DENSITY ESTIMATION BY ADAPTIVE PARTITIONING AND NONPARAMETRIC MAXIMUM ENTROPY[*]

ZACH D. MERINO[†], JENNY FARMER[‡], AND DONALD JACOBS [†]

Abstract. An automated data driven method for univariate probability density estimation using a divide and conquer approach is developed as a stitching estimator (SE). The SE combines estimates from a nonparametric maximum entropy method (NMEM) that was recently developed (PLOS ONE, 13(5): e0196937, 2018). Data is divided into a number of blocks calculated on the fly using an optimized branching tree algorithm that maximizes uniformity in density within individual data blocks. Uniformity in density is a relative comparative measure between data blocks. Furthermore, the blocks containing different sample sizes are arranged in a staggered array. Density estimates for each data block using the NMEM is processed in parallel. Stitching together the probability density function (PDF) for each sub-sample of data between neighboring blocks yields a single smooth estimate over the entire sample. Stitching involves an averaging procedure over a right and left PDF with weight factors constructed from their respective cumulative distribution functions. The NMEM-SE is benchmarked against a test set of distributions with sample sizes ranging from 2^9 to 2^{20} with diverse difficulty. The test set spans across simple single mode distributions to exotic mixed distributions that combine a heavy tailed PDF with a PDF with a singularity. Estimates are assessed using the scaled quantile residual for a sample size invariant measure. The NMEM-SE achieves fast and accurate probability density estimates suitable for big-data and high throughput applications.

Key words. destiny estimation, probability density function estimation, adaptive partitioning, non-parametric, bootstrap sampling, sub-sampling, branching tree, heavy tails, divergent distributions

1. Introduction.

Being able to quickly and accurately estimate a probability density function (pdf) for univariant data is of fundamental importance to professional fields and academic research. For instance, in the field of bioinfomratics it is crucial to have methods that can provide high throughput for large data sets to aid in the elucidation of active sites, binding residues of proteins, genetic parameters for a population of genetic data, etc. [3, 12, 9, 16]. Density estimation has been an indispensable tool in the physics community for a myriad of applications. Some notable examples include determining confidence intervals in the search for the Higgs Boson, automation of cut optimization, estimating electrical characteristics of a device from experimental results, and estimating celestial object's orbital characteristics with as small of samples as can be achieved [4, 14, 15]. Among the example applications given there are numerous other areas where density estimation is of great importance, such as, damage detection in engineering [17], isotope analysis in archaeology [2], and econometric data analysis in economics [5].

The applications for density estimators exemplified hitherto ensure that the use of a density estimator is of paramount importance in many areas of analysis, however a continuing issue is that the method of estimation and means of implementation is left to the users discretion. If the annalist has a wide range of experience the form of estimation can be picked with a certain confidence that the method will accurately represent the sample of data under consideration. Although, if large data sets of different types are being analyzed then this would be an unreasonable approach with insurmountable edge cases needing consideration. When there is no knowledge a priori of the general trends that may be expected in the data then the annalist must still

[*]Submitted to the editors 7/14/2020.

[†]Department of Physics and Optical Science, University of North Carolina Charlotte, Charlotte, NC (zmerino@uncc.edu, djacobs1@uncc.edu).

[‡]University of North Carolina Charlotte, Charlotte, NC (jfarmer@carolina.rr.com).

choose an estimator, but with the use of some other ideally objective criteria; this can introduce varying levels of bias into the estimated densities. There are many methods for pdf estimation all of which contain their own advantages and disadvantages.

There are methods of density estimation that contain a set of parameters that are used to construct a parameter space where, with the aid of some specific metric, the parameters are estimated. However, in many cases the optimal parameter set needed is unknown or perhaps can never be know, therefore it is advantageous to use a non-parametric estimation method that can adapt the parameter set size based on specific criterion to approximate the correct density distribution for the underlying data. Previously, a robust and efficient estimator was developed, termed the non-parametric maximum entropy method (NMEM) [10, 8], which uses a method of funnel diffusion to explore and adapt the parameter set to accurately find the estimate density, $\hat{f}(x)$, for a given sample of univarient data.

The NMEM estimator method has shown to be robust and computationally efficient for a large number of common distribution types. However, as is the case when attempting to create a general estimator for a broad range of data sets, the method has its limitations. As with the majority of non-parametric pdf estimators in common use the NMEM estimator has difficulty with highly divergent and extremely heavy tailed distributions. To improve $\hat{f}(x)$ for theses difficult cases a new non-parametric density estimator has been developed that has been coined the stitch estimator (SE). The SE extends the capabilities of the NMEM by using it as an underlying sub-routine. The SE was appropriately named since the data sample is partitioned into sub-samples, the NMEM estimator is applied to find a $\hat{f}_{sub}(x)$ per sub-sample, and then the $\hat{f}_{sub}(x)$ are then stitched together using a weighted average approach. The partitioning of the sample into sub-samples is akin to how a histogram partitions the data sample into bins to generate a density estimate, although the SE contains an actual density estimate rather than a sum of data points per bin.

The NMEM estimator was chosen as the underlying estimator since it has already proven to be efficient and robust with a large range of density distribution styles, but most importantly the NMEM has characteristics that make it ideal to be used with a subroutine developed to adaptability partition the sample data. The code for the NMEM estimator was initially written in C++, but has since been wrapped into Matlab R2018a [6, 7, 13]. The choice to apply NMEM to partitioned sub-samples in principle seemed straight forward, but it is in fact not the case. The method in which the sample is partition vastly influences the quality of estimates produced for various samples from density distributions. For example, a partition sub-routine may produce good results for divergent data sets but absolutely fail to produce estimates for heavily tailed data sets. The SE was developed to extend the capabilities of NMEM to handle more exotic or difficult data sets for the purpose of high throughput/big data applications. For this reason, the method in which the sample is partitioned is crucial to ensure that the SE could handle data sets that contain a vastly variant underlying density structure without the need of a apriori knowledge.

By applying NMEM to the partitioned sample gives the added benefit that the SE code was able to be multithreaded which enabled the speed of the NMEM when finding $\hat{f}_{sub}(x)$ in parallel to be utilized. Having an efficient estimator and by making SE a multithreaded script allowed for the SE to increase the range of distributions that can be handled while maintaining the original speed of the NMEM estimator. Depending on the availability of computational resources the SE can surpass the NMEM estimator in speed.

2. Stitch Estimator.

The SE estimator is able to be so robust by using a design paradigm in computer science commonly referred to as "divide and conquer", by which a difficult problem is recursively partitioned into easier to solve sub-problems. In this case the pdf estimation of a whole sample becomes the difficult problem and partitioning the sample into easy to process sub-samples becomes the easy sub-problem. The sub-samples that are generated by the SE algorithm are chosen in such a way to be easily handled by the NMEM estimator. Doing this enables the sub-problems to be solved in parallel by applying the original NMEM algorithm to a smaller input sample; this enables the $\hat{f}_{sub}(x)$ to be computed substantially faster than the NMEM applied to the original sample.

