Statistics in Biosciences
https://doi.org/10.1007/s12561-021-09324-4

Simultaneous Learning the Dimension and Parameter of a Statistical Model with Big Data

Long Wang[1] · **Fangzheng Xie**[1] · **Yanxun Xu**[1]

Received: 15 June 2020 / Revised: 19 November 2020 / Accepted: 19 September 2021

Abstract
Estimating the dimension of a model along with its parameters is fundamental to many statistical learning problems. Traditional model selection methods often approach this task by a two-step procedure: first estimate model parameters under every candidate model dimension, then select the best model dimension based on certain information criterion. When the number of candidate models is large, however, this two-step procedure is highly inefficient and not scalable. We develop a novel automated and scalable approach with theoretical guarantees, called mixed-binary simultaneous perturbation stochastic approximation (MB-SPSA), to simultaneously estimate the dimension and parameters of a statistical model. To demonstrate the broad practicability of the MB-SPSA algorithm, we apply the MB-SPSA to various classic statistical models including *K*-means clustering, Gaussian mixture models with an unknown number of components, sparse linear regression, and latent factor models with an unknown number of factors. We evaluate the performance of the MB-SPSA through simulation studies and an application to a single-cell sequencing dataset in terms of accuracy, running time, and scalability. The code implementing the MB-SPSA is available at http://github.com/wanglong24/MB-SPSA.

Keywords Clustering · Mixed-binary optimization · Mini-batch learning · Single-cell sequencing · Stochastic optimization

1 Introduction

Advances in high-throughput biotechnologies have led to the generation of large amounts of biomedical data, providing researchers unprecedented opportunities and challenges to analyze such large-scale and complex data. For example, modern

✉ Yanxun Xu
yanxun.xu@jhu.edu

[1] Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, USA

Published online: 15 October 2021

single-nucleotide polymorphisms (SNP) arrays technique allows us to tag millions of SNPs, which enables genome-wide association studies (GWAS) to detect disease associated variants [1]. Also, due to the emergence of microfluidics techniques and combinatorial indexing strategies, thousands or even millions of cells can be generated in a single experiment [2]. The analysis of these big data needs to be handled in a computationally efficient and statistically sound manner. Therefore, in light of the emergence of big data in various domains, how to select the best statistical model under an unknown dimension of parameter space in the big data context is of great interest. Some classic statistical models that rely on the selection of parameter dimension include mixture models with an unknown number of components [3], factor models with an unknown number of latent factors [4], and random dot product graphs with an unknown dimension of the latent vertex positions [5]. In this paper, we refer to the dimension of the parameter space of a model as the dimension of the model. Our goal is to develop an automated and scalable approach that allows us to simultaneously learn the optimal model dimension along with model parameters.

Popular examples of commonly used criteria for model selection include the Akaike information criterion (AIC) [6] and Bayesian information criterion (BIC) [7]. To select the best model, researchers need to first estimate the model-specific parameters under every candidate model, then compare all candidate models based on a certain information criterion (e.g., AIC or BIC). When the number of candidate models is large (e.g., exponential in the sample size), the complete learning process requires lots of computational resources. Similar issues arise in neural network training as well. When the optimal network structure (e.g., number of layers and number of neurons per layer) is unknown, conventional approaches apply reinforcement learning [8] or evolution [9] over a discrete and non-differentiable search space to cover every possible network structure, leading to a long training time. Hence, the inefficiency of such a two-phase procedure motivates us to seek an integrated way to determine the optimal model dimension as well as estimating model parameters.

In Bayesian literature, simultaneously learning the model dimension and parameters can be addressed by dimension-changing sampling techniques such as reversible jump Markov chain Monte Carlo (MCMC) [10] that can search over different model dimensions. However, the practical implementation of the reversible jump MCMC is limited by the difficulty in choosing good jump proposals since there is no natural neighborhood structure among different model dimensions. Bayesian non-parametric models, such as Dirichlet process mixture models [11, 12] and Indian buffet processes for latent feature allocation models [13], can adapt to the model dimension based on the complexity of the data. The posterior inference is usually carried out using either MCMC [14] or variational inference [15]. However, these Bayesian methods do not scale to big data effectively and cannot be generalized to a wide range of statistical models including frequentist and model-free methods.

Model selection has also been tackled via optimization methods. For example, when the model dimension is not given, Markley and Miller [16] proposed an optimization-based method: Firstly, an expectation maximization (EM)-type algorithm was developed to minimize the BIC value when the model dimension was fixed. Then the optimal model dimension was selected by starting the EM-type algorithm

with a sufficiently large initial dimension, and repeating the fitting process for every reduced model. A similar approach was also developed in Huang et al. [17]. However, the computational cost of these methods increases significantly when the initial dimension is large.

To simultaneously learn the model dimension and parameters, we formulate the model selection problem as a mixed-binary optimization problem. Specifically, a set of auxiliary binary variables is introduced to indicate whether a certain component or covariate is active in the model or not. This formulation transfers the task of model selection to an optimization problem involving binary indicator variables and continuous model variables (i.e., parameters), called mixed-binary optimization problem. When the model dimension is unknown but the maximum model dimension is given, some non-linear programming methods can be useful. For example, Bertsimas et al. [18] studied a mixed-integer quadratic programming formulation with a cardinality constraint. Based on various information criteria, Miyashiro and Takano [19] proposed a mixed-integer second-order cone programming formulation. Although these mixed-integer programming methods can provide good-quality solutions for small- or medium-sized benchmark datasets, they slow down dramatically and sometimes fail to converge for larger datasets. Therefore, an efficient and scalable optimization method for mixed-binary optimization problem is desired.

Efficient optimization algorithms for large-scale problems have been majorly focusing on stochastic optimization thanks to the recent progress in machine learning techniques for big data. Despite the flourish developments in stochastic optimization, stochastic mixed-binary problems are relatively under-explored. The main technical challenge is that the gradient information is not available since the loss function is generally not differentiable due to the binary variables, which limits the direct use of any first-order methods, such as stochastic gradient descent, Adagrad [20], and AdaM [21]. Spall [22] developed a gradient-free stochastic approximation algorithm for continuous optimization problems called the simultaneous perturbation stochastic approximation (SPSA), which is very efficient and has been widely used in machine learning problems involving a large dimension and/or sample size [23–27]. Other recent applications based on SPSA-related algorithms include the dynamic origin-destination matrix estimation [28], data-driven controller for general discrete non-linear system [29], optimal experiment design for evoking a desired target brain state [30], fitting stochastic epidemiological models [31], and medical imaging with convolutional neural networks [32]. Aksakalli and Malekipirbazari [33] developed the binary SPSA for classification problems and obtained favorable results comparing with K-nearest neighbors, decision tree methods, and the support vector machine on big data containing 10, 000 predictors. However, the SPSA for mixed-binary optimization problem has not been studied, which is a gap we aim to fill.

In this paper, we develop a mixed-binary SPSA (MB-SPSA) to solve mixed-binary optimization problems with theoretical guarantees. The novelty of the proposed MB-SPSA comes in three ways: (i) the model dimension is learned adaptively during the fitting process so that there is no need to fit all possible candidate models; (ii) the loss function measurements can be noisy so that evaluations on a small batch of data is allowed, which enables both the batch learning and the mini-batch

learning for scalability; (iii) the MB-SPSA can be applied to a broad range of statistical models and machine learning algorithms. We consider four applications of the MB-SPSA in this paper: *K*-means clustering with an unknown number of clusters, Gaussian mixture models with an unknown number of components, sparse linear models, and latent factor models.

The remainder of the paper is organized as follows. In Sect. 2 we review the basic SPSA and develop the MB-SPSA with theoretical guarantees. Applications of the MB-SPSA to four examples including the *K*-means, Gaussian mixture models, sparse linear models, and latent factor models are discussed in Sect. 3. We demonstrate the utility of the MB-SPSA using simulation studies in Sect. 4 and an application to a single-cell RNA-sequencing dataset in Sect. 5. We conclude the paper in Sect. 6.

2 Methods

2.1 Background on SPSA

We first briefly review the generic SPSA algorithm for continuous optimization problems [22]. Consider a general minimization problem $\min_{\boldsymbol{\theta}\in\mathbb{R}^p} L(\boldsymbol{\theta})$ where $L : \mathbb{R}^p \to \mathbb{R}$ is a differentiable loss function and denote its gradient as $\boldsymbol{g}(\boldsymbol{\theta}) = \partial L(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$. Furthermore, assume that only the noisy measurement $\hat{L}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \epsilon(\boldsymbol{\theta})$ with a mean-zero noise $\epsilon(\boldsymbol{\theta})$ is available. The standard stochastic approximation update gives

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k), \tag{1}$$

where $\{a_k\}$ is a non-negative decreasing gain sequence and $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ is the gradient estimate at $\hat{\boldsymbol{\theta}}_k$. When the true gradient $\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$ is either unavailable or computationally expensive, an estimated gradient $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ is applied in place of the true gradient. To estimate the true gradient $\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$, the well-known finite different method was first developed in Dennis Jr and Schnabel [34]. However, it requires $2p$ (noisy) loss function evaluations per iteration to construct one gradient estimate, making it inefficient in high-dimensional problems. In contrast, Spall [22] proposed a more efficient method, called SPSA, that can estimate $\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$ as follows: One first computes the perturbations $\hat{\boldsymbol{\theta}}_k^{(\pm)}$ as

$$\hat{\boldsymbol{\theta}}_k^{(+)} = \hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k \;\text{ and }\; \hat{\boldsymbol{\theta}}_k^{(-)} = \hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k, \tag{2}$$

where $\{c_k\}$ is another non-negative decreasing gain sequence and $\boldsymbol{\Delta}_k$ is a random p-dimensional perturbation vector $\boldsymbol{\Delta}_k = [\Delta_{k1}, \ldots, \Delta_{kp}]^T$. Based on $\hat{\boldsymbol{\theta}}_k^{(+)}$ and $\hat{\boldsymbol{\theta}}_k^{(-)}$, the gradient is then estimated by

$$\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) = \frac{\hat{L}(\hat{\boldsymbol{\theta}}_k^{(+)}) - \hat{L}(\hat{\boldsymbol{\theta}}_k^{(-)})}{2c_k\boldsymbol{\Delta}_k}, \tag{3}$$

where $1/\boldsymbol{\Delta}_k = \boldsymbol{\Delta}_k^{-1} = \left[\Delta_{k1}^{-1}, \ldots, \Delta_{kp}^{-1}\right]^T$. The advantage of (3) is that only two (noisy) loss function measurements, i.e., $\hat{L}(\hat{\boldsymbol{\theta}}_k^{(+)})$ and $\hat{L}(\hat{\boldsymbol{\theta}}_k^{(-)})$, are required, making the gradient estimate highly efficient when p is large. The regularity conditions for the almost sure convergence were provided in Spall [22]. One common choice of the gain sequence is $a_k = a/(A+k)^{\alpha}$ and $c_k = c/k^{\gamma}$ for some positive scalars a, c, A, α, γ. Each coordinate of the perturbation vector $\boldsymbol{\Delta}_k$ can be sampled independently and uniformly from $\{-1, +1\}$ [35].

