# Fast Bootstrapping Nonparametric Maximum Likelihood for Latent Mixture Models

Shijie Wang, Minsuk Shin, and Ray Bai

***Abstract*—Estimating the mixing density of a latent mixture model is an important task in signal processing. Nonparametric maximum likelihood estimation is one popular approach to this problem. If the latent variable distribution is assumed to be continuous, then bootstrapping can be used to approximate it. However, traditional bootstrapping requires repeated evaluations on resampled data and is not scalable. In this letter, we construct a generative process to rapidly produce nonparametric maximum likelihood bootstrap estimates. Our method requires only a single evaluation of a novel two-stage optimization algorithm. Simulations and real data analyses demonstrate that our procedure accurately estimates the mixing density with little computational cost even when there are a hundred thousand observations.**

***Index Terms*—Bootstrap/resampling, deep neural network, generative process, mixing density estimation, nonparametric maximum likelihood estimation, two-stage algorithm.**

## I. Introduction

NONPARAMETRIC maximum likelihood estimation (*NPMLE*) is a popular methodology in signal processing applications such as pattern recognition [1], [2], channel estimation [3], signal recovery [4], and positron emission tomography [5], [6]. NPMLE also has applications in empirical Bayes and regression modeling [7], [8], [9]. Suppose that we observe $\boldsymbol{y} = (y_1, \ldots, y_n)$ where

$$y_i \mid \theta_i \sim f(y_i \mid \theta_i),\ \theta_i \sim \pi(\theta),\ i = 1, \ldots, n, \tag{1}$$

and the density $f$ is known but $\pi$ is unknown. NPMLE aims to estimate the mixing density (or the prior) $\pi$.

The NPMLE estimator $\widehat{\pi}$ in the latent mixture model (1) [10], [11] was introduced by [12] and solves the optimization,

$$\begin{aligned}\widehat{\pi} &= \underset{\pi \in \Pi}{\operatorname{argmax}} \sum_{i=1}^{n} \log[\mathbb{E}_{\pi}\{f(y_i \mid \theta_i)\}] \\ &= \underset{\pi \in \Pi}{\operatorname{argmax}} \sum_{i=1}^{n} \log \left\{ \int_{\Theta} f(y_i \mid \theta_i) d\pi(\theta_i) \right\}, \end{aligned} \tag{2}$$

where $\Pi$ denotes the family of all probability distributions on the parameter space $\Theta$. Under mild regularity conditions, [13] showed that the solution $\widehat{\pi}$ of (2) exists and is unique, and even if the true $\pi$ is continuous, $\widehat{\pi}$ is almost surely discrete with a support of at most $n$ points. Exploiting this property, there exist many algorithms to solve (2), including the EM algorithm [14], [15] and convex optimizers [16], [17].

Nevertheless, a discrete NPMLE estimator $\widehat{\pi}$ is unsatisfactory when the latent distribution is reasonably assumed to be continuous [5], [18]. As a result, smoothed variants of NPMLE have been proposed [2], [5], [7], [19]. However, these methods typically require careful tuning of smoothing parameters such as bandwidth [2], [19], roughness penalty term [5], [7], and/or spline degrees of freedom [2], [7].

As an alternative to smoothing, bootstrapping has been shown to be an effective way to simulate from both prior and posterior continuous densities [20], [21], [22], [23]. However, bootstrapping is seldomly employed for NPMLE. This may be because the bootstrap traditionally involves resampling the data with replacement and repeatedly optimizing weighted objective functions. In the context of NPMLE, the standard bootstrap procedure requires one to repetitively optimize

$$\widehat{\pi}^{(b)} = \underset{\pi}{\operatorname{argmax}} \sum_{i=1}^{n} w_i^{(b)} \log[\mathbb{E}_{\pi}\{f(y_i \mid \theta_i)\}],\ \boldsymbol{w}^{(b)} \overset{\text{iid}}{\sim} \mathbb{P}_{\boldsymbol{w}}, \tag{3}$$

for some probability measure $\mathbb{P}_{\boldsymbol{w}}$. For example, if $\boldsymbol{w} \sim \text{Multinomial}(n, \mathbb{1}_n/n)$, where $\mathbb{1}_n$ denotes an $n$-dimensional one vector, we have the nonparametric bootstrap [24]. If $\boldsymbol{w} \sim n \times \text{Dirichlet}(n, \mathbb{1}_n)$, we have the weighted likelihood bootstrap [22]. Repeatedly solving (3) can be time-consuming.

