Improved EM algorithm for MPT model analysis

Yuan You · Xiangen Hu · Huan Qi

Published online: 3 June 2011
© Psychonomic Society, Inc. 2011

Abstract Multinomial processing tree (MPT) models are in wide use as measurement models for analyzing categorical data in cognitive experiments. The approach involves estimating parameters and conducting hypothesis tests involving parameters that are arrayed in a tree structure designed to represent latent cognitive processes. The standard inference algorithm for these models is based on the well-known expectation maximization (EM) algorithm. On the basis of the original use of the EM algorithm for MPT models, this article presents an approach that accelerates the convergence speed of the algorithm by (1) adjusting suitable initial positions for certain parameters to reduce required iterative times and (2) using a series of operations between/among a set of matrices that are specific to the original model structure and information to reduce the time required for a single iteration. As compared with traditional algorithms, the simulation results show that the proposed algorithm has superior efficiency in interpreted languages and also has better algorithm readability and structure flexibility.

Keywords MPT models · EM algorithm · Convergence speed · Initial value · Matrix operation

Y. You (✉) · H. Qi
Institute of Systems Engineering,
Huazhong University of Science and Technology,
Wuhan, China
e-mail: yyster514@gmail.com

X. Hu
Department of Psychology, University of Memphis,
Memphis, USA

Introduction

Multinomial processing tree (MPT) models (Batchelder & Riefer, 1986; Riefer & Batchelder, 1988) have been widely used by researchers in cognitive psychology for modeling and analysis (Batchelder & Riefer, 1999; Erdfelder et al., 2009). The MPT modeling method can be described as using a special class of parametric multinomial models with links, branches, and parameters displayed in a set of tree-like structures to present every cognitive process occurring in the corresponding experiment according to their logical sequence consistent with a cognitive theory. These functionally independent parameters vary in the interval [0,1] placed at each link, and they can be viewed as the probabilities of cognitive processes in different stages. In addition to providing a quantitative analysis of parameter estimations, MPT models can also be used to test relationships between/among parameters including equality constraints, as well as order constraints (Knapp & Batchelder, 2004), which enables researchers to reveal latent interactions among the cognitive processes (Bishara & Payne, 2009; Honey et al., 2005; Hu, 2001; Woodward, Menon, Hu, & Keefe, 2006).

Hu and Batchelder (1994) provided a unified mathematical form for binary MPT models (all parameters are in the form of θ or $1 - \theta$) that can be summarized as follows: (1) Branch probability is a multiplication of link probability, and (2) category probability is a sum of branch probabilities. For a binary MPT model with J observable response categories $C_1, C_2, \dots, C_J$, let $\theta = (\theta_1, \dots, \theta_s) \in \Omega$ denote the model parameters, where Ω is the parameter space and $B_{j1}, \dots, B_{jI_j}$

denotes the collection of branches leading to category C_j . The probability of a category can be described as

$$\begin{aligned} Pr(C_j; \theta) &= \sum_{k=1}^{I_j} Pr(B_{jk}; \theta) \\ &= \sum_{k=1}^{I_j} c_{jk} \prod_{s=1}^S \theta_s^{a_{jks}} (1 - \theta_s)^{b_{jks}}, \end{aligned} \quad (1)$$

where c_{jk} is the product of the numerical values on the links of branch k leading to category j or set to unity if there are no numerical links. Parameter a_{jks} and b_{jks} , respectively, are the numbers of links on branch k with θ_s and $(1 - \theta_s)$, which can be interpreted as the probability of a successful/failed execution of a corresponding cognitive latent stage. Because they completely specify the structure of an MPT model, a_{jks} and b_{jks} are called *structural* parameters. Because $Pr(C_j; \theta)$ are probabilities, $j = 1, \dots, J$, it is necessary that $\sum_{j=1}^J Pr(C_j) = 1$. Hu and Batchelder (1994) also provided a general framework of hypothesis testing among model parameters. Different hypotheses are in the form of equality among model parameters. The equality constraints among model parameters will result only in the change of structure parameters a_{jks} and b_{jks} . Suppose that the category counts n_j for each corresponding category C_j are obtained from independently and identically distributed replicates. Then the likelihood function of any MPT model is given by

$$L(\theta; \langle n_1, n_2, \dots, n_J \rangle) = N! \prod_{j=1}^J \frac{[Pr(C_j; \theta)]^{n_j}}{n_j!}, \quad (2)$$

where $N = \sum_{j=1}^J n_j$.

The expectationmaximization (EM) algorithm (Dempster, Laird, & Rubin, 1977) is one of the earliest algorithms used for parameter estimation of MPT models (Hu & Batchelder, 1994; Hu & Phillips, 1999). Because of the complexity of the likelihood function of MPT models, it is usually impossible to solve the maximization problem explicitly. The EM algorithm provides an iterative way for numerical estimates of the parameters; however, efficiency can be a problem when the number of parameters is large. We have observed that it often requires over a thousand iterations (E-steps and M-steps) to achieve the necessary accuracy (Hu & Phillips, 1999). It may take a long time to analyze joint MPT models (with a large number of parameters in each MPT model). The efficiency issue also makes it difficult to implement the EM algorithm in popular statistical packages such as MATLAB or R (Tang, Mei, & You, 2008).

To provide a solution to the issue mentioned above, in this article we present a matrix-based EM algorithm with an initial value adjustment that is specific to the characteristics of the MPT model structure. A set of matrices has been employed to present information in MPT models, and the

original EM algorithm function can now be written as a series of matrix calculations. As compared with the original algorithm, it improves convergence efficiency by reducing both the required iterative times and the time required for a single iteration in interpreted languages (such as MATLAB and R). It also has better structure flexibility, which means that the structures of MPT models are more flexible for modification when parameter constraints or model components have been changed.

