Open Journal of Statistics, **2013, 3, 344-355**
http://dx.doi.org/10.4236/ojs.2013.35040 Published Online October 2013 (http://www.scirp.org/journal/ojs)

Aggregating Density Estimators: An Empirical Study

Mathias Bourel[1,2], Badih Ghattas[2]
[1]Instituto de Matemática y Estadística, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay
[2]Institut de Mathématiques de Luminy, Université d'Aix-Marseille, Marseille, France
Email: mbourel@fing.edu.uy, badih.ghattas@univ-amu.fr

Received June 20, 2013; revised July 20, 2013; accepted July 27, 2013

ABSTRACT

Density estimation methods based on aggregating several estimators are described and compared over several simulation models. We show that aggregation gives rise in general to better estimators than simple methods like histograms or kernel density estimators. We suggest three new simple algorithms which aggregate histograms and compare very well to all the existing methods.

Keywords: Machine Learning; Histogram; Kernel Density Estimator; Bagging; Boosting; Stacking

1. Introduction

Ensemble learning or aggregation methods are among the most challenging recent approaches in statistical learning. In a supervised framework, the main goal is to estimate a function $f:\mathcal{X}\to\mathcal{Y}$ using a data set of independent observations of both variables $x\in\mathcal{X}$ and $y\in\mathcal{Y}$. Ensemble learning constructs several such estimates $g_1,\cdots,g_M$, which are often called weak learners, and combines them to obtain the aggregated model $f=g(g_1,\cdots,g_M)$ where g may be a simple or a weighted average when $\mathcal{Y}\subseteq\Re$ such as for regression, or a simple or a weighted majority voting rule when $\mathcal{Y}\subseteq\{1,\cdots,J\}$ such as for classification. In this framework, Bagging ([1]), Boosting ([2]), Stacking ([3]), and Random Forests ([4]) have been declared to be the best of the shelf classifiers achieving very high performances when tested over tens of various datasets selected from the machine learning benchmark. All these algorithms had been designed for supervised learning, and sometimes initially restricted to regression or binary classification. Several extensions are still under study: multivariate regression, multiclass learning, and adaptation to functional data or time series.

Very few developments exist for ensemble learning in the unsupervised framework, clustering analysis and density estimation. Our work concerns the latter case which may be seen as a fundamental problem in statistics. Among the latest developments, we found some extensions of Boosting ([2]) and Stacking ([5]) to density estimation. The existing methods seem to be quite complex, often combining kernel density estimators and whose parameters seem to be arbitrary. In particular, most of the methods combine a fixed number of weak learners.

In this paper we show extensive simulations that aggregation gives rise to effective better estimates than simple classical density estimators. We suggest three simple algorithms for density estimation in the same spirit of bagging and stacking, where the weak learners are histograms. We compare our algorithms to several algorithms for density estimation, some of them are simple like Histogram and Kernel Density Estimators (*Kde*) and others rather complex like stacking and boosting, which will be described in details. As we will show in the experiments, although the accuracy of our algorithms is not systematically higher than other ensemble methods, they are simpler, more intuitive and computationally less expensive. Up to our knowledge, the existing algorithms have never been compared over a common benchmark simulation data.

Aggregating methods for density estimation are described in Section 2. Section 3 describes our algorithms. Simulations and results are given in Section 4 and concluding remarks and future work are described in Section 5.

2. A Review of the Existing Algorithms

In this Section we review some density estimators obtained by aggregation. They may be classified in two categories depending on the aggregation form.

The first type has the form of linear or convex combination:

$$f_M(x) = \sum_{m=1}^{M} \alpha_m g_m(x) \quad (1)$$

where $\alpha_m \in \Re$ and g_m is typically a parametric or non parametric density model, and in general different values of *m* refer typically to different parameters values in the parametric case or different kernels or different bandwidths for a chosen kernel for the kernel density estimators.

The second type of aggregation is multiplicative and is based on the ideas of high order bias reduction for kernel density estimation as in [6]. The aggregated density estimator has the form:

$$f_M(x) = \prod_{m=1}^{M} \alpha_m g_m(x) \quad (2)$$

2.1. Linear or Convex Combination of Density Estimators

This kind of estimators (1) has been used in several works with different construction schemes.

- In [7-9] the weak learners g_m are introduced sequentially in the combination. At step *m*, g_m is a density selected among a fixed class H and is chosen to maximize the log likelihood of

$$f_m(x) = (1-\alpha) f_{m-1}(x) + \alpha g_m(x), \ \alpha \in [0,1] \quad (3)$$

In [8], g_m is selected among a non parametric family of estimators, and in [7,9], it is taken to be a Gaussian density or a mixture of Gaussian densities whose parameters are estimated. Different methods are used to estimate both density g_m and the mixture coefficient α.

In [7], g_m is a Gaussian density and the log likelihood of (3) is maximized using a special version of Expectation Maximization (EM) taking into account that a part of the mixture is known.

The main idea underlying the algorithms given by [8,9] is to use Taylor expansion around the negative log likelihood that we wish to minimize:

$$\sum_i -\log f_m(x_i) \approx \sum_i -\log f_{m-1}(x_i) - \alpha \sum_i \frac{g_m(x_i)}{f_{m-1}(x_i)}$$

where $\mathsf{T} = \{x_1, \cdots, x_n\}$ is the data set. Thus, minimizing the left side term is equivalent to maximizing $\sum_i \frac{g_m(x_i)}{f_{m-1}(x_i)}$, and the output of this method converges to a global minimum.

All the algorithms described above are sequential and the number of weak learners aggregated may be fixed by the user.

- In [5], Smith and Wolpert used stacked density estimator applying the same aggregation scheme as in stacked regression and classification ([10]). The M densities estimators $g_1, \cdots, g_M$ are fixed in advance (KDE with different bandwidths). The data set $\mathsf{T} = \{x_1, \cdots, x_n\}$ is divided into *V* cross validation subsets $L_1, \cdots, L_V$. For $v = 1, \cdots, V$, denote $L^{(-v)} = L \setminus L_v$.

The *M* models $g_1, \cdots, g_M$ are fitted using the training samples $L^{(-1)}, \cdots, L^{(-V)}$ and the obtained estimates are denoted by for all $m = 1, \cdots, M$. These models are then evaluated over the test samples $L_1, \cdots, L_V$ getting the vectors $g_m^{(-v)}(L_v)$ for $m = 1, \cdots, M$, $v = 1, \cdots, V$ put within a $n \times M$ block matrix:

$$A = \begin{pmatrix} g_1^{(-1)}(L_1) & g_2^{(-1)}(L_1) & \cdots & g_M^{(-1)}(L_1) \\ g_1^{(-2)}(L_2) & g_2^{(-2)}(L_2) & \cdots & g_M^{(-2)}(L_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{(-v)}(L_v) & g_2^{(-v)}(L_v) & \cdots & g_M^{(-v)}(L_v) \end{pmatrix}$$

This matrix is used to compute the coefficients $\alpha_1, \cdots, \alpha_M$ of the aggregated model (1) using the EM algorithm. Finally, for the output model, we re-estimate the individual densities $g_1, \cdots, g_M$ from the whole data.