To resolve this computational bottleneck and broaden the applicability of bootstrapping, we build upon the generative bootstrap sampling (GBS) approach of [25]. We introduce a new generative framework called Generative Bootstrapping for NPMLE (*GB-NPMLE*). Our contributions are as follows:

1) We propose GB-NPMLE for estimating a continuous prior density via bootstrapping. Different from GBS which only requires weights $\boldsymbol{w}$ as inputs [25], GB-NPMLE needs *additional* noise inputs $\boldsymbol{z}$ to capture the latent representation of $\pi$ in (1).
2) To optimize GB-NPMLE, we introduce a novel two-stage algorithm. In contrast to traditional bootstrapping, GB-NPMLE requires only a *single* evaluation of this two-stage algorithm instead of repetitive evaluations of (3).

Manuscript received 7 January 2024; revised 28 February 2024; accepted 6 March 2024. Date of publication 12 March 2024; date of current version 28 March 2024. The work of Shijie Wang and Ray Bai was supported by the U.S. National Science Foundation under Grant DMS-2015528. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Jeremy Thomas Reed. *(Corresponding author: Ray Bai.)*

Shijie Wang and Ray Bai are with the Department of Statistics, University of South Carolina, Columbia, SC 29208 USA (e-mail: shijiew@email.sc.edu; rbai@mailbox.sc.edu).

Minsuk Shin is with the Gauss Labs, Palo Alto, CA 94301 USA (e-mail: minsuk000@gmail.com).

Supplementary materials for this article are available online. This paper supersedes a previously circulated technical report by S. Wang and M. Shin (https://arxiv.org/pdf/2006.00767v2.pdf).

Digital Object Identifier 10.1109/LSP.2024.3376205

## II. Proposed Method

### A. GB-NPMLE

Recent developments in generative learning such as variational autoencoders (VAEs) [26] and generative adversarial networks (GANs) [27], [28], [29] provide another perspective for NPMLE. Instead of estimating $\pi$, we can construct a generator $T$ that matches to the target density $\pi$, i.e. $T(\boldsymbol{z}) \sim \pi$ where $\boldsymbol{z} \in \mathbb{R}^q$ follows a known distribution such as a Gaussian or a uniform. We then reformulate NPMLE optimization (2) via a generative framework,

$$\widehat{T} = \text{argmax}_{T \in \mathcal{T}} \sum_{i=1}^{n} \log[\mathbb{E}_{\boldsymbol{z}}\{f(y_i \mid T(\boldsymbol{z})\}], \tag{4}$$

where $\boldsymbol{z} \in \mathbb{R}^q$ follows (for example) a standard uniform. [30] considered (4) where the generator $T$ was constructed by a VAE [26] with a Langevin bias-correction algorithm [31]. Unfortunately, both the classical NPMLE $\widehat{\pi}$ in (2) and the generative NPMLE $\widehat{T}$ in (4) result in discrete solutions [13]. This discreteness is less attractive when the true latent $\pi$ is continuous, as in many signal processing applications [5], [6].

To efficiently obtain a bootstrapped NPMLE distribution as a smooth estimator for $\pi$ in (1), we introduce GB-NPMLE. Our framework (approximately) generates NPMLE bootstrap estimates via a generator function. The generator $G := G(\boldsymbol{w}, \boldsymbol{z})$ takes *both* noise $\boldsymbol{z} \in \mathbb{R}^q$ *and* bootstrap weights $\boldsymbol{w} \in \mathbb{R}^n$ as inputs. We construct $G$ using a feedforward neural network (FNN) [32], [33], because of the FNN's universal approximation properties for a large class of functions [32], [33].

GB-NPMLE incorporates the GBS approach of [25]. However, GBS is only geared towards uncertainty quantification of a single *point estimate*. Thus, GBS takes *only* weights $\boldsymbol{w}$ as inputs and *cannot* be directly used to learn an entire *probability density* $\pi$ as in (3). In order to learn an unknown density $\pi$, GB-NPMLE *also* requires noise inputs $\boldsymbol{z}$, as in (4). In summary, GB-NPMLE approximates the bootstrap distribution for NPMLE with the objective function,

$$\widehat{G} = \text{argmax}_G \, \mathbb{E}_{\boldsymbol{w}} \left[ \sum_{i=1}^{n} w_i \log[\mathbb{E}_{\boldsymbol{z}}\{f(y_i \mid G(\boldsymbol{w}, \boldsymbol{z})\}] \right], \tag{5}$$

where $G(\boldsymbol{w}, \boldsymbol{z}) : \mathbb{R}^{n+q} \mapsto \mathbb{R}$. Once we have optimized $G$ (i.e. the weights and biases of the FNN), it is then effortless to generate novel NPMLE bootstrap estimates as $\widehat{G}(\boldsymbol{w}_{\text{new}}, \boldsymbol{z}_{\text{new}})$.

To optimize (5) in practice, the expectations $\mathbb{E}_{\boldsymbol{z}}$ and $\mathbb{E}_{\boldsymbol{w}}$ are approximated by Monte Carlo averaging, similarly as in VAEs [26] and GANs [27]. Specifically, we independently sample $z \sim \text{Unif}(0, 1)$ and $\boldsymbol{w} \sim n \times \text{Dirichlet}(n, \mathbb{1}_n)$ a sufficiently large number of times and then approximate the expectations in (5) with sample averages.