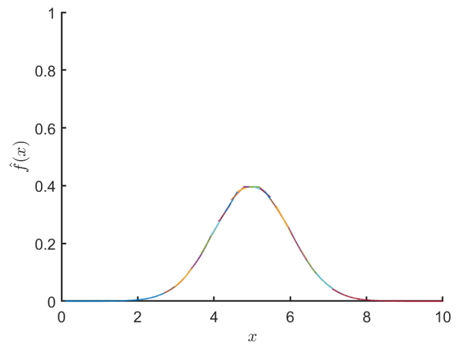
2.1. Sub-sample Definition.

$$n = \lceil cN \rceil \tag{2.1}$$

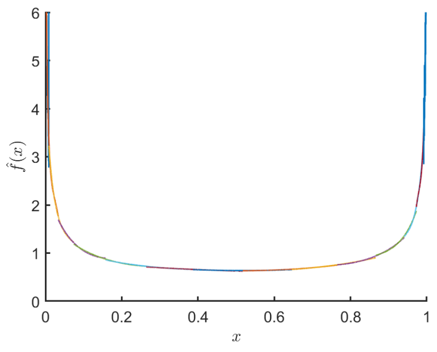
Initially the approach for defining the partitioned sample into sub-samples was simply to divide the sample as symmetrically as possible. This was achieved by requiring the size per sub-sample to be proportional to the sample size, as shown in equation 2.1, where c is a percentage coefficient of the sample size. The subroutine which employs the partitioning of equal sized sub-samples, starts by defining the boundaries of two sub-samples at the center of the sorted sample then proceeds to move towards the exterior of the sample creating sub-samples of the size n. Once sub-samples of size n are no longer able to be constructed, the subroutine defines the size of the exterior sub-samples to be the data points that are left. It is important to note that there was a minimum sub-sample size conditioned implement to ensure that no block would have insufficient information for the NMEM estimator; this was a subjective user defined parameter. The benefit to this approach was in its simplicity to implement and straight forward analogy to defining bins in a non-adaptive histogram approach. However, there were two apparent draw backs 1) What is the minimum amount of information per sub-sample that should be enforced to adequately employee the NMEM estimator? 2) What is the appropriate choice of c that would be adequate to handle general samples from varying kinds of data sets? It was discovered that this approach could not be used for partitioning the sample since different data sets required various c's to give an improved $\hat{f}(x)$ for the SE over the NMEM estimator. Undoubtedly, it was not always a benefit to evenly partition the sample for highly divergent or heavy tailed data sets in the same manor. After consideration, a new method for block definition was developed to eliminate the subjective or sample set dependent choices previously mention. This new method has been given the name: optimal branching tree (OBT) algorithm.

2.2. Create secondary sub-sample set.

Once the initial sub-samples sizes have been established a secondary set of sub-samples is created to further improve $\hat{f}(x)$. The second set of sub-samples is created to overlap the boundaries of the first sub-sample set with the second set of sub-samples' boundaries being located at the mean data point position of the first set. Figure 1 shows the resultant estimates for a Norma(5,1) and Beta(0.5,0.5) after stitching all the $\hat{f}_{sub}(x)$ together. Establishing the second set of sub-samples helps ensure more accurate predictions of $\hat{f}(x)$ are constructed from the stitching of $\hat{f}_k(x)$ $\forall$ k sub-samples by reducing errors near the exterior of the sub-sample . This error would be introduced since all $\hat{f}_{sub}(x)$ are generated with a boundary constraint; interior $\hat{f}_{sub}(x)$ are bounded on both ends while the two exterior $\hat{f}_{sub}(x)$ are only bounded on the interior edge.

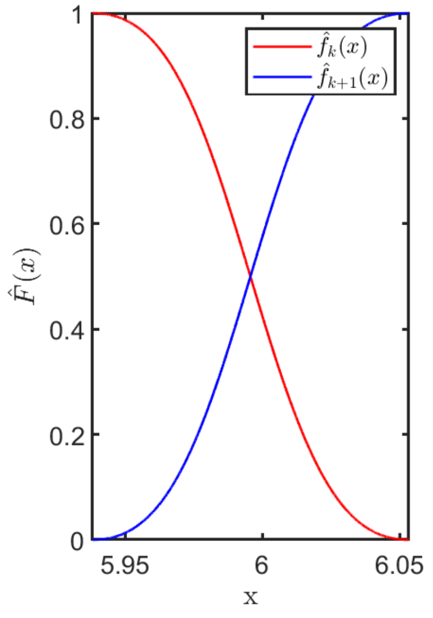


(a)

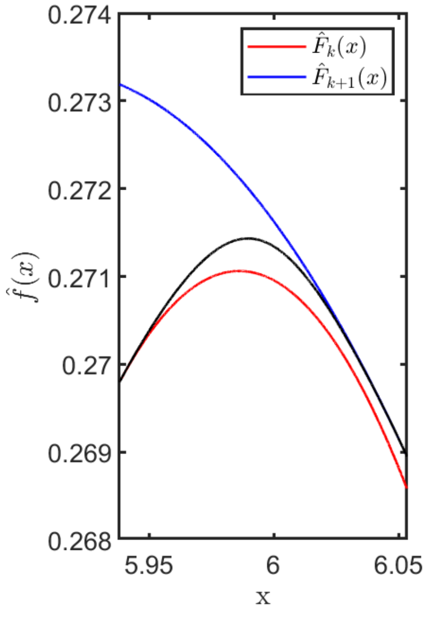


(b)

Fig. 1: All transformed $\hat{f}_{sub}(x)$ are shown after being stitched together for a (a) Normal(5,1) distribution (b) Beta(0.5,0.5) distribution



(a)



(b)

Fig. 2: (a) Shows the cdf etimates, $\hat{F}(x)$, for the kth and k+1 sub-sample's overlap region. (b) Displays the pdf estimates, $\hat{f}(x)$, for the adjacent sub-sample, as well as, the weighted averaged estimate labeled stitched pdf.

2.3. Stitching Adjacent $\hat{f}_{sub}(x)$. The density estimates for each adjacent sub-sample are stitched together using a weighted average, defined by equation 2.2, and an example of the method is shown in figure 2. For each sub-sample the cumulative distribution function (cdf) is calculated from the density estimate then the density estimate of the overlap region, $\hat{f}_s(x)$, for the two blocks is calculated by using the cdf, $\hat{F}_k(x)$, for each kth and $kth+1$ sub-sample as weights. The weighted average presented in equation 2.2 yields a smooth curve which joins the two adjacent sub-sample estimates in the overlapping region.

$$\hat{f}_s(x) = \hat{f}_k(x)(1-\hat{F}_k(x)) + \hat{f}_{k+1}(x)\hat{F}_{k+1}(x) \tag{2.2}$$

3. Optimal branching tree.

The OBT is an adaptive algorithm which optimally partitions the sample into sub-samples where the within sub-sample pseudo variance is minimized. The OBT algorithm initially evaluates the variation in the density of the original sample, then recursively makes partition decisions based on a comparison of the parameter ξ with the sample size dependent hyper parameter Γ. Γ and ξ are defined in equations 3.2 and 3.1, where the hyper parameter set p is heuristically determined to pick the appropriate Γ for general density estimate applications. The parameter set that the OBT sub-routine uses is shown in equation 3.5. n represents the number of data points per sub-sample while R is a parameter that represents the variation in the density in a sub-sample in the range [0,1]; 0 being an infinitely sparse sub-sample and 1 being a uniformly spread sub-sample.

$$\xi = n^{(p_1-p3)}R^{p_2} \tag{3.1}$$
$$\Gamma = p_5 n^{p_4} \tag{3.2}$$
$$w = \lceil p_6 n^{p_7} \rceil \tag{3.3}$$
$$m = p_8 \tag{3.4}$$
$$p = \{1,\ 0.55,\ 1,\ 0.33,\ 2,\ 0.0625,\ 0.5,\ 40\} \tag{3.5}$$

