

# Accuracy proof for algorithm and Ordinal tensor model simulation

Chanwoo Lee

## 1 Accuracy proof for algorithm

Our goal is to recover tensor  $\mathcal{A} = \mathcal{C} \times_1 M^{(1)} \times_2 M^{(2)} \times_3 M^{(3)}$  from a tensor  $\mathcal{D} = \mathcal{A} + \mathcal{E}$  with a noise whose element is drawn from  $N(0, \sigma^2)$  using following algorithms. To formulate the problem clear, we set dimension of  $\mathcal{A} \in R^{d_1 \times d_2 \times d_3}$  and  $\mathcal{C} \in R^{r_1 \times r_2 \times r_3}$

---

**Algorithm 1** Approx tensor SVD 1

---

```

1: procedure SVD( $\mathcal{A}$ )
2:   Step A: Approximate SVD of  $\mathcal{A}$ 
3:   for  $n \leftarrow 1 : N$  do
4:     Unfold  $\mathcal{A}$  as  $A_{(n)}$ 
5:     Generate an  $d_1 \cdots d_{n-1} d_{n+1} \cdots d_N \times (r_i)$  Gaussian test matrix  $\Omega^{(n)}$  from  $N(0, 1)$ 
6:     For  $\mathbf{Y}^{(n)} = \mathbf{A}_{(n)} \Omega^{(n)}$ 
7:       Construct a matrix  $\hat{M}^{(n)}$  whose columns form an orthonormal basis for the range
       of  $\mathbf{Y}^{(n)}$ 
8:     get  $\hat{\mathcal{C}} = \mathcal{A} \times_1 \hat{M}^{(1)} \times_2 \hat{M}^{(2)} \cdots \times_N \hat{M}^{(N)}$ 
9:   Step B: Get approximated SVD
10:  return  $(\hat{\mathcal{C}}, \hat{M}^{(1)}, \dots, \hat{M}^{(N)})$ 

```

---

I will show that under some good conditions, our estimation  $\hat{M}^{(i)}$  is converging to our parameter  $M^{(i)}$  with regards to principle angle which is defined by as follows

**Definition 1.** For nonzero subspaces  $\mathcal{R}, \mathcal{N} \subset \mathbb{R}^n$ , the minimal angle between  $\mathcal{R}$  and  $\mathcal{N}$  is defined to be the number  $0 \leq \theta \leq \pi/2$  that satisfies

$$\cos \theta = \max_{u \in \mathcal{R}, v \in \mathcal{N} \|u\|=\|v\|=1} v^t u.$$

Our main theorem is as follows

**Theorem 1.** Let  $\mathcal{A} = \mathcal{C} \times_1 M^{(1)} \times_2 M^{(2)} \times_3 M^{(3)}$  be a traget tensor where each  $M^{(i)} \in R^{d_i \times r_i}$  is orthonormal matrix for each  $i \in [3]$  and  $\mathcal{D} = \mathcal{A} + \mathcal{E}$  be a give tensor where noise elemens are drawn from  $N(0, \sigma^2)$ .

Suppose,  $\max_{k \in [r_i]} \|C_{(i)}^k\|_2 >> \sigma \sqrt{\max(d_i, \frac{d_1 d_2 d_3}{d_i}) \frac{d_1 d_2 d_3}{d_i}}$  as  $d_1, d_2, d_3 \rightarrow \infty$ , where  $C_{(i)}^k$  is  $k$ -th row of  $C_{(i)}$ .

If we implement Algorithm 1 with an input  $\mathcal{D}$ , we can get output  $(\hat{\mathcal{C}}, \hat{M}^{(1)}, \hat{M}^{(2)}, \hat{M}^{(3)})$  whose angle  $\cos \Theta(M^{(i)}, \hat{M}^{(i)}) \rightarrow 1$  in probability

Also add the convergence for hat  $\mathbf{A}$  vs.  $\mathbf{A}$ .

*Proof.* It suffices to show  $M^{(1)}$  case.

$$\begin{aligned} A_{(1)} &= M^{(1)}(\mathcal{C} \times_2 M^{(2)} \times_3 M^{(3)})_{(1)} \\ &= M^{(1)}C_{(1)}(M^{(3)} \otimes M^{(2)})^T \end{aligned}$$

Let's define  $B = (M^{(3)} \otimes M^{(2)})^T = \left( M_1^{(3)} \otimes M_1^{(2)}, M_1^{(3)} \otimes M_2^{(2)}, \dots, M_{r_3}^{(3)} \otimes M_{r_2}^{(2)} \right)$  where  $M_j^{(i)}$  is the  $j$ -th column of  $M^{(i)}$

Notice that  $B^T$  is again orthonormal matrix by orthonormality assumption on each  $M^{(i)}$ .

$$\begin{aligned} A_{(1)}\Omega &= M^{(1)}C_{(1)}(M^{(3)} \otimes M^{(2)})^T\Omega \\ &= M^{(1)}C_{(1)}B\Omega \quad \text{where } \Omega \in R^{d_2 d_3 \times r_1} \text{ whose elements from } i.i.d.N(0, 1) \\ &= M^{(1)}C_{(1)}(Z_1 \ Z_2, \ \dots, \ Z_{r_1}) \quad \text{where } Z_i^T = (\sum_{k=1}^{d_2 d_3} B_{1,k}\Omega_{k,i}, \ \dots, \ \sum_{k=1}^{d_2 d_3} B_{r_2 r_3, k}\Omega_{k,i}) \\ &= M^{(1)}C_{(1)}\mathbf{Z} \quad \text{where } \mathbf{Z} = (Z_1 \ Z_2, \ \dots, \ Z_{r_1}) \end{aligned} \tag{1}$$

I am going to use the fact that elements of  $\mathbf{Z} \in R^{r_2 r_3 \times r_1}$  are independent  $N(0, 1)$  later and I proved this in Lemma 1.

