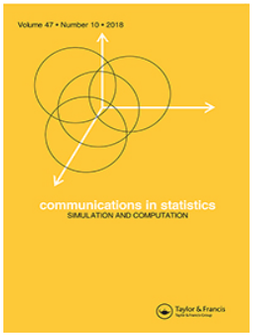


Taylor & Francis
Taylor & Francis Group

Communications in Statistics - Simulation and Computation

ISSN: 0361-0918 (Print) 1532-4141 (Online) Journal homepage: https://www.tandfonline.com/loi/lssp20

Beyond the EM algorithm: constrained optimization methods for latent class model

Hao Chen, Lanshan Han & Alvin Lim

To cite this article: Hao Chen, Lanshan Han & Alvin Lim (2020): Beyond the EM algorithm: constrained optimization methods for latent class model, Communications in Statistics - Simulation and Computation, DOI: 10.1080/03610918.2020.1764034

To link to this article: https://doi.org/10.1080/03610918.2020.1764034

COMMUNICATIONS IN STATISTICS - SIMULATION AND COMPUTATION®
https://doi.org/10.1080/03610918.2020.1764034

Beyond the EM algorithm: constrained optimization methods for latent class model

Hao Chen, Lanshan Han, and Alvin Lim

Research & Development, Precima, Chicago, IL, USA

ABSTRACT

Latent class model (LCM), which is a finite mixture of different categorical distributions, is one of the most widely used models in statistics and machine learning fields. Because of its noncontinuous nature and flexibility in shape, researchers in areas such as marketing and social sciences also frequently use LCM to gain insights from their data. One likelihood-based method, the expectation–maximization (EM) algorithm, is often used to obtain the model estimators. However, the EM algorithm is well-known for its notoriously slow convergence. In this research, we explore alternative likelihood-based methods that can potential remedy the slow convergence of the EM algorithm. More specifically, we regard likelihood-based approach as a constrained nonlinear optimization problem, and apply quasi-Newton type methods to solve them. We examine two different constrained optimization methods to maximize the log-likelihood function. We present simulation study results to show that the proposed methods not only converge in less iterations than the EM algorithm but also produce more accurate model estimators.

ARTICLE HISTORY
Received 8 May 2019
Accepted 28 April 2020

KEYWORDS
Constrained optimization; EM algorithm; Finite mixture model; Latent class model; Quasi-Newton's method; Quadratic programing

1. Introduction

Latent class model (LCM) (McCutcheon 1987) is a model to study latent (unobserved) categorical variables by examining a group of observed categorical variables which are regarded as the indictors of the underlying latent variables. It can be regarded as a special case of the finite mixture model (FMM) with component distributions being categorical distributions. It is widely used to analyze ordered survey data collected from real world applications. In many applications in econometrics, social sciences, biometrics, and business analytics (see for example, Hagenaars and McCutcheon 2002; Oser, Hooghe, and Marien 2013), finite mixture of categorical distributions arises naturally when we sample from a population with heterogeneous subgroups. LCM is a powerful tool to conduct statistical inference from the collected data in such situations.

We provide a motivating example from White and Murphy (2014) where an LCM is applied to analyze a dataset of patient symptoms recorded in the Mercer Institute of St. James' Hospital in Dublin, Ireland (Moran et al. 2004). The data is a recording of the presence of six symptoms displayed by 240 patients diagnosed with early onset Alzheimer's disease. The six symptoms are as follows: hallucination, activity, aggression,

CONTACT Hao Chen hao.chen@stat.ubc.ca Research & Development, Precima, Chicago, IL 60631, USA.

agitation, diurnal, and affective, and each symptom has two states: either present or absent. White and Murphy (2014) proposed to divide patients into $K = 3$ groups such that patients are homogeneous within each group and heterogeneous between groups. Each group's characteristics are summarized by the LCM parameters that help doctors prepare more specialized treatments. In this sense, LCM is a typical unsupervised statistical learning method that could "learn" the group labels based on the estimated parameters.

Due to its theoretical importance and practical relevance, many different approaches have been proposed to estimate the unknown parameters in LCMs from the observed data. In general, there are mainly two different paradigms. The first one is the frequentist's approach of maximum likelihood estimation (MLE), i.e. one maximizes the log-likelihood as a function of the unknown parameters. In contrast, a second paradigm – the Bayesian approach – where the unknown parameters obey a distribution and assumes prior distributions on them, then one either analytically or numerically obtains the posterior distributions and statistical inference is carried out based on the posterior distributions.

In recent years, significant progress has been made on Bayesian inference in LCM. White and Murphy (2014), by assuming the Dirichlet distribution on each unknown parameter, used Gibbs sampling to iteratively draw samples from the posterior distribution and then conduct inference on the LCM using the samples drawn. The authors also provided an implementation of the approach in Li et al. (2018) described a similar Bayesian approach to estimate the parameters and they also utilized the Dirichlet distribution as the prior distribution. Asparouhov and Muthén (2011) introduced a similar implementation package of Bayesian LCM in Mplus. However, compared to the fast development of the Bayesian inference via Markov chain Monte Carlo (MCMC), the frequentist's MLE approach for LCM has largely lagged. As far as we know, researchers still heavily rely on the expectation–maximization (EM) algorithm (Dempster, Laird, and Rubin 1977), even with its notoriously slow convergence (see for instance Meilijson (1989)), to maximize the log-likelihood function. It is known that some authors (Jamshidian and Jennrich 1997) use quasi-Newton methods as alternatives for the EM algorithm in Gaussian mixture models. However, the extension to LCM is not straightforward since LCM includes a lot more intrinsic constraints on the parameters than the general Gaussian mixture model when considered as an optimization problem. More sophisticated optimization methods need to be applied when maximizing the log-likelihood function.

This paper primarily focuses on the MLE paradigm. We propose the use of two widely used constrained optimization methods to maximize the likelihood function, namely, the projected quasi-Newton method and the sequential quadratic programing (SQP) method. Our contributions include not only exploring alternatives beyond the EM algorithm, but also demonstrating that better results could be obtained by using these alternatives. The rest of this paper is organized as follows: in Sec. 2, we present the preliminaries including the log-likelihood function and the classical EM algorithm. In Sec. 3, we introduce and discuss the two constrained optimization methods in detail. Some simulation studies and a real world data analysis are presented in Sec. 4 to compare the performance of the proposed methods with the EM algorithm. We make concluding remarks in Sec. 6.

2. Latent class models and the EM algorithm

In many applications, a finite mixture distribution arises naturally when we sample from a population with heterogeneous subgroups, indexed by k taking values in $\{1, ..., K\}$. Consider a population composed of K subgroups, mixed at random in proportion to the relative group sizes $\eta_1, ..., \eta_K$. There is a random feature y, heterogeneous across and homogeneous within the subgroups. The feature y obeys a different probability distribution, often from the same parametric family $p(y|\theta)$ with θ differing, for each subgroup. Now we sample from this population, if it is impossible to record the subgroup label, denoted by s, then the density $p(y)$ is:

$$p(y) = \sum_{k=1}^{K} \eta_k p(y|\theta_k),$$

which is a finite mixture distribution. In this situation, we often need to estimate the θ_k's as well as η_k based on the random samples of y, when the subgroup label s is known or unknown. Throughout this paper, we assume that K is known.

The LCM is a special case of the FMM. In LCM, the component densities are multivariate categorical distributions. That is, $\boldsymbol{y} = (\boldsymbol{y}_1, ..., \boldsymbol{y}_d)$ with each $\boldsymbol{y}_j$ being a categorical random variable, taking values from c_j categories $\{1, ..., c_j\}$. It is assumed that $\boldsymbol{y}_j$'s are independent within each subgroup with an indictor s (the latent variable), which is a categorical random variable taking values in $\{1, ..., K\}$, i.e. within each subgroup, the probability density function (PDF) is written as:

$$\prod_{j=1}^{d} \prod_{l=1}^{c_j} \pi_{k,j,l}^{\mathcal{I}(\boldsymbol{y}_j = l)},$$

where $\pi_{k,j,l} = \Pr(\boldsymbol{y}_j = l | s = k)$ and $\mathcal{I}(\cdot)$ is the Iverson bracket function, i.e.

$$\mathcal{I}(\mathrm{P}) = \begin{cases} 1 & \text{if P is true;} \\ 0 & \text{if P is false.} \end{cases}$$

Overall, the mixture density of LCMs is

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \left(\eta_k \prod_{j=1}^{d} \prod_{l=1}^{c_j} \pi_{k,j,l}^{\mathcal{I}(\boldsymbol{y}_j = l)} \right),$$

where, the parameters $\boldsymbol{\theta}$ include both the weight distribution η and the $\pi_{k,j,l}$'s that define the categorical distributions.

