

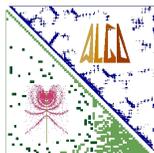


Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique

Optimization in latent class analysis

MARTIN FUCHS AND ARNOLD NEUMAIER

Technical Report TR/PA/10/89



Publications of the Parallel Algorithms Team

<http://www.cerfacs.fr/algor/publications/>

Optimization in latent class analysis

Martin Fuchs^{1*}, Arnold Neumaier²

¹CERFACS, Parallel Algorithms Team, Toulouse, France

²University of Vienna, Faculty of Mathematics, Vienna, Austria

*corresponding author: martin.fuchs81@gmail.com

October 13, 2010

Abstract. In latent class analysis (LCA) one seeks a clustering of categorical data, such as patterns of symptoms of a patient, in terms of locally independent stochastic models. This leads to practical definitions of criteria, e.g., whether to include patients in further diagnostic examinations. The clustering is often determined by parameters that are estimated by the maximum likelihood method. The likelihood function in LCA has in many cases – especially for sparse data sets – a complicated shape with many local extrema, even for small-scale problems. Hence a global optimization must be attempted.

This paper describes an algorithm and software for the global optimization of the likelihood function constrained by the requirement of a good fit of the data with a minimal number of classes. The problem is formulated in the algebraic modeling language AMPL and solved via state of the art optimization solvers.

The approach is successfully applied to three real-life problems. Remarkably, the goodness-of-fit constraint makes one of the three problems identifiable by eliminating all but one of the local minimizers.

Keywords. latent class analysis, global optimization, clustering

1 Introduction

This paper addresses the problem of explaining categorical data by means of stochastic models. The categories arise from responses to a criterion, e.g., responses to an intelligence test, or symptoms of a patient such as back pain or joint pain, or raters' statements such as caries diagnosis by dentists. In the latter case for example the categories could be 1 for healthy and 0 for sick. We are looking for means to assess the agreement of different raters and classify the ratings afterwards, e.g., we ask different dentists about different teeth and infer a classification of the state of a tooth from the associated responses by the dentists, the so-called response pattern. In the example of symptoms of a patient the responses would be answers to questions whether or not the patient suffers from, e.g., back pain, joint pain etc. The classification allows us to define criteria for including a patient in more intensive and expensive diagnostic examinations such as tomographies, or excluding him.

Mathematically speaking, **latent class analysis** (LCA) is a clustering method of categorical data assuming local stochastic independence. The questions asked to a patient, or the experts asked about the state of a tooth are called **items** in the abstract formulation of LCA. The term **local stochastic independence** means that the categorized items are independent in each class, i.e., different questions should not target related symptoms (e.g., alcohol consumption during the last week, and during the last month), and different experts should be independent in their decision. The parameters imposing the clustering are estimated by maximizing likelihood.

Latent class analysis was introduced in [26] and later studied in [3, 10, 17]. A survey is given in [25] whose notation we follow. An exhaustive online survey can be found in [35].

There are some fundamental problems associated with LCA that are still open to solve, see, e.g., [21]. First of all there is the problem of multiple locally optimal solutions. Ideally, the global maximum of the likelihood function leads to the best fit to the data. If a moderately large sample is given it is often possible to identify a unique optimal solution that fits the data well. However, in many applications only **scarce data** is available such that the likelihood function becomes multimodal, see, e.g., [12]. **Multiple extrema** require a more careful consideration of the aspect of global optimization of the likelihood function. Parameters estimated from a unique global solution allow to reproduce the data uniquely from the model, otherwise there may exist multiple sets of parameters that fit the data equally well. This is known as the issue of **identification**, cf. [18, 26]. The standard optimization technique used in LCA software is based on the EM-algorithm by Dempster, see [7].

Some current software packages that include LCA in their functionality are CDAS/MLLSA (shareware), see [19], Latent GOLD (commercial), see [38, 39], ℓ EM (free, predecessor of Latent GOLD), see [36, 37], MPlus (commercial), see [30, 31], MULTIMIX (free), see [22, 23], WinLTA (free), see [4, 5], and the R packages MCLUST, see [14, 15], and poLCA,

see [27, 28]. In the typical approach one runs the optimization phase with random multistart and checks each solution for goodness of fit in the postprocessing phase. As multiple extrema are very likely to exist in the case of scarce data one wishes to find **as many good optima as possible** such that additional information can eventually lead to the best model. More or less obvious redundancies in the set of items are removed, preferably in an automated fashion, by looking at item correlations. The number of classes is determined by running the multistart phase for different choices of this number and selecting the smallest number that yields a good fit. A small number is sought to facilitate an interpretation of the meaning of each class.

To simplify the optimization problem in several latent class models some probabilities may be constrained to be zero or equal, based on problem specific structural assumptions. Thus there are fewer free parameters left to fit, and the resulting problem becomes smaller. Since this is imposed by adding constraints to the model one speaks of **constrained LCA**. Examples of constrained LCA include the Rasch model, see [9, 32], the latent distance model, see [6, 26, 29], and workarounds for the EM-algorithm.

In this paper we will discuss so-called **unconstrained LCA**, i.e., we do not make any structural assumptions. The goal is to find a good data fit, so we look for the global solutions of the maximum likelihood problem. We will ensure a good fit – even if we only find a local solution – by way of adding goodness of fit constraints. To solve the final optimization problem we use state of the art optimization solvers with interface to the algebraic modeling languages AMPL, cf. [13], and GAMS, cf. [1]. Thus we are able to find quickly **many good solutions** if they exist. The number of latent classes is determined automatically by selecting the minimal number of classes for which we find a solution. Note that we do not have to check any goodness of fit a posteriori, since any solution that we find is already constrained to provide a good fit. If no satisfying solution could be found from the original data set we perform an automatic item subset selection which proves to be useful in our example applications. A closer look to non-zero item correlations, thus violating the local independence assumption, is out of the scope of the paper as well as comparing results in local optimization of the likelihood function for a fixed item set without goodness of fit constraints, which has been done in the past, e.g., in [20]. The numerical experiments will demonstrate the strength of our approach in finding multiple optima with ensured goodness of fit.

An implementation of our approach in Python can be found at [16].

This paper is organized as follows. In Section 2 we give the general problem exposition of LCA. In the following two sections we discuss two fundamental issues of LCA, i.e., goodness of fit in Section 4, and global and local extrema in Section 5. We present our approach to solve the introduced problems in Section 6. Numerical results are summarized in Section 7.

2 Problem formulation

Suppose that for each sample point x in an ordered sample X from a given population Ω one observes k variables, so-called **criteria** or **items**, I_1, \dots, I_k . For $i = 1, \dots, k$, we categorize the responses according criterion I_i into finitely many categories forming finite sets S_i . Writing $s_i(x)$ for the category of sample point x according to criterion I_i ($i = 1, \dots, k$) defines a response mapping $s : X \rightarrow S_1 \times \dots \times S_k$.