Directly optimizing $G$ in (5) using stochastic gradient descent (SGD) [34] is a nontrivial task. In practice, the Monte Carlo approximation for $\mathbb{E}_{\boldsymbol{z}}$ introduces high variance to the approximate gradient for SGD because the log functions in (5) greatly shatter the linearity of $\mathbb{E}_{\boldsymbol{z}}$ [35]. Consequently, this can lead to slow convergence when optimizing the GB-NPMLE objective function (5) [35]. We address these estimation difficulties with a novel two-stage algorithm.

### B. GB-NPMLE Two-Stage Algorithm

To efficiently optimize $G$ in (5), we generalize the generator output $\theta = G(\boldsymbol{w}, \boldsymbol{z})$ to be a multi-dimensional $\boldsymbol{\theta} \in \mathbb{R}^l$ where $l > 1$ is the number of realizations of $\theta$. The resulting sequence $\boldsymbol{\theta} = \{\theta^{(j)}\}_{j=1}^{l}$ serves as the bootstrap sample *candidates*. As we explain shortly, multiple realizations of $\theta$ help to stabilize the gradient approximation in SGD for the Monte Carlo estimate of $\mathbb{E}_{\boldsymbol{z}}$ in (5). Unfortunately, this generalization also leads to correlation across the bootstrap samples.

One way to overcome the correlation issue is to take a random draw of one $\theta^{(j)}$ from $\boldsymbol{\theta}$ according to mixing probabilities $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_l)$ for the entries of $\boldsymbol{\theta}$. Repeating this procedure $B$ times produces $B$ independent bootstrap samples. However, in addition to generator $G$, we now *also* need to estimate $\boldsymbol{\tau}$. Accordingly, we propose a two-stage algorithm where in Stage I, we train $G$, and in Stage II, we estimate $\boldsymbol{\tau}$.

*GB-NPMLE Algorithm Stage I:* Fix $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_l)$ where $\tau_1 = \ldots = \tau_l = 1/l$, and solve the modified GB-NPMLE objective function,

$$\widehat{G} = \text{argmax}_G \mathbb{E}_{\boldsymbol{w}} \left[ \sum_{i=1}^{n} w_i \log[\mathbb{E}_{\boldsymbol{z},\gamma}\{f(y_i \mid \boldsymbol{e}_\gamma^\top G(\boldsymbol{w}, \boldsymbol{z})\}] \right], \tag{6}$$

where $\gamma \sim \text{Multinomial}(1, \boldsymbol{\tau})$, $\boldsymbol{e}_\gamma$ is the $\gamma$-th unit vector, and $\boldsymbol{e}_\gamma^\top G(\boldsymbol{w}, \boldsymbol{z}) \in \mathbb{R}$ is the $\gamma$-th entry of generator output $\boldsymbol{\theta}$.

To see how the modified GB-NPMLE objective (6) helps to relieve high Monte Carlo variance, let $\mathbb{E}_{\boldsymbol{z},\gamma}\{f(y_i|\boldsymbol{e}_\gamma^\top G(\boldsymbol{w}, \boldsymbol{z})\}$ in (6) be denoted by $\mathbb{E}_{\boldsymbol{z}}\{\mathbb{E}_\gamma\{f(\boldsymbol{\theta}[\boldsymbol{w}, \boldsymbol{z}, \gamma])\}\}$, and similarly, let $\mathbb{E}_{\boldsymbol{z}}\{f(y_i|G(\boldsymbol{w}, \boldsymbol{z}))\}$ be $\mathbb{E}_{\boldsymbol{z}}\{f(\boldsymbol{\theta}[\boldsymbol{w}, \boldsymbol{z}])\}$ in (5). By the law of total variance, $\text{Var}\{f(\boldsymbol{\theta}[\boldsymbol{w}, \boldsymbol{z}])\} \geq \text{Var}\{\mathbb{E}_\gamma f(\boldsymbol{\theta}[\boldsymbol{w}, \boldsymbol{z}, \gamma])\}$. Hence, the generalization of the generator output dimension to $l > 1$ reduces Monte Carlo variance.