2.2 Mixed-Binary SPSA (MB-SPSA)

We propose the MB-SPSA to minimize loss functions with mixed-binary variables and prove the almost sure convergence. The applications under mini-batch learning framework are also discussed.

Consider the mixed-binary optimization problem with d binary variables and $p-d$ continuous variables, $\min_{\boldsymbol{\theta}\in\{0,1\}^d\times\mathbb{R}^{p-d}} L(\boldsymbol{\theta})$. Note that the original SPSA is no longer applicable since the binary constraint is not satisfied in (1) and (2), i.e., $\hat{\boldsymbol{\theta}}_k^{(\pm)} \notin \{0,1\}^d \times \mathbb{R}^{p-d}$. To overcome this issue, denoting $\lfloor x \rfloor$ to be the maximal integer no greater than x and

$$\boldsymbol{\pi}(\boldsymbol{\theta}) = \left(\lfloor \tau(\theta_1) + 1/2 \rfloor, \ldots, \lfloor \tau(\theta_d) + 1/2 \rfloor, \theta_{d+1}, \ldots, \theta_p\right)^T$$

with

$$\tau(\theta) = \begin{cases} 0, & \text{if } \theta < 0, \\ \theta, & \text{if } 0 \le \theta \le 1, \\ 1, & \text{if } \theta > 1, \end{cases}$$

we modify (2) as

$$\hat{\boldsymbol{\theta}}_k^{(+)} = \boldsymbol{\pi}(\hat{\boldsymbol{\theta}}_k + \boldsymbol{C}_k \circ \boldsymbol{\Delta}_k) \text{ and } \hat{\boldsymbol{\theta}}_k^{(-)} = \boldsymbol{\pi}(\hat{\boldsymbol{\theta}}_k - \boldsymbol{C}_k \circ \boldsymbol{\Delta}_k), \tag{4}$$

where the operator $\circ$ is the matrix Hadamard (element-wise) product and

$$\boldsymbol{C}_k = (\underbrace{b_k, \ldots, b_k}_{d \text{ comp.}}, \underbrace{c_k, \ldots, c_k}_{(p-d) \text{ comp.}})^T,$$

with $b_k = b/k^{\gamma}$. We then propose to compute the following pseudo-gradient estimate

$$\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) = \frac{\hat{L}(\hat{\boldsymbol{\theta}}_k^{(+)}) - \hat{L}\left(\hat{\boldsymbol{\theta}}_k^{(-)}\right)}{2\boldsymbol{C}_k \circ \boldsymbol{\Delta}_k}, \tag{5}$$

which is analogous to the gradient estimate $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ in (3) for continuous optimization problems. Since there is no "true gradient" of $L(\boldsymbol{\theta})$ due to the existence of binary variables and only noisy loss function measurements are used, our MB-SPSA can

be viewed as a gradient-free stochastic optimization. The SPSA idea of generating $\hat{\boldsymbol{\theta}}_k^{(\pm)}$ is used here to essentially estimate the "pseudo-gradient" while maintaining the scalability to high dimensionality. If the model dimension is known and fixed, our method performs similarly to classical first-order gradient descent algorithms. At a higher level, one can view the MB-SPSA as a pseudo-gradient descent method that jumps among different dimensions. We present the generic MB-SPSA in Algorithm 1 below.

Algorithm 1 Generic MB-SPSA algorithm

Input: initial condition $\hat{\boldsymbol{\theta}}_0$, gain sequence parameters $a, b, c, A, \alpha, \gamma$
Output: final estimate $\boldsymbol{\pi}(\hat{\boldsymbol{\theta}}_k)$

set iteration index $k = 0$
while terminating conditions have not been satisfied **do**
  construct gain sequences $a_k, b_k, c_k, \boldsymbol{C}_k$ and perturbation vector $\boldsymbol{\Delta}_k$
  compute perturbation points by (2.4): $\hat{\boldsymbol{\theta}}_k^{(\pm)} = \boldsymbol{\pi}(\hat{\boldsymbol{\theta}}_k \pm \boldsymbol{C}_k \circ \boldsymbol{\Delta}_k)$
  estimate pseudo-gradient by (2.5): $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) = \frac{\hat{L}(\hat{\boldsymbol{\theta}}_k^{(+)}) - \hat{L}(\hat{\boldsymbol{\theta}}_k^{(-)})}{2\boldsymbol{C}_k \circ \boldsymbol{\Delta}_k}$
  compute new estimate by (2.1): $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$
  update iteration index $k \leftarrow k + 1$
end while
return final estimate $\boldsymbol{\pi}(\hat{\boldsymbol{\theta}}_k)$

Remark 1 Note that in the MB-SPSA, only $\hat{\boldsymbol{\theta}}_k^{(\pm)} \in \{0, 1\}^d \times \mathbb{R}^{p-d}$ are required since we need to take direct loss function measurements at those two points. Although it is generally the case that $\hat{\boldsymbol{\theta}}_k \notin \{0, 1\}^d \times \mathbb{R}^{p-d}$, it does not affect the MB-SPSA algorithm since we do not need to evaluate $L(\hat{\boldsymbol{\theta}}_k)$ during the training process. The pseudo-gradient descent step (1) can gradually push the first d components of $\hat{\boldsymbol{\theta}}_k$ toward the value 0 or 1 depending on corresponding components in the optimal point. At the terminal iteration, we project $\hat{\boldsymbol{\theta}}_k$ to $\{0, 1\}^d \times \mathbb{R}^{p-d}$ by calculating $\boldsymbol{\pi}(\hat{\boldsymbol{\theta}}_k)$ to form the final estimate.

2.3 Mini-Batch Learning Using MB-SPSA

In the MB-SPSA, we only require the noisy measurements of the loss function $L(\boldsymbol{\theta})$. When the loss function is based on the given data $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ such as the negative joint log-likelihood function, one can consider the noisy measurement as the log-likelihood evaluated at only a single data point or a small mini-batch of data points, thus motivating the use of the MB-SPSA for scalable learning with big data or mini-batch learning.

Let $\boldsymbol{\Theta}$ be the parameter space of a statistical model. When the data $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ are independently and identically distributed (i.i.d.), the loss function to be minimized can be the negative joint log-likelihood function,

$$\min_{\theta\in\mathbf{\Theta}} L(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n;\boldsymbol{\theta}) = \min_{\theta\in\mathbf{\Theta}} \sum_{i=1}^{n} \ell(\boldsymbol{y}_i;\boldsymbol{\theta}),$$

where $\ell(\boldsymbol{y}_i;\boldsymbol{\theta})$ is the negative log-likelihood of a single data point $\boldsymbol{y}_i$. For big data problems when n is extremely large, directly evaluating $L(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n;\boldsymbol{\theta})$ is computationally intractable. Instead, we utilize a uniformly randomly selected subset of the data. Denote I to be a random index variable that has a uniform distribution on integers $\{1,\ldots,n_b\}$. For a mini-batch size $n_b \ll n$, assuming $I_1,\ldots,I_{n_b}$ are i.i.d., we get a uniformly randomly selected subset of the data $\boldsymbol{y}_{I_1},\ldots,\boldsymbol{y}_{I_n}$ such that

$$\hat{L}(\boldsymbol{y}_{I_1},\ldots,\boldsymbol{y}_{I_{n_b}},\boldsymbol{\theta}) = \frac{n}{n_b}\sum_{i=1}^{n_b} \ell(\boldsymbol{y}_{I_i};\boldsymbol{\theta}),$$

where $\hat{L}(\boldsymbol{y}_{I_1},\ldots,\boldsymbol{y}_{I_{n_b}},\boldsymbol{\theta})$ can be viewed as a noisy measurement of $L(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n;\boldsymbol{\theta})$. It is easy to see that the "random noise" $\epsilon(\boldsymbol{\theta}) = \hat{L}(\boldsymbol{y}_{I_1},\ldots,\boldsymbol{y}_{I_{n_b}},\boldsymbol{\theta}) - L(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n;\boldsymbol{\theta})$ is mean-zero since

$$\mathbb{E}_I[\epsilon(\boldsymbol{\theta})] = \mathbb{E}_I\left[\frac{n}{n_b}\sum_{i=1}^{n_b} \ell(\boldsymbol{y}_{I_i};\boldsymbol{\theta})\right] - \sum_{i=1}^{n} \ell(\boldsymbol{y}_i;\boldsymbol{\theta}) = 0,$$

where the expected value is taken with respect to the random sampling of $I_1,\ldots,I_{n_b}$. By accessing only a small subset of data $\boldsymbol{y}_{I_1},\ldots,\boldsymbol{y}_{I_{n_b}}$ at each iteration, the total computational cost at each iteration can be reduced considerably. The variance of the random noise depends on the loss function and the size of the mini-batch. As long as the random noise introduced by each mini-batch does not overturn the average behavior over multiple iterations, the proposed algorithm will converge and be faster than using the full batch evaluations at every iteration [36].