This method has been compared with other algorithms including the best single model obtained by cross-validation, the best single model obtained over a test sample and a uniform average of the different *Kde* models. It is shown that stacking outperforms these methods for different criteria: log likelihood, L_1 and L_2 performance measures (for the two last criteria use the true density).

- In [11], Rigollet and Tsybakov fixed the densities $g_1, \cdots, g_M$ in advance like for stacking (*Kde* estimators with different bandwidths). The dataset is split in two parts. The first sample is used to estimate the densities g_m, whereas the coefficients α_m are optimized using the second sample. The splitting process is repeated and the aggregated estimators for each data split are averaged. The final model has the form

$$f_M(x) = \frac{1}{|S|} \sum_{s \in S} g_M^s(x) \quad (4)$$

where *S* is the set of all the splits used and

$$g_M^s(x) = \sum_{m=1}^{M} \alpha_m \hat{g}_m^s(x) \quad (5)$$

is the aggregated estimator obtained from one split s of the data, $\hat{g}_m^s$ is the individual kernel density function estimated over the learning sample obtained from the split *s*. This algorithm is called *AggPure*. The authors compared their algorithm using different choices for the *Kde*'s bandwidth that we describe in the simulations section. Oracle inequalities and risk bounds are given for this estimator.

2.2. Multiplicative Aggregation

The only algorithm giving rise to this form of aggrega-

tion is the one described in [12] called *BoostKde*. It is a sequential algorithm where at each step *m* the weak learner is computed as follows:

$$\hat{g}_m(x) = \sum_{i=1}^{n} \frac{w_m(i)}{h} K\left(\frac{x - x_i}{h}\right) \quad (6)$$

where *K* is a fixed kernel, *h* its bandwidth, and $w_m(i)$ the weight of observation *i* at step *m*. Like for boosting, the weight of each observation is updated by

$$w_{m+1}(i) = w_m(i) + \log\left(\frac{\hat{g}_m(x_i)}{\hat{g}_m^{(-i)}(x_i)}\right) \quad (7)$$

where $\hat{g}_m^{(-i)}(x_i) = \sum_{j \neq i} \frac{w_m(i)}{h} K\left(\frac{x_i - x_j}{h}\right)$

The output is given by

$$f_M(x) = C \prod_{m=1}^{M} \hat{g}_m(x) \quad (8)$$

where C is a normalization constant.

Using several simulation models the authors explore different values for the bandwidth *h* minimizing the Mean integrated Square Error (MISE) for few values of M. They show that a bias reduction is obtained for *M* = 2 but it is not clear how the algorithm behaves for more than two steps.

3. Aggregating Histograms

We suggest three new density estimators obtained by linear combination like in (1), all of them use histograms as weak learners. The first two algorithms aggregate randomized histograms and may be parallelized. The third one is just an adaptation of Stacking using histograms instead of kernel density estimators.

The first algorithm is similar to Bagging ([1]). Given a data set $\mathsf{T} = \{x_1, \cdots, x_n\}$ and an integer *L*, at each step $m = 1, \cdots, M$ of the algorithm a bootstrap sample of T is generated and used to construct an histogram g_m with *L* equally spaced breakpoints. The output of this method is an average of the *M* histograms. We will refer to this algorithm as *BagHist* and it is detailed in **Figure 1**.

The second algorithm, *AggregHist*, works as follows. Consider the data set $\mathsf{T} = \{x_1, \cdots, x_n\}$ and an integer *L*.

1. Let T the original sample and *L* an integer.
2. For $m = 1, \cdots, M$:
   a Let T^m be a booststrap sample of T.
   b Set g_m to be the histogram constructed over T^m with *L* equispaced breakpoints.
3. Output: $f_M(x) = \sum_{m=1}^{M} \alpha_m g_m(x)$

Figure 1. Bagging histograms (*BagHist*).

Let g_0 be the histogram obtained over T using equally spaced breakpoints denoted by $B = \{b_1, \cdots, b_L\}$. We denote by $h = b_l - b_{l-1}$ for all $l = 2, \cdots, L$ the bandwidth of g_0. At each step $m = 1, \cdots, M$ we add a random uniform noise $\varepsilon_{l,m} \sim U[0, h]$ to each breakpoint and construct an histogram g_m using T and the new set of breakpoints. The final output is an average of the histograms $g_1, \cdots, g_M$. The algorithm is detailed in **Figure 2**.

The value of the parameter *L* used for *AggregHist* and *BagHist* will be optimized and the procedure used for that will be described in the next Section.

Finally, we introduce a third algorithm called *StackHist* where we replace in the stacking algorithm the six kernel density estimators by histograms with different number of breakpoints.

4. Experiments

In this Section we present the simulations we have done to compare all the methods described in Section 2 together with our three algorithms. We consider several data generating models we have found in the literature. We first show how our algorithms adjust quite well for the different models, and that the adjustment error decreases monotonically with the numbers of histograms used. Finally we will compare our methods with ensemble methods for density estimation like *Stacking*, *AggPure*, *BoostKde* which aggregated non parametric density estimators. All these aggregating methods are compared to optimized Histogram (*Hist*) and *Kde* using different bandwidth optimization approaches.

4.1. Models Used for the Simulations

Twelve models found in the papers we have referenced are used in our simulations. We denote them by $\mathsf{M}_1, \cdots, \mathsf{M}_{12}$ and we group them according to their difficulty level.

- Some standard densities used in [11,12]:
  (M_1) standard Gaussian density $N(0,1)$,
  (M_2) standard exponential density,
  (M_3) a Chisquare density χ^2_{10},

1. Let T the original sample, fix an integer *L* and construct the histogram g_0 built over T, $B = \{b_1, \cdots, b_L\}$ the equally spaced breakpoints of g_0 and $h = b_l - b_{l-1}$.
2. For $m = 1, \cdots, M$:
   a Consider $B^m = \{b_1^*, \cdots, b_L^*\}$ the randomly modified set of breakpoints where $b_l^* = b_l + \varepsilon_{l,m}$, $\varepsilon_{l,m} \sim U[0, h]$
   b We compute $\hat{g}_m$ the histogram over T using these new breakpoints.
3. Output: $f_M(x) = \sum_{m=1}^{M} \alpha_m g_m(x)$

Figure 2. Aggregating histograms based on randomly perturbed breakpoints (*AggregHist*).