*GB-NPMLE Algorithm Stage II:* Fix the well-trained generator $\widehat{G}$ from (6) and estimate $\boldsymbol{\tau}$ via the Monte Carlo EM (MCEM) algorithm [36]. In particular, the MCEM updates for each $k$th entry of $\boldsymbol{\tau}$ have the closed form,

$$\tau_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \frac{\tau_k^{(t)} \cdot \mathbb{E}_{\boldsymbol{w},\boldsymbol{z}}\{f(y_i \mid \boldsymbol{e}_k^\top \widehat{G}(\boldsymbol{w}, \boldsymbol{z})\}}{\sum_{k=1}^{l} \tau_k^{(t)} \cdot \mathbb{E}_{\boldsymbol{w},\boldsymbol{z}}\{f(y_i \mid \boldsymbol{e}_k^\top \widehat{G}(\boldsymbol{w}, \boldsymbol{z})\}}. \tag{7}$$

Note that we *already* obtained an efficient estimator of $\boldsymbol{\theta}$ from Stage I, and hence, the MCEM algorithm tends to have very fast convergence. In the literature, slow convergence of MCEM mostly lies in obtaining a sequence of estimators $\boldsymbol{\theta}$ [30]. We circumvent this issue by *fixing* the generator from Stage I.

Once we have estimated both $G$ (in Stage I) and $\boldsymbol{\tau}$ (in Stage II), we can easily generate a new bootstrap estimate by sampling $\boldsymbol{w}_{\text{new}}$, $\boldsymbol{z}_{\text{new}}$, $\gamma_{\text{new}}$, and then taking $\theta_{\text{new}} = \boldsymbol{e}_{\gamma_{\text{new}}}^\top \widehat{G}(\boldsymbol{w}_{\text{new}}, \boldsymbol{z}_{\text{new}})$. Repeating this procedure $B$ times results in $B$ independent bootstrap estimates. Thus, in contrast to bootstrapped NPMLE which requires $B$ total repetitive evaluations of (3), GB-NPMLE requires only a *single* evaluation of the two-stage algorithm. The entire two-stage GB-NPMLE procedure is given in Algorithm 1. Empirical evidence (reported in the online Supplementary Material) demonstrates sufficient convergence of our algorithm.

## III. Experiments and Results

### A. Simulation Studies

We investigate the performance of GB-NPMLE optimized by the two-stage algorithm from Section II-B. The generator

**Algorithm 1:** GB-NPMLE Two-Stage Algorithm.

*Set* $l = 100$, $tol = 0.001$, $T = 2000$, $B = 1000$.
*Initialize* neural network parameters $\phi$ and $\boldsymbol{\tau} = (1/l, \ldots, 1/l)^\top$
**procedure:** Stage I
  **for** $t$ in $1, \ldots,$ Epoch $T$ **do**
    *Sample* $\boldsymbol{w} \sim n \times \text{Dirichlet}(n, \mathbb{1}_n)$
    *Sample* $z \sim$ Unif(0,1)
    *Sample* $\gamma \sim \text{Multinomial}(1, \boldsymbol{\tau})$
    *Update* $G_\phi$ by SGD using modified GB-NPMLE objective (6)
  **end for**
  **return** $G_\phi$
**end procedure**
**procedure:** Stage II
  *Initialize* $\boldsymbol{\tau}^{new} = (1/l, \ldots, 1/l)$ and $\boldsymbol{\tau}^{old} = (0, \ldots, 0)$
  **while** $\min_k(|\tau_k^{old} - \tau_k^{new}|) \geq tol$ **do**
    *set* $\boldsymbol{\tau}^{old} = \boldsymbol{\tau}^{new}$
    **for** $k$ in $1, \ldots, l$ **do**
      *Sample* $\boldsymbol{w} \sim n \times \text{Dirichlet}(n, \mathbb{1}_n)$
      *Sample* $z \sim$ Unif(0,1)
      *Update* $\tau_k^{new}$ as in (7)
    **end for**
  **end while**
  **return** $\boldsymbol{\tau}^{new}$
**end procedure**
**procedure:** Generate $B$ bootstrap estimates
  **for** $b$ in $1, \ldots, B$ **do**
    *Sample* $\boldsymbol{w}^{(b)} \sim n \times \text{Dirichlet}(n, \mathbb{1}_n)$
    *Sample* $z^{(b)} \sim$ Unif(0,1)
    *Sample* $\gamma^{(b)} \sim \text{Multinomial}(1, \boldsymbol{\tau}^{new})$
    *Generate* $\theta^{(b)} = e_{\gamma^{(b)}}^\top \widehat{G}(\boldsymbol{w}^{(b)}, z^{(b)})$
  **end for**
  **return** $\theta^{(1)}, \ldots, \theta^{(B)}$
**end procedure**

$G$ is an FNN with two hidden layers and 500 neurons per layer. Sensitivity analysis to this choice of architecture is presented in the Supplement. The expectations $\mathbb{E}_{\boldsymbol{w}}$, $\mathbb{E}_{\boldsymbol{z}}$ and $\mathbb{E}_\gamma$ in the modified GB-NPMLE objective (6) are approximated by Monte Carlo averages of 100 independent samples of $\boldsymbol{w} \sim n \times \text{Dirichlet}(n, \mathbb{1}_n)$, $z \sim \text{Unif}(0,1)$ and $\gamma \sim \text{Multinomial}(1, \boldsymbol{\tau})$, and the length of $\boldsymbol{\tau}$ is set to be $l = 100$. To train $G$, we use SGD with the Adam optimizer [37]. All experiments are conducted on a single NVIDIA GeForce RTX 2080 Ti graphics processing unit (GPU) with 11 GB RAM.