To calculate R a sub-sample's data is initially sorted then the difference of all adjacent pairs of data points, $\Delta x = \{x_{k+1}-x_{k-1}\ \forall\ k \in \{1,\ldots,n-1\}\}$, are calculated. The set of Δx values are then sorted. A sub-sample size dependent windowing function, w, defined in equation 3.3 is then used to select data points from the set of Δx to use when determining Δx_{Min} and Δx_{Max}. As shown in equations 3.6 and 3.7, Δx_{Min} is the mean difference for the w most dense pairs of data points in n and conversely Δx_{Max} is the mean differences for the w least dense pairs of data points in n. Therefore, R is defined as the ratio of Δx_{Min} to Δx_{Max} as shown in equation 3.8.

$$\Delta x_{Min} = \frac{1}{w}\sum_{k=1}^{w-1} x_{k+1} - x_k \tag{3.6}$$

$$\Delta x_{Max} = \frac{1}{w}\sum_{k=n-w+1}^{n-1} x_{k+1} - x_k \tag{3.7}$$

$$R = \frac{\Delta x_{Min}}{\Delta x_{Max}} \tag{3.8}$$

Figure 3 shows a visual representation of how the windowing function is used when calculating Δx_{Min} or Δx_{Max} for a sub-sample. The function w is a rounded up portion of the n where p_6 and p_7 were defined to elucidate the true variation in the n and ensure the effect on R from the choice of w would be near invariant with respect

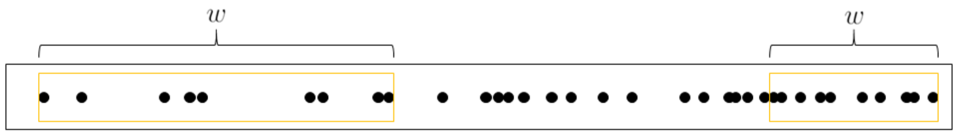


Fig. 3: A visual representation of a the distribution of Δx values for a sub-sample and how the windowing function is used to select the w most/least densely grouped data points.

to sub-sample size. If w was fixed for all sample sizes then as N is increased Δx_{Min} would increase substantially faster than Δx_{Max}, which would force R to tend towards one. This effect arises since a dense region of a distribution will generate many more data points than a sparse region when a uniform random sample is taken. If simply the variance of n was otherwise used then outlier data points in a given sub-sample would lead to a misrepresentation of the bulk behavior of the sample n.

The optimal branching tree algorithm starts by calculating ξ for the entire sample and if $\xi > \Gamma$ then the NMEM is applied to the entire sample without creating any sub-samples, otherwise a random partition, n_0, is picked and ξ is calculated for the two sub-samples created by the partitioning. Next the difference between the ξ parameter for the two blocks,

$$\Delta\xi_0 = \xi_1 - \xi_0 \tag{3.9}$$

is minimized to obtain two sub-samples that exhibit approximately the same ξ, where ξ_0 is defined as the first level of a branch for the tree, while ξ_1 is the first level and second branch of the tree. An example of how the optimized branching algorithm progresses is shown in figure 4 where a conventional tree diagram labeling scheme is employed. For every level of the tree ξ is compared to Γ and if $\xi < \Gamma$ a new branch in the tree is created, which creates two new sub-samples. Requiring the sub-samples to contain nearly uniform dense data ensures that the estimate made for each sub-sample is computationally easier for the NMEM estimator to obtain and less prone to an inferior estimate for difficult samples.

3.1. Stitching method algorithm.

The top level view of the SE algorithm is show in the algorithm 3.1. The algorithm starts by taking the sample and applying the OBT subroutine to find the optimum partitions for the first set of sub-samples. The second set of overlapping sub-samples is determined and then the set of all sub-samples ψ is stored in memory. A transformation is constructed to transform the sub-samples,S, to u for all sub-samples so that the domains fall in the range $[-1, 1]$ which aid in the efficiency and accuracy of the NMEM. Since there are $\alpha - 1$ sub-samples created for α partitions and similarly $\beta - 1$ sub-samples created for β partitions, then there are a total of $\alpha - 1 + \beta - 1 = 2\alpha - 3$ sub-samples to send to the NMEM. The NMEM takes the evaluates the transformed samples u in a parallel parfor loop then returns the pdf, $\hat{u}$, and the cdf, $\hat{U}$, for the total set of sub-samples. The inverse transform is applied to $\hat{u}$ and $\hat{U}$ prior to computing the overlap estimates $\hat{f}_{avg}(x)$. $\hat{f}_{avg}(x)$ is computed for all of the sub-sample estimates, $\hat{f}_s(x)$, using an average with the sub-samples' cdf,$\hat{F}_s(x)$, as weights. The algorithm ends by returning the ordered set of all $\hat{f}_s(x)$ and $\hat{f}_{avg}(x)$.

Algorithm 3.2 starts by taking the sample or sub-sample and using the set size $|S| = n$ to initialize the threshold parameter and starting partition set p for the sample

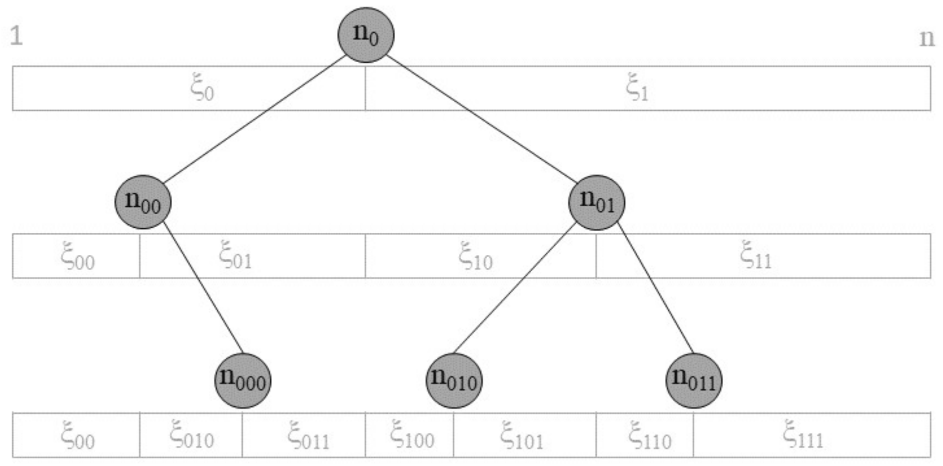


Fig. 4: Example of branching tree algorithm for a sub-sample or sample of size n. Partition one level zero,n_0, is initially chosen then $\Delta\xi$ is minimized. More levels are created and the procedure is repeated until all ξ are less than Γ.

Algorithm 3.1 Stitching method algorithm. Note $\circ$ is the Hadamard operator.

```
Input $X := \{x_1, \dots, x_N\}$ where $(x_{k+1} - x_k) > 0 \; \forall \; k \in \{1, N-1\}$
Execute $OBT(X) \to \alpha$
Define $\beta := \{\lfloor \frac{\alpha_{i+1} - \alpha_i}{2} + \frac{1}{2} \rfloor \mid i \in \alpha - 1\}$
Define $\psi := \{\alpha, \beta\}$
Define sub-samples as $S := \{X_{i+1} - X_i \; \forall \; i \in \psi\}$
Define $\nu = \{\frac{S_{i+1} + S_i}{2} \mid \; i \in \psi\}$ and $\omega = \{\frac{S_{i+1} - S_i}{2} \mid i \; \in \psi\}$
Apply transformation $T(S) := \nu^{(-1)} \circ S - \omega \to u$
for $j \in \{1, 2\alpha - 3\}$ do
  Apply $NMEM(u_j) = (\hat{u_j}, \hat{U_j})$
end for
Apply Inverse Transform $T^{-1}(\hat{u}) := \nu \circ \hat{u} + \omega \to \hat{f}_s(x)$
Apply Inverse Transform $T^{-1}(\hat{U}) := \nu \circ \hat{U} + \omega \to \hat{F}_s(x)$
for $j \in \{1, 2\alpha - 3)\}$ do
  $\hat{f}_{avg}(x)_j = \hat{F}_s(x)_{j+1} \hat{f}_s(x)_{j+1} + \hat{F}_s(x)_j \hat{f}_s(x)_j$
end for
return $\hat{f}(x) = \{\hat{f}_s(x)_1, \hat{f}_{avg}(x)_1, \dots, \hat{f}_s(x)_{2\alpha-3}, \hat{f}_{avg}(x)_{2\alpha-3}\}$
```