Suppose we have collected N samples drawn from the LCM distribution, denoted by $\{y^1, ..., y^N\}$. We write $Y = \left[y^1, ..., y^N\right]^{\mathrm{T}} \in \mathbb{R}^{N \times d}$ as the data matrix. The log-likelihood function is given by

$$
\begin{aligned}
L(\boldsymbol{\theta}|Y) &= \log\left(\prod_{i=1}^{N} p(\boldsymbol{y}^i|\boldsymbol{\theta})\right) \\
&= \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \eta_k \prod_{j=1}^{d}\prod_{l=1}^{c_d} \pi_{k,j,l}^{\mathcal{I}(y_j^i=l)}\right).
\end{aligned} \tag{1}
$$

The maximum likelihood principle is to find a $\boldsymbol{\theta}^*$ that maximizes the log-likelihood function (1) as the estimation of $\boldsymbol{\theta}$. Clearly, we can regard the problem of finding such a $\boldsymbol{\theta}^*$ as an optimization problem. At the same time, we notice that the LCM implies several constraints that need to be satisfied when maximizing the log-likelihood function (1). In particular, the η_k's are all nonnegative and sum up to 1. Also, for each $k = 1, ..., K$ and $j = 1, ..., d$, the $\pi_{k,j,l}$'s are all nonnegative and sum up to 1. Let $\eta = (\eta_k)_{k=1}^{K}$ be the vector of η_k's and $\pi = (\pi_{k,j,l})_{k=1,...,K;j=1,...,d;l=1,...,c_j}$ be the vector of $\pi_{k,j,l}$'s. From an optimization point of view, the MLE in the LCM case is the following optimization problem.

$$
\begin{aligned}
\max_{\eta,\,\pi} \quad & \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \eta_k \prod_{j=1}^{d}\prod_{l=1}^{c_d} \pi_{k,j,l}^{\mathcal{I}(y_j^i=l)}\right) \\
\text{s.t.} \quad & \sum_{k=1}^{K} \eta_k = 1, \\
& \sum_{l=1}^{c_j} \pi_{k,j,l} = 1, \ \forall\ k = 1, ..., K, j = 1, ..., d, \\
& \eta_k \geq 0, \forall\ k = 1, ..., K, \\
& \pi_{k,j,l} \geq 0, \forall\ k = 1, ..., K, j = 1, ..., d, l = 1, ..., c_j.
\end{aligned} \tag{2}
$$

As we can see, the optimization problem (2) possesses $K \times d + 1$ equality constraints together with nonnegativity constraints on all the individual decision variables. While there are considerable number of constraints, the feasible region in (2) is indeed the Cartesian product of $K \times d + 1$ probability simplexes. We recall that a probability simplex in n-dimensional space $\mathbb{R}^n$ is defined as

$$
\mathcal{P}^n = \left\{ x \in \mathbb{R}^n_+ \middle| \sum_{i=1}^{n} x_i = 1 \right\},
$$

where $\mathbb{R}^n_+$ is the nonnegative orthant of $\mathbb{R}^n$. Let $\pi_{k,j} = (\pi_{k,j,l})_{l=1}^{c_j}$ for all $k = 1, ..., K$ and $j = 1, ..., d$. The constraints in (2) can be written as $\eta \in \mathcal{P}^K$ and $\pi_{k,j} \in \mathcal{P}^{c_j}$, $\forall\ k = 1, ..., K; j = 1, ..., d$.

To maximize the log-likelihood function in (1), the EM algorithm is a classical approach. In statistics, the EM algorithm is a generic framework that is commonly used in obtaining maximum likelihood estimators. The reason why the EM algorithm enjoys its popularity in FMM is the fact that we can view FMM as an estimation problem with missing data. More specifically, if we know the true label of each observation, we could obtain the MLE in a fairly straightforward fashion. On the other hand, if we know the true model parameters, it is also trivial to compute the probability each observation belonging to each class. Therefore, a natural idea is that we begin the process with an initial random guess of the parameters, and compute the probability each observation belonging to each class

E(xpectation)-step. With those probabilities we compute the MLE, which is the M(aximization)-step. We iterate between the two steps until a convergence condition is reached. Particularly for the LCM, when the EM algorithm is applied to it, the constraints are implicitly satisfied for all the iterations thanks to the way the EM algorithm updates the values of the parameters. This nice property does not necessarily hold naturally when other non-linear optimization algorithms are applied to the optimization problem (2).

In the context of LCM, the details of the EM algorithm is given in Algorithm 1. We make two comments on Algorithm 1. First, Algorithm 1 does not produce standard errors of MLE as a by-product. In order to conduct statistical inference, one has to compute the observed Fisher information matrix and it could be algebraically tedious or might only apply to special cases. This is one of the criticisms often laid out against the EM algorithm as compared to Bayesian analysis using Gibbs samplers for example, where independent posterior samples are collected and statistical inference is easy under such circumstance. Second, the convergence of Algorithm 1 is typically slow. Wu (1983) studied the convergence issue of the EM algorithm and concluded that the convergence of the EM algorithm is sublinear when the Jacobian matrix of the unknown parameters is singular. Jamshidian and Jennrich (1997) also reported that the EM algorithm could well be accelerated by the Quasi-Newton method. In Sec. 4, we shall also empirically observe the two constrained optimization methods converge in less iterations than the EM algorithm.

Algorithm 1. EM Algorithm for LCM

1: Supply an initial guess of the parameters

$$\boldsymbol{\theta}^{(0)} = (\eta_k^{(0)}, \pi_{j,k,l}^{(0)})_{k=1,\ldots,K;j=1,\ldots,d;l=1,\ldots,c_j}$$

and a convergence tolerance ϵ.

2: Initialize with $t = 1$, and $\Delta > \epsilon$.

3: **While** $\Delta > \epsilon$:

4: (E-step) For each $i = 1, \ldots, N$ and $k = 1, \ldots, K$, compute:

$$D_{ik}^{(t)} = \frac{\eta_k^{(t-1)} \prod_{j=1}^{d} \prod_{l=1}^{c_j} \left(\pi_{k,j,l}^{(t-1)}\right)^{\mathcal{I}(y_j^i = l)}}{\sum_{k=1}^{K} \eta_k^{(t-1)} \prod_{j=1}^{d} \prod_{l=1}^{c_j} \left(\pi_{k,j,l}^{(t-1)}\right)^{\mathcal{I}(y_j^i = l)}}.$$

5: (M-step for weights) Compute:

$$\hat{\eta}_k^{(t)} = \frac{1}{N} \sum_{i=1}^{N} D_{ik}^{(t)}.$$

6: (M-step for categorical parameters) Compute:

$$\pi_{k,j,l}^{(t)} = \frac{\sum_{i=1}^{n} D_{ik}^{(t)} \left[\sum_{j=1}^{d} \sum_{l=1}^{c_j} \mathcal{I}(y_j^i = l)\right]}{\sum_{l=1}^{c_j} \left\{\sum_{i=1}^{n} D_{ik}^{(t)} \left[\sum_{j=1}^{m} \sum_{l=1}^{c_j} \mathcal{I}(y_j^i = l)\right]\right\}}.$$

7: Compute $\Delta = \|\boldsymbol{\theta}^{(h+1)} - \boldsymbol{\theta}^{(h)}\|_1$.

8: $t = t + 1$.

3. Constrained optimization methods

Motivated by the significant progress in constrained non-linear optimization, as well as the constrained nature of the LCM estimation problem, we propose to apply two nonlinear optimization approaches to solve the optimization problem (2). We notice that the EM algorithm is closely related to a gradient decent method Wu (1983), whose convergence rate is at most linear. On the other hand, it is known in optimization theory that if the second order information is utilized in the algorithm, quadratic convergence may be achieved, e.g. the classical Newton's method. However, in many applications, it is often computationally very expensive to obtain the second order information, i.e. the Hessian matrix. One remedy is to use computationally cheap approximation of the Hessian matrix. This idea leads to the family of quasi-Newton methods in the unconstrained case. While the convergence rate is typically only superlinear, the per iteration cost (both the execution time and the memory usage) is significantly reduced. In the constrained case, sophisticated methods have been developed to allow us to deal with the constraints. Given that it is relative easy to solve a constrained optimization problem when the objective function is quadratic and the constraints are all linear, one idea in constrained non-linear optimization is to approximate the objective function (or the Lagrangian function) by a quadratic function (via second-order Taylor expansion at the current solution) and approximate the constraints by linear constraints (via first-order Taylor expansion at the current solution). A new solution is obtained by solving the approximation and hence a new approximation can be constructed at the new solution. Analogous to the idea of quasi-Newton methods in the unconstrained case, in the constrained case, we can also consider an approximated Taylor expansion without having to compute the Hessian matrix exactly. Once an approximated quadratic program is obtained, one may use different approaches to solve it. For example, one can use an active set method or an interior point method to solve the quadratic program when it does not possess any specific structure. When the feasible region of the quadratic program is easily computable (typically in strongly polynomial time), a gradient projection method can be applied to solve the quadratic program approximation. As we have seen, the feasible region of optimization problem (2) is the Cartesian product of probability simplexes. It is known that projection on a probability simplex is computable in strongly polynomial time. Therefore, it is reasonable to apply a projection method to solve the quadratic program approximation. In the following subsections, the two approaches we propose are discussed in details. In both approaches, we need to evaluate the gradient of the LCM log-likelihood function. We provide the analytical expression below. For the η part, we have:

$$\frac{\partial L}{\partial \eta_k} = \sum_{i=1}^{n} \left[\frac{f(y^i|\theta_k)}{\left(\sum_{k=1}^{K} \eta_k f(y^i|\theta_k)\right)} \right], \quad k = 1, ..., K. \tag{3}$$