2.4 Theoretical Results of the MB-SPSA

In this subsection, we provide the theoretical guarantee of the MB-SPSA by establishing the almost sure convergence result.

Assumption 1 *(Gain sequences)* $a_k > 0, c_k > 0;\quad a_k \to 0, c_k \to 0;\quad \sum_{k=1}^{\infty} a_k = \infty, \sum_{k=1}^{\infty} a_k^2 < \infty, \sum_{k=1}^{\infty}(a_k/c_k)^2 < \infty.$

Assumption 2 *(Estimate boundedness)* For all $k, \|\hat{\boldsymbol{\theta}}_k\| < \infty$ a.s.

Assumption 3 *(Measurement noise)* Let $\mathcal{F}_k = \{\hat{\boldsymbol{\theta}}_1,\ldots,\hat{\boldsymbol{\theta}}_k\}$ and $\mathcal{G}_k = \{\boldsymbol{\Delta}_1,\ldots,\boldsymbol{\Delta}_k\}$. Denote $\epsilon_k^{(\pm)} = \hat{L}(\hat{\boldsymbol{\theta}}_k^{(\pm)}) - L(\hat{\boldsymbol{\theta}}_k^{(\pm)})$. For all k, there exists a real number B_0 such that $\mathbb{E}[\epsilon_k^{(+)} - \epsilon_k^{(-)} \mid \mathcal{F}_k, \mathcal{G}_k] = 0$ a.s., $\mathrm{Var}(\epsilon_k^{(\pm)}) \le B_0$.

Assumption 4 *(Perturbation vector)* For all i and k, the components of $\{\Delta_{ki}\}$ are independently and identically distributed (i.i.d.) and there exists B_1, B_2 and B_3 such that $0 < |\Delta_{ki}| \le B_1, \mathbb{E}[\Delta_{ki}] = 0, \mathbb{E}[|\Delta_{ki}^{-1}|] \le B_2$, and $\mathbb{E}\{[\hat{L}(\hat{\boldsymbol{\theta}}_k^{(-)}) - \hat{L}(\hat{\boldsymbol{\theta}}_k^{(-)})]^2\} \le B_3$.

Assumption 5 *(Loss function smoothness)* For all k, there exists B_4 such that $|L(\hat{\boldsymbol{\theta}}_k^{(+)}) - L(\hat{\boldsymbol{\theta}}_k^{(-)})| \leq B_4$ a.s.

Assumption 6 *(Search direction)* Let $\boldsymbol{\Theta}^*$ be the set of local minimizers such that for any $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}^*$, $\mathbb{E}\left[\hat{\boldsymbol{g}}_k(\boldsymbol{\theta}')\right]^T(\boldsymbol{\theta} - \boldsymbol{\theta}^*) > 0$ for all $\boldsymbol{\theta}' \in B_r(\boldsymbol{\theta})$, where $B_r(\boldsymbol{\theta})$ is the ball centered at $\boldsymbol{\theta}$ with radius $0 < r < 1/2$.

Theorem 1 *Under Assumptions 1–6, $\hat{\boldsymbol{\theta}}_k$ converges to some $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}^*$ a.s. when $k \to \infty$*
.

Proof See Appendix 1. □

Remark 2 Assumptions 1–5 are standard conditions for stochastic optimization and are similar to those provided in SPSA [22]. Assumption 6 is a common and necessary assumption for proving the local convergence. It generalizes the search directions for continuous problems [37, Sect. 4.3.2]. If the parameter dimension is fixed, similar assumptions can also be found in many EM-type optimization methods.

Remark 3 Spall [35] and Sect. 7.5 of Spall [37] provide a practical guidance on choosing the gain sequences: $a_k = a/(A + k)^\alpha$, $b_k = b/k^\gamma, c_k = c/k^\gamma$. Specifically, one can set $\alpha = 0.601$ and $\gamma = 0.101$ since they are practically effective and theoretically valid [35]. The value of A is often set to be around 10% of the total number of iterations. In addition, one can choose the values of a, b and c such that the elements of $\hat{\boldsymbol{\theta}}$ are moving by a desired magnitude (e.g., 10%) of the initial values in early iterations. The general philosophy is to make a_k large to increase the convergence speed and c_k small to obtain accurate gradient estimates. The variance of $\hat{\boldsymbol{\theta}}_k$ depends on the gain sequences. When the gain sequences are decaying at a moderate rate, the variance of the estimate is small.

Since the MB-SPSA is guaranteed to converge, the properties of the optimum depend only on the loss function used in the algorithm. If a traditional model selection criteria such as AIC or BIC is used in the loss function, the MB-SPSA performs comparably to these traditional model selection methods, but with a much shorter time. The added values of the MB-SPSA are the generality to a variety of models and the scalability to big data. In short, we are not proposing a new model selection method, but rather an efficient optimization technique that applies to optimization-based generic model selection problems.

3 Applying the MB-SPSA to Statistical Models

In this section, we illustrate how one can apply the MB-SPSA algorithm to the commonly used statistical models with four examples: K-means clustering, Gaussian mixture models (GMM), sparse linear models, and latent factor models.

3.1 K-Means Clustering

Clustering is a powerful machine learning tool for detecting structures in biomedical datasets, e.g., clustering single-cell sequencing data to identify different cell types [2], clustering gene expression data to discover disease subtypes [38], and clustering patients to find patients subgroups with the goal of facilitating precision medicine [39]. K-means clustering is one of the most popular and simplest unsupervised clustering algorithms. Given the data $\boldsymbol{y}_{1:n} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$, K-means clustering aims to partition the n data points into K clusters such that the data points in the same cluster are more similar with each other than those in other clusters. The objective function of K-means is

$$\mathcal{L}(\boldsymbol{\mu}_{1:K}) = \sum_{i=1}^{n} \min_{1 \le k \le K} \left\| \boldsymbol{y}_i - \boldsymbol{\mu}_k \right\|^2,$$

where $\boldsymbol{\mu}_{1:K} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$ are cluster centroids. Although finding the exact minimizer of the above objective function is NP-hard, efficient EM-type algorithms can be used to find a sequence of iterates that converges to a local optimum with a predefined K.

To select the number of clusters adaptively, one can apply the proposed MB-SPSA to the K-means clustering by considering the following modified objective function:

$$\mathcal{L}(z_{1:\overline{K}}, \boldsymbol{\mu}_{1:\overline{K}}) = \sum_{i=1}^{n} \min_{k:z_k=1} \left\| \boldsymbol{y}_i - \boldsymbol{\mu}_k \right\|^2 + \lambda \log(n) \sum_{k=1}^{\overline{K}} z_k, \tag{6}$$

where $\overline{K}$ is an upper bound for the number of clusters, $z_k \in \{0, 1\}$ is a binary variable to indicate whether the k-th cluster is included in the model or not, and $\boldsymbol{\mu}_k$ is the mean parameter of the k-th cluster. The last term in (6) that mimics the regular BIC-type regularizer is added to prevent overfitting and λ is a penalty parameter. Then Algorithm 1 can be applied directly to (6) for K-means clustering with an unknown number of clusters.

3.2 Gaussian Mixture Models (GMMs)

GMMs have been gaining popularity due to its flexibility and tractability for clustering and density estimation. Consider data $\boldsymbol{y}_{1:n} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ and the log-likelihood function

$$f(\boldsymbol{y}_{1:n}; w_{1:m}, \boldsymbol{\mu}_{1:m}, \boldsymbol{\Sigma}_{1:m}) = \sum_{i=1}^{n} \log \sum_{j=1}^{m} w_j \phi(\boldsymbol{y}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j),$$

where w_j is the weight parameter for the j-th cluster and $\phi(\cdot; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ is the multivariate normal density function with mean $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$. Standard

approaches first estimate the component-specific parameters $(\hat{w}_{1:m}, \hat{\boldsymbol{\mu}}_{1:m}, \hat{\boldsymbol{\Sigma}}_{1:m})$ under every possible value of m, then select the best model dimension as

$$\hat{m} = \arg\min_m -2\sum_{i=1}^{n}\log\sum_{j=1}^{m}\hat{w}_j\phi(\boldsymbol{y}_i;\hat{\boldsymbol{\mu}}_j,\hat{\boldsymbol{\Sigma}}_j) + m\log n,$$

where the last term is the BIC penalty and can be replaced with other information criterion. When possible values of m is large, the two-step procedure can be very inefficient. To overcome this computational challenge, we aim to estimate m and $(w_{1:m}, \boldsymbol{\mu}_{1:m}, \boldsymbol{\Sigma}_{1:m})$ simultaneously by solving the following mixed-binary optimization problem:

$$\begin{aligned}\min_{\substack{z_{1:M},\beta_{1:M}\\ \boldsymbol{\mu}_{1:M},\boldsymbol{L}_{1:M}}} \quad & L(z_{1:M},\beta_{1:M},\boldsymbol{\mu}_{1:M},\boldsymbol{L}_{1:M}) \\ &= -2\sum_{i=1}^{n}\log\sum_{j=1}^{M}\frac{z_j e^{\beta_j}}{\sum_{k=1}^{M} z_k e^{\beta_k}}\phi(\boldsymbol{y}_i;\boldsymbol{\mu}_j,\boldsymbol{L}_j\boldsymbol{L}_j^T) + \log n\sum_{j=1}^{M} z_j,\end{aligned} \tag{7}$$

where M is an upper bound for m, $z_{1:M}$ is a set of indicator variables such that $z_{1:M} \in \{0,1\}^M$, $\beta_{1:M}$ is the unconstrained weight parameter such that $\beta_k \in \mathbb{R}$ and $w_j = z_j e^{\beta_j} / \sum_{k=1}^{M} z_k e^{\beta_k}$, and $\boldsymbol{L}_j \in \mathbb{R}^{p\times p}$ ranges over the space of all lower-triangular p-by-p matrices such that $\boldsymbol{\Sigma}_j = \boldsymbol{L}_j\boldsymbol{L}_j^T$. The binary variable $z_j \in \{0,1\}$ indicates whether the j-th component is included in the model so that $m = \sum_{j=1}^{M} z_j$ represents the model dimension. The advantage of (7) is that the optimization can be performed over all parameters simultaneously. One may notice that a fixed upper bound M is required as an input. In practice, however, one can always run the algorithm at different values of M to see if it is large enough. To make the algorithm fully adaptive, we propose Algorithm 2 that allows an adaptive selection of M based on the current estimates $\hat{z}_{1:M}$. Intuitively, if M is not large enough, then the model space becomes restrictive, resulting $z_j = 1$ for many j's. On the other hand, if M is too large, we are likely to observe $z_j = 0$ for many j's. Hence, by increasing or decreasing the value of M adaptively, we can locate the best model gradually within the fitting process. Moreover, when searching between the models with K components and $K+1$ components, one expects that most of the component-wise parameters (e.g., means and covariance matrices) of the two models are close to each other. Hence, when moving from the current model to a newly updated model, one should be able to use the fitted parameter values in the previous model to achieve a faster convergence result. This idea is detailed in Algorithm 2 below.