S. To prevent excessive partitioning a maximum number of levels to the branching tree is set. It should be noted that this is set to 40 which would mean the maximum number of potential partitions would be 2^{40}. This is simply a parameter that can be used to control the error in the algorithm if needed, but in the current form is set so large that it can be discounted without loss of generality. The initial ξ_{00} is computed and the initial number of branches, b, is initialized to 1. If $\xi_{00} > T$ then the starting partition set is returned unchanged, otherwise the sub-routine starts to search for optimal partitions. The outer for loop iterates over the number of levels in the tree while the inner for loop iterates through all of the branches currently under evaluation per tree level. A sub-sample created from branches of the current lowest level of the tree is sent to the the golden-section search (GSS) sub-routine to compute a new partition p_{new} by minimizing ξ for the two new sub-samples created within the original sub-sample under consideration. The minimum ξ for the left and right newly generated sub-samples is stored in ξ_{ij}. The minimum is taken because all newly created sub-samples must obey the inequality $\xi < T$ for the tree to continue on down the given branching path. ξ_{ij} is evaluated against T and if $\xi_{ij} < T$ the new partition is accepted otherwise the partition is rejected. After all of the branches have been evaluated for a given level they are checked against T. If $\xi_{ij} > T \; \forall \; i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, b\}$ then the sub-routine ceases to make any further levels of the tree and returns the set p, otherwise the algorithm updates the number of current branches to evaluate and continues.

Algorithm 3.2 Optomized branching tree algorithm

```
Input $S := \{s_1, \ldots, s_n\}$ where $(s_{k+1} - s_k) > 0 \; \forall \; k \in \{1, n-1\}$
Define subsample size $T = an^{-b} + c$
Define partitions p for S $p := \{1, n\}$
Define max level $m$
Calculate initial $\xi_{00} = \Xi(S)$
Initialize $b = 1$
if $\xi_{00} < T$ then
  for $i \in \{1, \ldots, m\}$ do
    for $j \in \{1, \ldots, b\}$ do
      Golden-section search $GSS(\{S_{p_j}, \ldots, S_{p_{j+1}}\}) = (\{\xi_L, \xi_R\}, p_{new})$
      $\xi_{ij} = min\{\xi_L, \xi_R\}$
      if $\xi_{ij} < T$ then
        Update $p = \{p, p_{new}\}$
      end if
    end for
    if $\xi_{ij} > T \; \forall \; i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, b\}$ then
      Break loop
    end if
    Update $b = |p| - 1$
  end for
end if
return $p$
```

Algorithm 3.3 takes a sample, S, as input and then initializes the window parameter w. The differences, Δx, are computed for S then sorted. Δx_{min} and Δx_{max} are then defined as the mean for the sets of data points in Δx from the first and last w points of the set. R is computed and then ξ is returned.

4. Results.

The SE has been evaluated against various metrics to quantify the improvement of this method compared to using the NMEM alone as well as the quality of the estimator in general. There are four general metrics that were used to evaluate SE and these are: scaled quantile residuals (SQR), statistical distance, computational cost, and failure rate, and the ability for the estimators to handle highly divergent or heavily tailed distributions is reviewed.

Algorithm 3.3 Ξ algorithm

```
Input $S := \{s_1, \ldots, s_n\}$ where $(s_{k+1} - s_k) > 0 \; \forall \; k \in \{1, n-1\}$
Define window parameter $w$
Define $\Delta x := \{s_{k+1} - s_k \; \forall \; k \in \{1, n-1\}\}$
Sort $\Delta x = \{\Delta x_{k+1} - \Delta x_k \; \forall \; k \in \{1, n-1\}\}$
Define $\Delta x_{min} := mean\{\Delta x_1, \ldots, \Delta x_\omega\}$
Define $\Delta x_{max} := mean\{\Delta x_{n-1-\omega}, \ldots, \Delta x_{n-1}\}$
Compute $R = \frac{\Delta x_{min}}{\Delta x_{max}}$
**return** $\xi = n^{(p_1 - p3)} R^{p_2}$
```

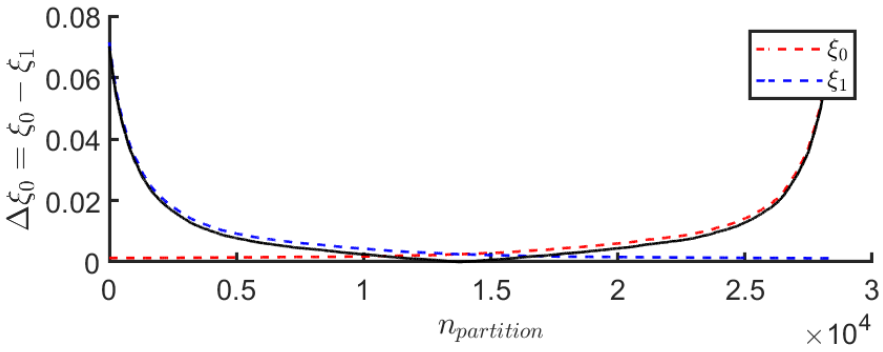


Fig. 5: Distribution type: Uniform for a sample size of 16,384. Blue/red lines indicate the left/right sub-sample's ξ. The solid black line $\Delta\xi$ is plotted as a function of the partition location over the sample.

4.1. OBT. The parameter ξ is a product of the sub-sample size and R, however in it's current form $\xi = R$. Take a sample where a single partition is made and allowed to search across the entirety of the sample's domain. As the one sub-sample size increases more data points are introduced into the ξ calculation which tends Δx_{Min} to be approximately constant or decrease and Δx_{Max} to stay constant or increase. This causes ξ for the left sub-sample to decrease when the partition is swept left to right and similarly for the right sub-sample when the partition is swept right to left. The competing characteristics of Δx_{Min} and Δx_{Max} help ensure that the $\Delta\xi$ has a single global minimum, which affords the SE to employee the efficient Fibonacci search algorithm. This search algorithm also is commonly referred to as the GSS algorithm. The use of the Fibonacci search algorithm proved useful when locating the minimum for $\Delta\xi$ since this algorithm is the most efficient method for finding a global minimum on a uni-modal function [11, 1]. Figure 5 shows the behavior of $\Delta\xi$ when two sub-samples are taken from a uniform distributions with original sample size of 16,384 and the sub-samples obey the equation $N = N_{left} + N_{right}$, where $N_{right} = N - N_{left}$.