($\mathcal{M}_4$) a student density t_4,

- Some Gaussian mixtures taken from [5,12]:

($\mathcal{M}_5$) $0.5N(-1,0.3)+0.5N(1,0.3)$,

($\mathcal{M}_6$) $0.25N(-3,0.5)+0.5N(0,1)+0.25N(3,0.5)$

($\mathcal{M}_7$) $0.55N(-3,0.5)+0.35N(0,1)+0.1N(3,0.5)$

- Gaussian mixtures used in [11] and taken from [13]:

($\mathcal{M}_8$) the Claw density,

($\mathcal{M}_9$) the Smooth Comb density,

- ($\mathcal{M}_{10}$) is a mixture density with highly inhomogeneous smoothness as in [12]
- Finally we include in our study two simple models known to be challenging for density estimators:

($\mathcal{M}_{11}$) a triangular density with support [0,1] and maximum at 1,

($\mathcal{M}_{12}$) the beta density with parameters 2 and 5.

All the simulations are done with the **R** software, and for models $\mathcal{M}_8$ and $\mathcal{M}_9$ we use the **benchden** package. **Figure 3** shows the shape of the densities we have used to generate the data sets.

Figure 4 shows, for some models, the estimator obtained using *AggregHist* (red curve), *BagHist* (green curve) and *StackHist* (blue curve) for $n = 1000$ observations and $M = 150$ histograms for the two first algorithms. A simple histogram is shown together with the three estimates. For *StackHist* we aggregate six histograms having 5, 10, 20, 30, 40 and 50 equally spaced breakpoints and a ten fold cross validation is used. Both *AggregHist* and *BagHist* give more smooth estimators than *StackHist*.

Figures 5 and **6** show the adjusted densities obtained from *AggregHist* and *BagHist* when increasing the number M of histograms for model $\mathcal{M}_7$.

4.2. Tuning the Algorithms

We compare the following algorithms *AggregHist*, *BagHist*, *StackHist*, *Stacking*, *AggPure* and *BoostKde* with some classical methods like *Hist* and *Kde*.

For *AggregHist* and *BagHist* we fix the number of histograms to $M = 200$. The number of breakpoints is optimized testing different values over a fixed grid of 10, 20 and 50 equally spaced breakpoints. The optimal value retained for each model is the one which maximizes the log likelihood over 100 independent test samples drawn

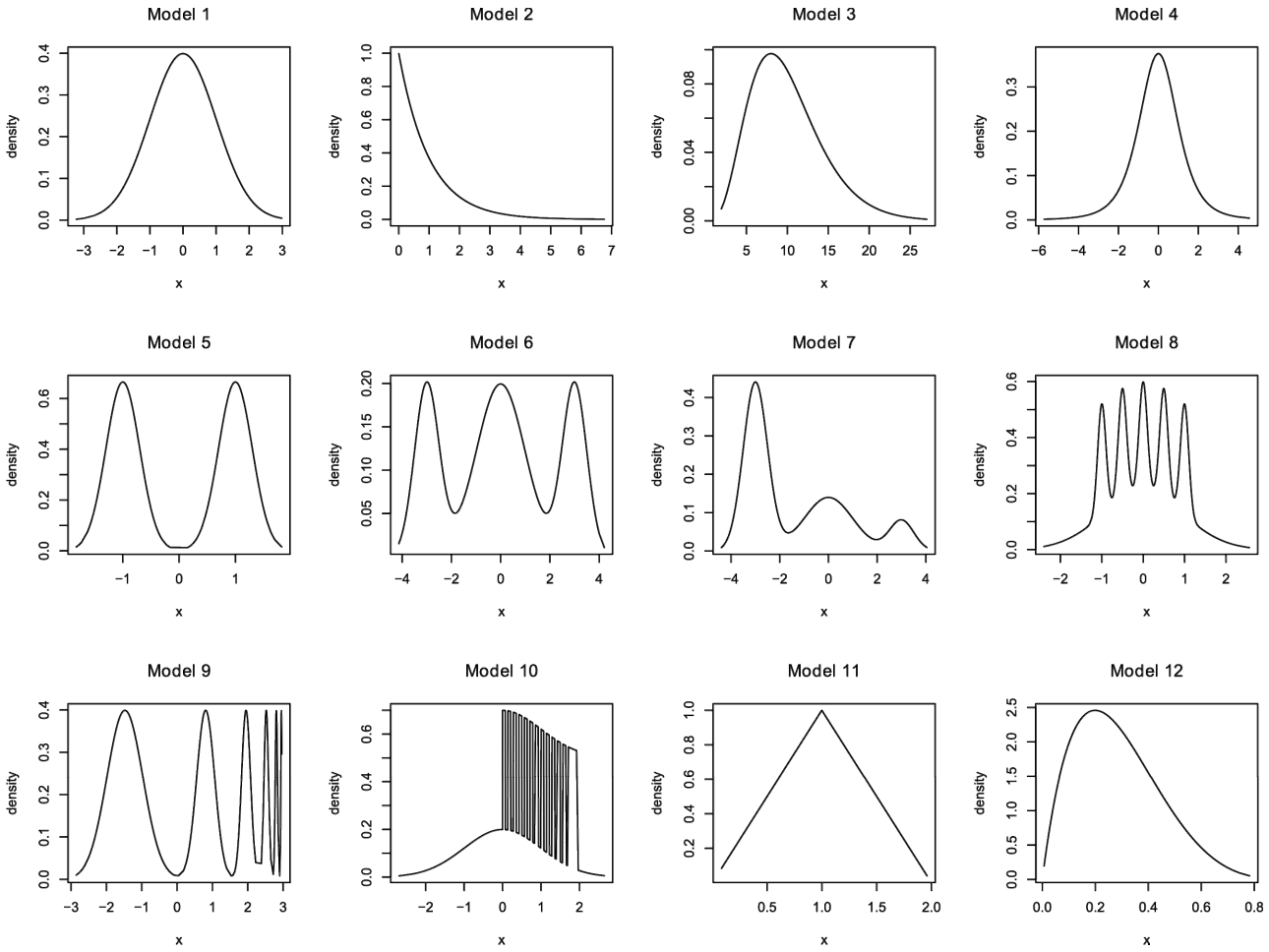

Figure 3. Densities used for the simulations.