For the π part, we have for all $i = 1, ..., n; k = 1, ..., K; j = 1, ..., m; l = 1, ..., c_j$:

$$\frac{\partial f(y^i|\pi_k)}{\partial \pi_{k,j,l}} = \mathcal{I}(y_j^i = l) \prod_{\neq j} \prod_{\ell=1}^{c_j} \pi_{k,,\ell}^{\mathcal{I}(y_i^\ell = \ell)},$$

where $\pi_k = (\pi_{k,j,l})_{j=1,...,d;l=1,...,c_j}$. And therefore, for all $k = 1, ..., K$,

$$\frac{\partial L}{\partial \pi_k} = \sum_{i=1}^{n} \left[\frac{\eta_k}{\left(\sum_{k=1}^{K} \eta_k f(y^i|\pi_k) \right)} \frac{\partial f(y^i|\pi_k)}{\partial \pi_k} \right]. \tag{4}$$

3.1. Limited memory projected Quasi-Newton method

We first present the projected quasi-Newton method which is proposed by Schmidt et al. (2009). We augment it with the algorithm proposed by Wang and Carreira-Perpinán (2013) to project parameters onto a probability simplex in strongly polynomial time. In general, we address the problem of minimizing a differentiable function $f(x)$ over a convex set $\mathcal{C}$ subject to m equality constraints:

$$\begin{array}{lll} \min_x & f(x) & \\ \text{s.t.} & h_j(x) = 0, & \forall\ j = 1, \ldots, m, \\ & x \in \mathcal{C}. & \end{array} \tag{5}$$

In an iterative algorithm, we update the next iteration as follows:

$$x^{(t+1)} = x^{(t)} + \alpha_t d^{(t)}, \tag{6}$$

where $x^{(t)}$ is the solution at the t-th iteration, α_t is the step length and $d^{(t)}$ is the moving direction at iteration t. Different algorithms differ in how $d^{(t)}$ and α_t are determined. In the Projected Quasi-Newton method, a quadratic approximation of the objective function around the current iterate $x^{(t)}$ is constructed as follows.

$$q_t(x) = f(x^{(t)}) + (x - x^{(t)})^{\mathrm{T}} g^{(t)} + \frac{1}{2}(x - x^{(t)})^{\mathrm{T}} B^{(t)} (x - x^{(t)}),$$

where $g^{(t)} = \nabla f(x^{(t)})$ and $B^{(t)}$ denotes a positive-definite approximation of the Hessian $\nabla^2 f(x^{(t)})$. The projected quasi-Newton method then compute a feasible descent direction by minimizing this quadratic approximation subject to the original constraints:

$$\begin{array}{lll} z^{(t)} = \operatorname{argmin}_x & q_t(x), & \\ \text{s.t.} & h_j(x) = 0, & \forall\ j = 1, \ldots, m, \\ & x \in \mathcal{C}. & \end{array} \tag{7}$$

Then the moving direction is $d^{(t)} = z^{(t)} - x^{(t)}$.

To determine the step length α_t, we ensure that a sufficient decrease condition, such as the Armijo condition is met

$$f(x^{(t)} + \alpha d_t) \leq f(x^{(t)}) + \nu \alpha (g^{(t)})^{\mathrm{T}} d^{(t)}, \tag{8}$$

where $\nu \in (0, 1)$.

Although there are many appealing theoretical properties of projected Newton method just summarized, many obstacles prevent its efficient implementation in its original form. A major shortcoming is that minimizing (7) could be as difficult as optimizing (5). In Schmidt et al. (2009), the projected Newton method was modified into a more practical version which uses the limited memory BFGS update to obtain $B^{(t)}$'s and a spectral projected gradient (SPG) algorithm (Birgin, Martínez, and Raydan 2000) to solve the quadratic approximation (7).

To apply this projected auasi-Newton method to (2), we let $f(\boldsymbol{\theta}) := -L(\boldsymbol{\theta}|Y)$. As we discussed in the previous section, we rewrite (2) as follows:

$$\begin{array}{ll} \min & f(\boldsymbol{\theta}) \\ \text{s.t} & \boldsymbol{\theta} \in \mathcal{F}, \end{array} \tag{9}$$

where $\mathcal{F} = \mathcal{P}^K \otimes \otimes_{k=1}^{K} \otimes_{j=1}^{d} \mathcal{P}^{c_j}$ is the feasible region given in the format of the Cartesian product of $K \times d + 1$ probability simplexes. This rewriting is to facilitate the projection operation. We denote $\Pi_S(x)$ as the projection of a vector $x \in \mathbb{R}^n$ on a closed convex set $S \subseteq \mathbb{R}^n$, i.e. $\Pi_s(x)$ is the unique solution of the following quadratic program:

$$\begin{array}{ll} \min & \|y - x\|_2^2 \\ \text{s.t.} & y \in S. \end{array} \tag{10}$$

As we can see, in general, a quadratic program needs to be solved to compute the projection onto a closed convex set, and hence is not computationally cheap. Fortunately, the feasible region in (9) allows for a projection computable in strongly polynomial time according to Wang and Carreira-Perpinán (2013). This algorithm is presented in Algorithm 4. This algorithm is the building block for the SPG algorithm to solve the quadratic approximation in each iteration. More specifically, in the tth iteration, let

$$q_t(\boldsymbol{\theta}) = f(\boldsymbol{\theta}^{(t)}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^{\mathrm{T}} g^{(t)} + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^{\mathrm{T}} B^{(t)} (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}),$$

where $g^{(t)} = \nabla f(\boldsymbol{\theta}^{(t)})$ and $B^{(t)}$ denotes a positive-definite approximation of the Hessian $\nabla^2 f(\boldsymbol{\theta}^{(t)})$. The quadratic approximation is now given by

$$\begin{array}{rcll} \vartheta(t) & = & \operatorname{argmin}_{\boldsymbol{\theta}} & q_t(\boldsymbol{\theta}), \\ & & \text{s.t.} & \boldsymbol{\theta} \in \mathcal{F}. \end{array} \tag{11}$$

The gradient of $q_t(\boldsymbol{\theta})$ is given by

$$\nabla q_t(\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}^{(t)}) + (B^{(t)})^{\mathrm{T}} (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}). \tag{12}$$

In our implementation, $\nabla f(\boldsymbol{\theta}^{(t)})$ is numerically approximated by the method of symmetric difference quotient with length chosen as 0.05. We can also compute $\nabla f(\theta^{(t)})$ using the analytical expressions (3) and (4).

We update $B^{(t)}$ using the limited memory version of BFGS. The non-limited memory BFGS update of B is given by

$$B^{(t+1)} = B^{(t)} - \frac{B^{(t)} s^{(t)} (s^{(t)})^{\mathrm{T}} B^{(t)}}{(s^{(t)})^{\mathrm{T}} B^{(t)} s^{(t)}} + \frac{y^{(t)} (y^{(t)})^{\mathrm{T}}}{(y^{(t)})^{\mathrm{T}} s^{(t)}}, \tag{13}$$

where $s^{(t)} = \boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}$ and $y^{(t)} = \nabla f(\boldsymbol{\theta}^{(t+1)}) - \nabla f(\boldsymbol{\theta}^{(t)})$. This will consume significant memory in storing $B^{(t)}$'s when the number of features increases dramatically. Therefore, in the proposed projected quasi-Newton algorithm we only keep the most recent $m = 5$ Y and S arrays (the definitions of Y and S are in Algorithm 2) and update $B^{(t)}$ using its compact representation described by Byrd, Nocedal, and Schnabel (1994):

$$B^{(t)} = \sigma_t I - N^{(t)} (M^{(t)})^{-1} (N^{(t)})^{\mathrm{T}}, \tag{14}$$

where $N^{(t)}$ and $M^{(t)}$ are explicitly given in Eq. (3.5) of Byrd, Nocedal, and Schnabel (1994).

In addition, running Algorithm 2 until convergence, the B matrix is outputted as a by-product. The $-B$ matrix is an approximation of the observed Fisher information of the unknown parameters, which will enable us to construct asymptotic confidence intervals using the following classical results:

$$\hat{\boldsymbol{\theta}} \to N(\boldsymbol{\theta}, -B_{\boldsymbol{\theta}}^{-1}). \tag{15}$$

This is way easier than the EM algorithm to conduct statistical inference. According to Gower and Richtárik (2017), when f is convex quadratic function with positive definite Hessian matrix, it is expected that $-B^{(t)}$ from the quasi-Newton method to converge to the true Hessian matrix. However, the log-likelihood function is obviously not a convex function and as far as we know there is no formal theory that guarantees the convergence. Nonetheless, in Section 6 of Jamshidian and Jennrich (1997), the authors empirically compared the estimates for standard errors to the true values and the results are satisfactory.

In our implementation of Algorithm 2, we use $m = 5, \epsilon = 10^{-4}$ and the default parameters are $\alpha_{\min} = 10^{-10}, \alpha_{\max} = 10^{10}$, $h = 1$ and $\nu = 10^{-4}$ in Algorithm 3.