Our estimator  $\hat{M}^{(1)}$  for  $M^{(1)}$  can be calculated replacing  $A_{(1)}$  by  $A_{(1)} + E_{(1)}$

$$\begin{aligned} (A_{(1)} + E_{(1)})\Omega &= M^{(1)}C_{(1)}\mathbf{Z} + E_{(1)}\Omega \\ &= \hat{M}^{(1)}R \quad (\text{QR decomposition}) \end{aligned} \tag{2}$$

Note that  $Im(A_{(1)}\Omega) \subset Im(M^{(1)})$  and  $Im(A_{(1)}\Omega + E_{(1)}\Omega) = Im(\hat{M}^{(1)})$  Therefore,

$$\begin{aligned} \cos \Theta(M^{(1)}, \hat{M}^{(1)}) &= \max_{u \in Im(M^{(1)}), v \in Im(\hat{M}^{(1)})} \cos(u, v) \\ &\geq \max_{u \in Im(A_{(1)}\Omega), v \in Im((A_{(1)} + E_{(1)})\Omega)} \cos(u, v) \\ &= \max_{x \in R^{r_1}, y \in R^{r_1}, \|x\|_2 = \|y\|_2 = 1} \cos(A_{(1)}\Omega x, (A_{(1)} + E_{(1)})\Omega y) \end{aligned} \tag{3}$$

To show  $\cos(A_{(1)}\Omega x, (A_{(1)} + E_{(1)})\Omega y) \rightarrow 1$ , it suffices to show that  $\cot(A_{(1)}\Omega x, (A_{(1)} + E_{(1)})\Omega y) \rightarrow \infty$ . We can get inequality as follows

$$\cot(A_{(1)}\Omega x, (A_{(1)} + E_{(1)})\Omega y) \geq \frac{\|A_{(1)}\Omega x\|_2}{\|E_{(1)}\Omega y\|_2} \geq \frac{\max_{k \in [r_1]} \|C_{(1)}^k\|_2 |Z|}{\|E\|_2 \|\Omega y\|_2} \tag{4}$$

To get numerator part in equation (4),

$$\begin{aligned} \|A_{(1)}\Omega x\|_2 &\stackrel{(1)}{=} \|M^{(1)}C_{(1)}\mathbf{Z}x\|_2 = \|C_{(1)} \sum_{i=1}^{r_1} Z_i x_i\|_2 \quad \text{by orthonormality of } M^{(1)} \\ &= \|C_{(1)} (\tilde{z}_1, \ \tilde{z}_2, \ \dots, \ \tilde{z}_{r_2 r_3})^T\|_2 \quad \text{where } \tilde{z}_i = \text{i-th row of } \sum_{i=1}^{r_1} Z_i x_i \end{aligned} \tag{5}$$

By Lemma 1, all elements of  $\mathbf{Z}$  are *i.i.d.* $N(0, 1)$ . Therefore, elements of  $\tilde{\mathbf{Z}} = (\tilde{z}_1, \tilde{z}_2, \dots, \tilde{z}_{r_2 r_3})^T$  from *i.i.d.* $N(0, 1)$  by the linearity of normal distribution and  $\|x\|_2 = 1$ . Finally we can have following inequality from (5) with a notation  $C_{(1)}^i$  =*i*-th row of  $C_{(1)}$

$$\begin{aligned} \|A_{(1)}\Omega x\|_2 &= \|C_{(1)}\tilde{\mathbf{Z}}\|_2 = \|([C_{(1)}^1]^T \tilde{\mathbf{Z}}, [C_{(1)}^2]^T \tilde{\mathbf{Z}}, \dots, [C_{(1)}^{r_1}]^T \tilde{\mathbf{Z}})^T\|_2 \\ &\geq \max_{k \in [r_1]} \|C_{(1)}^k\|_2 |Z| \quad \text{where } Z \sim N(0, 1) \end{aligned} \quad (6)$$

Finally, we can make numerator part in (4). Also, you note that  $\|\Omega y\|_2^2 \sim \chi^2(d_2 d_3)$  because  $\|y\|_2 = 1$ . Based on above,  $\|\Omega y\|_2 \asymp (1 + o(1))\sqrt{d_2 d_3}$ . Furthermore, we have that  $\|E\| \asymp (2 + o(1))\sigma\sqrt{\max(d_1, d_2 d_3)}$  by Lemma 2. Finally, for any fixed  $L > 0$

$$\begin{aligned} P(\cot(A_{(1)}\Omega x, (A_{(1)} + E_{(1)})\Omega y) \geq L) &\geq P\left(\frac{\max_{k \in [r_1]} \|C_{(1)}^k\|_2 |Z|}{\|E\|_2 \|\Omega y\|_2} \geq L\right) \\ &\geq P(|Z| \geq \frac{2L\sigma\sqrt{d_2 d_3 \max(d_1, d_2 d_3)}}{\max_{k \in [r_1]} \|C_{(1)}^k\|_2}) \\ &\geq 1 - \frac{4\lambda}{\sqrt{2\pi}} \end{aligned} \quad (7)$$

where  $\lambda \stackrel{\text{def}}{=} \frac{L\sigma\sqrt{d_2 d_3 \max(d_1, d_2 d_3)}}{\max_{k \in [r_1]} \|C_{(1)}^k\|_2}$ . The last line of equation (7) used the fact that  $P(|N(0, 1)| \geq t) \geq 1 - 2t\phi(0)$  for any  $t \geq 0$ , where  $\phi(\cdot)$  is the pdf for the standard normal. Based on assumption  $\lambda \rightarrow 0$ , we conclude that  $\cot(A_{(1)}\Omega x, (A_{(1)} + E_{(1)}) \geq L$  with high probability. Therefore, we can get desired results sending  $L \rightarrow \infty$  by the equation (3).  $\square$

**Lemma 1.** *In the proof of the Theorem 1, elements of  $\mathbf{Z} = (Z_1, Z_2, \dots, Z_{r_1})$  is from i.i.d. $N(0, 1)$  where  $Z_i^T = (\sum_{k=1}^{d_2 d_3} B_{1,k} \Omega_{k,i}, \dots, \sum_{k=1}^{d_2 d_3} B_{r_2 r_3, k} \Omega_{k,i})^T = (z_{1,i}, \dots, z_{r_1 r_2, i})^T$  and  $B = (M^{(3)} \otimes M^{(2)})^T = (M_1^{(3)} \otimes M_1^{(2)}, M_1^{(3)} \otimes M_2^{(2)}, \dots, M_{r_3}^{(3)} \otimes M_{r_2}^{(2)})$*

*Proof.* It's easy to check that  $Z_i$  and  $Z_j$  are independent where  $i \neq j$  because all elements of  $Z_i$  are made of  $\Omega_i$  = *i*-th column of  $\Omega$ . Therefore, it suffices to show all elements of  $Z_1$  are independent and from  $N(0, 1)$ .

1.  $z_{1,1} \sim N(0, 1)$

Note  $z_{1,1} = \sum_{k=1}^{d_2 d_3} B_{1,k} \Omega_{k,1} = [B^1]^T \Omega_1$ .

Since  $\|B^1\| = 1$  and  $\Omega_1 \stackrel{i.i.d.}{\sim} N(0, 1)$ ,  $z_{1,1}$  is from  $N(0, 1)$

2.  $z_{1,1}, \dots, z_{r_2 r_3, 1}$  are independent.

Let's define a function  $(ind_1, ind_2) : N \rightarrow N \times N$  which satisfies  $B_i = M_{ind_1(i)}^{(3)} \otimes M_{ind_2(i)}^{(2)}$ .

To give a simple example,  $(ind_1(1), ind_2(1)) = (1, 1)$  because  $B_1 = M_1^{(3)} \otimes M_1^{(2)}$ .