The uni-modal function $\Delta\xi$ is optimized using the Fibonacci search for all sub-samples and if $\Delta\xi$ for a given sub-sample is smaller than the cutoff threshold then the sub-sample is partitioned and the process of minimizing $\Delta\xi$ for the new sub-samples continues. Figure 6a shows the progression of ξ for all sub-samples by the tree level. As the sub-samples progressively shrink, ξ increases in the attempt to approach 1 but as more sub-samples are generated the individual ξ values tend to diverge from

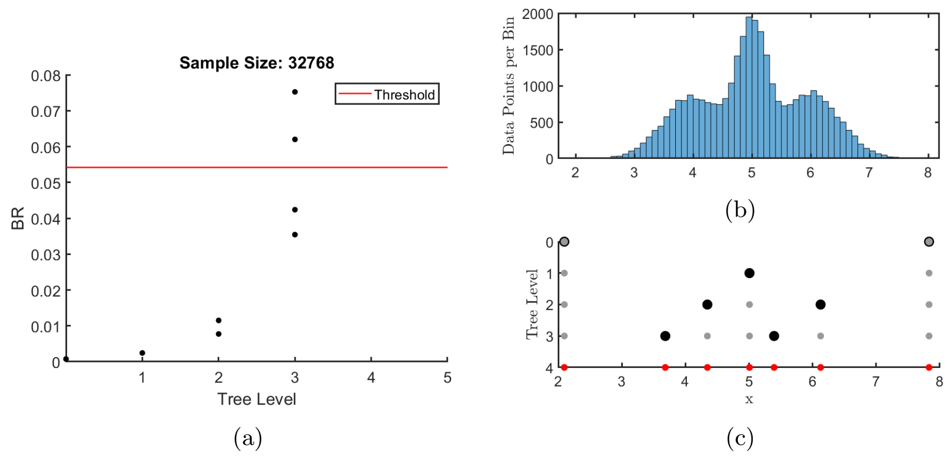


Fig. 6: (a) Shows the ξ value per level for each sub-sample that is created. (b) The histogram plot for the original sample (c) Shows the sub-sample partitions made for each individual level. The grey points show the initial range of the sample, the black points display the newly created partition for that level, and the red dots show the final sub-sample partitions.

one another. This is guaranteed since as the sub-sample size decreases the pseudo density estimate for said sub-sample will be sampling a progressively smaller region of the over all sample. This inevitably leads to the divergence in ξ for all sub-samples. Once ξ for all sub-samples fall above the designated cutoff threshold then the OBT sub-routine ceases to execute and returns the partition locations. This optimization and conditional branching sub-routine leads to an unbalanced binary tree structure which is shown in figure 6c. Figure 6b shows the distribution of the data with a generic histogram plot to help visualize where the partitions are made for each level of the tree. The initial partition is made where the difference in the sample's density is greatest; approximately at $x = 5$. The next group of partitions which are made are close to the midpoint between the exterior modes and troughs of the tri-modal distribution. These partitions are chosen here since the OBT algorithm minimizes all $\Delta\xi$ for the four sub-samples generated in level 2. It goes without stating that these partitions will not fall in the ideal position for the theoretical distribution due to random sampling fluctuations.

The threshold T displayed in 6a is sample size dependent and determined using the equation $T = aN^{-b} + c$ where N is the sample size and a, b, c are constant coefficients. This equation was determined by evaluating the average starting ξ values over a range of distributions and samples. Figure shows the mean ξ distributions for several distributions with samples ranging from $2^{10} - 2^{20}$. A fit line was computed for each distribution and found to be a decaying exponential function. From this information an appropriate exponential function was empirically determined to exhibit appropriate behavior with the distribution test set.

4.2. Comparing SE and NMEM. Comparing the SQR plots for both estimators shows that both produce good estimates from the sample data, however, there are less random fluctuations for the SE than the NMEM. This was predicted, since the NMEM is applied to the original sample, where as, the SE is applying the NMEM to

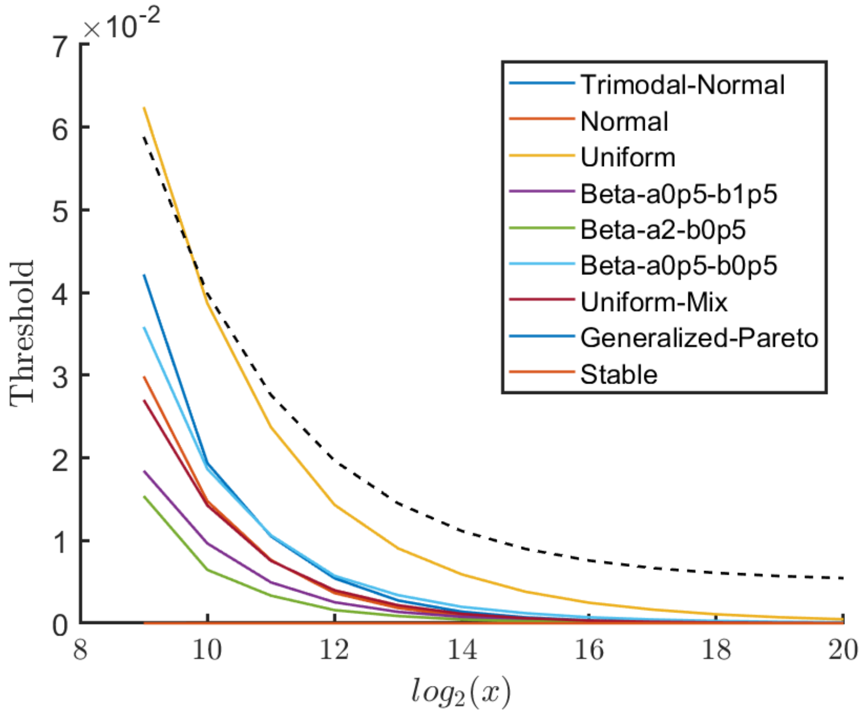


Fig. 7: The figure displays the mean ξ values for various distributions over a range of sample sizes. The dashed black line is the cut off threshold $T = aN^{-b} + c$ where $a = 2.6741$, $b = 0.626$, and $c = 0.005$.

sub-samples where the sub-sample sizes vary between trials. In figure 8 estimates from both estimators are displayed for samples of size 512 and 32,768 for a Normal(5,1). The left had side sub-figures show results from the SE and the right hand side results from NMEM. Visually the PDFs from both estimators show good results for many trial estimates, however, the SQR plots show that the SE tends to over fit compared to the NMEM. Although the SE is exhibiting over fitting behavior in the SQR plots the estimates generated still fall into the 99% confidence interval that the estimate accurately represents the under lying distribution from the sample data.

The Kullback-Leibler divergence test was utilized to evaluate the information retention between the two estimators to quantitatively evaluate the quality for the estimates produced for various distributions and sample sizes. Figure 9 shows box and whisker plots for the various samples and distributions considered over many trials. Figures 9a-9b display the KL divergence for multiple trials over range of samples from a normal distribution. These box and whisker plots show that for smaller sample sizes the SE retains less information about the underlying distribution that the NMEM, however, as the sample sizes increase the information retention converges between estimators. The SE estimator tends to have slightly smaller KL divergence values for sample sizes in the range $2^{12} - 2^{20}$. Similarly for figures 9c-9d the KL divergence values converge between estimators as the sample size increases for a tri-modal distribution. For the tri-modal distribution the KL divergence values are comparable to one another. Although the the KL divergence value between estimators are