One seeks a clustering of the categorized data based on conditional probabilities estimated as parameters by the maximum likelihood principle, cf. [40]. To this end we have to compute the likelihood function L , i.e., the probability of a known observation in terms of unknown parameters that will be estimated. In case of LCA we assume that we observe in our sample $N(s)$ times the vector $s = s(x)$ of categories, $s(x) = (s_1(x), \dots, s_k(x))^T \in S := S_1 \times \dots \times S_k$. Thus we know $s \rightarrow N(s)$ which maps each $s \in S$ to its number $N(s)$ of observations in the sample X . $N(s)$ is usually represented by a so-called **contingency table**. The likelihood function L is the probability of observing X , $|X| = \sum_{s \in S} N(s)$. Assuming that the observations in the sample are independent then

$$L(X) = \prod_{x \in X} \Pr(s = s(x)) = \prod_{a \in S} \Pr(s = a)^{N(a)}. \quad (2.1)$$

According to the maximum likelihood principle, the unknown probabilities are computed by maximizing the likelihood (2.1), or, equivalently, minimizing the negative loglikelihood

$$V(X) := -\log L(X) = -\sum_{a \in S} N(a) \log \Pr(s = a). \quad (2.2)$$

Let m be a fixed number of unknown, disjoint classes $C_1, \dots, C_m \subseteq \Omega$. The unknown proportional sizes

$$\Pr(\omega \in C_j) =: w_j \quad (2.3)$$

of the classes C_j , the so-called **class weights**, satisfy

$$0 < w_j \leq 1, \quad \sum_{j=1}^m w_j = 1.$$

We get the total probability

$$\begin{aligned}
\Pr(s(x) = a) &= \sum_{j=1}^m \Pr(s(x) = a, x \in C_j) \\
&= \sum_{j=1}^m \Pr(s(x) = a \mid x \in C_j) \Pr(x \in C_j) \\
&= \sum_{j=1}^m w_j \Pr(s(x) = a \mid x \in C_j). \tag{2.4}
\end{aligned}$$

In the following we suppress x in the formulas. The conditional probabilities, conditional on the class membership, are called **latent**. The classes are constructed according to the principle of **local stochastic independence**; i.e., the categories in each class are independent,

$$\Pr(s = a \mid x \in C_j) = \Pr(s_1 = a_1 \mid x \in C_j) \cdots \Pr(s_k = a_k \mid x \in C_j). \tag{2.5}$$

We denote the cardinality of S_i by $n_i := |S_i|$ and rewrite

$$\Pr(s_i = a_i \mid x \in C_j) =: p_{ij}(a_i). \tag{2.6}$$

Since for all i the n_i categories are alternatives (also called **dichotomous** if $n_i = 2$, and **polytomous** if $n_i > 2$), we have

$$\sum_{\alpha \in S_i} p_{ij}(\alpha) = 1$$

for all i, j . Thus (2.4), (2.5) and (2.6) imply

$$\Pr(s = a) = \sum_{j=1}^m w_j \prod_{i=1}^k p_{ij}(a_i). \tag{2.7}$$

2.1 Remark. In the special case of binary categories, i.e., $S_i = \{0, 1\}$ for all i , we may set $p_{ij} := p_{ij}(1)$ and get

$$p_{ij}(a_i) = \begin{cases} 1 - p_{ij} & \text{if } a_i = 0, \\ p_{ij} & \text{if } a_i = 1. \end{cases} \tag{2.8}$$

Hence $p_{ij}(a_i)$ can be written as

$$p_{ij}(a_i) = p_{ij}^{a_i} (1 - p_{ij})^{1-a_i}.$$

The full optimization problem for maximum likelihood estimation of the unknown parameters $w_j, p_{ij}(a_i), j = 1, \dots, m, i = 1, \dots, k, a_i \in S_i$, now reads as follows:

$$\left. \begin{aligned}
 \min_{w,p} \quad & - \sum_{a \in S} N(a) \log \Pr(s = a) \\
 \text{s.t.} \quad & \Pr(s = a) = \sum_{j=1}^m w_j \prod_{i=1}^k p_{ij}(a_i), \\
 & \sum_{\alpha \in S_i} p_{ij}(\alpha) = 1 \text{ for all } i, j, \\
 & \sum_{j=1}^m w_j = 1, \\
 & p_{ij}(a_i) \in [0, 1] \text{ for all } i, j, a_i, \\
 & w_j \in (0, 1] \text{ for all } j,
 \end{aligned} \right\} \quad (2.9)$$

with w_j as in (2.3) and $p_{ij}(a_i)$ as in (2.6).

The formulation (2.9) concerns the case of dense table data $N(s)$. For the sparse case one uses as objective function

$$\sum_{a \in S_0} N(a) \log \Pr(s = a), \quad (2.10)$$

where $S_0 = \{s(x) \mid x \in X, N(s(x)) > 0\}$ is the available categorized real sample in sparse format.

The number of variables in (2.9) is $n_{\text{var}} := m + m \sum_{i=1}^k n_i$. A feasible solution of (2.9) is given by $p_{ij}(a_i) = \frac{1}{n_i}$ for all i, j, a_i , and $w_j = \frac{1}{m}$ for all j , providing a possible starting point for the optimization.

2.2 Remark. The number of variables does not grow exponentially in k because the categories are separated by the local stochastic independence assumption.

2.3 Remark. In case all categories are binary, $n_i = 2$ for all i , thus $n_{\text{var}} = m + 2km$ which can be reduced to $n_{\text{var}} = m + km$ expressing $p_{ij}(0)$ in terms of $p_{ij}(1)$ as in (2.8).

The solution of (2.9) now allows the computation of the class membership $\Pr(x \in C_j \mid s = a)$ of a fixed $a \in S$ via the posterior probability

$$\begin{aligned}
 \Pr(x \in C_j \mid s = a) &= \Pr(s = a \mid x \in C_j) \frac{\Pr(x \in C_j)}{\Pr(s = a)} \\
 &= \frac{w_j \Pr(s = a \mid x \in C_j)}{\Pr(s = a)}, \quad (2.11)
 \end{aligned}$$

where $\Pr(s = a \mid x \in C_j)$ and $\Pr(s = a)$ can be computed from (2.5) and (2.7), respectively. The fixed $a \in S$ can then be classified uniquely to the class $\arg \max_{C_1, \dots, C_m} \Pr(s \in C_j \mid s = a)$, or in a fuzzy way, i.e., a belongs to each latent class with the membership degree (2.11).

The number of classes m is fixed a priori. To consider the quality of a choice for m we use goodness of fit test statistics in the next section, comparing for each $a \in S$ the fitted number $\Pr(s = a) \mid X$, cf. (2.7), with the available observations $N(a)$.

2.4 Remark. A simple idea for the choice of m in case that $|S_1| = \dots = |S_k|$ is $m = |S_1|$, i.e., the number of classes is the number of categories of each item. However, this recipe is not generally successful.

2.5 Remark. A bad selection of m causes a bad fit. A bad fit can also be caused simply by a bad concept in the item selection, e.g., items being arbitrarily related to each other, thus violating local stochastic independence, see, e.g., [24]. In general one seeks a latent class model with minimal m and a reasonable item (subset) selection, see Section 6.

3 Identifiability

Given a fixed item number k there is a maximum number of classes m_{\max} such that a reasonable fit is not possible if $m > m_{\max}$. In this case the number of estimated parameters is too large with respect to the available data points in X , i.e., there are different parameter sets giving exactly the same distribution. This issue is called the problem of **identification** of the parameters. There is a **necessary** condition on the degrees of freedom for the estimated parameters dof_{par} and for the data points dof_{data} , which gives us an upper bound for m_{\max} .

$$\text{dof}_{\text{par}} = n_{\text{var}} - 1 - mk, \quad (3.1)$$

$$\text{dof}_{\text{data}} = \prod_{i=1}^k n_i - 1. \quad (3.2)$$

Since identification of the parameters requires $\text{dof}_{\text{par}} \leq \text{dof}_{\text{data}}$, we have

$$m + m \sum_{i=1}^k n_i - 1 - mk \leq \prod_{i=1}^k n_i - 1 \quad (3.3)$$

$$\Leftrightarrow m \leq \frac{\prod_{i=1}^k n_i}{1 - k + \sum_{i=1}^k n_i}, \quad (3.4)$$

hence

$$m_{\max} \leq \left\lfloor \frac{\prod_{i=1}^k n_i}{1 - k + \sum_{i=1}^k n_i} \right\rfloor, \quad (3.5)$$

and for dichotomous data we have

$$m_{\max} \leq \left\lfloor \frac{2^k}{k + 1} \right\rfloor. \quad (3.6)$$

As described in [26] for the case of dichotomous data, m_{\max} is often far below this bound, as one investigates the minimal number k_{\min} of items that is required to identify a fixed number of classes m , i.e.,

$$k_{\min} = 2\lceil \log_2(m) \rceil + 1. \quad (3.7)$$

From this we infer a stricter limitation of m by

$$\tilde{m}_{\max} = 2^{\lfloor \frac{k-1}{2} \rfloor}, \quad (3.8)$$

which we will use in our algorithm for dichotomous data, cf. Section 6. The Table 3.1 shows some values of \tilde{m}_{\max} depending on k .