For  $i \neq j$ ,

$$\begin{aligned} Cov(z_{i,1}, z_{j,1}) &= Cov\left(\sum_{k=1}^{d_2 d_3} B_{i,k} \Omega_{k,1}, \sum_{k=1}^{d_2 d_3} B_{j,k} \Omega_{k,1}\right) \\ &= (B_i^T)^T (B_j^T) \\ &= 0 \end{aligned}$$

Therefore, all elements of  $\mathbf{Z}$  is from  $i.i.d.N(0, 1)$  by 1,2  $\square$

**Lemma 2** (Spectral norm of Gaussian matrix). *Let  $E \in R^{m \times n}$  be a random matrix with i.i.d.  $N(0, 1)$  entries. Then, we have, with very high probability,*

$$\|E\|_{\sigma} \asymp (2 + o(1)) \sqrt{\max(m, n)}$$

We can apply Theorem 1 to the lower dimension case.

**Corollary 1.** *Consider a noisy rank 1 matrix model  $D = \lambda a \otimes b + E$ , where  $\lambda \in R_+$  is a scalar,  $a \in R^{d_1}, b \in R^{d_2}$  are unit-1 vectors, and  $E \in R^{d_1 \times d_2}$  is a Gaussian matrix with i.i.d. $N(0, \sigma^2)$  entires. Define a random projection*

$$\hat{a} = D\Omega, \text{ where } \Omega = (z_1, \dots, z_{d_2})^T \stackrel{i.i.d.}{\sim} N(0, 1)$$

*Suppose  $\lambda >> \sigma \sqrt{d_2 \max(d_1, d_2)}$  as  $d_1, d_2 \rightarrow \infty$ . Then,  $\cos \Theta(a, \hat{a}) \rightarrow 1$  in probability.*

*Proof.* Put  $C = \lambda$ ,  $M^{(1)} = a$ ,  $M^{(2)} = b$  and  $M^{(3)} = 1$  into Theorem 1. Then you can get the result of Corollary 1.  $\square$

## 2 Ordinal tensor model simulation

### 2.1 Algorithm construction

In this section, we describe the algorithm which can be used to solve above optimization problem. We utilized a formulation of tucker decomposition, and turn the optimization into a block wise convex problem. We will divide cases into 2, when we know bin boundary  $\alpha_1, \dots, \alpha_K$  and when we don't have any information about bin boundary.

#### 2.1.1 Known Bin Boundary

From previous data or experience in the past we may know bin boundary parameter  $\alpha$ . When we know this bin boundary, finding an estimator becomes optimization problem of

$$\begin{aligned} \mathcal{L}_Y(\Theta, \alpha) &= - \sum_{i_1, \dots, i_N} \left[ \sum_{l=1}^K \mathbb{1}(y_{i_1, \dots, i_N} = l) \log(\pi_l(\theta, \alpha)) \right] \\ \pi_l &= P(Y_{i_1, \dots, i_N} = l | \theta_{i_1, \dots, i_N}, \alpha) = \text{logit}(\alpha_l + \theta_{i_1, \dots, i_N}) - \text{logit}(\alpha_{l-1} + \theta_{i_1, \dots, i_N}) \text{ for } l < K \\ \pi_K &= P(Y_{i_1, \dots, i_N} = K | \theta_{i_1, \dots, i_N}, \alpha) = 1 - \text{logit}(\alpha_{K-1} + \theta_{i_1, \dots, i_N}) \text{ for } l = K \end{aligned}$$

where  $\Theta = \mathcal{C} \times_1 A_1 \cdots \times_N A_N$ . Our strategy for this optimization is updating each block by fixing other blocks. To give you an simple example, let's assume that our tensor data has mode 3 i.e.  $\Theta = \mathcal{C} \times_1 A_1 \times_2 A_2 \times_3 A_3$ .

First, let's update  $A_1$  fixing  $A_2, A_3$  and  $\mathcal{C}$ . If you do mode 1 metricize  $\Theta$  and vectorize it , you can express this as an linear operation of vectorized  $A_1$  as follows.

$$Vec(\Theta_{(1)}) = ((A_3 \otimes A_2) \mathcal{C}_{(1)}^T \otimes I_{d_1}) Vec(A_1)$$

This equation can be decomposed into a total of  $d_1$  equations:

$$\text{Theta(i, ...)} = (\text{A3} \otimes \text{A2}) \text{C}^T \cdot \{(1)\} (\text{A1}(i, :))^T, \text{ where } i = 1, 2, \dots, d_1.$$

Speedup 1: solve A1 via parallel optimization.

Fact: the optimization over A1 can be decomposed into a total of d1 sub-optimizations. These sub-optimizations are separable, and each of them involves only one row of A1.

$\text{Ly}(\text{Vec}(A1)) = \text{Ly}(A1(1,:)) + \dots + \text{Ly}(A1(d1,:))$ . Therefore,  $L_y(\Theta)$  becomes  $L_y(\text{Vec}(A_1))$  which is a simple convex optimization problem.

Likewise, you can update  $A_2$  fixing  $A_1, A_3$  and  $\mathcal{C}$  using following formula, make it a convex optimization problem again.

$$\text{Vec}(\Theta_{(2)}) = ((A_3 \otimes A_1) \mathcal{C}_{(2)}^T \otimes I_{d_2}) \text{Vec}(A_2)$$

You can repeat this on  $A_3$  fixing  $A_2, A_3$  and  $\mathcal{C}$  using following formula.

$$\text{Vec}(\Theta_{(3)}) = ((A_2 \otimes A_1) \mathcal{C}_{(3)}^T \otimes I_{d_3}) \text{Vec}(A_3)$$

Finally, you use a formula below to update core tensor  $\mathcal{C}$  with fixed  $A_1, A_2$  and  $A_3$

$$\text{Vec}(\Theta_{(1)}) = (A_3 \otimes A_2 \otimes A_1) \text{Vec}(\mathcal{C}_{(1)})$$

By iterating this until it converges, you can get an estimator of  $\arg \min_{\Theta} L_y(\Theta)$ . I used method "BFGS", quasi-Newton method to find each axis optimizer. The full algorithm is described in Algorithm 2.