We consider three simulation settings for the prior $\pi$ in the mixture model (1): (i) bimodal, (ii) unimodal and bounded on (0,1), and (iii) skewed right. Additional simulations are conducted in the Supplement. Here, we consider:

1) *Gaussian mixture model (GMM):* $y \mid \theta \sim \mathcal{N}(\theta, 1)$ and $\theta = 0.5\mathcal{N}(-3, 2) + 0.5\mathcal{N}(3, 1)$;
2) *Gamma mixture model (GaMM):* $y \mid \theta \sim \text{Gamma}(10, \theta)$ and $\theta \sim \text{Beta}(10, 5)$;
3) *Poisson mixture model (PMM):* $y \mid \theta \sim \text{Poisson}(\theta)$ and $\theta \sim \text{Gamma}(3, 1)$.

We compare GB-NPMLE to bootstrapped NPMLE (3) and a smoothed version of the discrete NPMLE (2). We use $B = 10{,}000$ bootstrap estimates to estimate the prior for GB-NPMLE and bootstrapped NPMLE. For bootstrapped NPMLE, we use the nonparametric bootstrap, i.e. we sample $n$ instances of the data with replacement and then optimize (2) a total of $B$ times using the `R` package `REBayes` [38]. For the smoothed NPMLE, we use kernel smoothing of $\widehat{\pi}$ in (2) with the `KWsmooth` function in `REBayes` [38], where we select the optimal bandwidth from an equispaced grid of length 25 from 0.1 to 10 using leave-one-out cross-validation (LOOCV).

TABLE I
COMPARISONS OF PERFORMANCE OF DIFFERENT NPMLE METHODS ON SIMULATED DATASETS

| | GB-NPMLE | | Bootstrapped NPMLE | | Smoothed NPMLE | |
|---|---|---|---|---|---|---|
| Model | $W_1(\pi, \widehat{\pi})$ | ISE | $W_1(\pi, \widehat{\pi})$ | ISE | $W_1(\pi, \widehat{\pi})$ | ISE |
| GMM | 0.348 | 0.008 | 0.298 | 0.008 | 0.936 | 0.034 |
| GaMM | 0.032 | 0.263 | 0.037 | 0.497 | 0.223 | 1.731 |
| PMM | 0.400 | 0.045 | 0.389 | 0.036 | 1.763 | 0.059 |

For performance comparison of the three NPMLE methods, we generate artificial datasets of size $n = 1000$. We look at: a) Wasserstein-1 distance between $\pi$ and $\widehat{\pi}$, defined as $W_1(\pi, \widehat{\pi}) = \int |\mathbb{F}_\pi(x) - \mathbb{F}_{\widehat{\pi}}(x)| dx$, where $\mathbb{F}$ is the cumulative density function (CDF) of $\pi$, and b) Integrated Squared Error (ISE), defined as $\int (\widehat{\pi}(x) - \pi(x))^2 dx$. Lower $W_1(\pi, \widehat{\pi})$ and ISE indicate better performance. Our results for simulations (i)–(iii) averaged across 20 replications are shown in Table I.

From Table I, we first observe that GB-NPMLE has a very similar performance to bootstrapped NPMLE. On the other hand, smoothed NPMLE (with bandwidth parameter chosen by LOOCV) performs much worse than the bootstrapping approaches. This demonstrates the difficulty of optimally tuning smoothing parameters for smoothed NPMLE.

The results from one replication of simulations (i)–(iii) are plotted in Fig. 1, which also plots the true latent density $\pi$ (solid black) and the discrete NPMLE solution (2) (dotted purple). Fig. 1 shows that the classical (discrete) NPMLE (2) is inadequate for capturing $\pi$ when $\pi$ is truly continuous. For example, in the GaMM model, the classical NPMLE does not clearly indicate that $\pi$ is unimodal. On the other hand, GB-NPMLE (solid red) and bootstrapped NPMLE (solid blue) are quite close to the true $\pi$, capturing all inherent aspects of the true $\pi$ like bimodality, boundedness and unimodality, and skewness respectively in the GMM, GaMM, and PMM models. The smoothed NPMLE appears to be oversmoothed and is unable to capture bimodality of $\pi$ in the GMM model or the unimodality and boundedness of $\pi$ in the GaMM model. In the PMM model, the sharp peak of the skewed density $\pi$ is also not adequately captured by smoothed NPMLE.