Algorithm 2. Limited memory projected quasi-Newton method

Given $\boldsymbol{\theta}^{(0)}$, m and ϵ. Set $t = 0$.
While not converge:
  $f^{(t)} = f(\boldsymbol{\theta}^t)$ and $g^{(t)} = \nabla f(\boldsymbol{\theta}^{(t)})$
  Call **Algorithm 3** for $\vartheta^{(t)}$
  $d^{(t)} = \vartheta^{(t)} - \boldsymbol{\theta}^{(t)}$
  If $\left\| \Pi_{\mathcal{F}}(\boldsymbol{\theta}^{(t)} - g^{(t)}) - \boldsymbol{\theta}^{(t)} \right\|_1 \leq \epsilon$, where $\Pi_{\mathcal{F}}(\cdot)$ calls **Algorithm 4**:
    Converged; **Break**.
  $\alpha = 1$
  $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \alpha d^{(t)}$
  While $f(\boldsymbol{\theta}^{(t+1)}) > f^{(t)} + \nu\alpha(g^{(t)})^{\mathrm{T}} d(t)$:
    Select α randomly from Uniform distribution $U(0, \alpha)$
    $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \alpha d^{(t)}$
  $s^{(t)} = \boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}$
  $y^{(t)} = g^{(t+1)} - g^{(t)}$
  If $t = 0$:
    $S = [s^{(t)}], Y = [y^{(t)}]$
  Else:
    If $t \geq m$:
      Remove first column of S and Y
    $S = [S, s^{(t)}]$
    $Y = [Y, y^{(t)}]$
  $\sigma^{(t)} = \frac{(y^{(t)})^{\mathrm{T}} s^{(t)}}{(y^{(t)})^{\mathrm{T}} y^{(t)}}$
Form N and M for BFGS update
$t = t + 1$

Algorithm 3. Spectral projected gradient algorithm

Given x_0, step bounds $0 < \alpha_{\min} < \alpha_{\max}$
Initial step length $\alpha_{bb} \in [\alpha_{\min}, \alpha_{\max}]$, and history length h
While not converge:
    $\bar{\alpha}_k = \min\{\alpha_{\max}, \max\{\alpha_{\min}, \alpha_{bb}\}\}$
    $d_k = \mathcal{P}_c(x_k - \bar{\alpha}_k \nabla q_k(x_k)) - x_k$, where $\mathcal{P}_c(.)$ calls **Algorithm 4**.
    Set bound $f_b = \max\{f(x_k), ..., f(x_{k-h})\}$
    $\alpha = 1$
    While $q_k(x_k + \alpha d_k) > f_b + \nu\alpha \nabla q_k(x_k)^{\mathrm{T}} d_k$:
        Select α randomly from Uniform distribution $U(0, \alpha)$.
    $x_{k+1} = x_k + \alpha d_k$
    $s_k = x_{k+1} - x_k$
    $y_k = \nabla q_k(x_{k+1}) - \nabla q_k(x_k)$
    $\alpha_{bb} = y_k^{\mathrm{T}} y_k / s_k^{\mathrm{T}} s_k$
    $k = k + 1$

Algorithm 4. Euclidean Projection of a Vector onto the Probability Simplex.

Supply $\boldsymbol{x} \in R^D$.
Sort $\boldsymbol{x}$ into $\boldsymbol{u}$ such that $u_1 \geq u_2 \geq \cdots \geq u_D$
Find $\rho = \max\{1 \leq j \leq D : u_j + \frac{1}{j}(1 - \sum_{i=1}^{j} u_i) > 0\}$
Define $\lambda = \frac{1}{\rho}(1 - \sum_{i=1}^{\rho} u_i)$
Output $\boldsymbol{x}'$ such that $x_i' = \max\{x_i + \lambda, 0\}, i = 1, ..., D$.

3.2. Sequential quadratic programming

SQP is a generic method for non-linear optimization with constraints. It is known as one of the most efficient computational method to solve the general nonlinear programming problem in (5) subject to both equality and inequality constraints. There are many variants of this algorithm, we use the version considered in Kraft (1988). We give a brief review of this method and then we will specifically talk about how this method could be applied to optimization problem (2).

Consider the following minimization problem

$$\begin{array}{ll} \min_x & f(x) \\ \text{s.t.} & c_j(x) = 0, j = 1, 2, ..., m_e, \\ & c_j(x) \ \geq \ 0, j = m_e + 1, m_e + 2, ..., m, \\ & x_l \leq \ x \ \leq x_u, \end{array} \tag{16}$$

where the problem functions $f : \mathbb{R}^n \to \mathbb{R}$. SQP is also an iterative method and each iteration a quadratic approximation of the original problem is also constructed and solved to obtain the moving direction. Compared to the projected quasi-Newton method, in SQP, the quadratic approximations are typically solved by an active set method or an interior point method rather than a projection type method. This significantly complicates the algorithm, but also allows the algorithm to handle more general non-linear

optimization problems, especially when the feasible region is too complex to admit an efficient projection computation. In particular, starting with a given vector of parameters $x^{(0)}$, the moving direction $d^{(t)}$ at iteration t is determined by a quadratic programing problem, which is formulated by a quadratic approximation of the Lagrangian function and a linear approximation of the constraints. Note that, in contrast to the projected quasi-Newton method we presented in the previous subsection, the SQP algorithm here approximates the Lagrangian function instead of the objective function itself. An advantage is that the dual information can be incorporated in the algorithm to ensure better convergence property. Let

$$L(x;\lambda) \;=\; f(x) - \sum_{j=1}^{m} \lambda_j c_j(x), \tag{17}$$

be the Lagrangian function associated with this optimization problem. This approximation is of the following standard form of quadratic programing:

$$\begin{array}{ll} \min\limits_{x} & \frac{1}{2}(x - x^{(t)})^{\mathrm{T}} B^{(t)} (x - x^{(t)}) + \nabla f(x^{(t)})(x - x^{(t)}) \\ \text{s.t.} & (\nabla c_j(x^{(t)}))^{\mathrm{T}} (x - x^{(t)}) + c_j(x^{(t)}) \;=\; 0, j = 1, 2, ..., m_e, \\ & (\nabla c_j(x^{(t)}))^{\mathrm{T}} (x - x^{(t)}) + c_j(x^{(t)}) \geq 0, j = m_e + 1, m_e + 2, ..., m, \end{array} \tag{18}$$

with

$$B^{(t)} = \nabla^2_{xx} L(x^{(t)}, \lambda^{(t)}), \tag{19}$$

as proposed in Wilson (1963). The multiplier $\lambda^{(t)}$ is updated using the multipliers of the constraints in (18).

In terms of the step length α, Han (1977) proved that a one-dimensional minimization of the non-differential exact penalty function

$$\phi(x; \varrho) = f(x) + \sum_{j=1}^{m_e} \varrho_j |c_j(x)| + \sum_{j=m_e+1}^{m} \varrho_j |c_j(x)|_-$$

with $|c_j(x)|_- = |\min(0; c_j(x))|$, as a merit function $\varphi : \mathbb{R} \to \mathbb{R}$

$$\varphi(\alpha) = \phi(x^{(t)} + \alpha d^{(t)}),$$

with $x^{(t)}$ and $d^{(t)}$ fixed, leads to a step length α guaranteeing global convergence for values of the penalty parameters ϱ_j greater than some lower bounds. Then, Powell (1978) proposed to update the penalty parameters according to

$$\varrho_j = \max\left(\frac{1}{2}(\varrho_j^- + |\mu_j|), |\mu_j|\right), j = 1, ..., m,$$

where μ_j denotes the Lagrange multiplier of the jth constraint in the quadratic problem and ϱ_j^- is the jth penalty parameter of the previous iteration, starting with some $\varrho_j^0 = 0$.

It is important in practical applications to not evaluate $B^{(t)}$ in (19) in every iteration, but to use only first order information to approximate the Hessian matrix of the Lagrange function in (17). Powell (1978) proposed the following modification:

$$B^{(t+1)} = B^{(t)} + \frac{q^{(t)}(q^{(t)})^{\mathrm{T}}}{(q^{(t)})^{\mathrm{T}}s^{(t)}} - \frac{B^{(t)}s^{(t)}(s^{(t)})^{\mathrm{T}}B^{(t)}}{(s^{(t)})^{\mathrm{T}}B^{(t)}s^{(t)}},$$

with

$$s^{(t)} = x^{(t+1)} - x^{(t)}$$
$$q^{(t)} = \gamma_t \eta^{(t)} + (1-\gamma_t)B^{(t)}s^{(t)}$$

where

$$\eta^{(t)} = \nabla_x L(x^{(t+1)}, \lambda^{(t)}) - \nabla_x L(x^{(t)}, \lambda^{(t)})$$

and γ_t is chosen as

$$\gamma_t = \begin{cases} 1 & \text{if } (s^{(t)})^{\mathrm{T}}\eta^{(t)} \geq 0.2(s^{(t)})^{\mathrm{T}}B^{(t)}s^{(t)}, \\ \dfrac{0.8(s^{(t)})^{\mathrm{T}}B^{(t)}s^{(t)}}{(s^{(t)})^{\mathrm{T}}B^{(t)}s^{(t)} - (s^{(t)})^{\mathrm{T}}\eta^{(t)}} & \text{otherwise,} \end{cases}$$

which ensures that $B^{(t+1)}$ remains positive definite within the linear manifold defined by the tangent planes to active constraints at $x^{(t+1)}$.