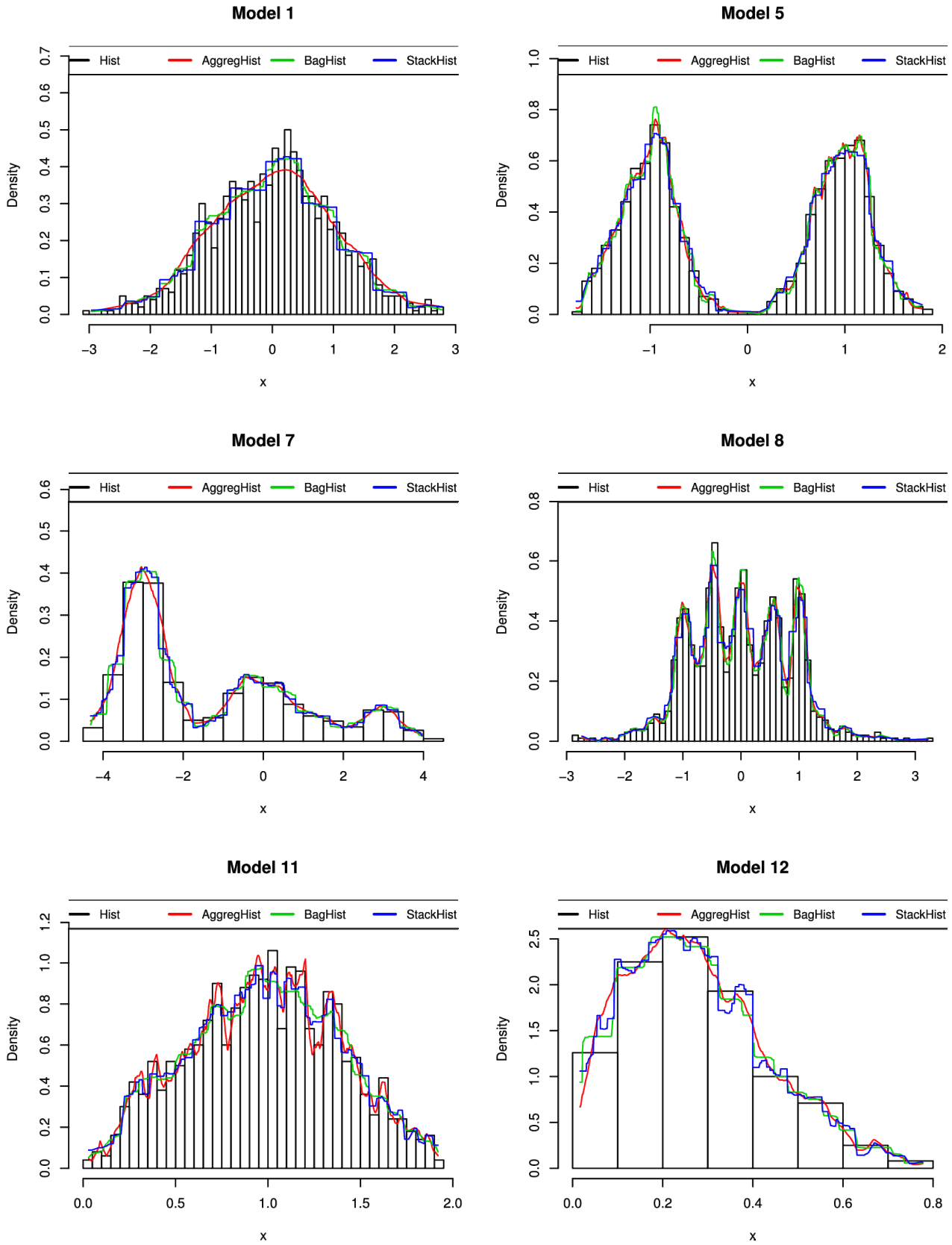

Figure 4. Density estimators obtained used for *Hist*, *AggregHist* (red curve), *BagHist* (green curve) and *StackHist* (blue curve) for 6 models among those used in the simulations.

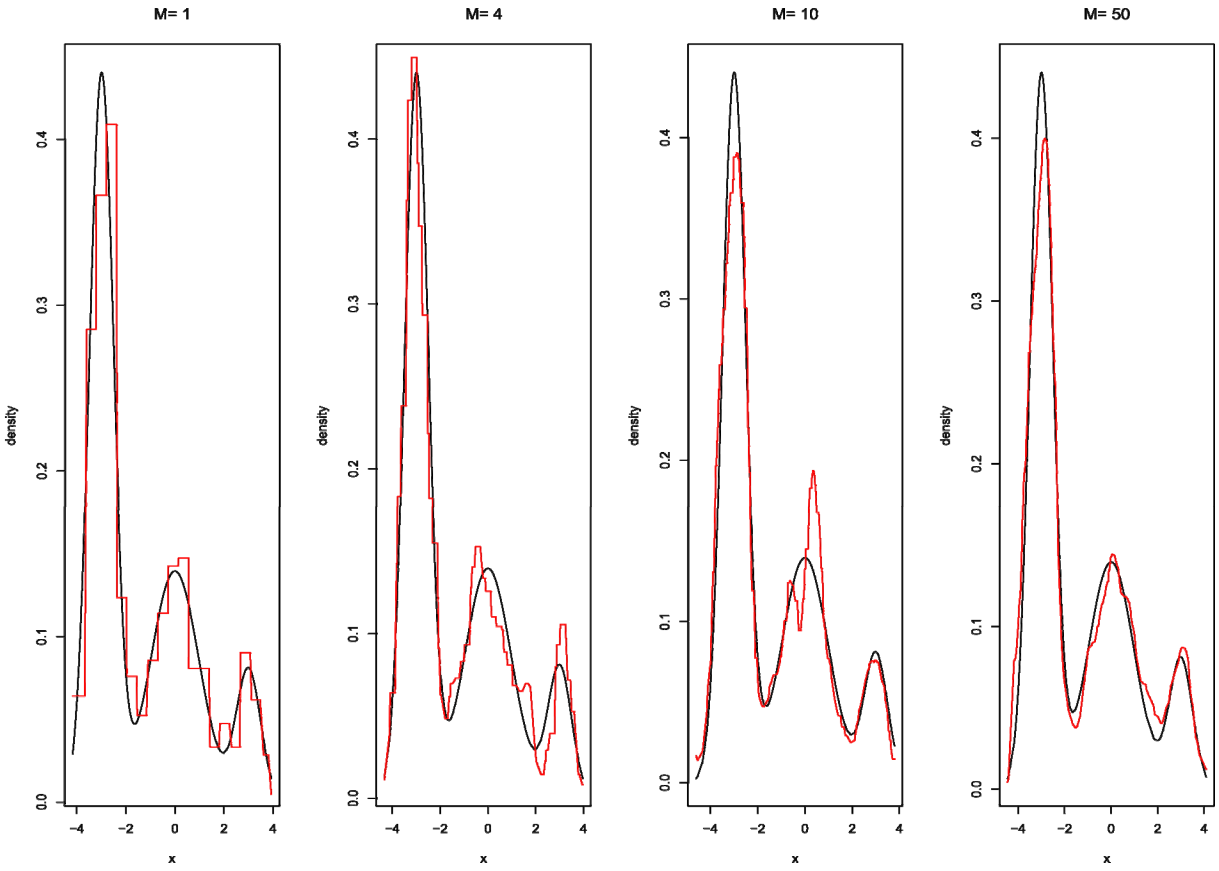

Figure 5. Density estimate given by *AggregHist* with different values of *M* for model M_7.