---

### Algorithm 2 Ordinal tensor optimization with known boundary $\alpha$

---

**Input:**  $\mathcal{C}^0 \in \mathbf{R}^{r_1 \times \dots \times r_N}, A_1^0 \in \mathbf{R}^{d_1 \times r_1}, \dots, A_N^0 \in \mathbf{R}^{d_N \times r_N}$

**Output:** Optimizor of  $\mathcal{L}_Z(\alpha, \Theta)$  given  $\alpha$

**for**  $t = 1, 2, \dots$ , **do** until convergence,

**Update**  $A_n$

**for**  $n = 1, 2, \dots, N$  **do**

$$\Theta_{(n)} = A_n^t \mathcal{C}_{(n)}^t (A_{n+1}^t \otimes \dots \otimes A_N^t \otimes A_1^{t+1} \otimes \dots \otimes A_{n-1}^{t+1})^T$$

$$\text{Vec}(\Theta_{(n)}) = \left( (A_{n+1}^t \otimes \dots \otimes A_N^t \otimes A_1^{t+1} \otimes \dots \otimes A_{n-1}^{t+1}) (\mathcal{C}_{(n)}^t)^T \otimes I_{d_n} \right) \text{Vec}(A_n^t)$$

$$\text{Vec}(A_n^{t+1}) = \arg \max(\mathcal{L}_y(\alpha, \text{Vec}(\Theta_{(n)})))$$

$$\text{Get } A_n^{t+1}$$

**Update**  $\mathcal{C}$

$$\Theta_{(1)} = A_1^{t+1} \mathcal{C}_{(1)}^t (A_N^{t+1} \otimes \dots \otimes A_2^{t+1})^T$$

$$\text{Vec}(\Theta_{(1)}) = (A_N^{t+1} \otimes \dots \otimes A_1^{t+1}) \text{Vec}(\mathcal{C}_{(1)}^t)$$

$$\text{Vec}(\mathcal{C}_{(1)}^t) = \arg \max(\mathcal{L}_y(\alpha, \text{Vec}(\Theta_{(1)})))$$

$$\text{Get } \mathcal{C}^{t+1}$$

**return**  $\alpha, \Theta$

---

### 2.1.2 Unknown Bin Boundary

In real world, knowing bin boundary rarely happens so  $\alpha$  also becomes parameter we have to estimate. In this case, we can add one more block of  $\alpha$  to Algorithm 2. So after updating  $\mathcal{C}, A_1, A_2$  and  $A_3$  in the example section 2.1.1, we can update  $\alpha$  by fixing  $\Theta = \mathcal{C} \times_1 A_1 \times_2 A_2 \times_3 A_3$ . This updating process for  $\alpha$  is just finding intercepts in ordinal logistic regression with fixed slope as 1. The full algorithm is described in Algorithm 3.

---

**Algorithm 3** Ordinal tensor optimization with unknown boundary  $\alpha$ 


---

**Input:**  $\mathcal{C}^0 \in \mathbf{R}^{r_1 \times \dots \times r_N}, A_1^0 \in \mathbf{R}^{d_1 \times r_1}, \dots, A_N^0 \in \mathbf{R}^{d_N \times r_N}$

**Output:** Optimizer of  $\mathcal{L}_Z(\alpha, \Theta)$  given  $\alpha$

**for**  $t = 1, 2, \dots$ , **do** until convergence,

**Update**  $A_n$

**for**  $n = 1, 2, \dots, N$  **do**

$$\Theta_{(n)} = A_n^t \mathcal{C}_{(n)}^t (A_{n+1}^t \otimes \dots \otimes A_N^t \otimes A_1^{t+1} \otimes \dots \otimes A_{n-1}^{t+1})^T$$

$$Vec(\Theta_{(n)}) = \left( (A_{n+1}^t \otimes \dots \otimes A_N^t \otimes A_1^{t+1} \otimes \dots \otimes A_{n-1}^{t+1}) (\mathcal{C}_{(n)}^t)^T \otimes I_{d_n} \right) Vec(A_n^t)$$

$$Vec(A_n^{t+1}) = \arg \max(\mathcal{L}_Y(\alpha^t, Vec(\Theta_{(n)})))$$

Get  $A_n^{t+1}$

**Update**  $\mathcal{C}$

$$\Theta_{(1)} = A_1^{t+1} \mathcal{C}_{(1)}^t (A_N^{t+1} \otimes \dots \otimes A_2^{t+1})^T$$

$$Vec(\Theta_{(1)}) = (A_N^{t+1} \otimes \dots \otimes A_1^{t+1}) Vec(\mathcal{C}_{(1)}^t)$$

$$Vec(\mathcal{C}_{(1)}^t) = \arg \max(\mathcal{L}_Y(\alpha^t, Vec(\Theta_{(1)})))$$

Get  $\mathcal{C}^{t+1}$

**Update**  $\alpha$

$$\alpha^{t+1} = \arg \max(\mathcal{L}_Y(\alpha, \Theta^{t+1}))$$

**return**  $\alpha, \Theta$

---

## 2.2 Simulation

For this section, I want to talk about simulation results of Algorithm 2 and Algorithm 3.

### 2.2.1 Known Bin boundary case

Our given ordinal tensor data  $D \in R^{d \times d \times d}$  has three values such that  $D_{i,j,k} \in \{1, 2, 3\}$ . This data  $D$  was made by following procedure.

1. Choose arbitrary core tensor  $\mathcal{C} \in R^{3 \times 3 \times 3}$ , orthonormal matrix  $A_1 \in R^{d \times 3}, A_2 \in R^{d \times 3}$  and  $A_3 \in R^{d \times 3}$

How does the maximum norm  
of 'Theta compare to 1?

2. Get tensor  $\Theta = \mathcal{C} \times_1 A_1 \times_2 A_2 \times_3 A_3 \in R^{d \times d \times d}$

3. Fix  $\alpha = (\alpha_1, \alpha_2) = (1, 2)$

4. Make ordinal tensor  $D \in R^{d \times d \times d}$  such that

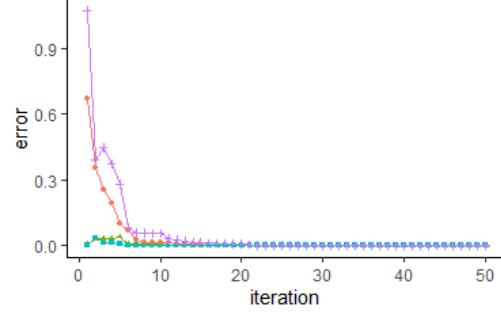
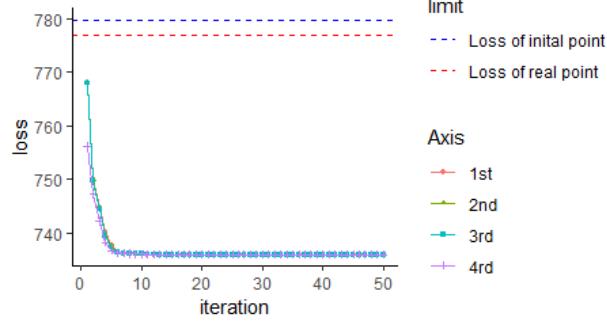
$$P(D_{i,j,k} = 1) = logistic(\alpha_1 + \Theta_{i,j,l}), \quad P(D_{i,j,k} = 2) = logistic(\alpha_2 + \Theta_{i,j,l}) - logistic(\alpha_1 + \Theta_{i,j,l}),$$