In LCM, the problem turns out to be simpler: the quadratic programing problem in (18) is only subject to m_e equality constraints. In addition, unless we use projected quasi-Newton method, for which we have to build our own solver, there is a popular implementation of SQP in Python's SciPy package. The package uses a variant of SQP: Sequential Least SQuares Programing (SLSQP): It replaces the quadratic programing problem in (18) by a linear least squares problem using a stable LDL^T factorization of the matrix $B^{(t)}$.

4. Simulation studies and real data analysis

In this section, we provide four example bundles and one real data analysis to demonstrate the performance of the proposed methods. The model specifications of the four example bundles as follows:

- Example bundle 1, $N = 500$: (A) $d = 1, K = 2$; (B) $d = 1, K = 3$; (C) $d = 2, K = 2$; (D) $d = 4, K = 2$
- Example bundle 2, $N = 1000$: (A) $d = 2, K = 2$; (B) $d = 2, K = 3$; (C) $d = 3, K = 2$; (D) $d = 3, K = 3$
- Example bundle 3, $N = 2000$: (A) $d = 3, K = 3$; (B) $d = 3, K = 4$; (C) $d = 4, K = 4$; (D) $d = 5, K = 3$
- Example bundle 4, $N = 5000$: (A) $d = 4, K = 4$; (B) $d = 4, K = 5$; (C) $d = 5, K = 4$; (D) $d = 5, K = 5$

One dataset is simulated from LCM for each combination. In total, we consider 16 datasets with different combinations of sample size, dimensionality and number of groups providing a comprehensive picture of the model performance.

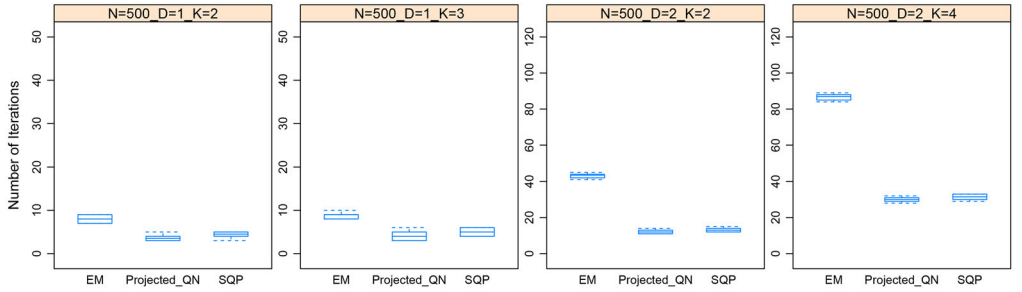


Figure 1. Example bundle 1, $N = 500$; number of iterations for (A) $d = 1, K = 2$; (B) $d = 1, K = 3$; (C) $d = 2, K = 2$; (D) $d = 4, K = 2$.

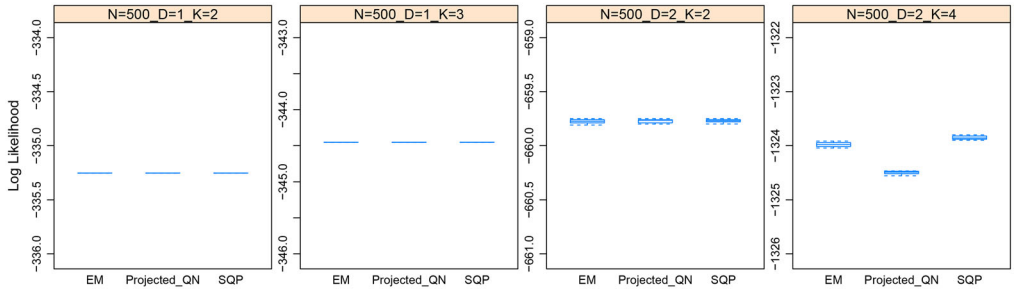


Figure 2. Example bundle 1, $N = 500$; log-likelihood values for (A) $d = 1, K = 2$; (B) $d = 1, K = 3$; (C) $d = 2, K = 2$; (D) $d = 4, K = 2$.

4.1. Example bundle 1

In this example bundle, we use the following three methods to maximize the log-likelihood function: (1) EM, (2) SQP, and (3) Projected Quasi-Newton (QN). Each method is repeated 10 times with different initial values across the 10 runs. At each run, the three methods begin with identical initial values. The true weights and categorical parameters are reported in Tables A1–A4 in the Appendix. Side by side boxplots are drawn and reported in Figures 1 and 2 showing number of iterations and log-likelihood values of the 10 runs, respectively. For each method, the best result based on the log-likelihood values across the 10 runs are given in Table 1. Results from the true parameters are also included in Table 1 as a comparison.

From Table 1, Figures 1 and 2, we observe that the proposed two optimization methods have good performance compared to the traditional EM algorithm: the log-likelihood values are very close to that of EM for all four datasets in this example bundle. Note that the vertical axis scales are different in Figure 1. The numbers of iterations of the two proposed optimization methods are obviously lower than that of EM, for example the number of iterations of SQP and projected QN are both 12 compared to 88 of the EM algorithm. This suggests that the two optimization methods are less likely to get stuck in local maxima.

In addition, there are no substantial differences between the final best solutions across the 10 runs. Actually, the final best results are quite close to the results obtained from the other 9 runs. Using scenario (A) with $d = 1, K = 2$ in this bundle as an example,

Table 1. Example bundle 1, $N = 500$; the best result based on the log-likelihood among the 10 runs for each method.

	True parameters	EM	SQP	Projected QN
(A) $d = 1, K = 2$				
Log-likelihood	−335.29	−335.25	−335.25	−335.25
Number of iterations	NA	8	4	4
(B) $d = 1, K = 3$				
Log-likelihood	−344.49	−344.45	−344.45	−344.45
Number of iterations	NA	9	5	4
(C) $d = 2, K = 2$				
Log-likelihood	−661.30	−659.75	−659.75	−659.75
Number of iterations	NA	43	13	12
(D) $d = 4, K = 2$				
Log-likelihood	−1323.29	−1323.91	−1323.80	−1324.46
Number of iterations	NA	88	31	30

we divide the 10 log-likelihood values into two groups, where the first group contains the largest log-likelihood value only while the second group contains the rest of the nine log-likelihood values, and then fit a nonparametric two-group Wilcoxon signed-rank test Bauer (1972). The p value is 0.20, which is clearly larger than the usual 0.05 threshold. The parametric t-test might not work well here because the group sizes are too small. Moreover, the estimated weights and categorical parameters from the 10 runs are also close to each other. We repeat the test for the log-likelihood values on the estimates for each of the weight and categorical parameters and none of the p values are larger than 0.05.

4.2. Example bundle 2

The true weights and categorical parameters are reported in Tables A5–A8 in the Appendix. As in example bundle 1, each method is repeated 10 times with different initial values across the 10 runs. At each run, the three methods begin with identical initial values. The simulation results for this bundle are summarized in Figures 3 and 4 for number of iterations and log-likelihood, respectively. Similarly, for each method, the best result based on the log-likelihood values among the 10 runs are given in Table 2. Results from the true parameters are also included in Table 2 for comparison.

From Table 2, Figures 3 and 4, we observed a similar pattern as in example bundle 1, i.e. the log-likelihood values are close to each other, however the number of iterations of the two optimization methods are smaller than that of EM, further showing the promise of using the proposed optimization methods as alternatives in practice.

4.3. Example bundle 3

With exactly the same settings, we report results for example bundle 3 in this section. The resulting number of iteration and log-likelihood values are reported in Figures 5 and 6, respectively. For each method, the best results based on the log-likelihood values among the 10 runs are given in Table 3. Results from the true parameters are also included in Table 3 for comparison. The true weights and categorical parameters are reported in Tables A9–A12 in the Appendix.

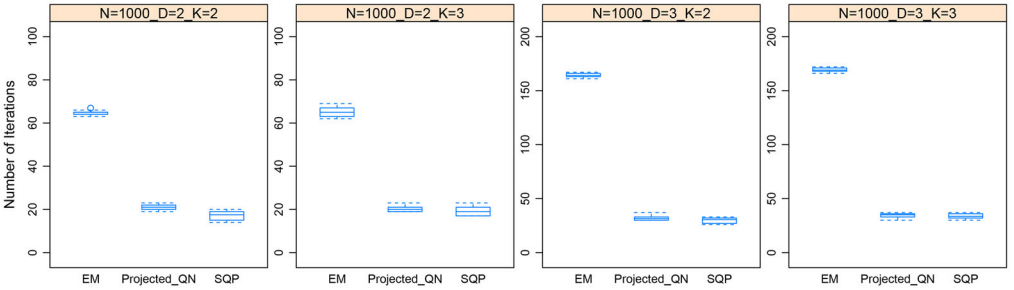


Figure 3. Example bundle 2, $N = 1000$; number of iterations for (A) $d = 2, K = 2$; (B) $d = 2, K = 3$; (C) $d = 3, K = 2$; (D) $d = 3, K = 3$.