Algorithm 2 Adaptively change the model dimension for GMM

Input: model dimension upper bound M; estimate $\hat{z}_{1:M}^{(k)}, \hat{\beta}_{1:M}^{(k)}, \hat{\boldsymbol{\mu}}_{1:M}^{(k)}, \hat{\boldsymbol{L}}_{1:M}^{(k)}$
Output: model dimension upper bound M; estimate $\hat{z}_{1:M}^{(k)}, \hat{\beta}_{1:M}^{(k)}, \hat{\boldsymbol{\mu}}_{1:M}^{(k)}, \hat{\boldsymbol{L}}_{1:M}^{(k)}$

1: **if** $\sum_{j=1}^{M} \pi(\hat{z}_j) = M$ **then**
2: **set** $M \leftarrow M + 1$ and **initialize** additional components in $\hat{z}_{1:M}^{(k)}$, $\hat{\beta}_{1:M}^{(k)}$, $\hat{\boldsymbol{\mu}}_{1:M}^{(k)}$, and $\hat{\boldsymbol{L}}_{1:M}^{(k)}$
3: **else if** $\sum_{j=1}^{M} \pi(\hat{z}_j) < 0.9M$ **then**
4: **find** the largest index j such that $\pi(\hat{z}_j^{(k)}) = 0$
5: **set** $M \leftarrow M - 1$ and **discard** the j-th components in $\hat{z}_{1:M}^{(k)}$, $\hat{\beta}_{1:M}^{(k)}$, $\hat{\boldsymbol{\mu}}_{1:M}^{(k)}$, and $\hat{\boldsymbol{L}}_{1:M}^{(k)}$
6: **end if**
7: **return** $\hat{z}_{1:M}^{(k)}, \hat{\beta}_{1:M}^{(k)}, \hat{\boldsymbol{\mu}}_{1:M}^{(k)}, \hat{\boldsymbol{L}}_{1:M}^{(k)}$

Remark 4 Changing the model dimension by at most 1 per iteration might slow down the algorithm if the initial M is extremely large or small. Our algorithm is flexible enough to allow M to increase or decrease by more than 1 per iteration according to the user's preference without affecting convergence results. For practical use, the algorithm can be implemented in the fashion that allows M to make big changes for early iterations (so that early detection of a neighborhood of the optimal point can be done) and small changes for later iterations (for the purpose of final convergence to the optimal point).

In mini-batch learning settings, we define the noisy loss function based on a subset of the data $\boldsymbol{y}_{1:n_b} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_b})$,

$$\begin{aligned}&\hat{L}(z_{1:M}, \beta_{1:M}, \boldsymbol{\mu}_{1:M}, \boldsymbol{L}_{1:M}) \\ &= -\frac{2n}{n_b} \sum_{i=1}^{n_b} \log \sum_{j=1}^{M} \frac{z_j e^{\beta_j}}{\sum_{k=1}^{M} z_k e^{\beta_k}} \phi(\boldsymbol{y}_i; \boldsymbol{\mu}_j, \boldsymbol{L}_j \boldsymbol{L}_j^T) + \log n \sum_{j=1}^{M} z_j. \end{aligned} \tag{8}$$

Algorithm 3 outlines the (mini-batch) MB-SPSA for GMMs.

Algorithm 3 (mini-batch) MB-SPSA for GMM

Input: upper bound M; initial conditions $\hat{z}_{1:M}^{(0)}, \hat{\beta}_{1:M}^{(0)}, \hat{\boldsymbol{\mu}}_{1:M}^{(0)}, \hat{\boldsymbol{L}}_{1:M}^{(0)}$; gain sequence parameters $a, b, c, A, \alpha, \gamma$; mini-batch size $n \leq N$

Output: final estimates $\hat{m}, \hat{w}_{1:\hat{m}}, \hat{\boldsymbol{\mu}}_{1:\hat{m}}, \hat{\boldsymbol{\Sigma}}_{1:\hat{m}}$

set iteration index $k = 0$
while terminating conditions have not been satisfied **do**
  construct gain sequences $a_k, b_k, c_k, \boldsymbol{C}_k$ and perturbation vector $\boldsymbol{\Delta}_k$
  compute perturbation points by (2.4): $\hat{\boldsymbol{\theta}}_k^{(\pm)} = (\hat{z}_{1:M}^{(k)}, \hat{\beta}_{1:M}^{(k)}, \hat{\boldsymbol{\mu}}_{1:M}^{(k)}, \hat{\boldsymbol{L}}_{1:M}^{(k)})^{(\pm)}$
  choose the proper loss function $\hat{L}$ based on the mini-batch size
  compute new estimate $\hat{\boldsymbol{\theta}}_{k+1}$ by (2.1): $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$
  update iteration index $k \leftarrow k + 1$
  apply Algorithm 2 for updating M
end while
normalize $\{\hat{w}_j : \pi(\hat{z}_j^{(k)}) = 1\}$ and **relabel** the index of $\hat{w}, \hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ to $1, ..., \hat{m}$
return final estimates $\hat{m}, \hat{w}_{1:\hat{m}}, \hat{\boldsymbol{\mu}}_{1:\hat{m}}, \hat{\boldsymbol{\Sigma}}_{1:\hat{m}}$

3.3 Sparse linear models

As an emerging area in biomedical applications, high-dimensional statistics refers to models where the number of parameters is much larger than the sample size in a dataset, i.e., $p \gg n$. For example, in genome-wide association studies (GWAS), researchers usually apply high-dimensional linear regression models to study the association between diseases or traits with SNPs, where the number of SNPs is about 10^6 but the number of individuals is in the range of thousands. A key assumption in high-dimensional models is the sparsity, and an appropriate sparsity assumption is essential to ensure that the inference problem has a well-posed solution.

Consider a linear regression model $y_i = \boldsymbol{x}_i^T \boldsymbol{\beta} + \epsilon_i$, $i = 1, \ldots, n$, with mean-zero noise ϵ_i. A sparse linear model imposes the following sparsity constraint: only $q < p$ regression coefficients are non-zero. We introduce a set of binary variables $z_{1:p} = (z_1, \ldots, z_p) \in \{0, 1\}^p$ such that $z_j = 1$ indicates $\beta_j \neq 0$ and $q = \sum_{j=1}^p z_j$. Using BIC as the model selection criterion, we obtain the loss function

$$\min_{z_{1:p}, \beta_{1:p}} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} z_j \beta_j x_{ij} \right)^2 + \log n \sum_{j=1}^{p} z_j. \tag{9}$$

Classical algorithms are often unable to learn both the sparsity and parameters at the same time since the penalty term $\log N \sum_{j=1}^p z_j$ is proportional to the ℓ_0-penalty function $\|\boldsymbol{\beta}\|_0 = \sum_{j=1}^m \mathbb{1}(z_j = 1)$, which is non-convex and discontinuous. Therefore, high-dimensional linear regression usually relies on the convex relaxation of $\|\boldsymbol{\beta}\|_0$ to the ℓ_1-penalization, giving rise to the least absolute shrinkage selection operator (LASSO) regression [40].

In contrast, the proposed MB-SPSA can optimize with respect to both the sparsity variable $z_{1:p}$ and the regression coefficients $\beta_{1:p}$ efficiently without relaxing the penalty function and is scalable to big data with extremely large N and p. For mini-batch learning, a new loss function can be define similar to (8). We present the detailed implementation of (mini-batch) MB-SPSA for sparse linear models in Algorithm 4.

Algorithm 4 (mini-batch) MB-SPSA for sparse linear models

Input: initial conditions $\hat{z}_{1:p}^{(0)}, \hat{\beta}_{1:p}^{(0)}$; gain sequence parameters $a, b, c, A, \alpha, \gamma$; mini-batch size $n_b \ll n$

Output: final estimates $\hat{z}_{1:p}, \hat{\beta}_{1:p}$

1: **set** iteration index $k = 0$
2: **while** terminating conditions have not been satisfied **do**
3: **construct** gain sequences $a_k, b_k, c_k, \boldsymbol{C}_k$ and perturbation vector $\boldsymbol{\Delta}_k$
4: **compute** perturbation points by (2.4): $\hat{\boldsymbol{\theta}}_k^{(\pm)} = (\hat{z}_{1:p}^{(k)}, \hat{\beta}_{1:p}^{(k)})^{(\pm)}$
5: **choose** the proper loss function $\hat{L}$ based on the mini-batch size
6: **compute** new estimate $\hat{\boldsymbol{\theta}}_{k+1}$ by (2.1): $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$
7: **update** iteration index $k \leftarrow k + 1$
8: **end while**
9: **return** final estimates $\hat{z}_{1:p}, \hat{\beta}_{1:p}$

3.4 Latent Factor Models

Another widely used model that involves an unknown model dimension is the latent factor model, which aims to decompose a data matrix into the product of two low-rank matrices. Mathematically, given a $p \times n$ data matrix $\boldsymbol{Y}$, one seeks to obtain a low-rank approximation $\boldsymbol{Y} \approx \boldsymbol{W}\boldsymbol{H}$ with $\boldsymbol{W}$ being a $p \times K$ matrix and $\boldsymbol{H}$ being a $K \times n$ matrix, where the number of latent factors K is referred to as the model dimension and usually required to be pre-defined for many machine learning algorithms (see, e.g., [41–43]).