Table 3.1: Values of \tilde{m}_{\max} depending on the item number k (for dichotomous data).

k	1	2	3	4	5	6	7	8	9
\tilde{m}_{\max}	1	1	2	2	4	4	8	8	16

4 Goodness of fit

In the statistics literature, see, e.g., [25], the goodness of fit for latent class models is typically estimated by the smallness of the X^2 or G^2 statistic, as defined in (4.1), (4.2), respectively.

$$X^2(N) := \sum_{a \in S} \frac{(N(a) - |X| \Pr(s = a))^2}{|X| \Pr(s = a)}, \quad (4.1)$$

$$G^2(N) := 2 \sum_{a \in S} N(a) \log \frac{N(a)}{|X| \Pr(s = a)}. \quad (4.2)$$

The statistics G^2 is suitable for sparse data N , since

$$G^2(N) = 2 \sum_{a \in S} N(a) \log \frac{N(a)}{|X| \Pr(s = a)} = 2 \sum_{a \in S_0} N(a) \log \frac{N(a)}{|X| \Pr(s = a)}. \quad (4.3)$$

On the other hand, X^2 does not exploit sparse data structures N , so we need to do some manipulations (4.4). Otherwise one would generally need to compute $\Pr(s = a)$ for all $a \in S$, causing exponential computational growth in k , since $|S| \geq 2^k$ if $n_i \geq 2$ for all i . That is why many examples of dense data in the literature are dealing with a limited number of items, e.g., $k = 5$.

$$\begin{aligned} X^2(\vec{0}) - X^2(N) &= \sum_{a \in S} \frac{(|X| \Pr(s = a))^2}{|X| \Pr(s = a)} - \sum_{a \in S} \frac{(N(a) - |X| \Pr(s = a))^2}{|X| \Pr(s = a)} \\ &= \sum_{a \in S} \frac{2N(a)|X| \Pr(s = a) - N(a)^2}{|X| \Pr(s = a)} = \sum_{a \in S} N(a) \left(2 - \frac{N(a)}{|X| \Pr(s = a)} \right) \\ &= \sum_{a \in S_0} N(a) \left(2 - \frac{N(a)}{|X| \Pr(s = a)} \right) \end{aligned} \quad (4.4)$$

with

$$X^2(\vec{0}) = \sum_{a \in S} |X| \Pr(s = a) = |X| \Pr(s \in S) = |X|, \quad (4.5)$$

hence

$$X^2(N) = |X| - \sum_{a \in S_0} N(a) \left(2 - \frac{N(a)}{|X| \Pr(s = a)} \right). \quad (4.6)$$

This representation of X^2 is suitable for sparse N , as now it is not necessary to compute $\Pr(s = a)$ for all possible patterns $a \in S$, but only for $a \in S_0$ to get X^2 .

Both statistics X^2 and G^2 are asymptotically χ^2 -distributed with

$$\text{dof} = \text{dof}_{\text{data}} - \text{dof}_{\text{par}} = \prod_{i=1}^k n_i - 1 - (n_{\text{var}} - 1 - mk) = \prod_{i=1}^k n_i - n_{\text{var}} + mk \quad (4.7)$$

degrees of freedom (i.e., $\text{dof} = 2^k - m(k + 1)$ for dichotomous data).

We define that a **good fit** means that the two criteria

$$F_{\chi^2, \text{dof}}(X^2) \leq 1 - \alpha_1, \quad (4.8)$$

$$F_{\chi^2, \text{dof}}(G^2) \leq 1 - \alpha_2 \quad (4.9)$$

are satisfied, where $F_{\chi^2, \text{dof}}$ is the cumulative distribution function (CDF) of the χ^2 -distribution with dof degrees of freedom, α_1, α_2 are some significance values, e.g., $\alpha_1 = \alpha_2 = 5\%$, and X^2, G^2 as in (4.6), (4.3), respectively. Increasing α_1 or α_2 makes the constraints (4.8) and (4.9) more difficult to achieve and thus strengthens the criterion for a good fit.

4.1 Remark. In [11, 12] one finds some well-known criticism on this kind of goodness of fit statistics, e.g., that a good fit according to our definition may have a large deviation in the 1-norm, or that the statistics X^2 and G^2 are **asymptotically** χ^2 -distributed, though we do not know what sample size $|X|$ is sufficient to justify the χ^2 distribution assumption. These questions are out of the scope of this paper, as we rather focus on solving the optimization problem (2.9), also see Section 5.

To ensure that our solution of (2.9) yields a good fit – no matter whether we have succeeded to find a global solution or only a local solution of (2.9) – we add (4.8) and (4.9) as new constraints to (2.9) with user-defined α_1, α_2 . By adding these constraints a good fit is **guaranteed** if we find a feasible solution. Moreover, if we can verify that no feasible solution exists (e.g., using a rigorous solver) we know that no good fit exists.

5 Global and local extrema

In the literature there are several examples of LCA problems with non-global optima reported, e.g., in [11] for constrained LCA, or in [24] for unconstrained LCA. Given the same data set and fixed m , multiple optima lead to different latent class models because of different parameter estimators. The estimator associated with the global optimum typically leads to the best fit, as opposed to suboptimal local extrema that may lead to poor fits and should then be avoided.

Unfortunately validation of global optimality is difficult, especially for scarce data sets. Intuitively scarce means that the data is spread over too many latent classes. As mentioned in Remark 4.1 the precise meaning of 'scarce data' or 'sufficient amount of data' with respect to k and m is difficult to formalize and out of the scope of this paper.

In a preliminary study, we looked for global solutions using our Test Environment, cf. [8], employing the global solver Baron, cf. [33, 34] with GAMS interface, cf. [1]. We made numerical tests with small LCA toy problems with plenty of data, trying to find a global solution claimed by Baron. However, within a timeout of 30 minutes, Baron could only

claim local optimality for its solutions, even for small LCA toy problems, which underlines that (2.9) is a very hard global optimization problem. We see that the **validation** of the global optimum is very difficult, but it may not be necessarily needed. A local optimum is sufficient in LCA if the resulting parameter estimation gives a sufficiently good fit, cf. Section 4.

Thus we use the local solver Knitro, cf. [2], with an interface to the algebraic modeling language AMPL, cf. [13], to repeatedly solve (2.9) with the additional goodness of fit constraints (4.8) and (4.9). As Problem 1 from Section 7 shows, this formulation is superior to the direct use of (2.9), followed by checking the goodness of fit statistics only in the postprocessing phase, since poor optima are automatically eliminated.

We use multistart rather to enlarge the set of good solutions found, and achieve N_{ms} many different optima $(w^1, p^1)^T, \dots, (w^{N_{\text{ms}}}, p^{N_{\text{ms}}})^T$. With this numerical tests we found out two typical features of the solution space: (i) the solutions were never observed to form an apparent continuum; (ii) we observe **symmetry classes** of discrete solutions, e.g., imposed by permutation of classes – hence there are at least $m!$ symmetry classes of solutions.