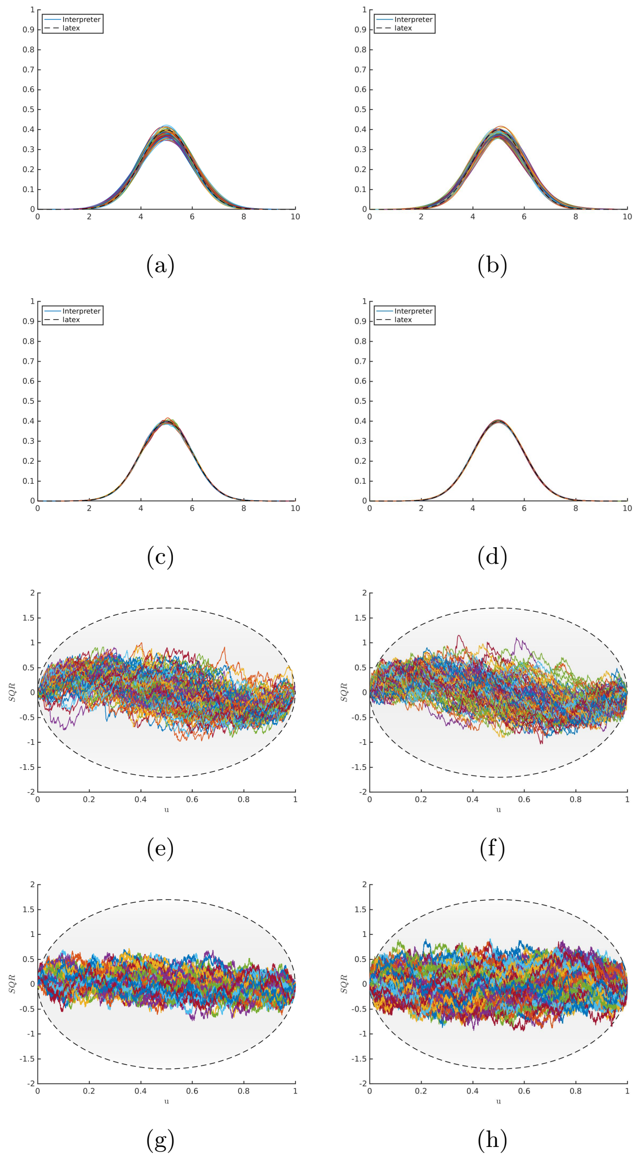
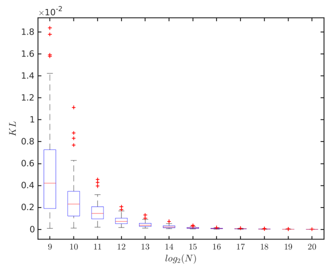


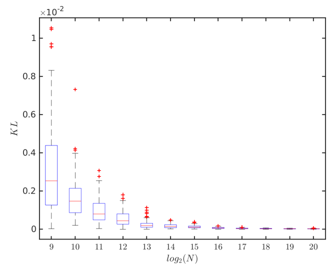
Fig. 8: 100 estimates from a Normal distribution centered at $x = 5$. (a) SE for $N = 512$. (b) NMEM $\hat{f}(x)$ for $N = 512$. (c) SE $\hat{f}(x)$ for $N = 32,768$. (d) SE $\hat{f}(x)$ for $N = 32,768$. (e) SE $SQR(u)$ for $N = 512$. (f) NMEM $SQR(u)$ for $N = 512$. (g) SE $SQR(u)$ for $N = 32,768$. (h) SE $SQR(u)$ for $N = 32,768$.

comparable for less exotic distributions, the trend in the behavior diverges for exotic distributions. Figures 9e-9f show that the SE KL divergence values for a heavy tailed generalize-Pareto distribution are notably smaller than the NMEM for increasingly larger samples. Similarly, for a highly divergent Beta distribution the KL divergence values for the SE are significantly lower than the NMEM.

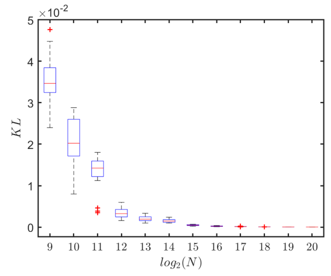
Figure 10 shows comparisons for the SE and NMEM when applied to a highly



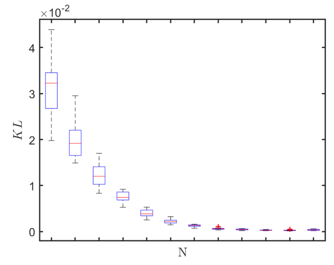
(a)



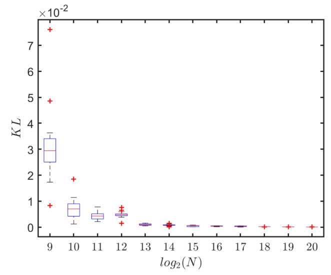
(b)



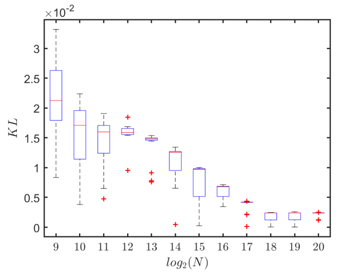
(c)



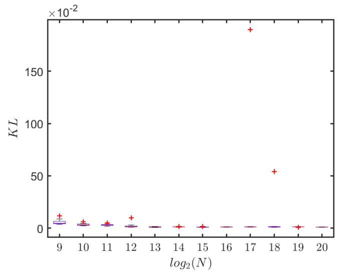
(d)



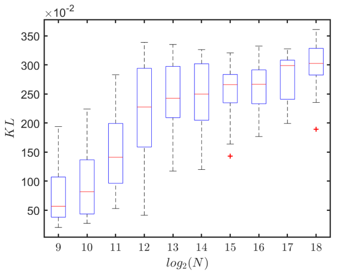
(e)



(f)



(g)



(h)

Fig. 9: KL divergence values from samples in the range from $2^{10} - 2^{20}$ for 100 estimates (a) SE for Normal(5,1) distribution. (b) NMEM for Normal(5,1) distribution. (c) SE for Tri-modal Normal distribution. (d) NMEM for tri-modal normal distribution. (e) SE for Generalized-Pareto(2,1,0) distribution. (f) NMEM for Generalized-Pareto(2,1,0) distribution. (g) SE for Beta(0.5,1.5) distribution. (h) NMEM for Beta(0.5,1.5) distribution.

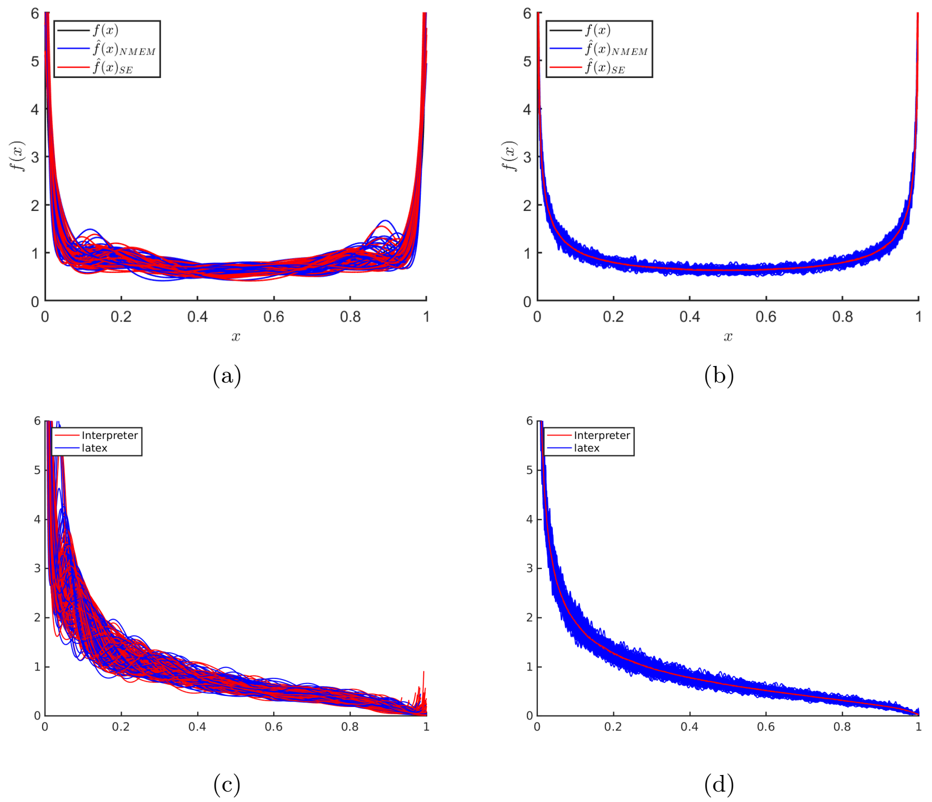


Fig. 10: Compares how the SE and NMEM handle highly divergent distributions for sample sizes of 1,024 and 1,048,576 for: (a)-(b) Beta(0.5,0.5) distribution and (c)-(d) Beta(0.5,1.5) distribution. The black line is the theoretical distribution.