$$P(D_{i,j,k} = 3) = 1 - logistic(\alpha_2 + \Theta_{i,j,l})$$

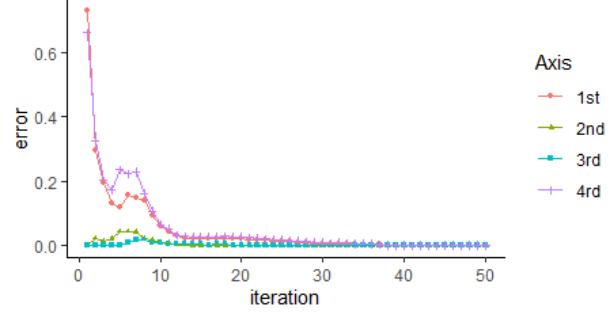
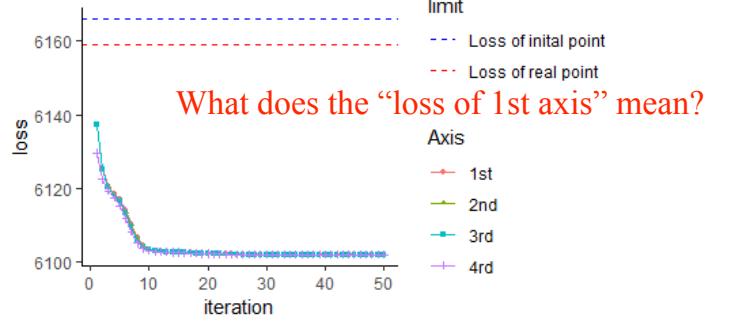
$$\text{where } logistic(x) = \frac{1}{1 + e^{-x}}$$

From this given data  $D$ , our goal is to estimate  $\mathcal{C}, A_1, A_2$  and  $A_3$  using Algorithm 2 with randomly selected initial values. I repeated above simulation for  $d = 10$  and  $d = 20$ . Results are in Figure 1.

How is the initial point chosen?



(a) When  $d = 10$



(b) When  $d = 20$

Figure 1: When we have knowledge that  $\alpha = (1, 2)$ . Y axis in upper part figures is values of loss function at each iteration. Blue dashed horizontal line is loss value evaluated at initial point and red dashed horizontal line is loss value evaluated at real parameter point. Y axis in lower part figures is value of norm difference between previous parameter and updated parameter. You can check our algorithm converges in both  $d = 10$  and  $d = 20$  cases and gives us parameters having smaller loss value than real parameters.

Also I could check difference norm between real  $\Theta$  and estimated  $\hat{\Theta}$  such that

$$diff(\Theta, \hat{\Theta}) = \|\Theta - \hat{\Theta}\|_F / \sqrt{d^3}$$

When  $d = 10$ ,  $diff(\Theta, \hat{\Theta}) = 0.7000825$ . When  $d = 20$ ,  $diff(\Theta, \hat{\Theta}) = 0.2789708$ . I think our estimators are quite close to real  $\Theta$

Please add the scatter plot of \Theta vs. \hat{\Theta}.  
The accuracy should depend on maximum norm of \Theta and dimension.

## 2.2.2 Unknown Bin boundary case

Our given ordinal tensor data  $D \in R^{d \times d \times d}$  has three values such that  $D_{i,j,k} \in \{1, 2, 3\}$ . This data  $D$  was made by following procedure.

1. Choose arbitrary core tensor  $\mathcal{C} \in R^{3 \times 3 \times 3}$ , orthonormal matrix  $A_1 \in R^{d \times 3}$ ,  $A_2 \in R^{d \times 3}$  and  $A_3 \in R^{d \times 3}$
2. Get tensor  $\Theta = \mathcal{C} \times_1 A_1 \times_2 A_2 \times_3 A_3 \in R^{d \times d \times d}$
3. Choose  $\alpha = (1, 2) = (\alpha_1, \alpha_2)$

4. Make ordinal tensor  $D \in R^{d \times d \times d}$  such that

$$P(D_{i,j,k} = 1) = \text{logistic}(\alpha_1 + \Theta_{i,j,l}), \quad P(D_{i,j,k} = 2) = \text{logistic}(\alpha_2 + \Theta_{i,j,l}) - \text{logistic}(\alpha_1 + \Theta_{i,j,l}),$$

$$P(D_{i,j,k} = 3) = 1 - \text{logistic}(\alpha_2 + \Theta_{i,j,l})$$

$$\text{where } \text{logistic}(x) = \frac{1}{1 + e^{-x}}$$

Only difference between the first and the second simulation is we don't know  $\alpha = (1, 2)$  in the second and will estimate  $\alpha$  including other parameters. From this given data  $D$ , we estimated  $\alpha, \mathcal{C}, A_1, A_2$  and  $A_3$  using Algorithm 3. I repeated above simulation for  $d = 10$  and  $d = 20$ . Results are in Figure 2.