GB-NPMLE and bootstrapped NPMLE both give similar performance, but GB-NPMLE is much faster because it only requires a single evaluation of Algorithm 1. We compare the average computational time of GB-NPMLE and bootstrapped NPMLE for sample sizes $n \in \{1000, 10{,}000, 100{,}000\}$ across simulations (i)–(iii). Our results are plotted (on the log scale) in Fig. 2. Fig. 2 shows that GB-NPMLE is much more scalable. In particular, GB-NPMLE takes only about five minutes to complete when $n = 100{,}000$; on the other hand, bootstrapped NPMLE requires almost two days to finish.

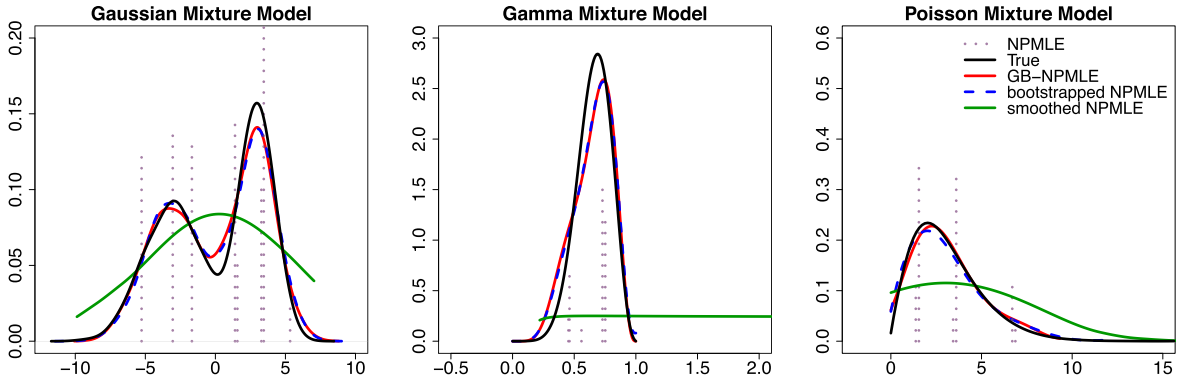


Fig. 1. Results from one replication of Simulations (i)–(iii). In addition to GB-NPMLE (solid red), bootstrapped NPMLE (dashed blue), and smoothed NPMLE (solid green), we also plot the true density (solid black) and the classical discrete NPMLE (dotted purple).

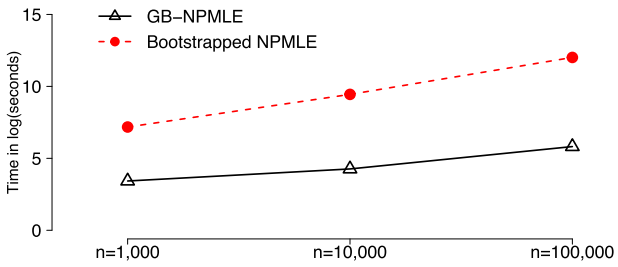


Fig. 2. Mean computation time in log(seconds) for the three simulations vs. sample size $n \in \{1000, 10{,}000, 100{,}000\}$. For $n = 100{,}000$, GB-NPMLE takes five minutes, whereas bootstrapped NPMLE takes almost two days.

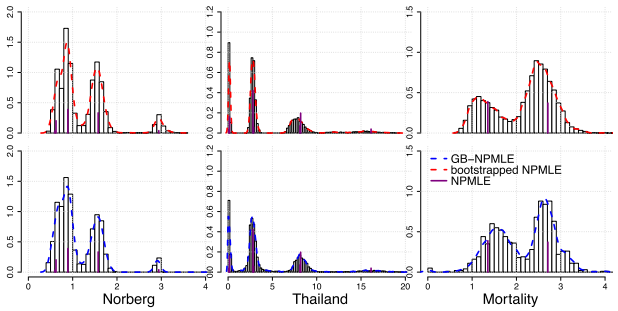


Fig. 3. Results for real datasets for GB-NPMLE (dashed red, top panel) and bootstrapped NPMLE (dotted blue, bottom panel). The pink vertical lines in both plots represent the discrete NPMLE solution.

### B. Evaluation on Real Datasets

We evaluate GB-NPMLE on three count datasets where the counts are assumed to follow a Poisson mixture model, $y_i \mid \theta_i \sim \text{Poisson}(\theta_i)$ and $\theta_i \sim \pi, i = 1, \ldots, n$. The *Norberg* dataset consists of 1125 group life insurance statistics in Norway [39], the *Thailland* dataset consists of 602 preschool children's health condition, and the *Mortality* dataset contains the number of deaths for women aged greater than eighty [40]. Fig. 3 compares GB-NPMLE (top panel) with bootstrapped NPMLE (bottom panel). We see that GB-NPMLE approximates bootstrapped NPMLE very well. Furthermore, both methods identify the same number of local modes (three in *Norberg*, four in *Thailand*, and two in *Mortality*).