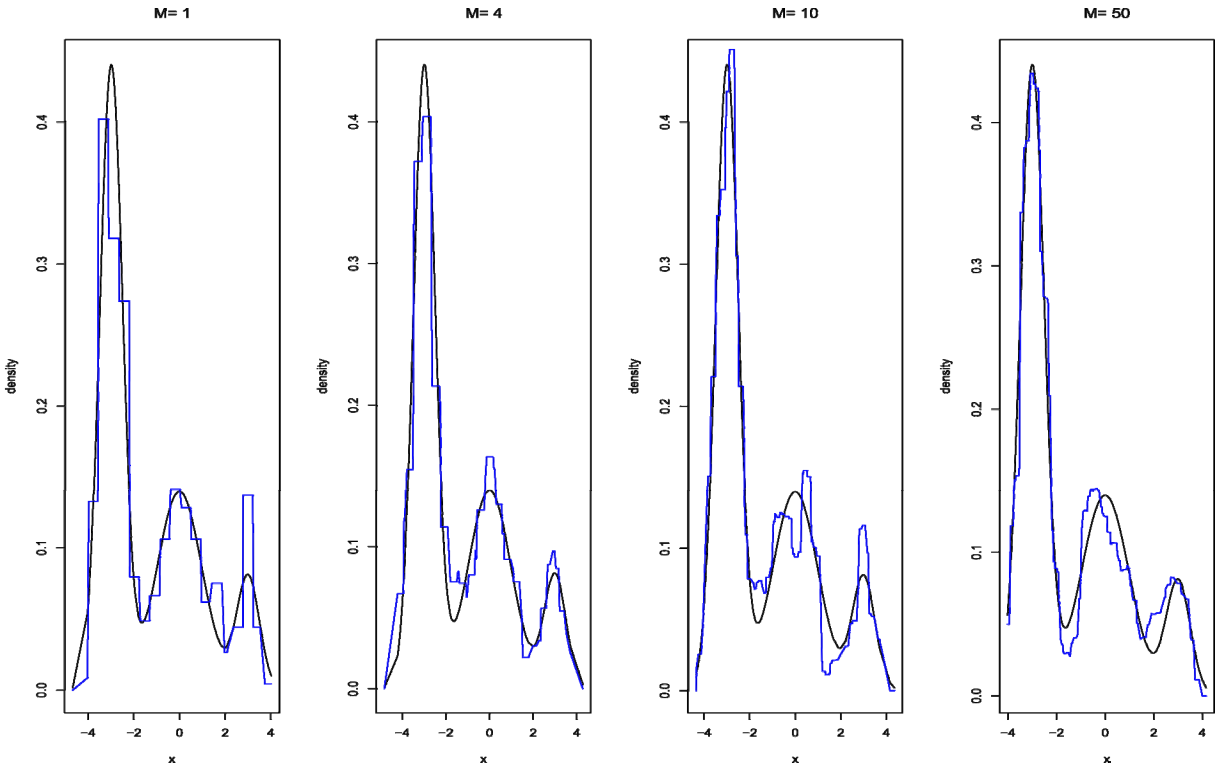

Figure 6. Density estimate given by *BagHist* with different values of *M* for model M_7.

from the corresponding model. We optimize the number of breakpoints of the histogram in the same way as for our algorithms. These optimal values are given in **Table 1** for differents values of n (100, 500 and 1000). We denote the optimal number of breakpoints L_H, L_{BH} and L_{AH} for *Hist*, *BagHist* and *AggregHist* respectively.

To make the comparisons as faithful as possible, we have used for the other methods the same values suggested by their corresponding authors:

- For *Stacking*, six kernel density estimators are aggregated, three of them use Gaussian kernels with fixed bandwidths h = 0.1, 0.2, 0.3 and the others use triangular kernels with bandwidths h = 0.1, 0.2, 0.3. The number of cross validation samples is V = 10.
- For *AggPure* six kernel density estimators are aggregated having bandwidths 0.001, 0.005, 0.01, 0.05, 0.1 and 0.5. Instead of the quadratic algorithm used by the authors in [11], we optimize the coefficients of the linear combination with the EM algorithm. The final estimator is a mean over $|S|$ = 10 random splits of the original data set.
- For *BoostKde*, we use 5 steps for the algorithm aggregating kernel density estimators whose bandwidths are optimized using Silverman rule of thumbs (see **Appendix**). Normalization of the output is done using numerical integration. Extensive simulations we have done using *BoostKde* showed that more steps give rise to less accurate estimators and unstable results.

For *Kde* we use a standard gaussian kernel and some common data driven bandwidth selectors:

Table 1. Optimal number of breakpoints used for *Hist* and our algorithms for each model and for each value of *n*.

Model	n = 100			n = 500			n = 1000		
	L_H	L_{BH}	L_{AH}	L_H	L_{BH}	L_{AH}	L_H	L_{BH}	L_{AH}
M_1	50	50	10	50	10	10	50	20	10
M_2	50	50	50	50	50	50	50	50	50
M_3	50	50	10	50	50	20	50	50	20
M_4	50	50	20	50	50	50	50	50	50
M_5	50	50	10	50	50	20	50	50	20
M_6	50	10	10	20	20	20	20	20	20
M_7	50	10	10	50	20	20	50	20	20
M_8	50	50	50	50	50	50	50	50	50
M_9	50	50	20	50	50	50	50	50	50
M_{10}	50	50	50	50	50	50	50	50	50
M_{11}	50	10	10	10	10	10	20	20	10
M_{12}	50	10	10	20	20	10	20	20	20

- the Silverman's rules of thumb ([14]) using factor 1.06 (*KdeNrd*) and factor 0.9 (*KdeNrd*0),
- the unbiased cross-validation rule (*KdeUCV*, [15]),
- the Sheater Johns plug-in method (*KdeSJ*, [16,17]).

These choices are described in details in the appendix.

4.3. Results

The performance of each estimator is evaluated using the Mean Integrated Squared Error (MISE):

$$MISE\left(\hat{f}(x)\right) = E\left(\int\left[\hat{f}(x) - f(x)\right]^2 \mathrm{d}x\right) \tag{9}$$

where f is the true density and $\hat{f}$ its estimate. It is computed as the average of the integrated squared error

$$ISE\left(\hat{f}(x)\right) = \int\left[\hat{f}(x) - f(x)\right]^2 \mathrm{d}x \tag{10}$$

over 100 Monte Carlo simulations.

For the same simulations, we have also computed the log likelihood criterion whose maximization is equivalent to reducing the Kullback-Leibler divergence between the true and the estimated densities (see P. Hall, "On Kullback-Leibler loss and density estimation", Ann. Stat., Vol. 15, 1987). The results obtained using this criterion are unstable due to numerical approximation of the log likelihood for small values of the densities. In particular the histogram has very good performance with respect to the log likelihood when compared to all the other methods. This is due to the fact that when computing the log likelihood we omit the points i for which $\hat{f}(x_i)$ equals zero, and such points appear much more for the histogram than for the other methods. For all these reasons we do not report these results.

Figure 7 shows how the MISE varies when increasing the number of histograms in *AggregHist* and *BagHist*. For all the models, the adjustment error decreases significantly for the first 100 iterations. In most of the cases *AggregHist* gives a better estimate than *BagHist*.