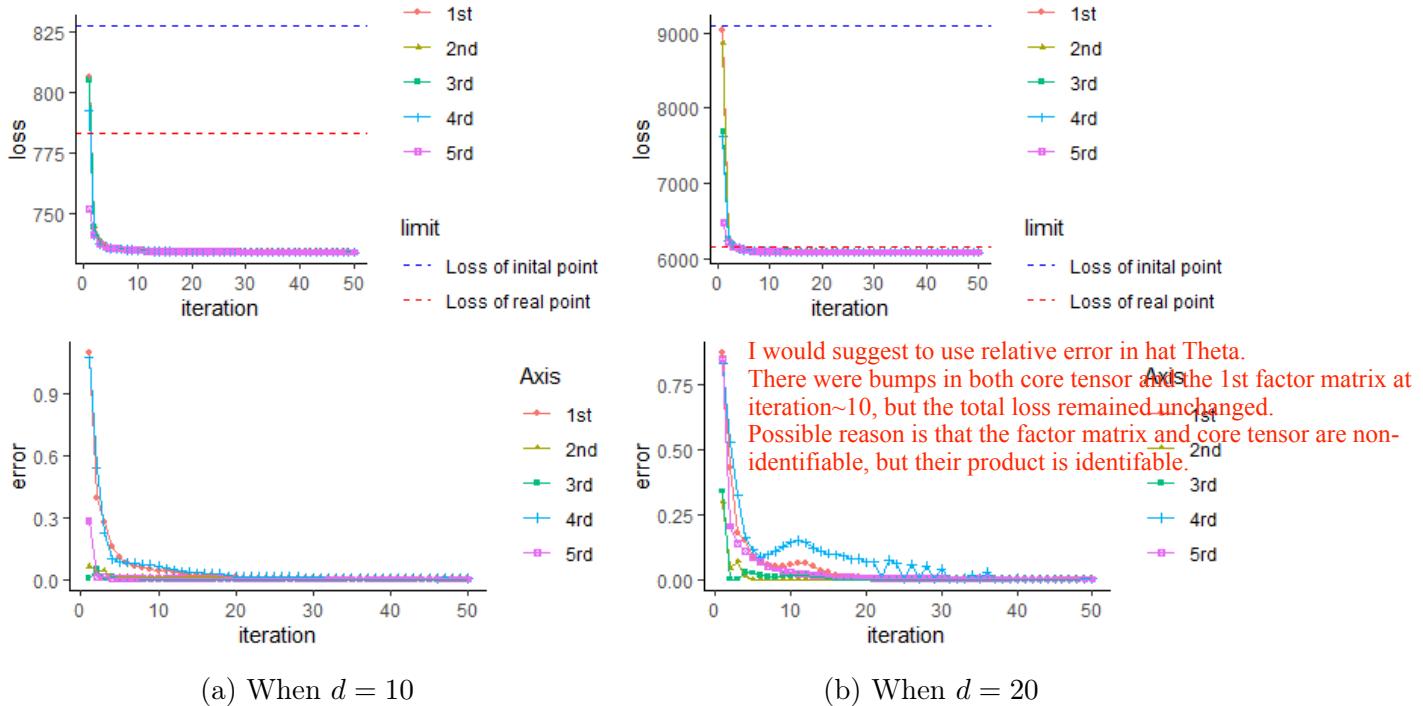


Figure 2: When we don't have knowledge that  $\alpha = (1, 2)$  Y axis in upper part figures is values of loss function at each iteration. Blue dashed horizontal line is loss value evaluated at initial point and red dashed horizontal line is loss value evaluated at real parameter point. Y axis in lower part figures is value of norm difference between previous parameter and updated parameter. You can check our algorithm converges in both  $d = 10$  and  $d = 20$  cases and gives us parameters having smaller loss value than real parameters.

Let's check how close it is between real  $\alpha = (1, 2)$  and estimated  $\hat{\alpha}$ . When  $d = 10$ , our estimated  $\hat{\alpha} = (0.9993238, 2.0961265)$  and when  $d = 20$ , our estimated  $\hat{\alpha} = (0.9484518, 1.9872800)$ . For difference norm defined above, we got  $\text{diff}(\Theta, \hat{\Theta}) = 0.8158596$  and  $\text{diff}(\Theta, \hat{\Theta}) = 0.3243989$  for each simulation. I think those simulation gave us moderate results.

To Q1. I cannot tell from the Figures the difference in ‘loss at real/true point’. Could you report the numerical value?  
I did see noticeable difference in “loss at initial point” between Figures 1 and 2 though.

### 3 Question and to do list

1. Actually, I can’t fully understand why graphs of Loss vs iteration look different according to knowledge of  $\alpha$ : **Loss value evaluated at real point** have a tendency to be small when we don’t know  $\alpha$ .
2. It took about 20 minutes for my algorithm to converge in the case of  $d = 20$  and when we don’t know the value of  $\alpha$ . Is it natural phenomenon or my algorithm is not efficient?( there’s no problem when  $d = 10$ )
3. I am now studying on estimation properties based on binary tensor research paper.

To Q2. Where is the bottleneck? Consider profiling your R code.