divergent distributions. For exotic distributions with highly divergent regions, such as 10a-10d, the NMEM produces estimates that have little to no bias for larger samples, but introduces numerical errors (oscillations) due to the need for many yet finite Lagrange multipliers in the series expansion. The SE avoids the need for so many series expansion terms by generating sub-samples which have approximately uniform density.

Figure 11 shows comparisons for the SE and NMEM when applied to a heavily tailed distributions. Figures 11a-11d display the behavior of the the cumulative probability distribution functions, $\hat{F}(x)$, for the two estimators for heavily tailed distributions plotted over a modified logarithmic scale. When looking at figure 11a it is observed that two estimators agree well with one another when $ln(x)$ falls in the range $[-6, -1]$ but the estimates from the estimators start to diverge from one another as $ln(x)$ increases. The same behavior is observed in figure 11b when the sample size increases and shows that the SE estimator better handles fitting to the distributions tails out to $x \approx 150$. Figures 11c shows that for small sample sizes the SE shows a greater variability for the quality of the estimates when compared to the NMEM. The SE estimator has several estimates that are not as well fitted to the theoretical distribution as the NMEM. Although there exist poorly fitted estimates, in general the

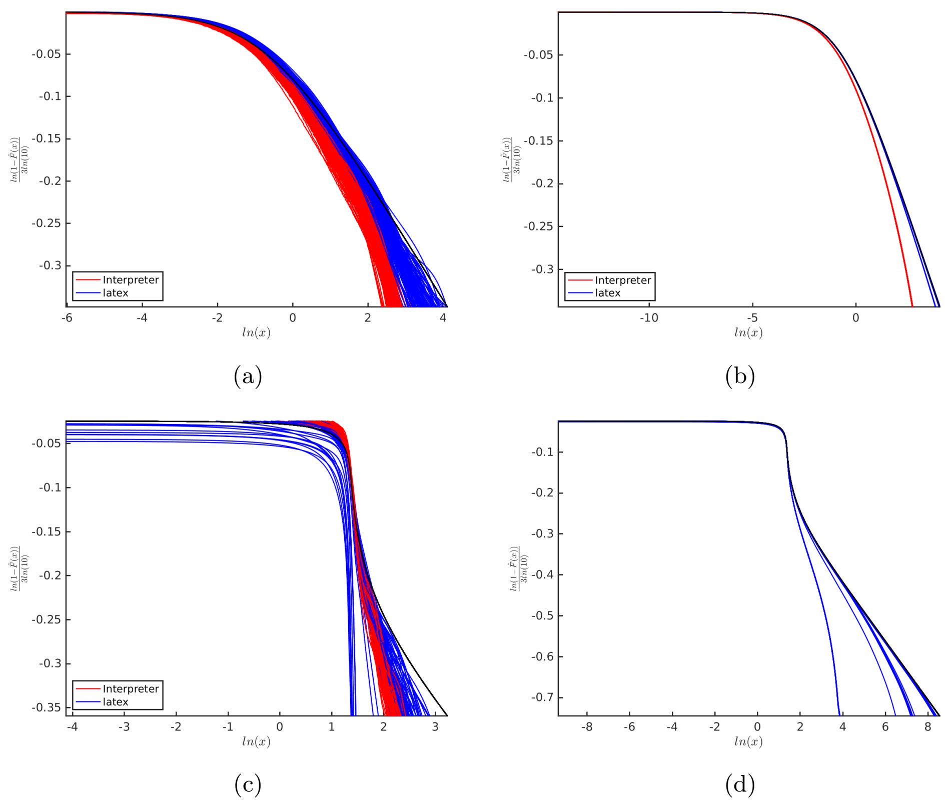


Fig. 11: Compares how the SE and NMEM handle heavily tailed distributions for sample sizes of 1,024 and 1,048,576 for: (a)-(b) Generalize-Pareto(2,1,0) distribution and (c)-(d) Stable(0.5,0.05,1,4) distribution. The black line is the theoretical distribution. Note: (c)-(d) only compare the tails of $\hat{f}_{SE}(x)$ and $\hat{f}_{NMEM}(x)$ for the domain $\{x \mid x \in (0, \infty)\}$.

SE better fits to the theoretical distribution about $ln(x) = 1$ compared to the NMEM and improves the fitting to the tails of the distribution. As seen in figure 11d the NMEM fails to produce estimates under default settings where the maximum number of Lagrange multipliers is limited to 200. These failures could be resolved by increasing the maximum Lagrange multiplier limit. This would allow for the large number of Lagrange multipliers required to fit the central behavior of Stable(0.5,0.05,1,4) distribution as well as the extremely heavy tails. The SE reduces the need for as many Lagrange multipliers which prevents the NMEM from failing in a given sub-sample's domain. The variability in the estimate quality improved significantly comparing to smaller samples around the central region of the distribution, but there still exist a greater variation in the quality of the estimate towards the tails.

Figure 12 displays the failure rates for the two estimators for various exotic distributions. Here it is seen that the SE has a smaller error rate for these various distributions compared to the NMEM. The failures for the NMEM occur for the distributions with large tails or divergent regions. The failures start to become more frequent as the sample size increases which in effect leads to more outlier like data points for heavy

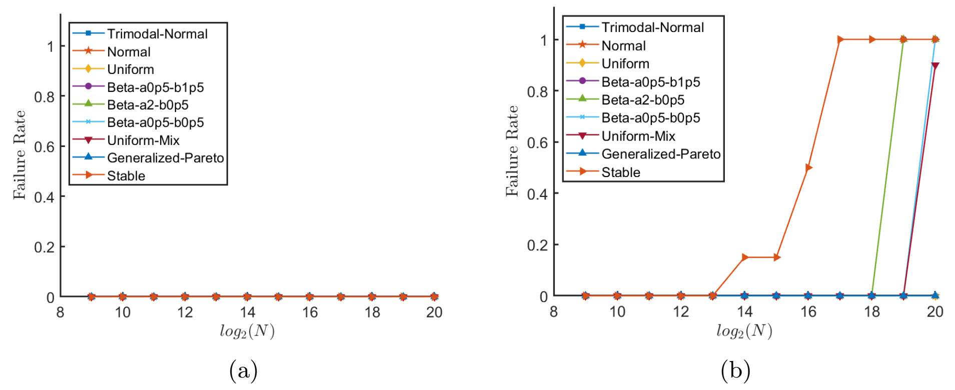


Fig. 12: Displays the failure rate for (a) the SE and (b) NMEM for a variety of distributions over a range of samples for 100 trials.

tailed distributions of causes the NMEM to have many more Lagrange multipliers to fit to the divergent curve. The SE is able to avoid these difficulties by partitioning the sample for the NMEM where by effectively removing what might be considered outlier data points or reduce the need to have so many Lagrange multipliers.