This article is organized as follows. The MPT modeling method and its original estimation algorithm will be introduced in the second section. In particular, we represent the detailed calculation processes of the E-step and M-step for performing EM iterations to estimate MPT models. Our new strategy is presented in the third section. First, we show how a parameter with the lowest convergence speed could influence the efficiency of the EM algorithm and then present an initial value adjustment algorithm to accelerate it, which can be considered as an extension of the M-step. Then we present the matrix notation of the classic EM algorithm to improve its calculation speed in interpreted languages. In the fourth section, we demonstrate our methods within the source-monitoring framework of source, followed by an efficiency comparison and a discussion.

Analysis of the original algorithm

To estimate the parameter vector θ on the basis of the observed category frequencies, the EM algorithm for MPT models treats the branch frequencies as missing data (Hu & Batchelder, 1994). To construct an EM algorithm for an MPT model, the complete data,

$$\mathbf{D} = \left\langle \langle m_{jk} \rangle_{k=1}^{I_j} \right\rangle_{j=1}^J, \quad (3)$$

are defined, where m_{jk} denotes the branch count in B_{jk} . The likelihood function given by Eq. 2 can be rewritten as

$$L(\theta; \mathbf{D}) = N! \prod_{j=1}^J \prod_{k=1}^{I_j} \frac{[Pr(B_{jk}; \theta)]^{m_{jk}}}{m_{jk}!}, \quad (4)$$

where $N = \sum_{j=1}^J \sum_{k=1}^{I_j} m_{jk}$.

In accord with Eqs. 1 and 4, suppose that $\theta^{(q)}$ denotes the current value of θ after q cycles of iteration. The expression of the EM algorithm for solving MPT models takes the following form:

E-step: For any value of $\theta^{(q)}$, the expected branch frequencies in this iteration are given by

$$m_{jk}^{(q)} = \frac{Pr(B_{jk}; \theta^{(q)})}{Pr(C_j; \theta^{(q)})} n_j. \quad (5)$$

M-step: Given the expected values of the missing data computed in the E-step, determine $\theta^{(q+1)} = (\theta_1^{(q+1)}, \dots, \theta_s^{(q+1)})$ as the solution of

$$\theta_s^{(q+1)} = \frac{\sum_{j=1}^J \sum_{k=1}^{I_j} a_{jks} m_{jk}^{(q)}}{\sum_{j=1}^J \sum_{k=1}^{I_j} (a_{jks} + b_{jks}) m_{jk}^{(q)}}. \quad (6)$$

The EMalgorithm iterates between the E-step and M-step until $L(\theta^{(q+1)}; D) - L(\theta^{(q)}; D)$ is lower than the selected accuracy, for it has been proven that a cycle of the EMalgorithm does not decrease the likelihood function (Dempster et al., 1977). Note that Eqs. 5 and 6 are also able to be extended for estimating multilink MPT models (Hu & Batchelder, 2007; Hu & Phillips, 1999; Tang et al., 2008).

Although the original EMalgorithm given by Eqs. 5 and 6 is able to estimate parameters in MPT models correctly, there are still some flexibility/efficiency challenges. First, the algorithm's flexibility is reduced because the MPT model structures in the original algorithm are described mainly by structure parameters, a_{jks} and b_{jks} , which are highly dependent on the equality constraints on parameters and need to be rewritten every time the hypothesis changes. Second, considering that the parameter estimations for MPT models usually require hundreds of EM iterations, a large amount of computing resources are consumed. These two issues have been resolved in the compiled computer program in Hu and Phillips (1999). However, these disadvantages will become very obvious when Eqs. 5 and 6 are used to perform parameter estimation in some interpreted languages such as MATLAB and R, especially in MPT models with a large number of parameters. Therefore, it is necessary to improve the convergence speed of the existing EMalgorithm for building a flexible and efficient light-weight package/plugin for the users of MPT models.

Proposed algorithm

As was discussed above, we will improve the convergence speed by (1) reducing the required iterative times with an initial value adjustment algorithm and (2) reducing the time required for each single iteration in interpreted languages with matrix operations for the E-step and M-step.

Algorithm for adjusting initial values

One of the causes for the reduced efficiency of an iterative optimization algorithm, such as the EMalgorithm, is the slow convergence for certain parameters starting with special initial points (Soto, Zavala, & Araneda, 2007). We have observed this in the case of MPT models. Figure 1

Fig. 1 Two typical influences on the convergence speed of different initial positions in MPT models. **a** Convergence curve of 100 iterations starting from 0.1. **b** Convergence curve of 100 iterations starting from 0

depicts the influence on the convergence speed of different initial points when parameter estimations are performed by the EMalgorithm. Some independent parameters in MPT models, such as θ_1 , θ_2 , θ_3 , and θ_5 , are able to reach their convergence points rapidly from any starting point; however, the convergence curve of another parameter, θ_4 , is very sensitive to its initial value and will slow down the global convergence speed greatly when a relatively distant initial value is selected. Therefore, an appropriate initial position for a “slow” parameter (the last parameter that is still varying significantly), which is close to its convergence point, will reduce the required iterative times significantly.

Next, we will focus on the behavior of the slowest converging parameter. While the other parameters are near constants over iterations, the likelihood given by Eq. 4 is thought of as a function of this “slow” parameter.