Table 2 shows the execution time for *AggregHist*, *BagHist*, *Stacking* and *AggPure* for n = 2000. The other algorithms need much less time as they combine very few simple estimators and do not use any resampling. Computing time is significantly lower for our algorithms.

We compare now all the algorithms cited above over the twelve models using n = 100, n = 500 and n = 1000. For *AggregHist* and *BagHist* we use M = 200 histograms. **Tables 3** to **5** summarize the results for each value of n = 100, 500 and 1000. For each model and each method we give the average of 100 × *MISE* over 100 Monte Carlo simulations. For the *Kde* we kept the best result among the four choices of bandwidth selectors (*nrd*, *nrd*0, *ucv* and *sj*), the best choice being between brackets. The best result for each simulation model is put in bold.

It is clear that no method outperforms all the others in all the cases and for all the methods the error decreases

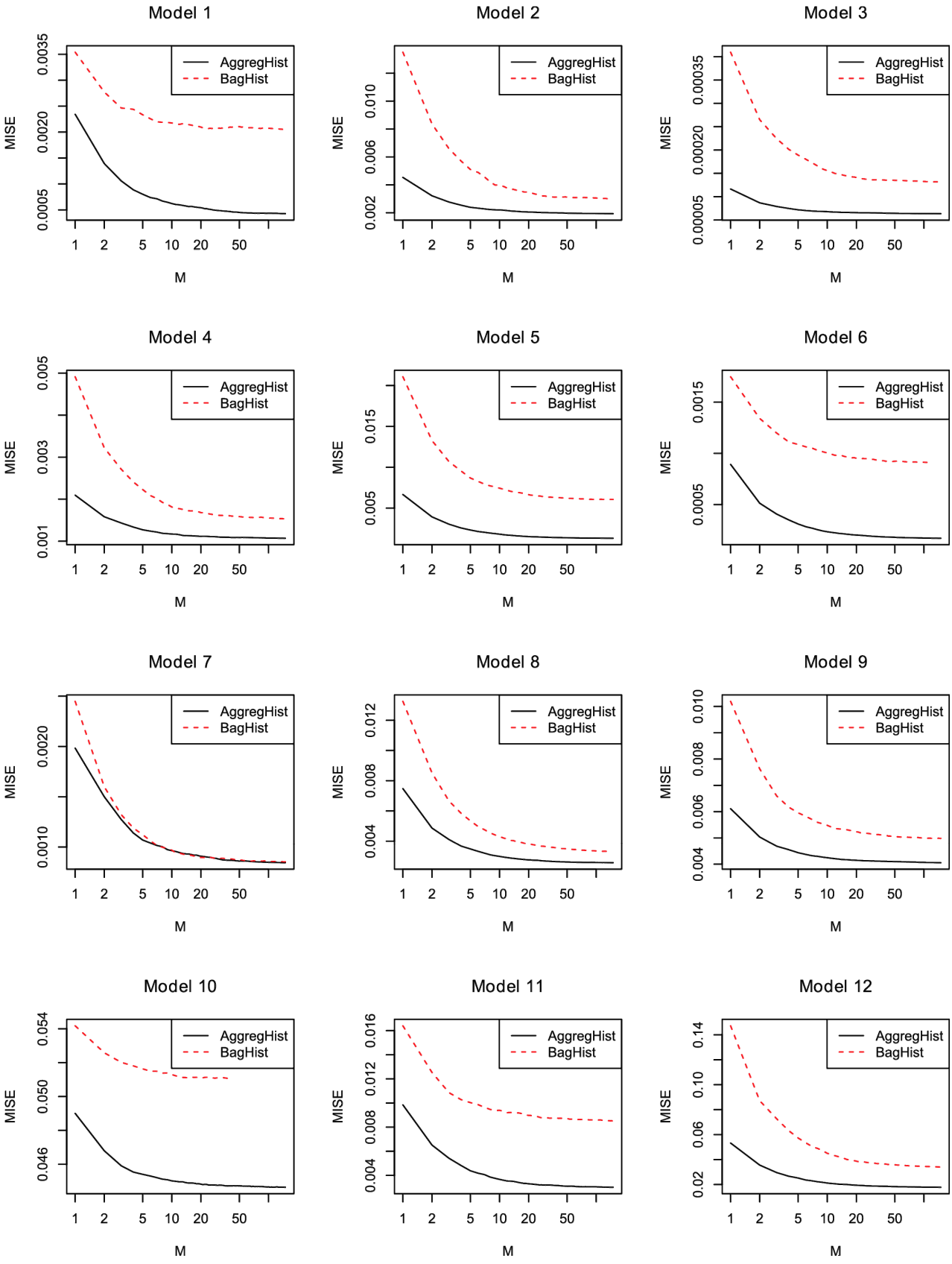

Figure 7. MISE error versus number of aggregated histograms in *AggregHist* and *BagHist* for models 1 to 12.

Table 2. Time (in seconds) elapsed for each model for *AggregHist, BagHist, Stacking* and *AggPure* taking $n = 2000$ and $M = 50$.

Model	*AggregHist*	*BagHist*	*Stacking*	*AggPure*
$\mathcal{M}_1$	18.3	19.6	22.3	161.4
$\mathcal{M}_2$	20.3	19.2	24.6	168.6
$\mathcal{M}_3$	17.4	17.6	27.7	160.2
$\mathcal{M}_4$	20.3	19.3	28.3	177.6
$\mathcal{M}_5$	19.1	20.5	31.8	177.6
$\mathcal{M}_6$	20.0	18.0	25.3	165
$\mathcal{M}_7$	20.1	19.6	27.7	163.2
$\mathcal{M}_8$	19.6	20.2	25.1	166.8
$\mathcal{M}_9$	18.8	20.4	25.3	170.4
$\mathcal{M}_{10}$	18.5	19.6	24.5	163.8
$\mathcal{M}_{11}$	20.0	19.8	25.1	175.8
$\mathcal{M}_{12}$	20.0	19.2	20.8	160.8

Table 3. 100 × MISE with $n = 100$ and $M = 200$.

Model	*Hist*	*Kde*	*Stacking*	*StackHist*	*BoostKde*	*AggPure*	*BagHist*	*AggregHist*
$\mathcal{M}_1$	2.94	**0.18**(nrd0)	0.268	0.546	0.441	0.407	2.33	0.268
$\mathcal{M}_2$	5.06	4.24(ucv)	2.35	**1.48**	8.19	2.58	4.38	3.31
$\mathcal{M}_3$	0.15	**0.0103**(nrd)	0.0612	0.0301	0.0306	0.0995	0.118	0.0148
$\mathcal{M}_4$	1.41	**0.189**(nrd0)	0.211	0.389	1.29	0.366	1.15	0.301
$\mathcal{M}_5$	6.72	3.02(ucv)	0.897	1.9	**0.515**	1.09	5.18	0.757
$\mathcal{M}_6$	0.843	0.156(ucv)	**0.112**	0.18	0.114	0.166	0.137	0.114
$\mathcal{M}_7$	1.3	0.542(ucv)	**0.196**	0.413	0.262	0.233	0.303	0.31
$\mathcal{M}_8$	3.99	2.16(ucv)	**1.51**	1.7	3.19	1.82	3.07	2.26
$\mathcal{M}_9$	2.51	1.35(ucv)	0.933	1.14	1.34	0.97	1.97	**0.841**
$\mathcal{M}_{10}$	6.24	6.72(nrd0)	5.76	5.79	6.21	5.22	4.91	**4.79**
$\mathcal{M}_{11}$	18.6	2.2(nrd0)	**1.22**	2.75	1.87	1.3	2.48	1.75
$\mathcal{M}_{12}$	133	58.1(nrd0)	**8.67**	21.3	16.6	12.9	18.4	13.8

Table 4. 100 × MISE with $n = 500$ and $M = 200$.

