree Algorithms for Spatial Pattern Search and Discovery*, Neural Information Processing Systems, Dec (2005).