Considering that there is a unique local maximum of likelihood that guarantees a unique convergence point for each latent parameter estimated by the EM algorithm (Hu & Batchelder, 1994) and every iteration will not decrease the likelihood value, the relationship between the likelihood and the “slow” parameter is demonstrated in Fig. 2.

There are two types of convergence curves of a “slow” parameter. One is the inflectional convergence shown in Fig. 2a; the other is the monotonic convergence shown in Fig. 2b. It is easy to see that the monotonic convergence will have a great effect on the global convergence speed, for its convergence curve can be so smooth that a large number of iterations are needed to approach its convergence point. Considering that, in this case, the last parameter left will approach its convergence point monotonously, it is possible to accelerate the approximating process for a “slow” parameter with a monotonic convergence curve.

Fig. 2 Relationship between likelihood and slowest parameter value. **a** Inflectional convergence. **b** Monotonic convergence

Given the model parameters $\theta = (\theta_1, \dots, \theta_S)$, suppose that the variation of all parameters, except θ_τ falls below a threshold ε , which is a small number in $[0, 1]$ in the last $(m-1)$ iterations. It is easy to see that the convergence point for the only parameter left is located in $[0, \theta_\tau^{(m)}]$ (if $\theta_\tau^{(m)} < \theta_\tau^{(m-1)}$) or in $[\theta_\tau^{(m)}, 1]$ (if $\theta_\tau^{(m)} > \theta_\tau^{(m-1)}$). For parameter θ_τ , we will reset the parameter value in the iteration on the basis of the following rule:

- In the case of $[0, \theta_\tau^{(m)}]$, set the initial value of θ_τ to the middle of the interval, which means that $\theta_\tau^{(m)} = \theta_\tau^{(m)}/2$, and then resume the EM iterations.
- At the next cycle, if the difference between $\theta_\tau^{(m+1)}$ and $\theta_\tau^{(m)}$ is still larger than ε , the interval will be updated to $[0, \theta_\tau^{(m+1)}]$ if $\theta_\tau^{(m+1)} < \theta_\tau^{(m)}$ or, otherwise, $[\theta_\tau^{(m+1)}, \theta_\tau^{(m)}]$.

Note that these steps can be executed repeatedly to obtain a narrower interval before the interval range falls below ε , demonstrated in Fig. 3, where ζ_τ is the convergence point of θ_τ .

As is shown in Fig. 3, an original smooth convergence curve has been replaced by an inflectional asymptote; therefore, the required iterative times can be reduced significantly. It is easy to see that the basic idea of this algorithm is very similar to the binary search (or half-interval search) algorithm, which is a method for finding a particular value in a sorted array (Cormen, Leiserson, Rivest, & Stein, 2009). A binary search finds the median, makes a comparison to determine whether the desired value comes before or after it, and then searches the remaining half in the same manner. Thus, it will successively narrow the search interval by half through a series of comparisons. This process of successive approximation is exactly the same as that in the initial algorithm. However, unlike the binary search method, in which we know the exact value we are searching for, we do not have any information about the position of the ending point. As a result, the calculations of the initial algorithm will be performed in the form of a series of intervals.

Since the length of the interval after t cycles of the initial algorithm is closely approximated by $(\frac{1}{2})^t$, which means that no more than $-\log_2 \varepsilon$ EM iterations are needed to compress the interval, t will not be a large number. For example, theoretically, fewer than 17 iterations are needed to give a slowly monotonically converging parameter an initial value that is within 10^{-5} to its convergence point. Note that high accuracy of ε is necessary to avoid the situation where some other parameters might shake loose after several iterations, and using this approaching algorithm at intervals is also helpful for reducing the influence on the other parameters every time a new initial value is given.

Now we will focus on the limitations of this initial algorithm. The purpose of this algorithm is to detect a monotonic “slow” parameter and accelerate its convergence

Fig. 3 Convergence curve for initial value adjustment algorithm

speed by reassigning the values (on the basis of the rules outlined above) in each iteration of the EM. Therefore, the series of “adjusted” values are getting closer and closer to its convergence point. In the case where there is no “slow” parameter existing in an MPT model, as we have observed, this algorithm will still capture the slowest parameter and give it a relatively distant starting point, so the global convergence speed will be even slower than that of the original EM iterations. The same thing will happen in the rare case in which there is more than one “slow” parameter with nearly the same convergence speed, while other parameters can quickly reach their convergence points. And in the case of a “slow” parameter with an inflectional convergence curve, shown in Fig. 2a, this algorithm will aggravate its oscillation convergence process. However, this negative influence on the convergence speed will not become obvious, for only a limited number of additional EM iterations will be involved. Considering that the convergence efficiency might improve, this initial position adjustment algorithm is still acceptable.

EMalgorithm in matrix notation

As was discussed above, there are two main operations in the original EMalgorithm. First, recalculate the probability of each branch leading to the corresponding category $Pr(B_{jk})$ according to the current value of category probability $Pr(C_j)$ and then distribute the observed category frequency to branches according to their expected probabilities. This procedure is the expectation (E-step), given by Eq. 5. Second, redistribute the expected branch frequencies acquired above to each parameter leading to a corresponding branch and then normalize them with their complements to get the updated values of parameters. This procedure is the maximization (M-step), given by Eq. 6. Since the matrix operation can be considered a process of

calculation and distribution based on different relationships, it is possible for us to replace the original EMalgorithm formulas with a series of matrix operations once all the information has been presented in matrices.