Model	*Hist*	*Kde*	*Stacking*	*StackHist*	*BoostKde*	*AggPure*	*BagHist*	*AggregHist*
$\mathcal{M}_1$	0.46	0.0839(nrd0)	0.0536	0.16	0.131	0.0825	0.085	**0.05**
$\mathcal{M}_2$	0.742	3.01(ucv)	1.25	0.605	6.6	1.41	0.684	**0.477**
$\mathcal{M}_3$	0.0241	**0.00292**(nrd)	0.0121	0.009	0.0192	0.0194	0.0185	0.00502
$\mathcal{M}_4$	0.182	0.091(nrd0)	**0.042**	0.113	1.57	0.0576	0.131	0.0861
$\mathcal{M}_5$	1.26	2.01(ucv)	0.219	0.57	**0.107**	0.41	0.916	0.231
$\mathcal{M}_6$	0.073	0.0843(ucv)	0.0311	0.0624	**0.022**	0.0329	0.0448	0.0294
$\mathcal{M}_7$	0.231	0.313(ucv)	0.0734	0.142	**0.0368**	0.0694	0.115	0.0675
$\mathcal{M}_8$	0.933	1.78(ucv)	0.603	0.707	2.13	0.765	0.606	**0.485**
$\mathcal{M}_9$	0.625	0.89(ucv)	0.403	0.449	1.06	0.449	0.507	**0.379**
$\mathcal{M}_{10}$	4.8	6.12(ucv)	4.47	4.94	5.43	**2.79**	4.3	4.13
$\mathcal{M}_{11}$	0.856	1.49(nrd0)	0.444	0.902	0.505	0.6	0.66	**0.367**
$\mathcal{M}_{12}$	9.46	40(ucv)	**2.11**	6.79	4.6	3.43	7.22	2.65

Table 5. 100 × MISE with n = 1000 and M = 200.

Model	*Hist*	*Kde*	*Stacking*	*StackHist*	*BoostKde*	*AggPure*	*BagHist*	*AggregHist*
$\mathcal{M}_1$	0.216	0.0694(nrd0)	0.0364	0.089	0.076	0.0407	0.0663	**0.0359**
$\mathcal{M}_2$	0.378	2.49(ucv)	0.864	0.365	6	0.918	0.333	**0.21**
$\mathcal{M}_3$	0.0112	**0.00178**(nrd)	0.00614	0.00556	0.0183	0.00971	0.0085	0.00252
$\mathcal{M}_4$	0.0893	0.0748(nrd0)	**0.0285**	0.0716	1.44	0.0315	0.0557	0.0381
$\mathcal{M}_5$	0.627	1.69(ucv)	0.155	0.349	**0.0674**	0.271	0.445	0.152
$\mathcal{M}_6$	0.0535	0.0662(ucv)	0.0218	0.0426	**0.0124**	0.0221	0.0321	0.0203
$\mathcal{M}_7$	0.137	0.227(ucv)	0.0381	0.084	**0.0198**	0.0401	0.0835	0.0453
$\mathcal{M}_8$	0.607	1.69(ucv)	0.37	0.522	1.54	0.482	0.337	**0.299**
$\mathcal{M}_9$	0.393	0.753(ucv)	0.277	0.321	0.889	0.299	0.32	**0.259**
$\mathcal{M}_{10}$	4.76	5.57(ucv)	4.09	4.78	5.39	**1.97**	4.44	4.16
$\mathcal{M}_{11}$	0.681	1.19(nrd0)	0.268	0.544	0.302	0.381	0.531	**0.202**
$\mathcal{M}_{12}$	4.99	32.7(ucv)	**1.28**	4.23	2.67	2.14	3.7	2.3

when increasing the sample size. Ensemble methods estimators outperform largely the *Hist* and *Kde* in most cases except three models (standard gaussian, χ^2_{10} and student distributions) for n = 100, and only the χ^2_{10} model for n = 500 and 1000. For the gaussian mixtures models *BoostKde* gives the best results for the largest values of n (500 and 1000). *Stacking* outperforms the other algorithms for the student and the triangular distributions for n = 1000. For model $\mathcal{M}_{10}$, *AggPure* achieves the best performance. For the remaining simulation models, *AggregHist* outperforms all the other methods when n = 1000. When it is doesn't achieve the lowest error, it is still very close to the best especially for the most challenging distributions used in models $\mathcal{M}_8$ to $\mathcal{M}_{12}$.

5. Conclusion

In this paper we have given a brief summary of most existing approaches for density estimation based on aggregation. We have shown using a wide range of simulations that, like for supervised learning, ensemble methods give rise to better density estimators than the Histogram and the Kernel Density Estimators. Among the existing methods we have tested direct extensions of stacking (*Stacking*) and of boosting (*BoostKde*) as well as the most recent approach found in the literature called *AggPure*.

We have also suggested three new algorithms among which two of them *BagHist* and *AggregHist* combine a large number of histograms. *BagHist* randomizes the data using bootstrap samples whereas *AggregHist* uses the original data set, randomizing the histogram breakpoints. *AggregHist* gives very good results for most of the situations especially for large sample sizes, and it is easy to implement and has the lowest computation cost among all the ensemble density estimators.

Most of the presented algorithms may be extended to the multivariate case and are now under study both empirically and theoretically. All the simulations and the methods have been implemented within an **R** package available upon request.

6. Acknowledgements

The authors would like to thank Claude Deniau and Ricardo Fraiman for many useful discussions and suggestions that have improved this work.

REFERENCES

[1] L. Breiman, "Bagging Predictors," *Machine Learning*, Vol. 24, No. 2, 1996, pp. 123-140. http://dx.doi.org/10.1007/BF00058655

[2] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and Application to Boosting," *Journal of Computer and System Sciences*, Vol. 55, No. 1, 1997, pp. 119-139. http://dx.doi.org/10.1006/jcss.1997.1504

[3] L. Breiman, "Stacked Regression," *Machine Learning*, Vol. 24, No. 1, 1996, pp. 49-64.