From the solutions $(w^1, p^1)^T, \dots, (w^{N_{\text{ms}}}, p^{N_{\text{ms}}})^T$ we remove the symmetries that arise from permutation of class weights as follows. For all class permutations $\pi \in \mathfrak{S}_m$ (with the symmetric group \mathfrak{S}_m), we compare solutions $(w^i, p^i), (w^j, p^j), i \neq j$ pairwise by applying π , i.e., two solutions i and j are considered to be symmetric if $w_\ell^i = w_{\pi(\ell)}^j$ and $p_{i\ell}^i(a_i) = p_{i\pi(\ell)}^j(a_i)$ for all $\ell = 1, \dots, m, i = 1, \dots, k, a_i \in S_i$. The class symmetry gives equivalence classes of solutions. To get a normalized set of solutions that is invariant of class permutations one considers the solutions modulo symmetry.

To make the comparison numerically more robust we use

$$\delta \cdot \text{round} \left(\frac{1}{\delta} (w^i, p^i) \right) \tag{5.1}$$

in place of (w^i, p^i) , where δ is a user-defined precision. In our examples we use $\delta = 10^{-2}$. We observe that the number of different solutions modulo symmetry does not change significantly, e.g., between $\delta = 10^{-4}$ and $\delta = 10^{-2}$ in numerical tests.

Based on the considerations in the last sections we now propose our algorithmic approach for LCA.

6 Algorithmic approach

Our strategy basically consists of the following three steps:

- **Phase A.** Find normalized solutions for a good fit, with the smallest possible m , cf. Section 4,
- **Phase B.** Find reasonable item subset selection if necessary, see Remark 2.5,
- **Phase C.** Investigate the local optima found with multistart, see Section 5.

6.1 Phase A: Find normalized solutions

To be able to solve (2.9) we first generate automatically a modified formulation of it in the modeling language AMPL, cf. [13]. As mentioned in Section 4 we add the goodness of fit constraints (4.8) and (4.9) with user-defined α_1, α_2 . If $\alpha_1 = 0$ the constraint (4.8) is skipped, if $\alpha_2 = 0$ the constraint (4.9) is skipped.

We also add a constraint such that the class weights in the solution are sorted, i.e., $w_1 \leq w_2 \leq \dots \leq w_m$. Thus we directly get normalized solutions except for the case in which two or more weights are equal. If this happens while analyzing multistart solutions in \mathbf{c} we remove all class symmetries, as described in Section 5.

Finally, since \log is singular at 0, we add the lower bound 10^{-20} on all variables in (2.9) to avoid that a solver chooses the initial point 0, which may cause problems in some solvers, e.g., for Baron. Thus we actually solve the following optimization problem.

$$\begin{aligned}
& \min_{w,p} - \sum_{a \in S_0} N(a) \log \Pr(s = a) \\
& \text{s.t. } \Pr(s = a) = \sum_{j=1}^m w_j \prod_{i=1}^k p_{ij}(a_i), \\
& \quad \sum_{\alpha \in S_i} p_{ij}(\alpha) = 1 \text{ for all } i, j, \\
& \quad \sum_{j=1}^m w_j = 1, \\
& \quad X^2 \leq F_{\chi^2, \text{dof}}^{-1}(1 - \alpha_1), \\
& \quad X^2 = |X| - \sum_{a \in S_0} N(a) \left(2 - \frac{N(a)}{|X| \Pr(s = a)} \right), \\
& \quad G^2 \leq F_{\chi^2, \text{dof}}^{-1}(1 - \alpha_2), \\
& \quad G^2 = 2 \sum_{a \in S_0} N(a) \log \frac{N(a)}{|X| \Pr(s = a)}, \\
& \quad w_1 \leq w_2 \leq \dots \leq w_m, \\
& \quad p_{ij}(a_i) \in [10^{-20}, 1] \text{ for all } i, j, a_i, \\
& \quad w_j \in [10^{-20}, 1] \text{ for all } j.
\end{aligned} \tag{6.1}$$

6.1 Remark. The automatic generation of (6.1) also considers Remark 2.3 to reduce the number of variables in case of dichotomous data. Also note that $F_{\chi^2, \text{dof}}^{-1}(1 - \alpha_1)$ and $F_{\chi^2, \text{dof}}^{-1}(1 - \alpha_2)$ are simply fixed real values.

To find normalized solutions for a good fit, with the smallest possible m we start looking for a **feasible solution** of (6.1) with $m = 1$. If we cannot find any we increase m by 1 and repeat this until we have found a feasible solution, i.e., we have determined a good fit. A feasible solution in our implementation is defined as a Knitro solution with exit status 'Locally optimal solution found.' and 'feasibility error' \leq maxinfeas, where maxinfeas is a user-defined value that is set by default to 10^{-5} . If we failed to find a feasible solution for all $m \leq \tilde{m}_{\max}$ we could not determine a good fit for the given data. The value of \tilde{m}_{\max} can also be restricted by the user to avoid large m that may be difficult to interpret. Users may also choose interactively not to accept a feasible solution found, if they think that the selected item set (cf. **b**), or the class number m belonging to the solution do not make sense.

6.2 Remark. We only check a **necessary** condition for identifiability of the parameters (as discussed in Section 3). In practice, lack of identifiability shows in the existence of multiple minimizers of (6.1).

6.3 Remark. Note that the user definable parts of (6.1) that have impact on the feasibility of the problem are α_1 , α_2 , \tilde{m}_{\max} , `maxinfeas`, and the selected item subset (see **b**).

Correctly claiming an infeasible solution is rather difficult. On the one hand, if (6.1) is infeasible, Knitro often runs until timeout. On the other hand, if (6.1) has a feasible solution though Knitro did not find any before timeout, the use of the timeout solution of Knitro as a starting point sometimes leads to a feasible solution in an iterative step. Hence we run with short timeout a few iterations (this number, `infeasniter`, is user-definable, with default value 2), rather than one run with long timeout. If we find no feasible solution within these iterations we try a completely new random starting point several times (this number, `randniter`, is also user-definable, with default value 5). To avoid a possibly very time consuming search for a feasible solution, one should use a small `randniter`; to decrease the likelihood of missing feasible solutions, one should use a high `randniter` (it may take 50 or more iterations in some cases). If no feasible solution was found in all iterations, we consider (6.1) infeasible and exit or increase m as described above.

6.2 Phase B: Subset selection

If we have found (6.1) infeasible for the initial item set $\{I_1, \dots, I_k\}$ we choose a subset of this set, i.e., if no good fit was feasible, then we leave out some items, and rerun the whole procedure. Selecting an item I_ℓ means taking the ℓ^{th} column from the contingency table.

After selecting a subset of items the new corresponding contingency table may have multiple entries for the same pattern, e.g., if we select I_1, I_2, I_3 from 5 dichotomous items then the entries $a^1 = (0, 0, 0, 0, 1)$ and $a^2 = (0, 0, 0, 1, 0)$, $a^1, a^2 \in S = \{0, 1\}^5$ both become $a^{\text{new}} = (0, 0, 0)$, $a^{\text{new}} \in S_{\text{new}} = \{0, 1\}^3$, so the observations $N(a^1)$ and $N(a^2)$ have to be summed up to get the new contingency table entry for $N_{\text{new}}(a^{\text{new}})$.

We perform the item subset selection automatically starting with all possible subsets with $k - 1$ elements, and decreasing the cardinality of the item subsets until we have found a feasible solution with the procedure described in Phase A.

6.3 Phase C: Multistart

Having found a feasible solution for some m and some fixed item subset, we perform a more exhaustive investigation of local extrema. This is done by multistart on (6.1) with the found item subset and a fixed m , using significantly more starting points than `randniter`. First do multistart on the found m decreased by 1, to safeguard against the case that a feasible solution for a lower m was lost. If multistart has found a feasible solution such that no smaller m leads to a feasible solution, we remove the symmetries of the multistart

solutions as described in Section 5. Thus we find normalized solutions, and use the best solution found as our final parameter estimator for LCA.

We also investigate the number of occurrences of the local solutions found, see, e.g., Table 7.6, to assess the detection rate of the local solutions by our solver.

An implementation of our approach in Python can be found at [16].