E-step Note that the branch probabilities in MPT models are dependent on the power of the parameters, such as a_{jks} and b_{jks} in Eq. 1. First, define an original MPT model as an MPT model in which a unique parameter is assigned to each link, which means no parameters are repeated in the tree. Given an original MPT model with P parameters that construct the complete parameter vector $\phi = (\phi_1, \dots, \phi_P)$, let B be the number of branches in an MPT model structure, and it is easy to see that $B = \sum_j I_j$. Then the power matrix can be written as

$$\mathbf{P} = (p_{ki})_{B \times P}, \quad (7)$$

where p_{ki} is the number of occurrences of parameter ϕ_i appearing in branch k . Please note that the matrix $\mathbf{P}$ has these two properties:

1. P equals the number of links.
2. Entries of $\mathbf{P} = (p_{ki})_{B \times P}$ are either 0 or 1.

To understand the meaning of the matrix $\mathbf{P}$, one can view it as an “occurrence” matrix: which parameter (a parameter or its complement) appears in which branch. Such an intuitive way of constructing the power matrix brings several advantages.

- When multiple trees are presented in an MPT model, the matrix $\mathbf{P}$ is a block diagonal, with each block corresponding to a subtree in its MPT model structure after the parameters are properly reordered. This feature makes the structures of an MPT model more clear, and the modification of the model structures (add/remove branches or subtrees) can be easily done by modifying corresponding blocks.
- Equality constraints are first added to the original model to identify it and to convert it to the model of interest in which the same parameter is assigned to multiple links. Setting parameters equal to each other is functionally constructing another “occurrence” matrix. With the matrix $\mathbf{P}$, the process is simplified dramatically: Add columns that correspond to the parameters being set equal. In the classic algorithm, these constraints were integrated in the original model structure by modifying corresponding powers of parameters directly, which means that all the values of each power need to be rewritten while the constraints are changed.

The matrix notation makes it possible to reflect the influence on constraints independently by building a series of matrices. Let $\lambda = (\omega_1, \dots, \omega_H)$ be the reduced parameter

vector according to equality constraints. The relationship matrix can be built directly by

$$\mathbf{R} = (\gamma_{ih})_{P \times H}, \quad (8)$$

where $\gamma_{ih} = 1$ means that the parameter φ_i is set to ω_h under the current constraints; otherwise, $\gamma_{ih} = 0$. Then the reduced power matrix $\mathbf{U} = (d_{kh})_{B \times H}$ can be represented as

$$\mathbf{U} = (\mathbf{P} \cdot \mathbf{R})_{B \times H}. \quad (9)$$

Intuitively, $\mathbf{U}$ is the result of adding certain columns (specified by $\mathbf{R}$), and it is an “occurrence” matrix corresponding to the equality constraints. In the case of a binary MPT model, each row of $\mathbf{U}$ can be considered to be the distribution of the structure parameter a_{jks} and b_{jks} for branch k . The differences between $\mathbf{U}$ and $\mathbf{P}$ are the following:

- The number of columns in $\mathbf{P}$ is the total number of links in the MPT models. The number of columns of $\mathbf{U}$ is the total number of *distinct* links in the MPT model.
- The entries of $\mathbf{P}$ can assume only the value 0 or 1. The entries of $\mathbf{U}$ can be an integer greater than 1. This can happen only when the same parameter occurs in one branch more than once.

Next, we consider the branch probability as a multiplication of the current link probabilities. The probability that each branch after q cycles of iteration can be represented by the branch probability vector is given by

$$\mathbf{b}^{(q)} = \left\langle b_k^{(q)} \right\rangle_{k=1}^B, \quad (10)$$

where $b_k^{(q)} = \prod_{h=1}^H [\omega_h^{(q)}]^{d_{kh}}$ and $\omega_h^{(q)}$ denotes the current estimation value of ω_h in $\lambda^{(q)}$.

Then we will calculate the probabilities of all the categories according to the branch probabilities obtained above. First, a matrix is built to describe the relationship between the observed categories and branches, given by

$$\mathbf{W} = (w_{jk})_{J \times B}, \quad (11)$$

where $w_{jk} = 1$ means that branch k leads to category j . Therefore, the expected probabilities of branches will be contained in this matrix, given by

$$\mathbf{V}^{(q)} = \left[\text{diag}(\mathbf{W} \cdot \mathbf{b}^{(q)}) \right]^{-1} \mathbf{W} \cdot \mathbf{B}^{(q)}, \quad (12)$$

where $\mathbf{B}^{(q)} = \text{diag}(\mathbf{b}^{(q)})$.

Intuitively, $\mathbf{V}^{(q)}$ is the result of row normalization of the product of $\mathbf{W}$ and $\mathbf{B}^{(q)}$, in which the following holds:

- Each cell can be considered as a branch probability.
- The sum of each row will be the categorical probability for a corresponding category.

Thus, there will be only one nonzero element, the value of the corresponding branch probability in each column of $\mathbf{V}^{(q)}$.

In order to distribute the observed category counts to branches according to their expected probabilities, first, the frequency row vector $\mathbf{f}$ can be built from the experimental data directly by

$$\mathbf{f}' = \langle n_j \rangle_{j=1}^J. \quad (13)$$

It is easy to see that each cell in the row vector

$$\mathbf{e}^{(q)} = \mathbf{f} \cdot \mathbf{V}^{(q)} \quad (14)$$

can be considered as the expected frequency of the corresponding branch after q cycles of the algorithm, given by $m_{jk}^{(q)}$ in Eq. 5. Thus, we have completed all the computational procedures of the E-step.