[4] L. Breiman, "Random Forests," *Machine Learning*, Vol. 45, No. 1, 2001, pp. 5-32. http://dx.doi.org/10.1023/A:1010933404324

[5] P. Smyth and D. H. Wolpert, "Linearly Combining Density Estimators via Stacking," *Machine Learning*, Vol. 36, No. 1, pp. 59-83.

[6] M. Jones, O. Linton and J. Nielsen, "A Simple Bias Reduction Method for Density Estimation," *Biometrika*, Vol. 82, No. 2, 1995, pp. 327-338. http://dx.doi.org/10.1093/biomet/82.2.327

[7] G. Ridgeway, "Looking for Lumps: Boosting and Bagging for Density Estimation," *Computational Statistics & Data Analysis*, Vol. 38, No. 4, 2002, pp. 379-392. http://dx.doi.org/10.1016/S0167-9473(01)00066-4

[8] S. Rosset and E. Segal, "Boosting Density Estimation," *Advances in Neural Information Processing Systems*, Vol. 15, MIT Press, 2002, pp. 641-648.

[9] X. Song, K. Yang and M. Pavel, "Density Boosting for Gaussian Mixtures," *Neural Information Processing*, Vol. 3316, 2004, pp. 508-515. http://dx.doi.org/10.1007/978-3-540-30499-9_78

[10] D. H. Wolpert, "Stacked Generalization," *Neural Networks*, Vol. 5, No. 2, 1992, pp. 241-259. http://dx.doi.org/10.1016/S0893-6080(05)80023-1

[11] P. Rigollet and A. B. Tsybakov, "Linear and Convex Aggregation of Density Estimators," *Mathematical Methods of Statistics*, Vo. 16, No. 3, 2007, pp. 260-280. http://dx.doi.org/10.3103/S1066530707030052

[12] M. Di Marzio and C. C. Taylor, "Boosting Kernel Density Estimates: A Bias Reduction Technique?" *Biometrika*, Vol. 91, No. 1, 2004, pp. 226-233. http://dx.doi.org/10.1093/biomet/91.1.226

[13] J. S. Marron and M. P. Wand, "Exact Mean Integrated Square Error," *The Annals of Statistics*, Vol. 20, No. 2, 2004, pp. 712-736. http://dx.doi.org/10.1214/aos/1176348653

[14] B. W. Silverman, "Density Estimation for Statistics and Data Analysis," Chapman and Hall, London, 1986.

[15] A. Bowman, "An Alternative Method of Cross-Validation for the Smoothing of Density Estimates," *Biometrika*, Vol. 71, No. 2, 1984 pp. 353-360. http://dx.doi.org/10.1093/biomet/71.2.353

[16] S. J. Sheather, "Density Estimation," *Statistical Science*, Vol. 19, No. 4, 2004, pp. 588-597. http://dx.doi.org/10.1214/088342304000000297

[17] W. Härdle, M. Müller, S. Sperlich and A. Werwatz, "Nonparametric and Semiparametric Models," Springer Verlag, Heidelberg, 2004. http://dx.doi.org/10.1007/978-3-642-17146-8

[18] S. J. Sheather and M. C. Jones, "A Reliable Data-Based Bandwidth Selection Method for Density Estimation," *Journal of the Royal Statistical Society*, Vol. 53, No. 3, 1991, pp. 683-690.

Appendix

Different Choices for the Bandwidth Selection in KDE

If $\mathsf{T} = \{x_1, \cdots, x_n\}$ denotes a sample obtained from a random variable X with density f, the kernel density estimation of f at point x is:

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{11}$$

where K is a kernel function.

A usual measure of the difference of the estimate density and the true density is the Mean integrated Squared Error (*MISE*) which may be written as:

$$\begin{aligned} MISE\left(\hat{f}_h\right) &= \int \mathrm{Bias}\left(\hat{f}_h(x)\right)^2 \mathrm{d}x \\ &\quad + \int \mathrm{Var}\left(\hat{f}_h(x)\right)\mathrm{d}x \end{aligned} \tag{12}$$

It is well known

$$\begin{aligned} MISE\left(\hat{f}_h\right) &= \frac{1}{nh}R(K) + \frac{h^4}{4}\mu_2^2(K)R(f'') \\ &\quad + o\left(\frac{1}{nh}\right) + o\left(h^4\right) \end{aligned}$$

with $h \to 0$ and $nh \to +\infty$ where $R(g) = \int g^2(x)\mathrm{d}x$ and $\mu_2(K) = \int x^2 K(x)\mathrm{d}x$ The Asymptotic Mean Integrated Squared Error (AMISE) is:

$$AMISE\left(\hat{f}_h\right) = \frac{1}{nh}R(K) + \frac{h^4}{4}\mu_2^2(K)R(f'') \tag{13}$$

and it is minimized for

$$h^* = \left[\frac{R(K)}{\mu_2^2(K)R(f'')}\right]^{1/5} n^{-1/5} \tag{14}$$

Taking Gaussian kernel K and assuming that the underlying distribution is normal, Silverman ([14]) showed that the expression of h^* is

$$h_{NRD} = 1.06 \min\left\{\hat{\sigma}, \frac{IQR}{1.34}\right\} n^{-1/5} \tag{15}$$

where $\hat{\sigma}$ and IQR are the standard deviation and the interquantile distance respectively of the sample. This is known as Silverman's rule of thumbs. Furthermore Silverman recommended to use for the constant the values 0.9 (*Nrd*0) or 1.06 (*Nrd*).

Another choice of the bandwidth is given by the cross validation method ([15]). Here we consider the Integrated squared error (ISE) which is given by

$$ISE\left(\hat{f}_h\right) = \int \hat{f}_h^2 - 2\int \hat{f}_h\, f + \int f^2 \tag{16}$$

Observe that the last term does not involve h. The least squares cross-validation is

$$LCSV(h) = \int \hat{f}_h^2 - \frac{2}{n}\sum_{i=1}^{n} \hat{f}_h^{(-i)}(x_i) \tag{17}$$

where $\hat{f}_h^{(-i)}$ denotes the kernel estimator constructed from the data without the observation i. In [17], it is proved rewriting

$$\int \hat{f}_h^2(x)\mathrm{d}x = \frac{1}{n^2 h}\sum_{i=1}^{n}\sum_{j=1}^{n} K * K\left(\frac{x_j - x_i}{h}\right) \tag{18}$$

where * denotes the convolution, that $LSCV(h)$ is an estimator of $ISE\left(\hat{f}_h\right) - \int f^2$. Moreover it is easy to verify that $E\left(LCSV(h)\right) = MISE\left(\hat{f}_h\right) - \int f^2(x)\mathrm{d}x$, thus the least squares cross-validation is also called unbiased cross-validation. We denote by h_{ucv} the value of h which minimizes $LSCV(h)$.

Finally, the plug-in method of Sheather and Jones replaces the unknown $R(f'')$ used in the optimal value of h by an estimator $R\left(\hat{f}''_{g(h)}\right)$ where g is a "pilot bandwidth" which depends on h (see [16-18] for more details).