7 Numerical results

We have run numerical experiments with three difficult problems from the literature, known for their multiple local minima: two problems with 5 items, i.e., Problem 1 taken from [11], Problem 2 taken from [25], and one problem with 8 items, i.e., Problem 3 taken from [24]. The data in all three cases are dichotomous. The timeout is set to 3 seconds. Time is of rather little importance as the primary goal is a good fit. We focus on exploiting recent optimization techniques to find a good fit (cf. Section 4) and an automated item subset selection, rather than minimizing the computational effort. Nevertheless the computation time is reasonably small according to the small timeout we selected.

The user-definable parameters in the algorithm, see Remark 6.3, have been chosen by $\alpha_1 = 0.05$, $\alpha_2 = 0.05$, \tilde{m}_{\max} as in Table 3.1, $\text{maxinfeas} = 10^{-5}$.

The results show that we find the best solutions of the problems at reasonable frequencies. Adding the goodness of fit constraints discussed in Section 4 pays off. Also the subset selection proved to be useful, in particular in Problem 3. The following sections 7.1 to 7.3 give the details on each problem.

7.1 Problem 1

Problem 1 is given by a sample of size $|X| = 3869$ with $k = 5$ items. Table 7.1 provides the associated contingency table.

We have reproduced the best solution with $m = 3$ classes for the subset $\{I_1, I_2, I_3, I_4, I_5\}$, i.e.,

Table 7.1: Problem 1: Contingency table $s \rightarrow N(s)$ for $N(s) > 0$.

s	$N(s)$	s	$N(s)$	s	$N(s)$	s	$N(s)$
[0,0,0,0,0]	1880	[0,1,0,0,0]	188	[1,0,0,0,0]	22	[1,1,0,0,0]	2
[0,0,0,0,1]	789	[0,1,0,0,1]	191	[1,0,0,0,1]	26	[1,1,0,0,1]	20
[0,0,0,1,0]	43	[0,1,0,1,0]	17	[1,0,0,1,0]	6	[1,1,0,1,0]	6
[0,0,0,1,1]	75	[0,1,0,1,1]	67	[1,0,0,1,1]	14	[1,1,0,1,1]	27
[0,0,1,0,0]	23	[0,1,1,0,0]	15	[1,0,1,0,0]	1	[1,1,1,0,0]	3
[0,0,1,0,1]	63	[0,1,1,0,1]	85	[1,0,1,0,1]	20	[1,1,1,0,1]	72
[0,0,1,1,0]	8	[0,1,1,1,0]	8	[1,0,1,1,0]	2	[1,1,1,1,0]	1
[0,0,1,1,1]	22	[0,1,1,1,1]	56	[1,0,1,1,1]	17	[1,1,1,1,1]	100

$$w = (0.0733, 0.2099, 0.7169), \quad (7.1)$$

$$p = \begin{pmatrix} 0.7436 & 0.1302 & 0.0081 \\ 0.8757 & 0.4916 & 0.0759 \\ 0.8764 & 0.2907 & 0.0041 \\ 0.5940 & 0.3245 & 0.0134 \\ 0.9959 & 0.7803 & 0.2625 \end{pmatrix}, \quad (7.2)$$

$$X^2 = 21.35, \quad (7.3)$$

$$G^2 = 21.53. \quad (7.4)$$

This is the result reported in [11]. The multistart phase with $N_{\text{ms}} = 50$ for $m = 3$ classes and subset $\{I_1, I_2, I_3, I_4, I_5\}$ gives just this single feasible solution modulo symmetry, cf. Table 7.2.

Table 7.2: Feasible solution modulo symmetry and frequency of occurrence, $N_{\text{ms}} = 50$.

solution	#1	none
obj. fct.	7411.2271	
w_1	0.0733	
w_2	0.2099	
w_3	0.7169	
freq.	11	39
rel. freq.	0.22	0.78

For the multistart phase with $N_{\text{ms}} = 50, 100, 500, 1000$ see Table 7.3, no further solutions were found. Observing convergence of the relative frequencies over N_{ms} we can expect to find the optimum in about 20% of the cases.

Table 7.3: Frequency of occurrence of the solution, $N_{\text{ms}} = 50, 100, 500, 1000$.

solution	#1	none
rel. freq. ($N_{\text{ms}} = 50$)	0.2200	0.7800
rel. freq. ($N_{\text{ms}} = 100$)	0.2300	0.7700
rel. freq. ($N_{\text{ms}} = 500$)	0.1860	0.8140
rel. freq. ($N_{\text{ms}} = 1000$)	0.1760	0.8240

Moreover, to investigate the utility of the constraints we added to ensure goodness of fit, we repeat the multistart phase without the goodness of fit constraints and find **16 different** feasible solutions modulo symmetry. However, the only one of these that yields a sufficiently good fit is the one stated above. Hence by adding the constraints we reduce the number of local minima significantly, and we find the one unambiguous solution that respects the goodness of fit.

7.2 Problem 2

Problem 2 is given by a sample of size $|X| = 1163$ with $k = 5$ items. Table 7.4 provides the associated contingency table.

Table 7.4: Problem 2: Contingency table $s \rightarrow N(s)$ for $N(s) > 0$.

s	$N(s)$	s	$N(s)$	s	$N(s)$	s	$N(s)$
[0,0,0,0,0]	53	[0,1,0,0,0]	30	[1,0,0,0,0]	83	[1,1,0,0,0]	133
[0,0,0,0,1]	6	[0,1,0,0,1]	4	[1,0,0,0,1]	7	[1,1,0,0,1]	22
[0,0,0,1,0]	10	[0,1,0,1,0]	10	[1,0,0,1,0]	31	[1,1,0,1,0]	66
[0,0,0,1,1]	1	[0,1,0,1,1]	1	[1,0,0,1,1]	6	[1,1,0,1,1]	17
[0,0,1,0,0]	28	[0,1,1,0,0]	29	[1,0,1,0,0]	74	[1,1,1,0,0]	189
[0,0,1,0,1]	2	[0,1,1,0,1]	9	[1,0,1,0,1]	17	[1,1,1,0,1]	59
[0,0,1,1,0]	10	[0,1,1,1,0]	4	[1,0,1,1,0]	36	[1,1,1,1,0]	120
[0,0,1,1,1]	4	[0,1,1,1,1]	5	[1,0,1,1,1]	11	[1,1,1,1,1]	86

We have reproduced the best solution with $m = 3$ classes for the subset $\{I_1, I_2, I_3, I_4, I_5\}$, i.e.,

$$w = (0.1303, 0.3830, 0.4866), \quad (7.5)$$

$$p = \begin{pmatrix} 0.9084 & 0.5688 & 1.0000 \\ 0.8731 & 0.4235 & 0.8181 \\ 0.8862 & 0.4008 & 0.6540 \\ 0.6292 & 0.2014 & 0.4115 \\ 1.0000 & 0.0915 & 0.1143 \end{pmatrix}, \quad (7.6)$$

$$X^2 = 14.12, \quad (7.7)$$

$$G^2 = 13.74. \quad (7.8)$$

The multistart phase with $N_{\text{ms}} = 50$ for $m = 3$ classes and subset $\{I_1, I_2, I_3, I_4, I_5\}$ gives 5 different feasible solutions modulo symmetry, cf. Table 7.5. The best solution found is #3. This is the result reported in [25].

Table 7.5: Different feasible solutions modulo symmetry and their frequency of occurrence, $N_{\text{ms}} = 50$.

solution	#1	#2	#3	#4	#5	none
obj. fct.	3336.6677	3336.8852	3335.5426	3336.8456	3335.8614	
w_1	0.2820	0.3333	0.1303	0.3229	0.1544	
w_2	0.2820	0.3333	0.3830	0.3229	0.4228	
w_3	0.4359	0.3333	0.4866	0.3542	0.4228	
freq.	10	17	10	4	7	2
rel. freq.	0.20	0.34	0.20	0.08	0.14	0.04

For the multistart phase with $N_{\text{ms}} = 50, 100, 500, 1000$ see Table 7.6. We observe that the relative frequencies over N_{ms} converge, if we increase N_{ms} , so we can expect to find the optimum in about 20% of the cases.