M-step First, it is easy to see that the distribution of the expected branch frequencies acquired for each parameter can be represented by a row vector

$$\mathbf{h}^{(q)} = \mathbf{e}^{(q)} \cdot \mathbf{U}, \quad (15)$$

where each cell value is the estimated frequency, just like the numerator of Eq. 6, for a corresponding parameter in $\lambda^{(q)}$.

Next, we will take into consideration the parameter normalizing conditions, which means that the probability summation of all the parameters in the same group (those that share the same parent parameter) in the model structure should be 1. Given a binary MPT model in which all the parameters will appear in pairs, the parameter group information is so clear that the normalization can be very easy to perform by rewriting the reduced parameter vector λ in the form of

$$(\lambda_1, \lambda_2) = (\mu_1, \dots, \mu_L, 1 - \mu_1, \dots, 1 - \mu_L), \quad (16)$$

where $H = 2L$. Thus, by separating its columns, $\mathbf{U}$ can be divided into two $B \times L$ matrices, $\mathbf{U}_1$ for $\mu_1, \dots, \mu_L$ and $\mathbf{U}_2$ for their complements. Therefore, the updated parameter values will be obtained directly by performing the normalization

$$\lambda_1^{(q+1)} = \mathbf{h}_1^{(q)} \cdot \left[\text{diag}(\mathbf{h}_1^{(q)} + \mathbf{h}_2^{(q)}) \right]^{-1}, \quad (17)$$

$$\lambda_2^{(q+1)} = 1 - \lambda_1^{(q+1)}, \quad (18)$$

where $\mathbf{h}_1^{(q)} = \mathbf{e}^{(q)} \cdot \mathbf{U}_1$ and $\mathbf{h}_2^{(q)} = \mathbf{e}^{(q)} \cdot \mathbf{U}_2$ are the expected category frequencies for parameters in λ_1 and λ_2 , respectively.

The steps described here provide a simple and efficient algorithm. However, it also brings two problems:

1. As was discussed above, because $\mathbf{U}$ is the result of adding certain columns of $\mathbf{P}$, in order to make sure the parameter order in λ is in the form of Eq. 16, the parameter order in the complete parameter set ϕ will also be changed, which means matrix $\mathbf{P}$ cannot be a block matrix.
2. In the case of a multilink MPT model (MMPT) where λ cannot be divided into several subsets with equal size, Eqs. 17 and 18 cannot be performed directly.

To address these problems, additional information about parameter groups will be needed. Although this information has been already contained in $\mathbf{P}$ and can be obtained by parsing it, it is much easier to use a symmetric matrix,

$$\mathbf{G} = (\delta_{iv})_{P \times G}, \quad (19)$$

where G is the count of parameter groups and $\delta_{iv} = 1$ means that the parameters ϕ_i and ϕ_v are in the same group (otherwise, $\delta_{iv} = 0$) to represent this information separately for reducing the computational expense. The relationships between $\mathbf{P}$ and $\mathbf{G}$ are the following:

- As was discussed above, $\mathbf{P}$ can be viewed as an “occurrence” matrix. In fact, it is a part of the reachability matrix of the directed graph of an original MPT model’s structure by selecting certain rows corresponding to branches. $\mathbf{G}$ demonstrates the unilateral connection relationship between each parameter and its parent, and it can be considered an adjacency matrix (Cormen et al., 2009).
- Since the parameter connection relationship carried by the adjacency matrix is necessary information to build a reachability matrix, $\mathbf{G}$ has already been built in the construction of $\mathbf{P}$; therefore, importing $\mathbf{G}$ will not bring any additional computations.

It is easy to see that if any parameters in different parameter groups are set to be equal, the parameter groups will also be combined. This combination procedure can be described by a matrix $\mathbf{T} = (t_{hl})_{H \times I}$, given by

$$\mathbf{T} = (\text{unique}(\mathbf{G}' \cdot \mathbf{R})), \quad (20)$$

where *unique* is a common function for finding unique elements of a vector/matrix in both MATLAB and R and I is the count of independent parameter groups under the current constraints, while all the extra groups with exactly the same elements have been removed.

Similarly, let $\mathbf{H}^{(q)} = \text{diag}(\mathbf{h}^{(q)})$; then there will be only one nonzero element placed in each row of the result of column normalization of the product of $\mathbf{H}^{(q)}$ and $\mathbf{T}$, which is the estimation of the parameter at the $(q + 1)$ iteration cycle;

therefore, the updated parameter value of ω_h can be obtained simply by

$$\lambda^{(q+1)} = \mathbf{H}^{(q)} \cdot \mathbf{T} \cdot [\text{diag}(\mathbf{h}^{(q)} \cdot \mathbf{T})]^{-1} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{I \times 1}, \quad (21)$$

where $\lambda^{(q+1)} = \langle \omega_h^{(q+1)} \rangle_{h=1}^H$. Thus, we have completed the M-step as given by Eq. 6.

Thus, all the computational steps of the original algorithm have been given in the framework of matrices, and we can update the parameter vector $\lambda^{(q)} \rightarrow \lambda^{(q+1)}$ to continue the iteration process.

Next, we will extend the matrix algorithm to the power divergence family. Hu and Batchelder (1994) defined the power divergence family of statistics for every real m by

$$2NI^{(m)}\left(\left\langle \frac{n_j}{N} : \tilde{p} \right\rangle\right) = \frac{2}{m(m+1)} \times \sum_{j=1}^J n_j \left[\left(\frac{n_j}{N\tilde{p}_j} \right)^m - 1 \right], \quad (22)$$

where $\tilde{p}$ is a probability distribution over the categories and $N = \sum_{j=1}^J n_j$. With the different m values, different test statistics can be obtained. When $m = 0$, the G^2 statistic test is given by

$$\lim_{m \rightarrow 0} 2NI^{(m)}\left(\left\langle \frac{n_j}{N} : \tilde{p} \right\rangle\right) = 2 \sum_{j=1}^J n_j \log \left(\frac{n_j}{N\tilde{p}_j} \right) = G^2, \quad (23)$$

where each $\tilde{p}_j$ can be replaced by $Pr(C_j; \theta)$ in Eq. 1.