The latent factor model has been applied to a wide range of genetics and genomics problems [44–46]. For example, several dimension reduction methods have been introduced in analyzing single-cell RNA-seq data to identify and characterize novel cell types and gene expression patterns before applying clustering methods [47–50]. Specifically, for a count data matrix $\boldsymbol{Y} \in \mathbb{N}^{p \times n}$ with p genes and n cells, one often assumes that $\boldsymbol{Y}$ follows a distribution $\mathscr{P}(\boldsymbol{\Lambda})$ parameterized by a matrix $\boldsymbol{\Lambda}$, then decomposes $\boldsymbol{\Lambda}$ into the product of two low-rank matrices $\boldsymbol{W} \in \mathbb{R}^{+,p,K}$ and $\boldsymbol{H} \in \mathbb{R}^{+,K,n}$, where K is typically much smaller than p and n. Various probability models $\mathscr{P}(\cdot)$ have been proposed, such as the standard Poisson non-negative matrix factorization [51], Gamma–Poisson factor models [47–49], and sparse Gamma–Poisson factor models [52]. However, in these models, the number of latent factors K needs to be pre-defined and fixed during the fitting process.

We now illustrate how one can apply the proposed MB-SPSA algorithm to latent factor models using the model proposed in Sun et al. [53] as an example. Denote $\boldsymbol{Y}$ a gene expression matrix. Specifically, Sun et al. [53] models its element y_{ij} for gene

i and cell j as a negative binomial distribution, i.e., $y_{ij} \sim \text{NB}(\mu_{ij}, \phi_i)$ with the rate parameter μ_{ij} denoting the mean expression level and the parameter ϕ_i representing the gene-specific over-dispersion. The rate parameter μ_{ij} is further assumed to follow a regression model $\log(\mu_{ij}) = \log(N_j) + \sum_{k=1}^{K} W_{ik}H_{kj}$, where N_j is the total read count for the individual cell j, W_{ik} is the loading matrix, and H_{kj} is the factor matrix representing the coordinates of the cells. While the values of W_{ik}'s and H_{kj}'s can be estimated using standard gradient descent methods, the choice of K must be pre-defined and fixed by the user. To estimate the model dimension K adaptively, we propose to use the MB-SPSA with a set of binary variable $\{z_k\} \in \{0, 1\}$ to indicate whether a specific column of $\boldsymbol{W}$ and a row of $\boldsymbol{H}$ is included in the low-rank approximation. Considering the Frobenius norm $\|\cdot\|_F$ as the measurement of an approximation error, we seek to minimize the following objective function

$$\hat{L}(z_{1:\overline{K}}, \boldsymbol{W}, \boldsymbol{H}) = \|\boldsymbol{Y} - \boldsymbol{WZH}\|_F^2 + \lambda \sum_{k=1}^{\overline{K}} z_k, \tag{10}$$

where $\boldsymbol{Z} = \text{diag}(z_1, \ldots, z_{\overline{K}})$ is a diagonal matrix with k-th diagonal being z_k, λ is a penalty parameter, and $\overline{K}$ is an upper bound of the model dimension. If a statistical model is assumed on $\boldsymbol{Y}$, such as the negative binomial distribution mentioned above, one could replace the Frobenius norm term with the corresponding negative log-likelihood function. Then Algorithm 1 can be directly applied with the loss function in (10) to learn the model dimension and parameter values $(z_{1:\overline{K}}, \boldsymbol{W}, \boldsymbol{H})$ simultaneously.

4 Simulation Study

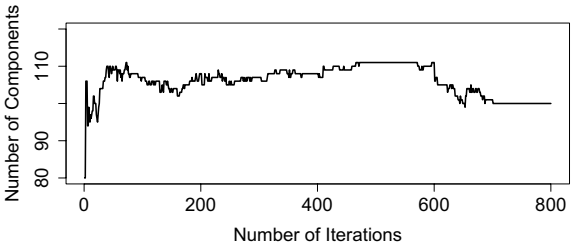
We evaluated the performance of the proposed MB-SPSA using two examples discussed in Sect. 3: Gaussian mixture models and sparse linear models. We also compared the MB-SPSA with alternative methods in terms of accuracy, running time, and scalability.

4.1 Gaussian Mixture Models

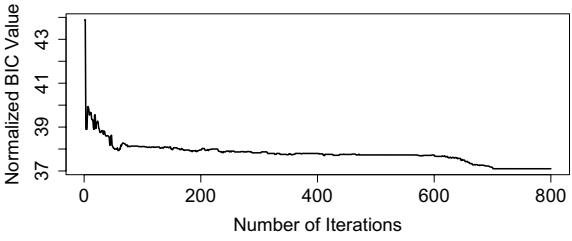
We generated data $\boldsymbol{y}_{1:n}$ from the density: $f(\boldsymbol{y}) = \sum_{j=1}^{100} w_j \phi(\boldsymbol{y}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ with $n = 100,000$, $w_j = 0.01$, $\boldsymbol{\mu}_j = (j, \ldots, j)^T$, and $\boldsymbol{\Sigma}_j = \boldsymbol{I}_{10}$ for all j. We used (7) as the loss function and restricted $\boldsymbol{\Sigma}_j$ to diagonal matrices with unknown diagonal elements. Following Spall [35], we set the gain sequences as $a = 0.1, b = 0.25, c = 0.5, A = 100, \alpha = 0.602, \gamma = 0.101$ and sampled the perturbation vector $\boldsymbol{\Delta}$ independently and uniformly from $\{-1, +1\}$ in the MB-SPSA. Initial means $\hat{\boldsymbol{\mu}}_{1:M}^{(0)}$ were sampled from $\mathcal{N}(0, 10^2)$ and initial covariances $\hat{\boldsymbol{\Sigma}}_{1:M}^{(0)}$ were set to be the identity with $M = 150$. We considered two settings: (i) the MB-SPSA and (ii) the MB-SPSA-mini-batch with a mini-batch size $n_b = 100$. The MB-SPSA was implemented for 800 iterations with 10 parallel runs. For comparison, we applied the R package `mclust`, which uses BIC for learning GMMs via an EM algorithm.

Table 1 Estimated number of components, terminal normalized BIC values, and total running times under true parameters, mclust, the MB-SPSA, and the MB-SPSA-mini-batch algorithms for a GMM with 100,000 data points

Algorithm	No. of comp	Terminal normalized BIC	Time (s)
True parameters	100	37.04	–
mclust	110	37.29	10979
MB-SPSA-batch	100	37.10	3920
MB-SPSA-mini-batch	105	37.28	46



(a) Number of components



(b) Normalized BIC values

Fig. 1 Number of components estimated by the MB-SPSA algorithm over 800 iterations with 100,000 data points

Table 1 shows the estimated number of components, terminal normalized BIC values (the standard BIC value divided by n), and total running times under the simulated true parameters, mclust, the MB-SPSA, and the MB-SPSA-mini-batch, respectively. The MB-SPSA correctly estimates the number of components and yields the best BIC value with only one-third of the time spent in `mclust`. The MB-SPSA-mini-batch also shows a reasonably good result under a significantly shorter time (0.4% of the time in `mclust`). Figure 1 plots the number of estimated components and the corresponding BIC values of the MB-SPSA algorithm versus the number of iterations, showing that the MB-SPSA algorithm gradually converges to the optimal number of components and BIC value.

4.2 Sparse Linear Models

Denote $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$, we generated the data y_i, $i = 1, \ldots, n$, independently using the following linear regression model:

$$y_i = \beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta} + \epsilon_i, \tag{11}$$

where $\epsilon_i \sim N(0, 0.5^2)$. The covariates in (11) were sampled independently from the uniform distribution, i.e., $x_{ij} \sim \text{Unif}(0, 1)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, p$. To evaluate the MB-SPSA algorithm for sparse linear models, we considered two scenarios, one with a large sample size n and the other one with a high dimension p.

In scenario 1, we assumed $n = 100 \times p$ and considered two cases: $p = 100$ and $p = 200$. The simulated true regression coefficients were set to be $\beta_j = 1$ for $j = 0, 1, \ldots, p/2$ and $\beta_j = 0$ for $j = p/2 + 1, \ldots, p$. Therefore, the simulated true number of active covariates was 51 when $p = 200$, and 101 when $p = 200$. We used the loss function (9) with the BIC penalty term $\log(n) \sum_{i=1}^{p} z_j$ to train the MB-SPSA. For comparison, we applied the stepwise regression with both directions (SW-both) [54, 55], which combined the forward and backward selections. Specifically, starting from no active covariates, SW-both sequentially adds the covariate that contributes the most to the model and removes covariates that cannot improve the model fitting. Table 2 shows that both the SW-both and MB-SPSA yield satisfactory results in terms of recovering non-zero β_j's. The SW-both performs slightly better than the proposed MB-SPSA in terms of the BIC since the SW-both is an exact method, while the MB-SPSA relies on stochastic approximation. However, the total running time of the MB-SPSA is much shorter and the speed advantage becomes more significant as p gets larger.

In scenario 2, we examined the performance of the MB-SPSA in high-dimensional case ($p \gg n$). We assumed $n = 100$ and set the regression coefficients to be $\beta_0 = 1, \beta_j = 5$ for $j = 1, \ldots, 5$, and $\beta_j = 0$ for $j = 6, \ldots, p$. We considered two cases: $p = 200$ and $p = 400$. In both cases, the simulated true number of active covariates was 6. The loss function for the MB-SPSA was still chosen to be (9) with the BIC penalty. For comparison, we applied the LASSO regression [40] to the simulated datasets with the penalty parameter chosen by the fivefold cross-validation. Since different loss functions were used for the LASSO and MB-SPSA, we computed the residual sum of squares (RSS) for comparison. Table 3 shows the RSS values and the estimated number of active covariates under the LASSO and MB-SPSA, respectively. Although LASSO achieved smaller RSS values under both cases, it selected a significantly larger number of active covariates compared to that under the MB-SPSA.