We also examine out-of-sample prediction using the log predictive score (LPS) $\mathbb{E}_\theta[-\log f(\boldsymbol{y}_{\text{new}}; \theta) \mid \boldsymbol{y}_{\text{obs}}]$. To approximate

TABLE II
COMPARISONS OF PERFORMANCE OF GB-NPMLE AND BOOTSTRAPPED NPMLE ON REAL COUNT DATASETS

| | | GB-NPMLE | | Bootstrapped NPMLE | |
|---|---|---|---|---|---|
| Dataset | $n$ | LPS | Time (sec) | LPS | Time (sec) |
| Norberg | 72 | 20.35 | 126.1 | 20.83 | 732.8 |
| Thailland | 602 | 156.26 | 140.9 | 156.21 | 3439.3 |
| Mortality | 1096 | 199.30 | 153.3 | 199.31 | 5279.2 |

LPS, we use $K$-fold cross validation (CV),

$$\text{LPS} = K^{-1} \sum_{k=1}^{K} \sum_{i \in I_k} -\log \left[ B^{-1} \sum_{b=1}^{B} f(y_i; \widehat{\theta}^b_{(-k)}) \right],$$

where $\widehat{\theta}^b_{(-k)} \sim \widehat{\pi}_{(-k)}$ and $\widehat{\pi}_{(-k)}$ is the bootstrap distribution obtained by excluding the $k$th fold $I_k$. We set $K = 10$ and $B = 500$. Table II shows that the GB-NPMLE predictions are quite close to their standard bootstrap counterparts.. Table II also shows that GB-NPMLE is much more scalable, with only a modest increase in computation time for 10-fold CV when $n$ increases from 72 to 1096. When $n = 1096$, GB-NPMLE is about 34 times faster than bootstrapped NPMLE.

## IV. CONCLUSION

In this letter, we introduced GB-NPMLE, a generative framework for estimating a mixing density through bootstrapping. A novel two-stage algorithm was proposed to rapidly and accurately obtain bootstrap estimates. Bootstrapping has rarely been employed for NPMLE in the past, and our work paves the way for bootstrapping to serve as an attractive alternative to smoothing NPMLE. Simulations and real data analyses demonstrated the effectiveness and scalability of our approach. Codes to implement GB-NPMLE and the real datasets analyzed in this article are available on GitHub at https://github.com/shijiew97/GBnpmle.

Our methodology is only presented for estimating a univariate latent density. However, it will be useful to extend GB-NPMLE to *multivariate* density estimation. If covariates are also observed, then GB-NPMLE can be extended to mixture of regression models [8]. Finally, theoretical analysis of the two-stage algorithm (e.g. the convergence rate) is needed.

## REFERENCES


[1] H. T. Huynh and L. Nguyen, "Nonparametric maximum likelihood estimation using neural networks," *Pattern Recognit. Lett.*, vol. 138, pp. 580–586, 2020.

[2] Y. Li and Z. Ye, "Boosting in univariate nonparametric maximum likelihood estimation," *IEEE Signal Process. Lett.*, vol. 28, pp. 623–627, 2021.

[3] V. Bhatia and B. Mulgrew, "Non-parametric likelihood based channel estimator for gaussian mixture noise," *Signal Process.*, vol. 87, no. 11, pp. 2569–2586, 2007.

[4] L. Feng, R. Ma, and L. H. Dicker, "Nonparametric maximum likelihood approximate message passing," in *Proc. IEEE 51st Annu. Conf. Inf. Sci. Syst.*, 2017, pp. 1–6.

[5] B. W. Silverman, M. C. Jones, J. D. Wilson, and D. W. Nychka, "A smoothed EM approach to indirect estimation problems, with particular, reference to stereology and emission tomography," *J. Roy. Stat. Soc. Ser. B. (Methodological)*, vol. 52, no. 2, pp. 271–324, 1990.

[6] Y. Vardi, L. A. Shepp, and L. Kaufman, "A statistical model for positron emission tomography," *J. Amer. Stat. Assoc.*, vol. 80, no. 389, pp. 8–20, 1985.

[7] B. Efron, "Empirical bayes deconvolution estimates," *Biometrika*, vol. 103, no. 1, pp. 1–20, 2016.

[8] H. Jiang and A. Guntuboyina, "A nonparametric likelihood approach to mixture of regression," 2021, *arXiv:2108.09816*.

[9] Z. Fan, L. Guan, Y. Shen, and Y. Wu, "Gradient flows for empirical Bayes in high-dimensional linear models," 2023, *arXiv:2312.12708*.

[10] S. Frühwirth-Schnatter, *Finite Mixture and Markov Switching Models*. New York, Ny, USA: Springer, 2006.

[11] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake, "Finite mixture models," *Annu. Rev. Statist. Appl.*, vol. 6, pp. 355–378, 2019.