It is easy to convert the G^2 formula, which is useful for testing the equality constraints on the parameters in MPT models, into our matrix notation according to Eqs. 11 and 12. If the EM iterations end after n cycles, the estimated category probabilities can be obtained by

$$\begin{aligned} \langle \tilde{p}_j \rangle &= \langle Pr(C_j; \theta) \rangle_{j=1}^J = \frac{\mathbf{W}}{\sum \mathbf{b}^{(n)}} \cdot \langle Pr(B_{jk}; \theta) \rangle_{k=1}^{I_j} \\ &= \frac{\mathbf{W} \cdot \mathbf{b}^{(n)}}{\sum \mathbf{b}^{(n)}}, \end{aligned} \quad (24)$$

where $\sum \mathbf{b}^{(n)}$ equals the count of independent subtrees in MPT models; then the G^2 function becomes

$$2\mathbf{f} \left[\log \left(\mathbf{f} \left[\text{diag}(\rho^{(n)}) \right]^{-1} \right) \right] = G^2, \quad (25)$$

where

$$\rho^{(n)} = N \langle \tilde{p}_j \rangle = \frac{N}{\sum \mathbf{b}^{(n)}} \mathbf{W} \cdot \mathbf{b}^{(n)} \quad (26)$$

are the expected frequencies based on estimated category

probabilities. Then the extension of our methods to the power divergence family, given by

$$\psi_s^{(m)} = \frac{\sum_{j=1}^J \left(\frac{n_j}{Np_j}\right)^m \sum_{k=1}^{I_j} a_{jks} m_{jk}}{\sum_{j=1}^J \left(\frac{n_j}{Np_j}\right)^m \sum_{k=1}^{I_j} (a_{jks} + b_{jks}) m_{jk}}, \quad (27)$$

also can be defined by rewriting Eq. 15 in the M-step to

$$\mathbf{h}^{(q)} = \mathbf{f} \cdot \mathbf{r}^{(m)} \cdot \mathbf{V}^{(q)} \cdot \mathbf{U}, \quad (28)$$

where

$$\begin{aligned} \mathbf{r}^{(m)} &= \left(\left[\frac{n_j}{Np_j} \right]_{j,j}^m \right)_{J \times J} \\ &= \text{diag} \left[\left(\mathbf{f} \left[\text{diag}(\rho^{(q)}) \right]^{-1} \right)^m \right]. \end{aligned} \quad (29)$$

The asymptotic approximations to the variance–covariances matrix are able to be obtained by the observed Fisher information matrix $\mathbf{I}(\theta)$ defined by Hu and Batchelder (1994). Given a binary MPT model with the parameter vector defined by Eq. 16, if the iterations end after n cycles, it is easy to see that the structure parameters a_{jkl} and b_{jkl} for $\mu_l \in \lambda$ are the counts of $\mathbf{U}[k, l]$ and $\mathbf{U}[k, L + l]$, $l = 1, \dots, L$, respectively, which means that the basic factor of $\mathbf{I}(\theta)$ can be rewritten as

$$\frac{a_{jkl}}{\mu_l} - \frac{b_{jkl}}{1 - \mu_l} = \frac{\mathbf{U}[k, l]}{\lambda^{(n)}[l]} - \frac{\mathbf{U}[k, L + l]}{\lambda^{(n)}[L + l]}, \quad (30)$$

where $\lambda^{(n)}$ is the updated parameter values calculated by Eqs. 17 and 18. Considering that the other factors, $Pr(P_j; \theta)$ and $Pr(B_{jk}; \theta)$, have also been given in Eq. 24, it is easy to convert the Fisher information matrix to our matrix notation. Because the formula of $\mathbf{I}(\theta)$ is complex while the conversion is direct, the detailed expression is omitted here.

Note that although many different matrices/vectors have been defined in the matrix expression of the E-step and the M-step, respectively, only four independent matrices/vectors are used to describe all the information of a corresponding MPT model: $\mathbf{P}$, $\mathbf{W}$, $\mathbf{f}$, and $\mathbf{R}$. The first two matrices reflect the MPT model structure, and the frequency vector $\mathbf{f}$ is the record of the experimental observations. Only one matrix, $\mathbf{R}$, is used to describe the equality constraints on parameters. These separate representations make it possible for us to modify the existing computational process by simply replacing some of the corresponding matrices, instead of rewriting all the power values of each parameter, as the classic algorithm does, when the parameter hypothesis has been changed. Therefore, the matrix expression has better readability and flexibility.

Another main advantage of the matrix algorithm is that it offers much greater execution speed in interpreted lan-

guages such as MATLAB and R, where the matrix operations are much more efficient due to their code interpretation methods. As one part of the GPT-R library project¹(Tang et al., 2008), the matrix algorithm listed above has been implemented in R to replace the original EM functions for accelerating the running speed of a single EM iteration while performing parameter estimation of MPT models. The improvement of the convergence speed can be very prominent and will become more obvious with the rise in convergence accuracy, as will be demonstrated later in this article.