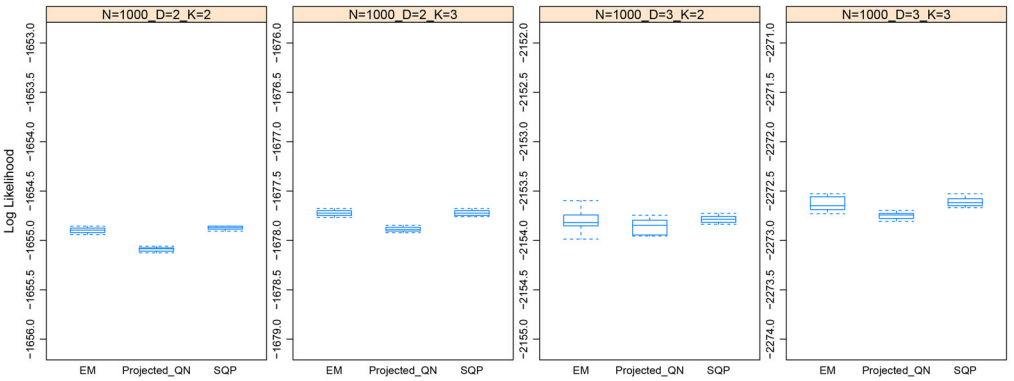


Figure 4. Example bundle 2, $N = 1000$; log-likelihood values for (A) $d = 2, K = 2$; (B) $d = 2, K = 3$; (C) $d = 3, K = 2$; (D) $d = 3, K = 3$.

Table 2. Example bundle 2, $N = 1000$; the best result based on the log-likelihood values among the 10 runs for each method.

	True parameters	EM	SQP	Projected QN
(A) $d = 2, K = 2$				
Log-likelihood	−1656.59	−1654.86	−1654.86	−1655.06
Number of iterations	NA	65	17	21
(B) $d = 2, K = 3$				
Log-likelihood	−1679.85	−1677.68	−1677.68	−1677.85
Number of iterations	NA	75	21	22
(C) $d = 3, K = 2$				
Log-likelihood	−2156.11	−2153.60	−2153.73	−2153.74
Number of iterations	NA	165	29	32
(D) $d = 3, K = 3$				
Log-likelihood	−2274.72	−2272.53	−2272.53	−2272.70
Number of iterations	NA	169	34	35

From Table 3, Figures 5 and 6, we observed a similar pattern as in the previous examples: the number of iterations of the two optimization methods are much smaller than that of EM, while the log-likelihood values are quite close to each other for the three methods.

Table 3. Example bundle 3, $N = 2000$; the best result based on the log-likelihood values among the 10 runs for each method.

	True parameters	EM	SQP	Projected QN
(A) $d = 3, K = 3$				
Log-likelihood	−3888.77	−3887.30	−3887.30	−3887.30
Number of iterations	NA	355	23	28
(B) $d = 3, K = 4$				
Log-likelihood	−3906.01	−3905.22	−3905.22	−3905.38
Number of iterations	NA	464	26	34
(C) $d = 4, K = 3$				
Log-likelihood	−5241.99	−5237.39	−5237.37	−5237.41
Number of iterations	NA	526	51	46
(D) $d = 5, K = 3$				
Log-likelihood	−6437.05	−6431.05	−6431.07	−6431.07
Number of iterations	NA	533	53	48

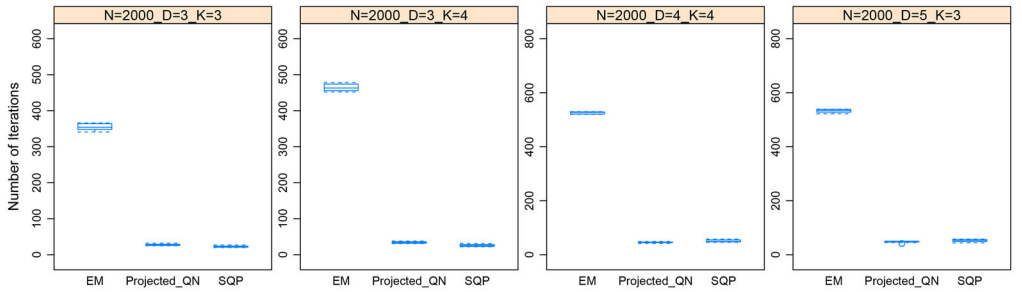


Figure 5. Example bundle 3, $N = 2000$; number of iterations for (A) $d = 3, K = 3$; (B) $d = 3, K = 4$; (C) $d = 4, K = 4$; (D) $d = 5, K = 3$.

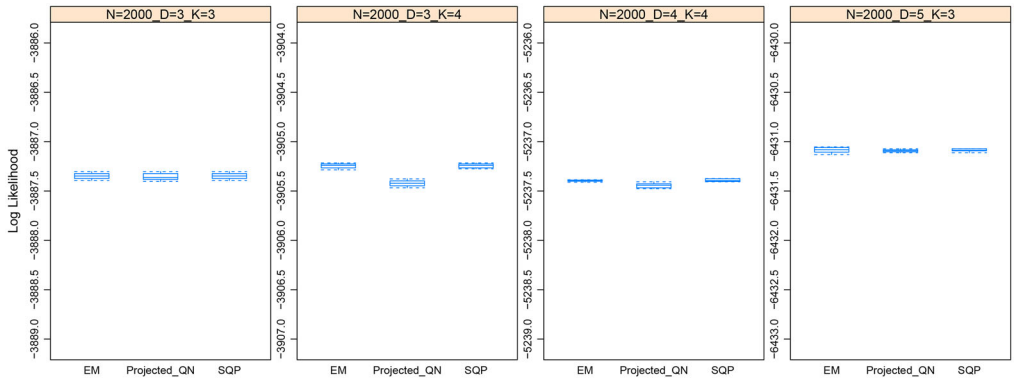


Figure 6. Example bundle 3, $N = 2000$; log-likelihood values for (A) $d = 3, K = 3$; (B) $d = 3, K = 4$; (C) $d = 4, K = 4$; (D) $d = 5, K = 3$.

4.4. Example bundle 4

The resulting number of iterations and log-likelihood values of example bundle 4 are reported in Figures 7 and 8, respectively. For each method, the best results based on the log-likelihood values among the ten runs are given in Table 4. Results from the true

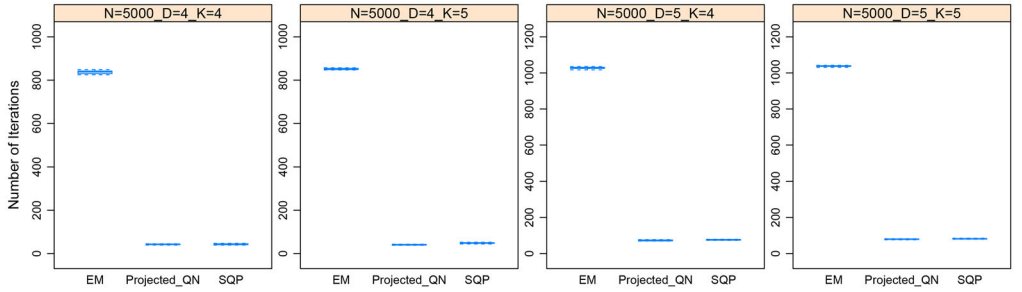


Figure 7. Example bundle 4, number of iterations.

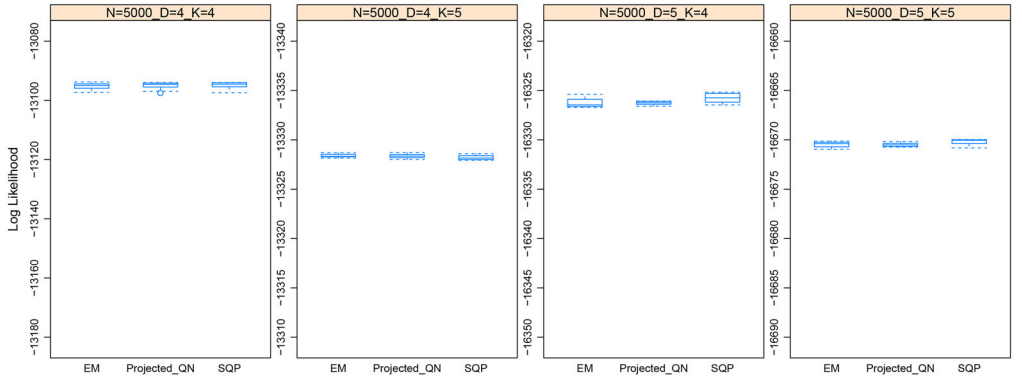


Figure 8. Example bundle 4, log-likelihood. (a) Number of iterations, (b) log-likelihood.

Table 4. Example bundle 4, the best result based on the log-likelihood values among the 10 runs for each method.

