Title

Learning Diagonal Gaussian Mixture Models and Incomplete Tensor Decompositions

Author

Guo, Bingni

Publication Date

2024

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

Learning Diagonal Gaussian Mixture Models and Incomplete Tensor Decompositions

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Mathematics

by

Bingni Guo

Committee in charge:

Professor Jiawang Nie, Chair
Professor Bhaskar Rao
Professor Rose Yu
Professor Tianyi Zheng
Professor Wenxin Zhou

2024

The Dissertation of Bingni Guo is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Firstly, I would like to express my greatest appreciation to my advisor Professor Jiawang Nie. His unwavering support, invaluable guidance and insightful feedback have been instrumental throughout my doctoral journey. I am very fortunate to be guided by him in my Ph.D study and I am deeply grateful for his mentorship.

Next, I hope to thank Professor Zi Yang for his contribution on the research project for his generous suggestions. I would also like to thank Professor Bhaskar Rao, Professor Rose Yu, Professor Tianyi Zheng and Professor Wenxin Zhou for serving on my committee and providing impassive comments.

No fortune is more valuable than a good friend. I would like to thank my colleagues and friends in UCSD, including Toni Gui, Jingwen Liang, Jiaqi Liu, Zhiling Liu, Tuo Lin, Chunyi Lyu, Jiajie Shi, Xindong Tang, Xuyu Zhang, Muhan Zhao, Suhan Zhong, Ziyan Zhu, etc. for their companionship. I am also grateful to my friends outside UCSD. In particular, I want to thank Diyue Guo, Yuyan Guo, Tianning Lu and Zijun Luo.

Finally, I would like to express my deepest gratitude to my parents for always being by my side. The love from them is my strongest support.

In this dissertation, some materials have been published, or been submitted for publication.

The Chapter 2, in full, is a reprint of the material as it appears in *Vietnam Journal of Mathematics* 2021 [64]. The dissertation author coauthored this paper with Nie, Jiawang and Yang, Zi.

The Chapter 3, in full, has been submitted for publication. The dissertation author is the coauthor of the preprint "B. Guo, J. Nie and Z. Yang, Diagonal Gaussian Mixture Models and Higher Order Tensor Decompositions, 2024. arXiv preprint arXiv:2401.01337".

VITA

2012–2016 Bachelor of Science, Xi'an Jiaotong University

2016–2023 Master of Arts, University of California San Diego

2018–2024 Doctor of Philosophy, University of California San Diego

PUBLICATIONS

"Learning Diagonal Gaussian Mixture Models and Incomplete Tensor Decompositions" Joint with J. Nie and Z. Yang on Vietnam Journal of Mathematics, 50(2), pp 421–446, 2022.

"Diagonal Gaussian Mixture Models and Higher Order Tensor Decompositions" Joint with J. Nie and Z. Yang, Submitted, 2024.

FIELDS OF STUDY

Major Field: Mathematics

Studies in Applied Mathematics
Professors Jiawang Nie

ABSTRACT OF THE DISSERTATION

Learning Diagonal Gaussian Mixture Models and Incomplete Tensor Decompositions

by

Bingni Guo

Doctor of Philosophy in Mathematics

University of California San Diego, 2024

Professor Jiawang Nie, Chair

Gaussian mixture models are widely used in statistics and machine learning because of their simple formulation and superior fitting ability. High order moments of the Gaussian mixture model form incomplete symmetric tensors generated by hidden parameters in the model. This thesis studies how to recover unknown parameters in diagonal Gaussian mixture models using high order moment tensors. The problem can be formulated as computing incomplete symmetric tensor decompositions. We propose to use generating polynomials to compute incomplete symmetric tensor approximations and approximations. The obtained decomposition is utilized to recover parameters in the model.

In the first part of thesis, we propose a learning algorithm using the first and third order moment tensors and require that the number of components $r \leq \frac{d}{2} - 1$ for mixture of d-dimensional Gaussians. In the second part of thesis, we generalize the previous algorithm using higher order moment tensors and therefore we can recover the unknown parameters of the model when the number of components $r > \frac{d}{2} - 1$. We provide an upper bound of the number of components in the Gaussian mixture model that the generalized algorithm can compute. For both algorithms, we prove that our recovered parameters are accurate when the estimated moments are accurate. Numerical simulations and comparisons with EM algorithm are presented to show the performance of our algorithms.