Table 2 Terminal BIC values, number of active covariates, and total running times under the SW-both and MB-SPSA for sparse linear models with $n = 100p$ data points

p	Algorithm	BIC	No. of active covariates	Time(s)
100	SW-both	2979.80	51	279.28
	MB-SPSA	2990.98	51	20.07
200	SW-both	5931.78	101	8621.80
	MB-SPSA	5952.76	101	76.57

Table 3 RSS values and the number of active covariates from fitted models under the LASSO and MB-SPSA for sparse linear models

p	Algorithm	RSS	No. of active covariates
200	LASSO	11.01	48
	MB-SPSA	21.61	6
400	LASSO	19.98	25
	MB-SPSA	28.44	10

5 Real Data: Single-Cell RNA-Seq Data Analysis

As mentioned in Sect. 3.4, an important task in the single-cell transcriptome analysis is to identify distinct cell types with different gene expression patterns by clustering cells. Many methods have been proposed for detecting cell types from single-cell RNA-Seq data, such as K-means, iterative clustering [56], or first projecting the high-dimensional data to a lower-dimensional space, then using clustering methods to detect cell types. Commonly used dimension reduction methods include principal component analysis (PCA) [57], non-negative matrix factorization [58], and t-distributed stochastic neighbor embedding algorithm (t-SNE) [59].

We applied the proposed MB-SPSA to detect cell types by clustering cells using a benchmark single-cell RNA-seq dataset [60], which is available at the data repository Gene Expression Omnibus (GSE67853, https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE67835). After initial processing (i.e., filtering out the hybrid cells and the low expression genes, whose total expression over all non-hybrid cells was less or equal than 10), we ended up with $p = 18,568$ genes and $n = 420$ cells over 8 different cell types, including astrocytes cells (62 cells), endothelial cells (20 cells), fetal quiescent cells (110 cells), fetal replicating cells (25 cells), oligodendrocytes cells (38 cells), OPC cells (18 cells), microglia cells (16 cells), and neurons cells (131 cells). Denote y_{ij} to be the count of gene i in cell j, we transformed y_{ij} into continuous data by using base 2 and pseudo count 1, i.e., $\log_2(y_{ij} + 1)$. After the transformation, for each gene, we normalize the values by its total expression over all cells, i.e., $y_{ij} / \sum_{j=1}^{n} y_{ij}$.

Using (6) as the objective function, we implemented the MB-SPSA for K-means clustering with the gain sequence being $a = 0.01, A = 50, \alpha = 0.602, c_z = 0.05$, $c_\mu = 0.001, \gamma = 0.101$. The penalty parameter was set to $\lambda = 0.0004$. The initial number of clusters was set to $\overline{K} = 10$. Figure 2 plots the estimated number of clusters versus the number of iterations, showing that the number of clusters learned by the MB-SPSA successfully converges within 200 iterations to 8, which is the true number of clusters in the dataset.

To measure the performance of the clustering result, we used the normalized mutual information (NMI) [61] and the adjusted rand index (ARI) [62]. Specifically, assume that K_{t} and K_{e} were the true and estimated number of clusters, respectively. For each cluster, denote n_k^{t} to be the number of cells assigned to the k-th true cluster for $k = 1, \ldots, K_{\mathrm{t}}$, and $n_{k'}^{\mathrm{e}}$ to be the number of cells assigned to the k'-th estimated cluster for $k' = 1, \ldots, K_{\mathrm{e}}$. We further denoted $n_{k,k'}$ to be the

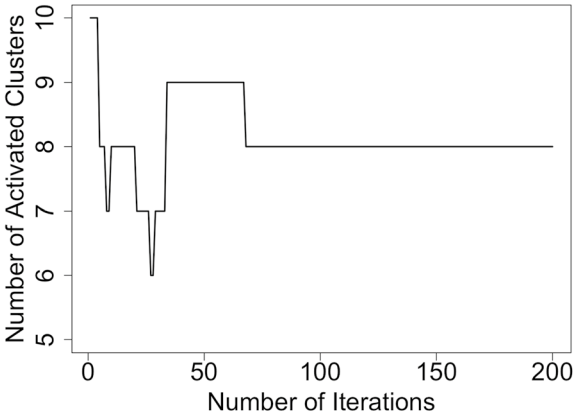


Fig. 2 Number of active clusters estimated by the MB-SPSA

number of cells assigned to both the k-th true cluster and the k'-th estimated cluster. Then the NMI was defined by

$$\mathrm{NMI} = \frac{\sum_{k=1}^{K_t}\sum_{k'=1}^{K_e}\frac{n_{k,k'}}{n}\log\left(\frac{n_{k,k'}}{n}\right) - \sum_{k=1}^{K_t}\frac{n_k^t}{n}\log\left(\frac{n_k^t}{n}\right) - \sum_{t=1}^{K_e}\frac{n_{k'}^e}{n}\log\left(\frac{n_{k'}^e}{n}\right)}{\sqrt{\sum_{k=1}^{K_t}\frac{n_k^t}{n}\log\left(\frac{n_k^t}{n}\right)\sum_{k'=1}^{K_e}\frac{n_{k'}^e}{n}\log\left(\frac{n_{k'}^e}{n}\right)}},$$

and the ARI was defined by

$$\mathrm{ARI} = \frac{\sum_{k=1}^{K_t}\sum_{k'=1}^{K_e}\binom{n_{k,k'}}{2} - \sum_{k=1}^{K_t}\binom{n_k}{2}\sum_{k'=1}^{K_e}\binom{n_{k'}}{2}/\binom{n}{2}}{\left(\sum_{k=1}^{K_t}\binom{n_k}{2} + \sum_{k'=1}^{K_e}\binom{n_{k'}}{2}\right)/2 - \sum_{k=1}^{K_t}\binom{n_k}{2}\sum_{k'=1}^{K_e}\binom{n_{k'}}{2}/\binom{n}{2}}.$$

For comparison, we implemented several alternative methods including the classical K-means, the K-means after PCA, the K-means after non-negative matrix factorization via EM algorithm (NMF-EM) [58], and the K-means after the t-SNE [63]. For K-means, we fixed the number of cluster to be 7, 8 and 9, respectively. For PCA and NMF-EM, we first embedded the original data into the top 10-dimensional feature space, then computed the NMI and ARI values after applying the K-means to the low-dimensional features with the true number of clusters ($K = 8$). For t-SNE, we first embedded the data to the top-2 dimensional feature space, then applied K-means with the true number of clusters ($K = 8$). Table 4 reports the NMI and ARI values under the proposed MB-SPSA as well as all alternative methods, showing that the MB-SPSA yields the highest values of NMI and ARI compared to alternative methods. This example demonstrates that the proposed MB-SPSA outperforms alternative methods even without pre-specifying the correct number of clusters.

Table 4 NMI and ARI values estimated by K-means with different number of clusters, PCA, NMF-EM, t-SNE, and the MB-SPSA

Methods	NMI	ARI
K-means ($K = 7$)	0.729	0.738
K-means ($K = 8$)	0.792	0.787
K-means ($K = 9$)	0.768	0.772
PCA and K-means ($K = 8$)	0.703	0.521
NMF-EM and K-means ($K = 8$)	0.456	0.264
t-SNE and K-means ($K = 8$)	0.742	0.572
MB-SPSA	0.808	0.829

6 Conclusion

We developed an automated and scalable MB-SPSA algorithm to simultaneously learn the model dimension and parameters for a general class of statistical models. A mini-batch learning framework of the MB-SPSA is also discussed when evaluating the loss function on the complete data and/or the gradient information is computationally expensive or infeasible. Theoretically, the MB-SPSA is guaranteed to converge locally under certain regularity conditions. Applications including K-means clustering, GMMs with an unknown number of components, sparse linear models with an unknown number of active covariates, and latent factor models with an unknown number of factors are demonstrated. Through simulation studies and real data analyses, the MB-SPSA yields favorable results compared to alternatives in terms of efficiency and accuracy.

There are many other potential applications of the proposed MB-SPSA algorithm, mainly in dimensionality reduction, including but not limited to, low-rank random graph models [64], manifold learning [65], and topic models [66]. The common question of interest in these dimensionality reduction problems is how to identify the dimension of the latent space. For example, in low-rank random graph models, the dimension of the latent positions could be unknown. In manifold learning, the intrinsic dimension of the underlying unobserved manifold could be unknown. In topic models, it is impractical to assume a pre-determined number of topics for a collection of documents. Applying the proposed MB-SPSA algorithm to learn the latent space dimension for various dimensionality reduction problems with appropriate loss functions is an interesting future research direction.