Table 7.6: Different feasible solutions modulo symmetry and their frequency of occurrence, $N_{\text{ms}} = 50, 100, 500, 1000$.

solution	#1	#2	#3	#4	#5	none
rel. freq. ($N_{\text{ms}} = 50$)	0.2000	0.3400	0.2000	0.0800	0.1400	0.0400
rel. freq. ($N_{\text{ms}} = 100$)	0.1700	0.3000	0.2500	0.0600	0.1900	0.0300
rel. freq. ($N_{\text{ms}} = 500$)	0.1600	0.2720	0.2160	0.1080	0.2100	0.0340
rel. freq. ($N_{\text{ms}} = 1000$)	0.1520	0.2720	0.2440	0.1020	0.2010	0.0290

As in Problem 1, we also investigate the utility of the constraints we added to ensure goodness of fit. We solve (6.1) two times with the fixed starting point $w_j = 1, p_{ij} = 1$

for all i, j : one time including the goodness of fit constraints, and one time without them. Including the constraints we find the solution #3 reported above. However, neglecting them, we could only find a **non-global solution**. This underlines the importance to make use of all constraints in (6.1).

7.3 Problem 3

Problem 3 is given by a sample of size $|X| = 7161$ with $k = 8$ items. The associated contingency table is given in the appendix. We reproduced the best solution from the literature with $m = 4$ classes for the subset $\{I_1, I_2, I_3, I_4, I_5\}$, i.e.,

$$w = (0.0574, 0.1226, 0.2108, 0.6092), \quad (7.9)$$

$$p = \begin{pmatrix} 0.9144 & 0.4045 & 0.6864 & 0.0658 \\ 0.8519 & 0.2334 & 0.5561 & 0.0435 \\ 0.9539 & 0.8965 & 0.4024 & 0.0673 \\ 0.5669 & 0.4526 & 0.0448 & 0.0200 \\ 0.9160 & 0.4164 & 0.1177 & 0.0137 \end{pmatrix}, \quad (7.10)$$

$$X^2 = 8.53, \quad (7.11)$$

$$G^2 = 8.43. \quad (7.12)$$

The multistart phase with $N_{\text{ms}} = 50$ for $m = 4$ classes and subset $\{I_1, I_2, I_3, I_4, I_5\}$ gives 3 different feasible solutions modulo symmetry, cf. Table 7.7. The best solution found is #1, reproducing the result reported in [24].

Table 7.7: Different feasible solutions modulo symmetry and their frequency of occurrence, $N_{\text{ms}} = 50$.

solution	#1	#2	#3	none
obj. fct.	15317.8513	15320.7927	15320.1922	
w_1	0.0574	0.0842	0.0487	
w_2	0.1226	0.0842	0.1593	
w_3	0.2108	0.2473	0.1593	
w_4	0.6092	0.5842	0.6326	
freq.	2	4	3	41
rel. freq.	0.04	0.08	0.06	0.82

For the multistart phase with $N_{\text{ms}} = 50, 100, 500, 1000, 10000$ see Table 7.8. Observing convergence of the relative frequencies over N_{ms} we can expect to find the optimum in about 10% of the cases.

Table 7.8: Different feasible solutions modulo symmetry and their frequency of occurrence, $N_{\text{ms}} = 50, 100, 500, 1000, 10000$.

solution	#1	#2	#3	none
rel. freq. ($N_{\text{ms}} = 50$)	0.0400	0.0800	0.0600	0.8200
rel. freq. ($N_{\text{ms}} = 100$)	0.0500	0.1000	0.0700	0.7800
rel. freq. ($N_{\text{ms}} = 500$)	0.0820	0.1140	0.0440	0.7600
rel. freq. ($N_{\text{ms}} = 1000$)	0.1000	0.1050	0.0500	0.7450
rel. freq. ($N_{\text{ms}} = 10000$)	0.0904	0.0959	0.0643	0.7494

We also found a **different** item subset leading to a good fit with a higher number of items and classes, which shows a benefit of the subset selection in Phase B of our algorithm. The best solution with $m = 8$ classes for the subset $\{I_1, I_2, I_3, I_4, I_5, I_6, I_8\}$ is

$$w = (0.0258, 0.0273, 0.0599, 0.0815, 0.0884, 0.0886, 0.0949, 0.5337), \quad (7.13)$$

$$p = \begin{pmatrix} 0.0000 & 0.4827 & 0.7972 & 0.8812 & 0.4178 & 0.0686 & 0.6952 & 0.0845 \\ 0.1816 & 0.3842 & 0.6207 & 0.6811 & 0.2677 & 0.0961 & 0.6249 & 0.0410 \\ 0.9237 & 0.3471 & 0.7247 & 0.9670 & 1.0000 & 0.1070 & 0.1111 & 0.0496 \\ 0.5823 & 0.1167 & 0.0806 & 0.5083 & 0.2782 & 0.0771 & 0.0655 & 0.0141 \\ 0.6946 & 1.0000 & 0.0000 & 0.9110 & 0.0862 & 0.0061 & 0.0627 & 0.0059 \\ 0.8055 & 0.4273 & 0.9396 & 0.9384 & 0.4550 & 0.8988 & 0.0249 & 0.0000 \\ 0.8584 & 0.8900 & 0.4785 & 0.9132 & 0.1190 & 0.2660 & 0.0604 & 0.0077 \end{pmatrix}, \quad (7.14)$$

$$X^2 = 81.00, \quad (7.15)$$

$$G^2 = 81.26. \quad (7.16)$$

The multistart phase in this scenario with $N_{\text{ms}} = 10000$ gives 49 different feasible solutions, with small frequencies of occurrence between 1 and 6. In total we find the very best solution in 1 case, and generally a feasible solution and thus a good fit in about 1% of the runs.

Acknowledgments

The paper is dedicated to the memory of Anton Formann, whose cooperation concerning the statistical side of the problem is highly appreciated.

Partial funding of the research by the research grant FSP 506/003 of the University of Vienna is gratefully acknowledged.

References

- [1] Brooke, A., Kendrick, D., and Meeraus, A. (1988). *GAMS: A User's Guide*. The Scientific Press.
- [2] Byrd, R., Nocedal, J., and Waltz, R. (2006). *Large-Scale Nonlinear Optimization*, chapter KNITRO: An Integrated Package for Nonlinear Optimization, pages 35–59. Springer.
- [3] Clogg, C. (1988). *Latent trait and latent class models*, chapter Latent class models for measuring, pages 173–205. Plenum Press New York.
- [4] Collins, L., Lanza, S., Schafer, J., and Flaherty, B. (2002). *WinLTA Users Guide. Version 3.0*. University Park: The Pennsylvania State University, The Methodology Center.
- [5] Collins, L., Lanza, S., Schafer, J., and Flaherty, B. (2010). WinLTA for Latent Transition Analysis. Web document, <http://methodology.psu.edu/downloads/winlta>.
- [6] Dayton, C. and Macready, G. (1976). A probabilistic model for validation of behavioral hierarchies. *Psychometrika* **41**, 189–204.
- [7] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* **39**, 1–38.
- [8] Domes, F., Fuchs, M., and Schichl, H. (2010). The optimization test environment. Submitted. Preprint available on-line at: <http://www.mat.univie.ac.at/~dferi/testenv.html>.
- [9] Fischer, G. and Molenaar, I. (1995). *Rasch models: Foundations, recent developments, and applications*. Springer.
- [10] Formann, A. (1985). Constrained latent class models: Theory and applications. *British Journal of Mathematical and Statistical Psychology* **38**, 87–111.
- [11] Formann, A. (1994). Measurement errors in caries diagnosis: some further latent class models. *Biometrics* **50**, 865–871.
- [12] Formann, A. (2003). Latent class model diagnosis from a frequentist point of view. *Biometrics* **59**, 189–196.
- [13] Fourer, R., Gay, D., and Kernighan, B. (2002). *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press/Brooks/Cole Publishing Company.
- [14] Fraley, C. and Raftery, A. (2006). MCLUST Version 3 for R: Normal Mixture Modeling and Model-Based Clustering. Technical Report 504, Department of Statistics, University of Washington, Seattle, USA.