[12] J. Kiefer and J. Wolfowitz, "Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters," *Ann. Math. Statist.*, vol. 27, no. 4, pp. 887–906, 1956.

[13] B. G. Lindsay, "Mixture models: Theory, geometry and applications," in *Proc. NSF-CBMS Regional Conf. Ser. Probability Statist.*, 1995, pp. 1–163.

[14] N. Laird, "Nonparametric maximum likelihood estimation of a mixing distribution," *J. Amer. Stat. Assoc.*, vol. 73, no. 364, pp. 805–811, 1978.

[15] C. H. Zhang, "Compound decision theory and empirical Bayes methods," *Ann. Statist.*, vol. 31, no. 2, pp. 379–390, 2003.

[16] R. Koenker and I. Mizera, "Convex optimization, shape constraints, compound decisions, and empirical Bayes rules," *J. Amer. Stat. Assoc.*, vol. 109, no. 506, pp. 674–685, 2014.

[17] L. Feng and L. H. Dicker, "Approximate nonparametric maximum likelihood for mixture models: A convex optimization approach to fitting arbitrary multivariate mixing distributions," *Comput. Statist. Data Anal.*, vol. 122, pp. 80–91, 2018.

[18] L. Liu, M. Levine, and Y. Zhu, "A functional EM algorithm for mixing density estimation via nonparametric penalized likelihood maximization," *J. Comput. Graphical Statist.*, vol. 18, no. 2, pp. 481–504, 2009.

[19] R. Koenker and J. Gu, "Comment: Minimalist G-modeling," *Stat. Sci.*, vol. 34, no. 2, pp. 209–213, 2019.

[20] M. Knott and P. Tzamourani, "Bootstrapping the estimated latent distribution of the two-parameter latent trait model," *Brit. J. Math. Stat. Psychol.*, vol. 60, no. 1, pp. 175–191, 2007.

[21] D. B. Rubin, "The bayesian bootstrap," *Ann. Statist.*, vol. 9, no. 1, 1981, Art. no. 130434.

[22] M. A. Newton and A. E. Raftery, "Approximate Bayesian inference with the weighted likelihood bootstrap," *J. Roy. Stat. Soc.: Ser. B. (Methodological)*, vol. 56, no. 1, pp. 3–26, 1994.

[23] M. A. Newton, N. G. Polson, and J. Xu, "Weighted Bayesian bootstrap for scalable posterior distributions," *Can. J. Statist.*, vol. 49, no. 2, pp. 421–437, 2021.

[24] B. Efron, "Bootstrap methods: Another look at the jackknife," *Ann. Statist.*, vol. 7, no. 1, pp. 1–26, 1979.

[25] M. Shin, S. Wang, and J. S. Liu, "Generative multiple-purpose sampler for weighted M-estimation," *J. Comput. Graphical Statist.*, to be published.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2022, *arXiv:1312.6114*.

[27] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[28] X. Zhou, Y. Jiao, J. Liu, and J. Huang, "A deep generative approach to conditional sampling," *J. Amer. Stat. Assoc.*, vol. 118, no. 543, pp. 1837–1848, 2023.

[29] S. Liu, X. Zhou, Y. Jiao, and J. Huang, "Wasserstein generative learning of conditional distribution," 2021, *arXiv:2112.10039*.

[30] Y. Qiu and X. Wang, "ALMOND: Adaptive latent modeling and optimization via neural networks and Langevin diffusion," *J. Amer. Stat. Assoc.*, vol. 116, no. 535, pp. 1224–1236, 2021.

[31] G. O. Roberts and R. L. Tweedie, "Exponential convergence of Langevin distributions and their discrete approximations," *Bernoulli*, vol. 2, no. 4, pp. 341–363, 1996.

[32] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[33] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[34] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An introductory review of deep learning for prediction models with Big Data," *Front. Artif. Intell.*, vol. 3, no. 4, 2020, doi: 10.3389/frai.2020.00004.

[35] X. Lian, M. Wang, and J. Liu, "Finite-sum composition optimization via variance reduced gradient descent," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1159–1167.

[36] R. A. Levine and G. Casella, "Implementations of the Monte Carlo EM algorithm," *J. Comput. Graphical Statist.*, vol. 10, no. 3, pp. 422–439, 2001.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*.

[38] R. Koenker and J. Gu, "REBayes: An R package for empirical Bayes mixture methods," *J. Stat. Softw.*, vol. 82, no. 8, pp. 1–26, 2017.

[39] R. Norberg, "Experience rating in group life insurance," *Scand. Actuarial J.*, vol. 1989, no. 4, pp. 194–224, 1989.

[40] D. Böhning, *Computer-Assisted Analysis of Mixtures and Applications: Meta-Analysis, Disease Mapping and Others*. Boca Raton, FL, USA: CRC Press, 1999.