Appendix: Proof of Theorem 1

Proof Denote $L(\hat{\boldsymbol{\theta}}_k^{(+)}) = L_k^{(+)}$ and $L_k^{(-)} = L(\hat{\boldsymbol{\theta}}_k^{(-)})$. Before starting the main proof, we first define some useful notations below

$$\bar{\boldsymbol{g}}_k = \mathbb{E}\big[\hat{\boldsymbol{g}}_k \mid \hat{\boldsymbol{\theta}}_k\big], \tag{12}$$

$$\hat{\boldsymbol{b}}_k = \frac{1}{2}(\boldsymbol{C}_k \circ \boldsymbol{\Delta}_k)\left[L_k^{(+)} - L_k^{(-)}\right] - \bar{\boldsymbol{g}}_k, \tag{13}$$

$$\hat{\boldsymbol{e}}_k = \hat{\boldsymbol{g}}_k - \frac{1}{2}(\boldsymbol{C}_k \circ \boldsymbol{\Delta}_k)\left[L_k^{(+)} - L_k^{(-)}\right] = \frac{1}{2}(\boldsymbol{C}_k \circ \boldsymbol{\Delta}_k)\left[\epsilon_k^{(+)} - \epsilon_k^{(-)}\right], \tag{14}$$

where the expectation in (12) is taken over both the perturbation vector $\boldsymbol{\Delta}_k$ and the noise term ϵ_k. Using (12), (13) and (14), we can write the updating equation as

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{k+1} &= \hat{\boldsymbol{\theta}}_k - a_k \hat{\boldsymbol{g}}_k \\ &= \hat{\boldsymbol{\theta}}_k - a_k\left(\frac{1}{2}(\boldsymbol{C}_k \circ \boldsymbol{\Delta}_k)\left[L_k^{(+)} - L_k^{(-)}\right] + \frac{1}{2}(\boldsymbol{C}_k \circ \boldsymbol{\Delta}_k)\left[\epsilon_k^{(+)} - \epsilon_k^{(-)}\right]\right) \\ &= \hat{\boldsymbol{\theta}}_k - a_k\left(\bar{\boldsymbol{g}}_k + \hat{\boldsymbol{b}}_k + \bar{\boldsymbol{g}}_k\right).\end{aligned} \tag{15}$$

For any $\omega \in \Omega_0$ such that $P(\Omega_0) = 1$, since $\{\hat{\boldsymbol{\theta}}_k(\omega)\}$ is a bounded sequence by Assumption 2, the Bolzano–Weierstrass Theorem implies that there exists $\Omega_1 \subset \Omega$ such that $P(\Omega_1) = 1$ and for any $\omega \in \Omega_1$ there exists a convergent subsequence $\{\hat{\boldsymbol{\theta}}_{k_s}(\omega)\}$. Denote the limiting point of the convergent subsequence as $\boldsymbol{\theta}'(\omega)$. For simplicity, the notation ω is suppressed below.

According to (15), we can write

$$\boldsymbol{\theta}' - \hat{\boldsymbol{\theta}}_{k_s} = \lim_{s\to\infty}\sum_{i=s}^{n}(\hat{\boldsymbol{\theta}}_{k_{i+1}} - \hat{\boldsymbol{\theta}}_{k_i}) = -\lim_{s\to\infty}\sum_{i=s}^{n} a_{k_i}\left(\bar{\boldsymbol{g}}_{k_i} + \hat{\boldsymbol{b}}_{k_i} + \bar{\boldsymbol{g}}_{k_i}\right), \tag{16}$$

Since $\boldsymbol{\theta}' - \hat{\boldsymbol{\theta}}_{k_s} \to \mathbf{0}$ as $s \to \infty$, we will show below that all the three terms of the right-hand side of (16) must also converge to $\mathbf{0}$.

First note that by Assumption 3 and (13), we have

$$\mathbb{E}\left[\hat{\boldsymbol{b}}_{k+1} \mid \mathcal{F}_k\right] = \mathbf{0},$$

which implies that $\{\sum_{i=k}^{m} a_i \boldsymbol{b}_i\}_{m\geq k}$ is a martingale sequence as

$$\mathbb{E}\left[\sum_{i=k}^{m+1} a_i \hat{\boldsymbol{b}}_i \mid \mathcal{F}_m\right] = \sum_{i=k}^{m} a_i \hat{\boldsymbol{b}}_i + a_{m+1}\mathbb{E}\left[\hat{\boldsymbol{b}}_{m+1} \mid \mathcal{F}_m\right] = \sum_{i=k}^{m} a_i \hat{\boldsymbol{b}}_i.$$

Given that $\{\sum_{i=k}^{m} a_i \boldsymbol{b}_i\}_{m\geq k}$ is a martingale sequence, the Doob's martingale inequality implies that for any $\eta > 0$

$$P\left(\sup_{m\geq k}\left\|\sum_{i=k}^{m} a_i \hat{\boldsymbol{b}}_i\right\| \geq \eta\right) \leq \eta^{-2}\mathbb{E}\left[\left\|\sum_{i=k}^{\infty} a_i \hat{\boldsymbol{b}}_i\right\|^2\right] = \eta^{-2}\sum_{i=k}^{\infty} a_i^2 \mathbb{E}\left[\left\|\hat{\boldsymbol{b}}_i\right\|^2\right], \tag{17}$$

where the last equality is due to Assumption 3, since

$$\mathbb{E}\left[\hat{\boldsymbol{b}}_i^T \hat{\boldsymbol{b}}_i\right] = \mathbb{E}\left[\mathbb{E}\left[\hat{\boldsymbol{b}}_i^T \hat{\boldsymbol{b}}_i \mid \mathcal{F}_j, \mathcal{G}_{j-1}\right]\right] = \mathbb{E}\left[\hat{\boldsymbol{b}}_i^T \mathbb{E}\left[\hat{\boldsymbol{b}}_i \mid \mathcal{F}_j\right]\right] = 0.$$

By Assumption 1, we have $b_k > 0$ and $c_k \le c_0$. Hence, there exist a constant $\bar{c}$ such that we can write (17) as

$$\sum_{i=k}^{\infty} a_i^2 \mathbb{E}\left[\left\|\hat{\boldsymbol{b}}_i\right\|^2\right] \le \sum_{i=k}^{\infty} a_i^2 \mathbb{E}\left[\left(L_k^{(+)} - L_k^{(-)}\right)^2 (2\boldsymbol{C}_i \circ \boldsymbol{\Delta}_i)^{-T} (2\boldsymbol{C}_i \circ \boldsymbol{\Delta}_i)^{-1}\right]$$
$$\le \sum_{i=k}^{\infty} \bar{c}^2 \frac{a_i^2}{c_i^2} \mathbb{E}\left[\left(L_k^{(+)} - L_k^{(-)}\right)^2 \boldsymbol{\Delta}_i^{-T} \boldsymbol{\Delta}_i^{-1}\right] < \infty,$$

which further implies that

$$\lim_{k\to\infty} a_i^2 \mathbb{E}\left[\left\|\hat{\boldsymbol{b}}_i\right\|^2\right] = 0.$$

For any $\eta > 0$ and all $k \ge n$, since

$$\left\{\sup_k \left\|\sum_{i=k}^{m} a_i \hat{\boldsymbol{b}}_i\right\| \ge \eta\right\} \subset \left\{\sup_{m\ge k} \left\|\sum_{i=k}^{m} a_i \hat{\boldsymbol{b}}_i\right\| \ge \eta\right\},$$

we can use (17) to get

$$P\left(\sup_k \left\|\sum_{i=k}^{m} a_i \hat{\boldsymbol{b}}_i\right\| \ge \eta\right) \le P\left(\sup_{m\ge k} \left\|\sum_{i=k}^{m} a_i \hat{\boldsymbol{b}}_i\right\| \ge \eta\right) \le \eta^{-2} \sum_{i=k}^{\infty} a_i^2 \mathbb{E}\left[\left\|\hat{\boldsymbol{b}}_i\right\|^2\right].$$

As $n \to \infty$, for all $k \ge n$,

$$\lim_{k\to\infty} \eta^{-2} \sum_{i=k}^{\infty} a_i^2 \mathbb{E}\left[\left\|\hat{\boldsymbol{b}}_i\right\|^2\right] = 0,$$

and

$$\lim_{n\to\infty} P\left(\sup_{k\ge n} \left\|\sum_{i=k}^{\infty} a_i \hat{\boldsymbol{b}}_i\right\| \ge \eta\right) = 0.$$

Therefore, we conclude

$$\lim_{k\to\infty} \sum_{i=k}^{\infty} a_i \hat{\boldsymbol{b}}_i = \mathbf{0},$$

and

$$\lim_{s\to\infty} \sum_{i=s}^{\infty} a_{k_i} \hat{\boldsymbol{b}}_{k_i} = \mathbf{0}. \tag{18}$$

Similarly, we can also show that

$$\lim_{s\to\infty}\sum_{i=s}^{\infty} a_{k_i}\hat{\boldsymbol{e}}_{k_i} = \mathbf{0}. \tag{19}$$

Combining (16) with results in (18) and (19), we have

$$\lim_{s\to\infty}\sum_{i=s}^{n} a_{k_i}\bar{\boldsymbol{g}}_{k_i} = \mathbf{0}. \tag{20}$$

Suppose $\boldsymbol{\theta}' \neq \boldsymbol{\theta}^*$. Given $\lim_{s\to\infty}\hat{\boldsymbol{\theta}}_{k_s} = \boldsymbol{\theta}'$, for any $\delta > 0$, there exists a S such that for any $s > S$, $\|\hat{\boldsymbol{\theta}}_{k_s} - \boldsymbol{\theta}'\| \leq \delta$. Let δ be sufficiently small, we have $\hat{\boldsymbol{\theta}}_{k_s} \in B_r(\boldsymbol{\theta}')$. By Assumption 1 and 6, we must have $\sum_{i=s}^{\infty} a_{k_i} = \infty$ implies

$$\lim_{s\to\infty}\sum_{i=s}^{n} a_{k_i}\bar{\boldsymbol{g}}_{k_i}^T(\boldsymbol{\theta}' - \boldsymbol{\theta}^*) = \infty,$$

which contradicts with (20). Hence, we conclude that $\boldsymbol{\theta}' = \boldsymbol{\theta}^*$. Since $\boldsymbol{\theta}'$ is chosen to be the limiting point of any convergent subsequence, we have all the convergent subsequence converges to the same liming points and consequently $\hat{\boldsymbol{\theta}}_k \to \boldsymbol{\theta}^*$ a.s. when $k \to \infty$. □