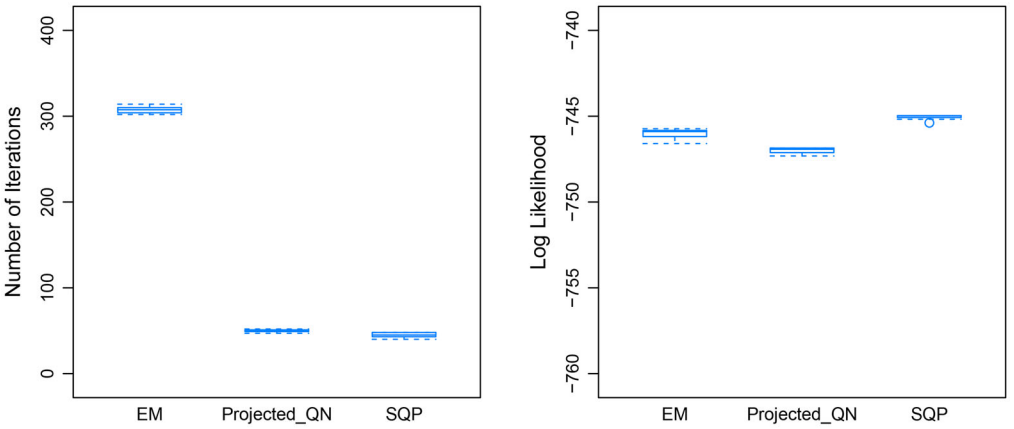
	True parameters	EM	SQP	Projected QN
(A) $d = 4, K = 4$				
Log-likelihood	−13105.82	−13093.78	−13093.98	−13093.92
Number of iterations	NA	837	43	42
(B) $d = 4, K = 5$				
Log-likelihood	−13335.47	−13328.15	−13327.94	−13328.02
Number of iterations	NA	852	48	40
(C) $d = 5, K = 4$				
Log-likelihood	−16336.15	−16325.38	−16325.18	−16326.06
Number of iterations	NA	1028	76	73
(D) $d = 5, K = 5$				
Log-likelihood	−16684.59	−16670.12	−16669.97	−16670.17
Number of iterations	NA	1038	82	80

parameters are also included in Table 4 for comparison. The true weights and categorical parameters are repeated in Tables A13–A16 in the Appendix.

These results in example bundle 4 further confirm what we have observed: with the same settings, the two optimization methods converge in less iterations than EM, while they still yield comparable log-likelihood values as EM. This strengthens the promise of using the two proposed optimization methods as alternatives to EM when estimating a LCM.

Table 5. Performance of the three methods based on 10 runs for the application example.

	BayesLCA	EM	SQP	Projected QN
Log-likelihood	−781.8063	−745.7291	−744.9672	−746.8557
Number of iterations	NA	302	44	50



(a) Number of Iterations (b) Log-Likelihood

Figure 9. Boxplots for the application example.

4.5. An application

We now go back to the motivating example discussed in Sec. 1. The data set is available in the R package *BayesLCA*. White and Murphy (2014) used a $K = 3$ LCM to fit the data using Gibbs sampler. It is clear that this is an $n = 240, d = 6$ binary data set. We follow the recommendation of Moran et al. (2004) and fit a $K = 3$ LCM with (1) EM, (2) SQP, and (3) projected quasi-Newton methods and 10 different initial points, and the best result of each method is recorded based on the log-likelihood value. The results are summarized in Table 5. The result from BayesLCA package (White and Murphy 2014) is also included. The side-by-side boxplots for number of iterations and log-likelihood values of the 10 runs are reported in Figure 9.

From Table 5, SQP has the best performance in terms both the log-likelihood value and the number of iterations. The results from EM and projected quasi-Newton are very similar although EM needs way more iterations to converge. This agrees with the previous observations. We also note that all the three methods considered have larger log-likelihood values than that of BayesLCA. The method proposed by White and Murphy (2014) actually has the smallest log-likelihood value.

In addition, since we do not know the true values, we computed pairwise root mean squared error (RMSE) based on the estimates, i.e. we compute RMSE of estimates for every two methods. Since we have considered four different methods, we will have six RMSEs, one number for each pair of methods. The results are reported in Table 6.

The results in Table 6 are consistent with the observations we have made: Since the log-likelihood values are closer for EM, SQP, and projected QN, the pairwise RMSE of these three methods are way lower than those when paired with BayesLCA, for example

Table 6. Pairwise root mean squared error (RMSE) of the four methods considered for the application example.

	Bayes LCA	EM	SQP	Projected QN
Bayes LCA	0	0.237	0.241	0.233
EM	0.237	0	0.029	0.046
SQP	0.241	0.029	0	0.045
Projected QN	0.233	0.046	0.045	0

Table 7. Comparison of CPU time (in seconds).

	EM	SQP	Projected QN
CPU time per iteration	0.08	0.31	0.39
Number of iterations	302	44	50
Overall runtime	24.2	13.6	19.5

the RMSE of SQP and EM is 0.029, while the RMSE for SQP and BayesLCA is 0.241, which is over eight times larger.

5. Discussion

In the previous section, we have shown the number of iterations of the proposed methods is smaller than that of the EM algorithm. In this section, we report the comparison of CPU times. Taking the application as an example, the runtime are reported in Table 7.

From Table 7, we can see that the EM algorithm indeed has the lowest CPU time per iteration. However, when taking the number of iterations into account, the story is different: using SQP as example, the number of iterations of EM and SQP are 302 and 44, respectively. The number of iterations for the SQP algorithm is around 1/8 of the EM algorithm, although the CPU time per iteration is merely about four times longer. Therefore, the total computational time of the SQP algorithm is significantly less than that of the EM algorithm. In the application example, the computational times of the SQP and projected QN methods are respectively 43% and 19% better compared to the EM algorithm.

6. Concluding remarks

The primary research objective of the paper is to provide alternative methods to learn the unknown parameters of the LCM. Given the log-likelihood as a function of the parameters, we aim to find estimators that can maximize the log-likelihood function. The traditional way is to use the EM algorithm. However, it is observed that the EM algorithm converges slowly. Therefore, in this paper, we propose the use of two constrained optimization methods, namely the SQP and the projected quasi-Newton methods as alternatives. Simulation studies and the real example in Sec. 4 reveal that the two proposed methods perform well. The obvious advantages we observed are as follows: (1) the two optimization methods produced slightly larger log-likelihood values compared to the EM algorithm; (2) they converge in significantly less iterations than the EM algorithm. That being said, we want to make it clear that the aim is not to completely replace the EM algorithm, rather we would like to provide alternative ways of

achieving the same goal using some optimization methods. Inter-disciplinary collaboration between researchers in statistics and mathematical optimization has never been as important as in the big data era.

Acknowledgment

We thank the editor and the anonymous reviewer for suggestions that improved the manuscript.

Appendix

The Python source codes for EM and projected quasi-Newton for LCM are available upon request. The implementation of SQP is available in Python SciPy package.

The true weights and parameters used in Sec. 4 are given below.

Example Bundle 1

Table A1. True weights and categorical parameters for example bundle 1, $d = 1, K = 2$.

		$d=1$	
	Weights	0	1
$K=1$	0.5	0.4	0.6
$K=2$	0.5	0.8	0.2

Table A2. True weights and categorical parameters for example bundle 1, $d = 1, K = 3$.

		$d=1$	
	Weights	0	1
$K=1$	0.5	0.4	0.6
$K=2$	0.3	0.8	0.2
$K=3$	0.2	0.1	0.9

Table A3. True weights and categorical parameters for example bundle 1, $d = 2, K = 2$.

		$d=1$		$d=2$	
	Weights	0	1	0	1
$K=1$	0.5	0.4	0.6	0.1	0.9
$K=2$	0.5	0.8	0.2	0.6	0.4

Table A4. True weights and categorical parameters for example bundle 1, $d = 4, K = 2$.

		$d=1$		$d=2$		$d=2$		$d=2$	
	Weights	0	1	0	1	0	1	0	1
$K=1$	0.5	0.4	0.6	0.1	0.9	0.5	0.5	0.6	0.4
$K=2$	0.5	0.8	0.2	0.6	0.4	0.4	0.6	0.7	0.3

Example Bundle 2

Table A5. True weights and categorical parameters for example bundle 2, $d = 2, K = 2$.

		$d=1$		$d=2$		
	Weights	0	1	0	1	2
$K=1$	0.4	0.1	0.9	0.8	0.1	0.1
$K=2$	0.6	0.8	0.2	0.3	0.4	0.3

Table A6. True weights and categorical parameters for example bundle 2, $d = 2, K = 3$.

		$d=1$		$d=2$		
	Weights	0	1	0	1	2
$K=1$	0.4	0.1	0.9	0.8	0.1	0.1
$K=2$	0.4	0.8	0.2	0.3	0.4	0.3
$K=3$	0.2	0.6	0.4	0.5	0.3	0.2

Table A7. True weights and categorical parameters for example bundle 2, $d = 3, K = 2$.

		$d=1$		$d=2$			$d=3$	
	Weights	0	1	0	1	2	0	1
$K=1$	0.4	0.1	0.9	0.8	0.1	0.1	0.6	0.4
$K=2$	0.6	0.8	0.2	0.3	0.4	0.3	0.9	0.1

Table A8. True weights and categorical parameters for example bundle 2, $d = 3, K = 3$.