Integration of two algorithms

As was discussed above, the initial value adjustment algorithm is essentially a kind of effective accelerating algorithm to speed up the convergence of the EM algorithm, which can be presented in the form of matrix expressions. Thus, it is easy for us to combine these two algorithms by monitoring the difference in distance between $\omega_h^{(q+1)}$ and $\omega_h^{(q)}$, obtained by Eq. 21, after $(q + 1)$ cycles of iteration, interrupting the convergence processes of the slowest parameter group, setting their initial values according to the adjustment algorithm and the parameter normalizing conditions, then resuming the EM iterations, starting from Eq. 10. Note that the initial value adjustment algorithm will be applied each time the M-step finishes. Indeed, it can be considered an extension of the classic M-step, which is able to be combined with the M-step as the “adjusted M-step”; therefore, the calculation steps of the combined algorithm can be considered iterations between “matrix E-step” and “adjusted M-step.” As a result, we can improve the convergence speed of MPT model estimation for cognitive analysis more efficiently by reducing the required iterative times, as well as increasing the convergence speed for a single iteration simultaneously.

Simulations

To illustrate the performance of the proposed algorithms, an example will now be considered. MPT models have been widely used for analyzing experimental data in cognitive psychology, and one of their most typical and famous applications is source-monitoring modeling. The source-monitoring paradigm (Johnson & Raye, 1981) has been used in cognitive psychology to explore human memory processes. The resulting data of a source-monitoring experiment with two original sources and an interference source (new items) will be arranged in a series of 3×3

¹ Available at <http://sites.google.com/site/gptnr/>.

Fig. 4 Tree representation of the source-monitoring model. θ_1 and θ_2 are the detections of source A and source B, respectively, at the stimulus detection stage that happens first. Then θ_3 and θ_4 are the discriminations of source A and source B, respectively, in the following source detection stage. And the other parameters are

considered the response biases placed in the two cognitive stages above. θ_5 is the response bias toward old items, θ_6 is the response bias toward source A when the item is detected as an old item, and θ_7 is the response bias toward source A when the item is not detected

Fig. 5 Impact of initial value adjustment algorithm on convergence procedures for source-monitoring estimations. **a** Three hundred iterations with fixed initial position. **b** One hundred iterations with initial value adjustment algorithm

contingency tables corresponding to different groups of subjects.

The MPT model structure for source monitoring is presented in Fig. 4 (Batchelder & Riefer, 1990, 1999), which contains three separated subtrees for the items from source A, source B, and new distractors (N) separately. This model assumes that there are a series of hypothetical cognitive processes occurring before subjects make their responses, including stimulus detection, source discrimination, and various response biases.

There are a total of seven parameters involved in the MPT model shown in Fig. 4 in the parameter space

$$\Omega_7 = \{\langle \theta_1, \theta_2, \dots, \theta_7 \rangle\}, \quad (31)$$

and each of these parameters is valuable for analyzing the cognitive ability of subjects at different stages in source monitoring. In order to make parameters identifiable, Batchelder and Riefer (1990) constructed a nested hierarchy for the MPT model depicted in Fig. 4, with seven additional submodels corresponding to different equality restrictions between parameters that are both of psychological and data-analytic significance. To perform the algorithm, first we choose submodel M_{6c} , which assumes that $\theta_6 = \theta_7$ is in Ω_7 ; therefore, there are six independent parameters. In addition, we select one of the frequency tables from Harvey's experiment (Harvey, 1985) as the observed response categories. The simulation results are presented in the following figures.

Figure 5 shows the influence of the initial value adjustment algorithm on the convergence efficiency with the threshold $\varepsilon = 10^{-4}$. The smooth convergence curve for θ_3 , shown in Fig. 5a, has been replaced by an inflectional asymptote, shown in Fig. 5b. As was discussed above,

Fig. 6 Convergence time required for comparison of source-monitoring estimations

Fig. 7 Convergence time required for comparison of large multigroup source-monitoring estimations under accuracy 10^{-10}

the initial value adjustment algorithm will accelerate a monotonic “slow” parameter so that it reaches the convergence point with fewer required iteration times.

Figure 6 presents the convergence times required for each listed algorithm to estimate latent parameters in the proposed MPT model in Fig. 4 under different levels of required accuracies. Although the matrix expression algorithm did not change the original computational procedure of the EM algorithm and required the same number of iterations to reach the convergence point, it still had a much faster computation speed. This confirms that the matrix operations in the interpreted languages have a higher operating speed that will reduce the time required for a single EM iteration. And as was expected, the combined algorithm, which merged the advantages of both the initial value adjustment algorithm and the matrix algorithm, obtained the fastest convergence speed. It is also easy to see that the initial value adjustment algorithm will have a relatively constant improvement on the base of the matrix algorithm, which will significantly accelerate the convergence speed. For example, it can save up to 91.4% of the convergence time under the precision of 10^{-20} , as compared with the original algorithm.

Note that the convergence time saved by the initial value adjustment algorithm is not very obvious when the matrix notation of the EM algorithm is already fast enough to have all the parameters reach their states of convergence. Moreover, in the cases we discussed, it will be even slower than the EM iterations. However, given a complex MPT model with very slow convergence speed, the advantage of the initial value adjustment algorithm will be much clearer.