Acknowledgements The work of Xu was supported by NSF 1918854 and NSF 1940107.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Visscher PM, Brown MA, McCarthy MI, Yang J (2012) Five years of GWAS discovery. Am J Hum Genet 90(1):7–24
2. Cao J, Spielmann M, Qiu X, Huang X, Ibrahim DM, Hill AJ, Zhang F, Mundlos S, Christiansen L, Steemers FJ et al (2019) The single-cell transcriptional landscape of mammalian organogenesis. Nature 566(7745):496–502
3. Richardson S, Green PJ (1997) On Bayesian analysis of mixtures with an unknown number of components (with discussion). J R Stat Soc Ser B (Methodol) 59(4):731–792
4. Bhattacharya A, Dunson DB (2011) Sparse Bayesian infinite factor models. Biometrika 98(2):291–306
5. Athreya A, Fishkind DE, Tang M, Priebe CE, Park Y, Vogelstein JT, Levin K, Lyzinski V, Qin Y (2017) Statistical inference on random dot product graphs: a survey. J Mach Learn Res 18(1):8393–8484
6. Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19(6):716–723
7. Schwarz G et al (1978) Estimating the dimension of a model. Ann Stat 6(2):461–464
8. Zoph B et al (2018) Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on computer vision and pattern recognition, pp 8697–8710
9. Real E, Aggarwal A, Huang Y, Le QV (2019) Regularized evolution for image classifier architecture search. Proc AAAI Conf Artif Intell 33:4780–4789

10. Green PJ (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 82(4):711–732
11. Antoniak CE (1974) Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. Ann Stat 2(6):1152–1174
12. Neal RM (2000) Markov chain sampling methods for Dirichlet process mixture models. J Comput Graph Stat 9(2):249–265
13. Ghahramani Z, Griffiths TL (2006) Infinite latent feature models and the Indian buffet process. In: Advances in neural information processing systems, pp 475–482
14. Walker SG (2007) Sampling the dirichlet mixture model with slices. Commun Stat-Simul Comput 36(1):45–54
15. Blei DM, Jordan MI et al (2006) Variational inference for dirichlet process mixtures. Bayesian Anal 1(1):121–143
16. Markley SC, Miller DJ (2010) Joint parsimonious modeling and model order selection for multivariate gaussian mixtures. IEEE J Select Top Signal Proces 4(3):548–559
17. Huang T, Peng H, Zhang K (2017) Model selection for Gaussian mixture models. Stat Sin 27(1):147–169
18. Bertsimas D, King A, Mazumder R et al (2016) Best subset selection via a modern optimization lens. Ann Stat 44(2):813–852
19. Miyashiro R, Takano Y (2015) Mixed integer second-order cone programming formulations for variable selection in linear regression. Eur J Oper Res 247(3):721–731
20. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12:2121–2159
21. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. http://arxiv.org/abs/1412.6980
22. Spall JC (1992) Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Trans Autom Control 37(3):332–341
23. Alessandri A, Parisini T (1997) Nonlinear modeling of complex large-scale plants using neural networks and stochastic approximation. IEEE Trans Syst Man Cybern A Syst Hum 27(6):750–757
24. Balakrishna R, Antoniou C, Ben-Akiva M, Koutsopoulos HN, Wen Y (2007) Calibration of microscopic traffic simulation models: methods and application. Transp Res Rec 1999(1):198–207
25. Kocsis L, Szepesvári C (2006) Universal parameter optimisation in games based on spsa. Mach Learn 63(3):249–286
26. Sidorov KA, Richmond S and Marshall D (2009) An efficient stochastic approach to groupwise non-rigid image registration. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2208–2213
27. Wang L, Zhu J and Spall JC (2018) Mixed simultaneous perturbation stochastic approximation for gradient-free optimization with noisy measurements. In Proceedings of the annual american control conference, pp 3774–3779
28. Tympakianaki A, Koutsopoulos HN, Jenelius E (2015) C-SPSA: cluster-wise simultaneous perturbation stochastic approximation algorithm and its application to dynamic origin-destination matrix estimation. Transp Res C Emerg Technol 55:231–245
29. Dong N, Wu C-H, Gao Z-K, Chen Z-Q, Ip W-H (2016) Data-driven control based on simultaneous perturbation stochastic approximation with adaptive weighted gradient estimation. IET Control Theory Appl 10(2):201–209
30. Lorenz R, Monti RP, Violante IR, Anagnostopoulos C, Faisal AA, Montana G, Leech R (2016) The automatic neuroscientist: a framework for optimizing experimental design with closed-loop real-time fmri. Neuroimage 129:320–334
31. Alaeddini A, Klein DJ (2017) Application of a second-order stochastic optimization algorithm for fitting stochastic epidemiological models. In: Proceedings of the winter simulation conference, pp 2194–2206
32. Khatami A, Nazari A, Khosravi A, Lim CP, Nahavandi S (2020) A weight perturbation-based regularisation technique for convolutional neural networks and the application in medical imaging. Expert Syst Appl 149:113196
33. Aksakalli V, Malekipirbazari M (2016) Feature selection via binary simultaneous perturbation stochastic approximation. Pattern Recogn Lett 75:41–47

34. Dennis J Jr, Schnabel RB (1989) Chapter ia view of unconstrained optimization. Handb Oper Res Manage Sci 1:1–72
35. Spall JC (1998) Implementation of the simultaneous perturbation algorithm for stochastic optimization. IEEE Trans Aerosp Electron Syst 34(3):817–823
36. Bottou L, Cun YL (2004) Large scale online learning. In: Proceedings of the advances in neural information processing systems, pp 217–224
37. Spall JC (2005) Introduction to stochastic search and optimization: estimation, simulation, and control, vol 65. Wiley, Berlin
38. Shukla AK, Muhuri PK (2019) Big-data clustering with interval type-2 fuzzy uncertainty modeling in gene expression datasets. Eng Appl Artif Intell 77:268–282
39. de la Fuente-Tomas L, Arranz B, Safont G, Sierra P, Sanchez-Autet M, Garcia-Blanco A, Garcia-Portilla MP (2019) Classification of patients with bipolar disorder using k-means clustering. PLoS ONE 14(1):e0210314
40. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J Roy Stat Soc Ser B (Methodol) 58(1):267–288
41. Brunet J-P, Tamayo P, Golub TR, Mesirov JP (2004) Metagenes and molecular pattern discovery using matrix factorization. Proc Natl Acad Sci 101(12):4164–4169
42. Lee DD, Seung HS (2001) Algorithms for non-negative matrix factorization. In: Proceedings of the advances in neural information processing systems, pp 556–562
43. Pascual-Montano A, Carazo JM, Kochi K, Lehmann D, Pascual-Marqui RD (2006) Nonsmooth nonnegative matrix factorization (nsnmf). IEEE Trans Pattern Anal Mach Intell 28(3):403–415
44. Alexandrov LB, Nik-Zainal S, Wedge DC, Campbell PJ, Stratton MR (2013) Deciphering signatures of mutational processes operative in human cancer. Cell Rep 3(1):246–259
45. Frichot E, Mathieu F, Trouillon T, Bouchard G, François O (2014) Fast and efficient estimation of individual ancestry coefficients. Genetics 196(4):973–983
46. Stein-O'Brien GL, Arora R, Culhane AC, Favorov AV, Garmire LX, Greene CS, Goff LA, Li Y, Ngom A, Ochs MF et al (2018) Enter the matrix: factorization uncovers knowledge from omics. Trends Genet 34(10):790–805
47. Cemgil AT (2009) Bayesian inference for nonnegative matrix factorisation models. Comput Intell Neurosci 2009:785152
48. Févotte C, Cemgil AT (2009) Nonnegative matrix factorizations as probabilistic inference in composite models. In: Proceedings of the European signal processing conference, pp 1913–1917
49. Landgraf AJ, Lee Y (2020) Generalized principal component analysis: projection of saturated model parameters. Technometrics 62(4):459–472
50. Zhang S et al (2020) Review of single-cell rna-seq data clustering for cell type identification and characterization. http://arxiv.org/abs/2001.01006
51. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788–791
52. Durif G, Modolo L, Mold JE, Lambert-Lacroix S, Picard F (2019) Probabilistic count matrix factorization for single cell expression data analysis. Bioinformatics 35(20):4011–4019
53. Sun S, Chen Y, Liu Y, Shang X (2019) A fast and efficient count-based matrix factorization method for detecting cell types from single-cell rnaseq data. BMC Syst Biol 13(2):28
54. Bruce P, Bruce A (2017) Practical statistics for data scientists: 50 essential concepts. O'Reilly Media, Inc, Newton
55. James G, Witten D, Hastie T, Tibshirani R (2013) An introduction to statistical learning, vol 112. Springer, New York
56. Yang L, Liu J, Lu Q, Riggs AD, Wu X (2017) SAIC: an iterative clustering approach for analysis of single cell RNA-seq data. BMC Genomics 18(6):689
57. Jiang L, Chen H, Pinello L, Yuan G-C (2016) Giniclust: detecting rare cell types from single-cell gene expression data with gini index. Genome Biol 17(1):144
58. Zhu X, Ching T, Pan X, Weissman SM, Garmire L (2017) Detecting heterogeneity in single-cell rna-seq data by non-negative matrix factorization. PeerJ 5:e2888
59. Linderman GC, Rachh M, Hoskins JG, Steinerberger S, Kluger Y (2019) Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. Nat Methods 16(3):243–245
60. Darmanis S, Sloan SA, Zhang Y, Enge M, Caneda C, Shuer LM, Gephart MGH, Barres BA, Quake SR (2015) A survey of human brain transcriptome diversity at the single cell level. Proc Natl Acad Sci 112(23):7285–7290

61. Ghosh J, Acharya A (2011) Cluster ensembles. Wiley Interdiscipl Rev Data Mining Knowl Discov 1(4):305–315
62. Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218
63. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
64. Xie F, Xu Y (2019) Optimal Bayesian estimation for random dot product graphs. http://arxiv.org/abs/1904.12070
65. Huang H, Shi G, He H, Duan Y, Luo F (2019) Dimensionality reduction of hyperspectral imagery based on spatial-spectral manifold learning. IEEE Trans Cybern 50(6):2604–2616
66. Bing X, Bunea F, Wegkamp M et al (2020) A fast algorithm with minimax optimal guarantees for topic models with an unknown number of topics. Bernoulli 26(3):1765–1796