		$d=1$		$d=2$			$d=3$	
	Weights	0	1	0	1	2	0	1
$K=1$	0.4	0.1	0.9	0.8	0.1	0.1	0.6	0.4
$K=2$	0.4	0.8	0.2	0.3	0.4	0.3	0.9	0.1
$K=3$	0.2	0.6	0.4	0.6	0.3	0.1	0.2	0.8

Example Bundle 3

Table A9. True weights and categorical parameters for example bundle 3, $d = 3, K = 3$.

		$d=1$		$d=2$		$d=3$	
	Weights	0	1	0	1	0	1
$K=1$	0.3	0.9	0.1	0.3	0.7	0.1	0.9
$K=2$	0.4	0.2	0.8	0.5	0.5	0.55	0.45
$K=3$	0.3	0.1	0.9	0.4	0.6	0.3	0.7

Table A10. True weights and categorical parameters for example bundle 3, $d = 3, K = 4$.

		$d=1$		$d=2$		$d=3$	
	Weights	0	1	0	1	0	1
$K=1$	0.3	0.9	0.1	0.3	0.7	0.1	0.9
$K=2$	0.2	0.2	0.8	0.5	0.5	0.55	0.45
$K=3$	0.3	0.1	0.9	0.4	0.6	0.3	0.7
$K=4$	0.2	0.5	0.5	0.9	0.1	0.2	0.8

Table A11. True weights and categorical parameters for example bundle 3, $d = 4, K = 4$.

		$d=1$		$d=2$		$d=3$		$d=4$	
	Weights	0	1	0	1	0	1	0	1
$K=1$	0.3	0.9	0.1	0.3	0.7	0.1	0.9	0.6	0.4
$K=2$	0.2	0.2	0.8	0.5	0.5	0.55	0.45	0.5	0.5
$K=3$	0.3	0.1	0.9	0.4	0.6	0.3	0.7	0.7	0.3
$K=4$	0.2	0.5	0.5	0.9	0.1	0.2	0.8	0.5	0.5

Table A12. True weights and categorical parameters for example bundle 3, $d = 5, K = 3$.

		$d=1$		$d=2$		$d=3$		$d=4$		$d=5$	
	Weights	0	1	0	1	0	1	0	1	0	1
$K=1$	0.3	0.9	0.1	0.3	0.7	0.1	0.9	0.6	0.4	0.7	0.3
$K=2$	0.4	0.2	0.8	0.5	0.5	0.55	0.45	0.5	0.5	0.3	0.7
$K=3$	0.3	0.1	0.9	0.4	0.6	0.3	0.7	0.9	0.1	0.2	0.8

Example Bundle 4

Table A13. True weights and categorical parameters for example bundle 4, $d = 4, K = 4$.

		$d=1$		$d=2$		$d=3$		$d=4$	
	Weights	0	1	0	1	0	1	0	1
$K=1$	0.3	0.9	0.1	0.3	0.7	0.1	0.9	0.6	0.4
$K=2$	0.2	0.2	0.8	0.5	0.5	0.55	0.45	0.5	0.5
$K=3$	0.3	0.1	0.9	0.4	0.6	0.3	0.7	0.7	0.3
$K=4$	0.2	0.5	0.5	0.9	0.1	0.2	0.8	0.5	0.5

Table A14. True weights and categorical parameters for example bundle 4, $d = 4, K = 5$.

		$d=1$		$d=2$		$d=3$		$d=4$	
	Weights	0	1	0	1	0	1	0	1
$K=1$	0.3	0.9	0.1	0.3	0.7	0.1	0.9	0.6	0.4
$K=2$	0.2	0.2	0.8	0.5	0.5	0.55	0.45	0.5	0.5
$K=3$	0.3	0.1	0.9	0.4	0.6	0.3	0.7	0.7	0.3
$K=4$	0.1	0.5	0.5	0.9	0.1	0.2	0.8	0.5	0.5
$K=5$	0.1	0.8	0.2	0.1	0.9	0.9	0.1	0.7	0.3

Table A15. True weights and categorical parameters for example bundle 4, $d = 5, K = 4$.

		$d=1$		$d=2$		$d=3$		$d=4$		$d=5$	
	Weights	0	1	0	1	0	1	0	1	0	1
$K=1$	0.3	0.9	0.1	0.3	0.7	0.1	0.9	0.6	0.4	0.2	0.8
$K=2$	0.2	0.2	0.8	0.5	0.5	0.55	0.45	0.5	0.5	0.8	0.2
$K=3$	0.3	0.1	0.9	0.4	0.6	0.3	0.7	0.7	0.3	0.3	0.7
$K=4$	0.2	0.5	0.5	0.9	0.1	0.2	0.8	0.5	0.5	0.9	0.1

Table A16. True weights and categorical parameters for example bundle 4, $d = 5, K = 5$.

		$d=1$		$d=2$		$d=3$		$d=4$		$d=5$	
	Weights	0	1	0	1	0	1	0	1	0	1
$K=1$	0.3	0.9	0.1	0.3	0.7	0.1	0.9	0.6	0.4	0.4	0.6
$K=2$	0.2	0.2	0.8	0.5	0.5	0.55	0.45	0.5	0.5	0.7	0.3
$K=3$	0.3	0.1	0.9	0.4	0.6	0.3	0.7	0.7	0.3	0.4	0.6
$K=4$	0.1	0.5	0.5	0.9	0.1	0.2	0.8	0.5	0.5	0.8	0.2
$K=5$	0.1	0.8	0.2	0.1	0.9	0.9	0.1	0.7	0.3	0.9	0.1

References

Asparouhov, T., and B. Muthén. 2011. *Using Bayesian priors for more flexible latent class analysis*. Alexandria, VA: American Statistical Association.

Bauer, D. 1972. Constructing confidence sets using rank statistics. *Journal of the American Statistical Association* 67 (339):687–90. doi:10.1080/01621459.1972.10481279.

Birgin, E. G., J. M. Martínez, and M. Raydan. 2000. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization* 10 (4):1196–211. doi:10.1137/S1052623497330963.

Byrd, R. H., J. Nocedal, and R. B. Schnabel. 1994. Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming* 63 (1–3):129–56. doi:10.1007/BF01582063.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39 (1): 1–38. doi:10.1111/j.2517-6161.1977.tb01600.x.

Gower, R. M., and P. Richtárik. 2017. Randomized Quasi-Newton updates are linearly convergent matrix inversion algorithms. *SIAM Journal on Matrix Analysis and Applications* 38 (4): 1380–409. doi:10.1137/16M1062053.

Hagenaars, J. A., and A. L. McCutcheon. 2002. *Applied latent class analysis*. Vol. 64, Cambridge: Cambridge University Press.

Han, S. P. 1977. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications* 22 (3):297–309. doi:10.1007/BF00932858.

Jamshidian, M., and R. I. Jennrich. 1997. Acceleration of the EM algorithm by using quasi-newton methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 59 (3): 569–87. doi:10.1111/1467-9868.00083.

Kraft, D. 1988. A software package for sequential quadratic programming. Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt.

Li, Y., J. Lord-Bessen, M. Shiyko, and R. Loeb. 2018. Bayesian latent class analysis tutorial. *Multivariate Behavioral Research* 53 (3):430–51. doi:10.1080/00273171.2018.1428892.

McCutcheon, A. L. 1987. *Latent class analysis*. Vol. 64, Thousand Oaks, CA: Sage.

Meilijson, I. 1989. A fast improvement to the EM algorithm on its own terms. *Journal of the Royal Statistical Society: Series B (Methodological)* 51 (1):127–38. doi:10.1111/j.2517-6161.1989.tb01754.x.

Moran, M., C. Walsh, A. Lynch, R. Coen, D. Coakley, and B. Lawlor. 2004. Syndromes of behavioural and psychological symptoms in mild Alzheimer's disease. *International Journal of Geriatric Psychiatry* 19 (4):359–64. doi:10.1002/gps.1091.

Oser, J., M. Hooghe, and S. Marien. 2013. Is online participation distinct from offline participation? a latent class analysis of participation types and their stratification. *Political Research Quarterly* 66 (1):91–101. doi:10.1177/1065912912436695.

Powell, M. J. 1978. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical analysis*, ed. G. A. Watson. Berlin: Springer, 144–57.

Schmidt, M., E. Berg, M. Friedlander, and K. Murphy. 2009. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. *Artificial Intelligence and Statistics* 56:456–63.

Wang, W., and M. A. Carreira-Perpinán. 2013. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. arXiv preprint arXiv:1309.1541.

White, A., and T. B. Murphy. 2014. Bayeslca: An r package for Bayesian latent class analysis. *Journal of Statistical Software* 61 (13):5683. doi:10.18637/jss.v061.i13.

Wilson, R. B. 1963. A simplicial algorithm for concave programming. PhD diss., Graduate School of Bussiness Administration.

Wu, C. J. 1983. On the convergence properties of the EM algorithm. *The Annals of Statistics* 11 (1):95–103. doi:10.1214/aos/1176346060.