Figure 7 presents the impact of the initial value adjustment algorithm on the growth of the MPT model's size in the framework of source monitoring, where submodel M_{5a} assumes $\theta_1 = \theta_2$, $\theta_3 = \theta_4$ in Ω_7 , and $\theta_3 = \theta_4$ in

M_{6b} . As was discussed above, the adjustment algorithm has only a modest effect on the convergence speed in submodel M_{6b} , and it even slightly increases the convergence time in M_{5a} . However, it might give a huge improvement of the convergence speed while estimating complex MPT models with very slow convergence speed caused by a monotonic “slow” parameter, such as the combined submodel $M_{5a} + M_{6b}$. Note that the convergence speed of a large complex MPT model is not always slow, for it depends highly on both the model structures described by the power matrix $\mathbf{P}$ and the observed data set described by the frequency vector $\mathbf{f}$. Since we cannot predict whether the convergence time for an MPT model is long before performing estimations, it is worthwhile to employ the initial value adjustment algorithm to eliminate the influence of the possible existence of a monotonic “slow” parameter, especially for very complex MPT models. In addition, considering that our algorithm has good structure flexibility due to its matrix-based algorithm expression, both the submodel conversions (constraint changes) and submodel combinations above are able to be performed very easily by replacing the relationship matrix $\mathbf{R}$ defined by Eq. 8 and merging their power matrices.

Conclusion

In this article, we have presented a matrix-based EM algorithm with an initial value adjustment to improve the parameter estimation of MPT models. It has two main advantages for improvement: (1) It will reduce the required iterative times when a monotonic “slow” parameter exists, and (2) it will accelerate the convergence speed of the EM algorithm for estimating MPT models in interpreted languages. The result of the simulation shows that this algorithm has significantly improved the convergence of the EM algorithm for estimating MPT models, especially for solving complex MPT models with a large number of parameters. In addition, the proposed algorithm has better flexibility for using a block diagonal power matrix $\mathbf{P}$ and an individual relationship matrix $\mathbf{R}$ to represent model structure and parameter constraints, respectively.

We believe that the algorithm presented in this article² can be used as a core algorithm for MPT model analysis. Especially, the improved algorithm can be implemented as library/packages in MATLAB and R.

Author Note The first and the third authors are supported by National Science Foundation of China (NSFC) Grant 60774036. The second author is supported by NSF Grant 0616657. The authors are grateful for comments and suggestions by the action editor and two anonymous referees.

² Basic code available at <http://sites.google.com/site/gptnr/instructions-of-how-to-use-gpt-r/files-located-here>.

References

- Batchelder, W. H., & Riefer, D. M. (1986). The statistical analysis of a model for storage and retrieval: Processes in human memory. *The British Journal of Mathematical and Statistical Psychology*, 39, 129–149.
- Batchelder, W. H., & Riefer, D. M. (1990). Multinomial processing models of source monitoring. *Psychological Review*, 97, 548–564.
- Batchelder, W. H., & Riefer, D. M. (1999). Theoretical and empirical review of multinomial process tree modeling. *Psychonomic Bulletin & Review*, 6, 57–86.
- Bishara, A. J., & Payne, B. K. (2009). Multinomial process tree models of control and automaticity in weapon misidentification. *Journal of Experimental Social Psychology*, 15, 524–534.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). Cambridge, MA: MIT Press.
- Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society B*, 39, 1–38.
- Erdfelder, E., Auer, T. S., Hilbig, B. E., Aßfalg, A., Moshagen, M., & Nadarevic, L. (2009). Multinomial processing tree models: A review of the literature. *Zeitschrift für Psychologie*, 217, 108–124.
- Harvey, P. D. (1985). Reality monitoring in mania and schizophrenia. *The Journal of Nervous and Mental Disease*, 173, 67–72.
- Honey, G. D., Honey, R., Sharar, S. R., Turner, D. C., Pomarol-Clotet, E., Kumaran, D., et al. (2005). Impairment of specific episodic memory processes by sub-psychotic doses of ketamine: The effects of levels of processing at encoding and of the subsequent retrieval task. *Psychopharmacology*, 181, 445–457.
- Hu, X. (2001). Extending general processing tree models to analyze reaction time experiments. *Journal of Mathematical Psychology*, 45, 603–634.
- Hu, X., & Batchelder, W. H. (1994). The statistical analysis of general processing tree models use the em algorithm. *Psychometrika*, 59, 21–47.
- Hu, X., & Batchelder, W. H. (2007, July). *Statistical closure of MPT models under parameter constraints*. Paper presented at 40th Annual Meeting of the Society for Mathematical Psychology, Cista Meta, CA.
- Hu, X., & Phillips, G. A. (1999). GPT.EXE: A powerful tool for the visualization and analysis of general processing tree models. *Behavior Research Methods, Instrumentation, & Computers*, 31, 220–234.
- Johnson, M. K., & Raye, C. L. (1981). Reality monitoring. *Psychological Review*, 88, 67–85.
- Knapp, B. R., & Batchelder, W. H. (2004). Representing parametric order constraints in multi-trial applications of multinomial processing tree models. *Journal of Mathematical Psychology*, 48, 215–229.
- Riefer, D. M., & Batchelder, W. H. (1988). Multinomial modeling and the measurement of cognitive processes. *Psychological Review*, 95, 318–339.
- Soto, A., Zavala, F., & Araneda, A. (2007). An accelerated algorithm for density estimation in large databases using gaussian mixtures. *Cybernetics and Systems*, 38, 123–139.
- Tang, Q., Mei, X., & You, Y. (2008, November). *GPT models implemented in R*. Paper presented at 38th Annual Meeting of the Society for Computers in Psychology (SCiP), Chicago.
- Woodward, T., Menon, M., Hu, X., & Keefe, R. S. E. (2006). Optimization of a multinomial model for investigating hallucinations and delusions with source monitoring. *Schizophrenia Research*, 85, 106–112.