<https://www.r-bloggers.com/profiling-r-code/>

or **4 R codes**

<http://adv-r.had.co.nz/Profiling.html#performance-profiling>

#### 4.1 Algorithm 2

Possible reasons are (1) per-iteration cost of updating C; (2) per-iter cost of factor matrices; (3) per-iter cost of alpha; (4) too many iterations? Please confirm the main bottle neck.

```

1 library(MASS)
2 library(rTensor)
3 library(pracma)
4
5
6 normf = function(tnsr){
7   return(rTensor::fnorm(tnsr))
8 }
9
10 logistic = function(x){
11   return(1/(1+exp(-x)))
12 }
13
14 alpha = c(1,2)
15
16 likelihood = function(ttnsr,thet,alpha){
17   p1 = logistic(c(thet@data) + alpha[1])
18   p2 = logistic(c(thet@data) + alpha[2])
19   p = cbind(p1,p2-p1,1-p2)
20   return(-sum(log(c(p[which(c(ttnsr)==1),1],p[which(c(ttnsr)==2),2],p[
21     which(c(ttnsr)==3),3]))))
22 }
23
24 mleordinal = function(ttnsr,C,A_1,A_2,A_3){
25   result = list()
26   alpha = c(1,2)
27   error<- 3
28   iter = 0
29   d1 <- nrow(A_1); d2 <- nrow(A_2); d3 <- nrow(A_3)
30   r1 <- ncol(A_1); r2 <- ncol(A_2); r3 <- ncol(A_3)
31   while (error > 10^-3) {                                Speed-up 2: use the relative error
32     iter = iter +1                                         in Theta
33     #update A_1
34     tmp1 <- A_1
35     W1 = kronecker(kronecker(A_3,A_2)%*%t(k_unfold(C,1)@data),diag(1,d1))
```

```

35 h1 = function(A_1){
36   thet = W1%*%c(A_1)
37   p1 = logistic(thet + alpha[1])
38   p2 = logistic(thet + alpha[2])
39   p = cbind(p1,p2-p1,1-p2)
40   return(-sum(log(c(p[which(c(ttnsr)==1),1],p[which(c(ttnsr)==2),2],p[
41 which(c(ttnsr)==3),3]))))
42 }
43 g1 = function(A_1){
44   thet = W1%*%c(A_1)
45   p1 = logistic(thet + alpha[1])
46   p2 = logistic(thet + alpha[2])
47   q1 <- p1-1
48   q2 <- (p2*(1-p2)-p1*(1-p1))/(p1-p2) Speed-up 3: write a single function that implements h1/h2/h3. Put the
49   sub-functions outside the function mleordinal.
50   q3 <- p2
51   gd = apply(diag(q1[which(c(ttnsr)==1)]))%*%W1[which(c(ttnsr)==1),],2,
52   sum)+ apply(diag(q2[which(c(ttnsr)==2)]))%*%W1[which(c(ttnsr)==2),],2,sum
53 )+
54   apply(diag(q3[which(c(ttnsr)==3)]))%*%W1[which(c(ttnsr)==3),],2,sum
55 )
56   return(gd)
57 }
58
59 a <- optim(c(A_1),h1,g1,method = "BFGS")
60 A_1 <- matrix(a$par,nrow = d1,ncol = r1)
61 error1 <- norm(A_1-tmp1,"F")/sqrt(length(c(A_1)))
62
63 # update A_2
64 tmp2 <- A_2
65 W2 <- kronecker(kronecker(A_3,A_1)%*%t(k_unfold(C,2)@data),diag(1,d2))
66 h2 = function(A_2){
67   tmpthet = W2%*%c(A_2)
68   thet = c(k_unfold(k_fold(matrix(tmpthet,nrow = d2),2,modes = c(d1,d2
69 ,d3)),1)@data)
70   p1 = logistic(thet + alpha[1])
71   p2 = logistic(thet + alpha[2])
72   p = cbind(p1,p2-p1,1-p2)
73   return(-sum(log(c(p[which(c(ttnsr)==1),1],p[which(c(ttnsr)==2),2],p[
74 which(c(ttnsr)==3),3]))))
75 }
76 g2 = function(A_2){
77   tmpthet = W2%*%c(A_2)
78   thet = c(k_unfold(k_fold(matrix(tmpthet,nrow = d2),2,modes = c(d1,d2
79 ,d3)),1)@data)
80   p1 = logistic(thet + alpha[1])
81   p2 = logistic(thet + alpha[2])
82   q1 <- p1-1
83   q2 <- (p2*(1-p2)-p1*(1-p1))/(p1-p2)
84   q3 <- p2
85   gd = apply(diag(q1[which(c(ttnsr)==1)]))%*%W2[which(c(ttnsr)==1),],2,
86   sum)+
```

```

80      apply(diag(q2[which(c(ttnsr)==2)])%*%W2[which(c(ttnsr)==2),],2,sum
81  )+
82      apply(diag(q3[which(c(ttnsr)==3)])%*%W2[which(c(ttnsr)==3),],2,sum
83  )
84      return(gd)
85  }
86 b <- optim(c(A_2),h2,g2,method = "BFGS")
87 A_2 <- matrix(b$par,nrow = d2,ncol = r2)
88 error2 <- norm(A_2-tmp2,"F")/sqrt(length(c(A_2)))
89
90 # update A_3
91 tmp3 <- A_3
92 W3 <- kronecker(kronecker(A_2,A_1)%*%t(k_unfold(C,3)@data),diag(1,d3))
93 h3 = function(A_3){
94   tmpthet = W3%*%c(A_3)
95   thet = c(k_unfold(k_fold(matrix(tmpthet,nrow = d3),3,modes = c(d1,d2
96 ,d3)),1)@data)
97   p1 = logistic(theta + alpha[1])
98   p2 = logistic(theta + alpha[2])
99   p = cbind(p1,p2-p1,1-p2)
100  return(-sum(log(c(p[which(c(ttnsr)==1),1],p[which(c(ttnsr)==2),2],p[
101 which(c(ttnsr)==3),3]))))
102 }
103 g3 = function(A_2){
104   tmpthet = W3%*%c(A_3)
105   thet = c(k_unfold(k_fold(matrix(tmpthet,nrow = d3),3,modes = c(d1,d2
106 ,d3)),1)@data)
107   p1 = logistic(theta + alpha[1])
108   p2 = logistic(theta + alpha[2])
109   q1 <- p1-1
110   q2 <- (p2*(1-p2)-p1*(1-p1))/(p1-p2)
111   q3 <- p2
112   gd = apply(diag(q1[which(c(ttnsr)==1)])%*%W3[which(c(ttnsr)==1),],2,sum
113 )+
114     apply(diag(q2[which(c(ttnsr)==2)])%*%W3[which(c(ttnsr)==2),],2,sum
115 )+
116     apply(diag(q3[which(c(ttnsr)==3)])%*%W3[which(c(ttnsr)==3),],2,sum
117 )
118   return(gd)
119 }
120 c <- optim(c(A_3),h3,g3,method = "BFGS")
121 A_3 <- matrix(c$par,nrow = d3,ncol = r3)
122 error3 <- norm(A_3-tmp3,"F")/sqrt(length(c(A_3)))
123 # update C
124 tmp4 <- C
125 W4 <- kronecker(kronecker(A_3,A_2),A_1)
126 h4 = function(Cvec){
127   theta = W4%*%Cvec
128   p1 = logistic(theta + alpha[1])
129   p2 = logistic(theta + alpha[2])
130   p = cbind(p1,p2-p1,1-p2)
131   return(-sum(log(c(p[which(c(ttnsr)==1),1],p[which(c(ttnsr)==2),2],p[
132 which(c(ttnsr)==3),3]))))
133 }

```

```

125 g4 = function(Cvec){
126   thet = W4%*%Cvec
127   p1 = logistic(thet + alpha[1])
128   p2 = logistic(thet + alpha[2])
129   q1 <- p1-1
130   q2 <- (p2*(1-p2)-p1*(1-p1))/(p1-p2)
131   q3 <- p2
132   gd = apply(diag(q1[which(c(ttnsr)==1)]))%*%W4[which(c(ttnsr)==1),],2,sum) +
133     apply(diag(q2[which(c(ttnsr)==2)]))%*%W4[which(c(ttnsr)==2),],2,sum) +
134     apply(diag(q3[which(c(ttnsr)==3)]))%*%W4[which(c(ttnsr)==3),],2,sum)
135   )
136   return(gd)
137 }
138 d <- optim(c(C@data),h4,g4,method = "BFGS")
139 C <- new("Tensor",C@num_modes,C@modes,data =d$par)
140 error4 <- normf(C-tmp4)/sqrt(length(c(C@data)))
141 error <- max(error1,error2,error3,error4)
142 }
143 result$C <- C; result$A_1 <- A_1; result$A_2 <- A_2; result$A_3 <- A_3;
144 result$iteration <- iter
145 return(result)
146 }
```