4.3. Computation time. To evaluate the efficiency of the SE and NMEM the CPU times were obtained for a variety of sample sizes and distributions. The time of computation was calculated using the cputime function in Matlab R2018a. The cputime function returns the elapsed CPU time which sums across all threads. Tracking the CPU time gives information on the efficiency for the parallel threads. For the computational efficiency analysis of the SE method 20 threads were used. Figure 13a

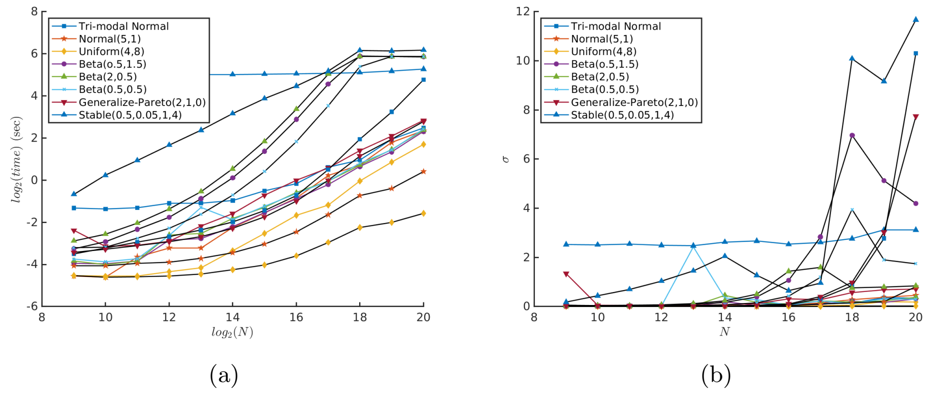


Fig. 13: Colored lines represent the SE while black lines represent the NMEM (a) Mean CPU times for various distributions (b) Standard deviation for the CPU times for various distributions

displays the mean CPU time for a given sample size and distribution while figure 13a shows the corresponding standard deviations for the mean CPU times. Figure 13a shows that the SE approaches a linear log CPU time over this range of samples for all of the distributions while the NMEM exhibits the same behavior for less exotic distributions. However, for more exotic and difficult distributions this behavior starts to break down and exhibit a logarithmic exponential growth with samples size. The SE method may be sped up significantly from the analysis here depending on the users available computer hardware and ability to add more threads to compute pdf estimates per sub-sample.

It is shown in figure 13b that the standard deviation of the computations times between estimators are similar for smaller samples but as sample sizes increase the standard deviation for the NMEM has a trend to increases compared to the SE. The increase in standard deviation is in part due to failed estimates for the NMEM, which would take a longer time searching the parameter space prior to failing. A more predominant cause for the increased variation is due to the funnel diffusion search on average requiring more time to search for the optimal number of Lagrange multipliers. However, due to the random search method of the funnel diffusion search the NMEM can start much closer to the optimal solution in the parameter space.

5. Conclusions.

Implementing a divide and conquer methodology to the problem of density estimation, via the SE, has proven to be generally effective, in particular, handling exotic distributions that are highly divergent or have infinite variance. The adaptive partitioning OBT algorithm greatly extended the capabilities of the SE to a larger class of distribution types and improved the ability to accurately the underlying data with the user having no a priori knowledge. With these two approaches to the estimation problem this body of work was able to extend the the NMEM's ability to accurately produce density estimates. By partitioning the sample into more manageable sub-samples the SE was designed to be more efficient than the NMEM with the option to improve the speed even further given the computational resources. The OBT algorithm is a novel approach to representing information about a sample when searching for regions that would be manageable for many density estimation methods. Further more, the SE's method of stitching sub-estimates together greatly retains information about the underlying distribution while not over fitting to the given sample.

While the SE does use several empirically derived parameters, this should not be presumed to be a point of user subjection. The general functional form of the threshold Γ was determined through data driven simulation and then was refined empirically through simulation. However, the parameters for the functional form could likely be derived analytically for various distributions and the threshold Γ could be better chosen through employing a machine learning model.

REFERENCES

[1] M. Avriel and D. J. Wilde, *Optimality proof for the symmetric Fibonacci search technique*, The Fibonacci Quarterly. Official Organ of the Fibonacci Association, 4 (1966), pp. 265–269.

[2] M. J. Baxter, C. C. Beardah, and S. Westwood, *Sample size and related issues in the analysis of lead isotope data*, Journal of Archaeological Science, 27 (2000), pp. 973–980.

[3] A. P. Boyle, J. Guinney, G. E. Crawford, and T. S. Furey, *F-Seq: a feature density estimator for high-throughput sequence tags*, Bioinformatics, 24 (2008), pp. 2537–2538.

[4] K. Cranmer, *Kernel estimation in high-energy physics*, Computer Physics Communications, 136 (2001), pp. 198 – 207.

[5] J. DiNardo, N. M. Fortin, and T. Lemieux, *Labor market institutions and the distribution of wages, 1973-1992: a semiparametric approach*, Econometrica, 64 (1996), p. 1001.

[6] J. Farmer, *PDFAnalyze*, Mar. 2020.

[7] ———, *PDFAnalyze for Probability Density Estimation*, Mar. 2020.

[8] J. Farmer and D. Jacobs, *High throughput nonparametric probability density estimation.(research article)(report)*, PLoS ONE, 13 (2018), p. e0196937.

[9] J. D. Fischer, C. E. Mayer, and J. Soding, *Prediction of protein functional residues from sequence by probability density estimation*, Bioinformatics, 24 (2008), pp. 613–620.

[10] D. J. Jacobs, *Best probability density function for random sampled data*, Entropy (Basel, Switzerland), 11 (2009), p. 1001.

[11] J. Kiefer, *Sequential minimax search for a maximum*, Proceedings of the American Mathematical Society, 4 (1953), pp. 502–506.

[12] M. K. Kuhner, *LAMARC 2.0: maximum likelihood and Bayesian estimation of population parameters*, Bioinformatics, 22 (2006), pp. 768–770.

[13] MATLAB, *9.7.0.1190202 (R2019a)*, The MathWorks Inc., Natick, Massachusetts, 2018.

[14] L. Raczyński, W. Wiślicki, W. Krzemień, P. Kowalski, D. Alfs, T. Bednarski, P. Białas, C. Curceanu, E. Czerwiński, K. Dulski, A. Gajos, B. Głowacz, M. Gorgol, B. Hiesmayr, B. Jasińska, D. Kamińska, G. Korcyl, T. Kozik, N. Krawczyk, E. Kubicz, M. Mohammed, M. Pawlik-Niedźwiecka, S. Niedźwiecki, M. Pałka, Z. Rudy, O. Rundel, N. G. Sharma, M. Silarski, J. Smyrski, A. Strzelecki, A. Wieczorek, B. Zgardzińska, M. Zieliński, and P. Moskal, *Calculation of the time resolution of the j-pet tomograph using kernel density estimation*, Physics in Medicine and Biology, 62 (2017), pp. 5076–5097.

[15] U. Von Toussaint, *Bayesian inference in physics*, Reviews of Modern Physics, 83 (2011), pp. 943–999.

[16] M. F. Wanderley, V. Gardeux, R. Natowicz, and A. Braga, *Ga-kde-bayes: An evolutionary wrapper method based on non-parametric density estimation applied to bioinformatics problems*, 04 2013.

[17] L. Yu and Z. Su, *Application of kernel density estimation in lamb wave-based damage detection*, Mathematical Problems in Engineering, 2012 (2012).