- [15] Fraley, C. and Raftery, A. (2010). MCLUST: Multivariate Normal Mixture Modeling and Model-Based Clustering. Web document, <http://www.stat.washington.edu/mclust>.
- [16] Fuchs, M. (2010). Latent class analysis. Web document, <http://www.martin-fuchs.net/lca.php>.
- [17] Goodman, L. (1974a). The analysis of systems of qualitative variables when some of the variables are unobservable. Part I – A modified latent structure approach. *American Journal of Sociology* **79**, 1179–1259.
- [18] Goodman, L. (1974b). Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika* **61**, 215–231.
- [19] Hanneman, R. (2010). CDAS Program MLLSA. Web document, <http://faculty.ucr.edu/~hanneman/cdas/mllsa.htm>.
- [20] Haughton, D., Legrand, P., and Woolford, S. (2009). Review of three latent class cluster analysis packages: Latent Gold, poLCA, and MCLUST. *The American Statistician* **63**, 81–91.
- [21] Huang, G. and Bandeen-Roche, K. (2004). Building an identifiable latent class model with covariate effects on underlying and measured variables. *Psychometrika* **69**, 5–32.
- [22] Hunt, L. and Jorgensen, M. (1999). Mixture model clustering using the MULTIMIX program. *Australian & New Zealand Journal of Statistics* **41**, 154–171.
- [23] Hunt, L. and Jorgensen, M. (2010). MULTIMIX. Web document, <http://www.stats.waikato.ac.nz/Staff/maj/multimix>.
- [24] Kohlmann, T. and Formann, A. (1997). *Applications of latent trait and latent class models in the social sciences*, chapter Using latent class models to analyze response patterns in epidemiologic mail surveys, pages 345–352. Waxmann Münster, Germany.
- [25] Kotz, S., Read, C., and Banks, D. (1999). *Encyclopedia of statistical sciences, update volume 3*. John Wiley & Sons.
- [26] Lazarsfeld, P. and Henry, N. (1968). *Latent structure analysis*. Houghton, Mifflin.
- [27] Linzer, D. and Lewis, J. (2010a). poLCA: Polytomous Variable Latent Class Analysis. Web document, <http://userwww.service.emory.edu/~dlinzer/poLCA>.
- [28] Linzer, D. and Lewis, J. (2010b). poLCA: Polytomous Variable Latent Class Analysis. Version 1.2. Manuscript.
- [29] Macready, G. and Dayton, C. (1977). The use of probabilistic models in the assessment of mastery. *Journal of Educational Statistics* **2**, 99–120.

- [30] Muthen, L. and Muthen, B. (2010a). Mplus. Web document, <http://www.statmodel.com>.
- [31] Muthen, L. and Muthen, B. (2010b). *Mplus: Statistical Analysis With Latent Variables – User’s Guide*. Muthen & Muthen.
- [32] Rasch, G. (1993). *Probabilistic models for some intelligence and attainment tests*. MESA Press.
- [33] Sahinidis, N. and Tawarmalani, M. (2005). BARON 7.2.5: Global optimization of mixed-integer nonlinear programs. User’s Manual. Available on-line at: <http://www.gams.com/dd/docs/solvers/baron.pdf>.
- [34] Tawarmalani, M. and Sahinidis, N. (2004). Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming* **99**, 563–591.
- [35] Uebersax, J. (2009). Latent Structure Analysis. Web document, <http://www.john-uebersax.com/stat/index.htm>.
- [36] Vermunt, J. (1997). *LEM: A general program for the analysis of categorical data*. Department of Methodology and Statistics, Tilburg University.
- [37] Vermunt, J. (2010). LEM. Web document, <http://www.uvt.nl/faculteiten/fsw/organisatie/departementen/mto/software2.html>.
- [38] Vermunt, J. and Magidson, J. (2010a). Latent GOLD 4.5. Web document, http://www.statisticalinnovations.com/products/latentgold_v4.html.
- [39] Vermunt, J. and Magidson, J. (2010b). *Latent GOLD User’s Guide*. Statistical Innovations.
- [40] Weisstein, E. (2010). Maximum Likelihood. MathWorld – A Wolfram Web Resource. Available on-line at: <http://mathworld.wolfram.com/MaximumLikelihood.html>.

Appendix

Problem 3 contingency table $s \rightarrow N(s)$ for $N(s) > 0$

s	$N(s)$	s	$N(s)$	s	$N(s)$
[0,0,0,0,0,0,0]	3145	[0,1,0,0,1,1,0,0]	2	[1,0,1,0,1,1,0,1]	67
[0,0,0,0,0,0,0,1]	37	[0,1,0,0,1,1,0,1]	7	[1,0,1,0,1,1,1,0]	1
[0,0,0,0,0,0,1,0]	46	[0,1,0,0,1,1,1,1]	1	[1,0,1,0,1,1,1,1]	18
[0,0,0,0,0,0,1,1]	3	[0,1,0,1,0,0,0,0]	4	[1,0,1,1,0,0,0,0]	10
[0,0,0,0,0,1,0,0]	242	[0,1,0,1,0,0,1,0]	6	[1,0,1,1,0,0,0,1]	1
[0,0,0,0,0,1,0,1]	64	[0,1,0,1,0,1,0,0]	1	[1,0,1,1,0,0,1,0]	12
[0,0,0,0,0,1,1,0]	49	[0,1,0,1,0,1,0,1]	1	[1,0,1,1,0,0,1,1]	3
[0,0,0,0,0,1,1,1]	48	[0,1,0,1,0,1,1,0]	1	[1,0,1,1,0,1,0,0]	6
[0,0,0,0,1,0,0,0]	24	[0,1,0,1,0,1,1,1]	2	[1,0,1,1,0,1,0,1]	5
[0,0,0,0,1,0,0,1]	24	[0,1,0,1,1,0,0,0]	1	[1,0,1,1,0,1,1,0]	22
[0,0,0,0,1,0,1,0]	1	[0,1,0,1,1,0,1,1]	1	[1,0,1,1,0,1,1,1]	12
[0,0,0,0,1,0,1,1]	1	[0,1,0,1,1,1,1,0]	1	[1,0,1,1,1,0,0,0]	2
[0,0,0,0,1,1,0,0]	4	[0,1,0,1,1,1,1,1]	2	[1,0,1,1,1,0,0,1]	2
[0,0,0,0,1,1,0,1]	13	[0,1,1,0,0,0,0,0]	50	[1,0,1,1,1,0,1,1]	2
[0,0,0,0,1,1,1,1]	6	[0,1,1,0,0,0,0,1]	9	[1,0,1,1,1,1,0,0]	1
[0,0,0,1,0,0,0,0]	23	[0,1,1,0,0,0,1,0]	3	[1,0,1,1,1,1,0,1]	10
[0,0,0,1,0,0,0,1]	1	[0,1,1,0,0,0,1,1]	1	[1,0,1,1,1,1,1,0]	3
[0,0,0,1,0,0,1,0]	26	[0,1,1,0,0,1,0,0]	42	[1,0,1,1,1,1,1,1]	45
[0,0,0,1,0,0,1,1]	1	[0,1,1,0,0,1,0,1]	23	[1,1,0,0,0,0,0,0]	224
[0,0,0,1,0,1,0,0]	9	[0,1,1,0,0,1,1,0]	9	[1,1,0,0,0,0,0,1]	16
[0,0,0,1,0,1,0,1]	2	[0,1,1,0,1,0,0,0]	4	[1,1,0,0,0,0,1,0]	3
[0,0,0,1,0,1,1,0]	18	[0,1,1,0,1,0,0,1]	7	[1,1,0,0,0,1,0,0]	27
[0,0,0,1,0,1,1,1]	7	[0,1,1,0,1,1,0,0]	3	[1,1,0,0,0,1,0,1]	20
[0,0,0,1,1,0,0,0]	1	[0,1,1,0,1,1,0,1]	19	[1,1,0,0,0,1,1,0]	8
[0,0,0,1,1,0,0,1]	1	[0,1,1,0,1,1,1,1]	9	[1,1,0,0,0,1,1,1]	8
[0,0,0,1,1,0,1,0]	1	[0,1,1,1,0,0,0,0]	5	[1,1,0,0,1,0,0,0]	16
[0,0,0,1,1,0,1,1]	1	[0,1,1,1,0,0,1,0]	2	[1,1,0,0,1,0,0,1]	13
[0,0,0,1,1,1,0,1]	1	[0,1,1,1,0,0,1,1]	1	[1,1,0,0,1,1,0,0]	1
[0,0,0,1,1,1,1,1]	5	[0,1,1,1,0,1,0,0]	6	[1,1,0,0,1,1,0,1]	16
[0,0,1,0,0,0,0,0]	245	[0,1,1,1,0,1,0,1]	2	[1,1,0,0,1,1,1,1]	2
[0,0,1,0,0,0,0,1]	14	[0,1,1,1,0,1,1,0]	10	[1,1,0,1,0,0,0,0]	6
[0,0,1,0,0,0,1,0]	13	[0,1,1,1,0,1,1,1]	9	[1,1,0,1,0,0,1,0]	7
[0,0,1,0,0,0,1,1]	3	[0,1,1,1,1,0,0,0]	1	[1,1,0,1,0,0,1,1]	1
[0,0,1,0,0,1,0,0]	105	[0,1,1,1,1,0,0,1]	1	[1,1,0,1,0,1,1,0]	2
[0,0,1,0,0,1,0,1]	30	[0,1,1,1,1,0,1,1]	2	[1,1,0,1,1,0,0,0]	1
...					