## 4.2 Algorithm 3

```

1 mleordinal2 = function(ttnsr,C,A_1,A_2,A_3,alpha){
2   result = list()
3   error <- 3
4   iter = 0
5   d1 <- nrow(A_1); d2 <- nrow(A_2); d3 <- nrow(A_3)
6   r1 <- ncol(A_1); r2 <- ncol(A_2); r3 <- ncol(A_3)
7   while (error > 10^-3) {
8     iter = iter +1
9     #update A_1
10    tmp1 <- A_1
11    W1 = kronecker(kronecker(A_3,A_2)%*%t(k_unfold(C,1)@data),diag(1,d1))
12    h1 = function(A_1){
13      thet =W1%*%c(A_1)
14      p1 = logistic(thet + alpha[1])
15      p2 = logistic(thet + alpha[2])
16      p = cbind(p1,p2-p1,1-p2)
17      return(-sum(log(c(p[which(c(ttnsr)==1),1],p[which(c(ttnsr)==2),2],p[which(c(ttnsr)==3),3])))))
18    }
19    g1 = function(A_1){
20      thet =W1%*%c(A_1)
21      p1 = logistic(thet + alpha[1])
22      p2 = logistic(thet + alpha[2])
23      q1 <- p1-1
24      q2 <- (p2*(1-p2)-p1*(1-p1))/(p1-p2)
```

```

25   q3 <- p2
26   gd = apply(diag(q1[which(c(ttnsr)==1)])%*%W1[which(c(ttnsr)==1),],2,
27   sum)+
28     apply(diag(q2[which(c(ttnsr)==2)])%*%W1[which(c(ttnsr)==2),],2,sum
29   )+
30     apply(diag(q3[which(c(ttnsr)==3)])%*%W1[which(c(ttnsr)==3),],2,sum
31   )
32   return(gd)
33 }
34
35
36
37 # update A_2
38 tmp2 <- A_2
39 W2 <- kronecker(kronecker(A_3,A_1)%*%t(k_unfold(C,2)@data),diag(1,d2))
40 h2 = function(A_2){
41   tmpthet = W2%*%c(A_2)
42   thet = c(k_unfold(k_fold(matrix(tmpthet,nrow = d2),2,modes = c(d1,d2
43 ,d3)),1)@data)
44   p1 = logistic(theta + alpha[1])
45   p2 = logistic(theta + alpha[2])
46   p = cbind(p1,p2-p1,1-p2)
47   return(-sum(log(c(p[which(c(ttnsr)==1),1],p[which(c(ttnsr)==2),2],p[
48 which(c(ttnsr)==3),3]))))
49 }
50 g2 = function(A_2){
51   tmpthet = W2%*%c(A_2)
52   thet = c(k_unfold(k_fold(matrix(tmpthet,nrow = d2),2,modes = c(d1,d2
53 ,d3)),1)@data)
54   p1 = logistic(theta + alpha[1])
55   p2 = logistic(theta + alpha[2])
56   q1 <- p1-1
57   q2 <- (p2*(1-p2)-p1*(1-p1))/(p1-p2)
58   q3 <- p2
59   gd = apply(diag(q1[which(c(ttnsr)==1)])%*%W2[which(c(ttnsr)==1),],2,
60   sum)+
61     apply(diag(q2[which(c(ttnsr)==2)])%*%W2[which(c(ttnsr)==2),],2,sum
62   )+
63     apply(diag(q3[which(c(ttnsr)==3)])%*%W2[which(c(ttnsr)==3),],2,sum
64   )
65   return(gd)
66 }
67 b <- optim(c(A_2),h2,g2,method = "BFGS")
68 A_2 <- matrix(b$par,nrow = d2,ncol = r2)
69 error2 <- norm(A_2-tmp2,"F")/sqrt(length(c(A_2)))
70
71
72 # update A_3
73 tmp3 <- A_3
74 W3 <- kronecker(kronecker(A_2,A_1)%*%t(k_unfold(C,3)@data),diag(1,d3))
75 h3 = function(A_3){
76   tmpthet = W3%*%c(A_3)

```

```

70     thet = c(k_unfold(k_fold(matrix(tmpthet, nrow = d3), 3, modes = c(d1, d2
71 ,d3)), 1)@data)
72     p1 = logistic(thet + alpha[1])
73     p2 = logistic(thet + alpha[2])
74     p = cbind(p1, p2 - p1, 1 - p2)
75     return(-sum(log(c(p[which(c(ttnsr)==1), 1], p[which(c(ttnsr)==2), 2], p[
76 which(c(ttnsr)==3), 3])))))
77 }
78 g3 = function(A_2){
79   tmpthet = W3%*%c(A_3)
80   thet = c(k_unfold(k_fold(matrix(tmpthet, nrow = d3), 3, modes = c(d1, d2
81 ,d3)), 1)@data)
82   p1 = logistic(thet + alpha[1])
83   p2 = logistic(thet + alpha[2])
84   q1 <- p1 - 1
85   q2 <- (p2 * (1 - p2) - p1 * (1 - p1)) / (p1 - p2)
86   q3 <- p2
87   gd = apply(diag(q1[which(c(ttnsr)==1)]))%*%W3[which(c(ttnsr)==1), ] ,2 ,
88 sum) +
89     apply(diag(q2[which(c(ttnsr)==2)]))%*%W3[which(c(ttnsr)==2), ] ,2 ,sum
90 ) +
91   apply(diag(q3[which(c(ttnsr)==3)]))%*%W3[which(c(ttnsr)==3), ] ,2 ,sum
92 )
93   return(gd)
94 }
95 c <- optim(c(A_3), h3, g3, method = "BFGS")
96 A_3 <- matrix(c$par, nrow = d3, ncol = r3)
97 error3 <- norm(A_3 - tmp3, "F") / sqrt(length(c(A_3)))
98 # update C
99 tmp4 <- C
100 W4 <- kronecker(kronecker(A_3, A_2), A_1)
101 h4 = function(Cvec){
102   thet = W4%*%Cvec
103   p1 = logistic(thet + alpha[1])
104   p2 = logistic(thet + alpha[2])
105   p = cbind(p1, p2 - p1, 1 - p2)
106   return(-sum(log(c(p[which(c(ttnsr)==1), 1], p[which(c(ttnsr)==2), 2], p[
107 which(c(ttnsr)==3), 3])))))
108 }
109 g4 = function(Cvec){
110   thet = W4%*%Cvec
111   p1 = logistic(thet + alpha[1])
112   p2 = logistic(thet + alpha[2])
113   q1 <- p1 - 1
114   q2 <- (p2 * (1 - p2) - p1 * (1 - p1)) / (p1 - p2)
115   q3 <- p2
116   gd = apply(diag(q1[which(c(ttnsr)==1)]))%*%W4[which(c(ttnsr)==1), ] ,2 ,
117 sum) +
118     apply(diag(q2[which(c(ttnsr)==2)]))%*%W4[which(c(ttnsr)==2), ] ,2 ,sum
119 ) +
120   apply(diag(q3[which(c(ttnsr)==3)]))%*%W4[which(c(ttnsr)==3), ] ,2 ,sum
121 )
122   return(gd)
123 }
```

```

114 d <- optim(c(C@data), h4, g4, method = "BFGS")
115 C <- new("Tensor", C@num_modes, C@modes, data = d$par)
116 error4 <- normf(C-tmp4)/sqrt(length(c(C@data)))
117
118 #update alpha
119 tmp5 <- alpha
120 theta <- ttm(ttm(C, A_1, 1), A_2, 2), A_3, 3)
121 m <- polr(as.factor(c(ttnsr)) ~ offset(-c(theta@data)))
122 alpha <- m$zeta
123 error5 <- sqrt(sum((alpha - tmp5)^2)/2)
124 error <- max(error1, error2, error3, error4, error5)
125 }
126 result$C <- C; result$A_1 <- A_1; result$A_2 <- A_2; result$A_3 <- A_3
127 result$alpha <- alpha ; result$iteration <- iter
128 return(result)
129 }
```