[0,0,1,0,0,1,1,0]	16	[0,1,1,1,1,1,0,0]	3	[1,1,0,1,1,0,0,1]	3
[0,0,1,0,0,1,1,1]	14	[0,1,1,1,1,1,0,1]	6	[1,1,0,1,1,0,1,1]	1
[0,0,1,0,1,0,0,0]	11	[0,1,1,1,1,1,1,0]	3	[1,1,0,1,1,1,0,1]	1
[0,0,1,0,1,0,0,1]	16	[0,1,1,1,1,1,1,1]	23	[1,1,0,1,1,1,1,0]	1
[0,0,1,0,1,0,1,0]	1	[1,0,0,0,0,0,0,0]	404	[1,1,0,1,1,1,1,1]	4
[0,0,1,0,1,0,1,1]	2	[1,0,0,0,0,0,0,1]	11	[1,1,1,0,0,0,0,0]	45
[0,0,1,0,1,1,0,0]	11	[1,0,0,0,0,0,1,0]	11	[1,1,1,0,0,0,0,1]	8
[0,0,1,0,1,1,0,1]	31	[1,0,0,0,0,1,0,0]	36	[1,1,1,0,0,0,1,0]	4
[0,0,1,0,1,1,1,0]	1	[1,0,0,0,0,1,0,1]	15	[1,1,1,0,0,0,1,1]	2
[0,0,1,0,1,1,1,1]	16	[1,0,0,0,0,1,1,0]	6	[1,1,1,0,0,1,0,0]	78
[0,0,1,1,0,0,0,0]	23	[1,0,0,0,0,1,1,1]	6	[1,1,1,0,0,1,0,1]	63
[0,0,1,1,0,0,0,1]	1	[1,0,0,0,1,0,0,0]	9	[1,1,1,0,0,1,1,0]	12
[0,0,1,1,0,0,1,0]	21	[1,0,0,0,1,0,0,1]	14	[1,1,1,0,0,1,1,1]	17
[0,0,1,1,0,0,1,1]	6	[1,0,0,0,1,0,1,0]	1	[1,1,1,0,1,0,0,0]	9
[0,0,1,1,0,1,0,0]	5	[1,0,0,0,1,1,0,0]	2	[1,1,1,0,1,0,0,1]	16
[0,0,1,1,0,1,0,1]	4	[1,0,0,0,1,1,0,1]	10	[1,1,1,0,1,0,1,0]	1
[0,0,1,1,0,1,1,0]	25	[1,0,0,0,1,1,1,1]	1	[1,1,1,0,1,0,1,1]	1
[0,0,1,1,0,1,1,1]	21	[1,0,0,1,0,0,0,0]	6	[1,1,1,0,1,1,0,0]	12
[0,0,1,1,1,0,0,0]	2	[1,0,0,1,0,0,1,0]	10	[1,1,1,0,1,1,0,1]	93
[0,0,1,1,1,0,0,1]	3	[1,0,0,1,0,1,0,0]	1	[1,1,1,0,1,1,1,0]	3
[0,0,1,1,1,0,1,0]	3	[1,0,0,1,0,1,1,0]	1	[1,1,1,0,1,1,1,1]	26
[0,0,1,1,1,0,1,1]	12	[1,0,0,1,0,1,1,1]	5	[1,1,1,1,0,0,0,0]	1
[0,0,1,1,1,1,0,0]	1	[1,0,0,1,1,0,0,0]	2	[1,1,1,1,0,0,1,0]	9
[0,0,1,1,1,1,0,1]	8	[1,0,0,1,1,0,1,1]	3	[1,1,1,1,0,0,1,1]	3
[0,0,1,1,1,1,1,0]	10	[1,0,0,1,1,1,0,1]	2	[1,1,1,1,0,1,0,0]	8
[0,0,1,1,1,1,1,1]	36	[1,0,1,0,0,0,0,0]	94	[1,1,1,1,0,1,0,1]	7
[0,1,0,0,0,0,0,0]	222	[1,0,1,0,0,0,0,1]	10	[1,1,1,1,0,1,1,0]	8
[0,1,0,0,0,0,0,1]	8	[1,0,1,0,0,0,1,0]	4	[1,1,1,1,0,1,1,1]	8
[0,1,0,0,0,0,1,0]	6	[1,0,1,0,0,0,1,1]	2	[1,1,1,1,1,0,0,0]	4
[0,1,0,0,0,0,1,1]	2	[1,0,1,0,0,1,0,0]	82	[1,1,1,1,1,0,0,1]	4
[0,1,0,0,0,1,0,0]	40	[1,0,1,0,0,1,0,1]	37	[1,1,1,1,1,0,1,0]	2
[0,1,0,0,0,1,0,1]	12	[1,0,1,0,0,1,1,0]	10	[1,1,1,1,1,0,1,1]	3
[0,1,0,0,0,1,1,0]	2	[1,0,1,0,0,1,1,1]	16	[1,1,1,1,1,1,0,0]	7
[0,1,0,0,0,1,1,1]	3	[1,0,1,0,1,0,0,0]	11	[1,1,1,1,1,1,0,1]	21
[0,1,0,0,1,0,0,0]	6	[1,0,1,0,1,0,0,1]	12	[1,1,1,1,1,1,1,0]	9
[0,1,0,0,1,0,0,1]	8	[1,0,1,0,1,0,1,1]	1	[1,1,1,1,1,1,1,1]	126
[0,1,0,0,1,0,1,1]	1	[1,0,1,0,1,1,0,0]	6		