Chapter 1

Introduction

1.1 Tensors

Let m be positive integers. Let $\mathbb{F} = \mathbb{C}$ (the complex field) or $\mathbb{R}$ (the real field). Denote $X_1, \ldots, X_m$ finite dimensional vector spaces over the field $\mathbb{F}$. The dual space X_i^* of vector space X_i is defined to be the space of all linear functionals $x_i^* : X_i \to \mathbb{R}$. Denote the linear functional $x_1 \otimes \cdots \otimes x_m$ on $X_1^* \times \cdots \times X_m^*$ such that

$$(x_1 \otimes \cdots \otimes x_m)(z_1, \ldots, z_m) = z_1(x_1) \cdots z_m(x_m),$$

for arbitrary $z_i \in X_i^*$. The tensor product space of $X_1, \ldots, X_m$, denote by $X_1 \otimes \cdots \otimes X_m$, is a vector space of such $x_1 \otimes \cdots \otimes x_m$.

If $\{e_j^i : j = 1, \ldots, n_i\}$ is a basis for X_i, then the set of all tensors of the form

$$e_{j_1}^1 \otimes \cdots \otimes e_{j_m}^m,$$

where $1 \leq j_i \leq n_i$ induces a basis of $X_1 \otimes \cdots \otimes X_m$. By universal property, there exists a isomorphism between a tensor space $\mathbb{F}^{n_1} \otimes \cdots \otimes \mathbb{F}^{n_m}$ and $\mathbb{F}^{n_1 \times \cdots \times n_m}$. Therefore we may represent a tensor $\mathcal{T} \in \mathbb{F}^{n_1} \otimes \cdots \otimes \mathbb{F}^{n_m}$ by a multidimensional array in $\mathbb{F}^{n_1 \times \cdots \times n_m}$.

Let $\mathrm{S}^m(\mathbb{C}^d)$ (resp., $\mathrm{S}^m(\mathbb{R}^d)$) denote the space of mth order symmetric tensors over the vector space $\mathbb{C}^d$ (resp., $\mathbb{R}^d$). For convenience of notation, we set $d = n + 1$ and the labels for tensors start with 0. A symmetric tensor $\mathcal{F}$ of order m and dimension $n+1$ can be represented by an array indexed by integer tuples $(i_1, \ldots, i_m)$, that is,

$$\mathcal{F} = (\mathcal{F}_{i_1 \ldots i_m})_{0 \le i_1, \ldots, i_m \le n},$$

where the entry $\mathcal{F}_{i_1 \ldots i_m}$ is invariant for all permutations of $(i_1, \ldots, i_m)$.

For a vector $u := (u_0, u_1, \ldots, u_n) \in \mathbb{C}^{n+1}$, the tensor power $u^{\otimes m} := u \otimes \cdots \otimes u$, where u is repeated m times, is defined such that

$$(u^{\otimes m})_{i_1 \ldots i_m} = u_{i_1} \times \cdots \times u_{i_m}.$$

An outer product like $u^{\otimes m}$ is called a rank-1 symmetric tensor.

For every $\mathcal{F} \in \mathrm{S}^m(\mathbb{C}^d)$, there exist $u_1, \ldots, u_r \in \mathbb{C}^{n+1}$ such that

$$\mathcal{F} = (u_1)^{\otimes m} + \cdots + (u_r)^{\otimes m}.$$

The smallest such r is defined as the symmetric rank of $\mathcal{F}$, denoted as $\mathrm{rank}_\mathrm{S}(\mathcal{F})$, i.e.

$$\mathrm{rank}_\mathrm{S}(\mathcal{F}) := \min \left\{ r \mid \mathcal{F} = \sum_{i=1}^{r} u_i^{\otimes m} \right\}.$$

There are other types of tensor ranks [34, 36]. We refer to [11, 16, 24, 29, 34, 36] for general work about tensors and their ranks. In this thesis, we only deal with symmetric tensors and symmetric ranks. For convenience, if $r = \mathrm{rank}_\mathrm{S}(\mathcal{F})$, we call $\mathcal{F}$ a rank-r tensor and $\mathcal{F} = \sum_{i=1}^{r} u_i^{\otimes m}$ is called a rank decomposition.

1.2 Generating Polynomials

For a power $\alpha := (\alpha_1, \alpha_2, \cdots, \alpha_n) \in \mathbb{N}^n$ and $x := (x_1, x_2, \cdots, x_n)$, denote

$$|\alpha| := \alpha_1 + \alpha_2 + \cdots + \alpha_n, \quad x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}, \quad x_0 := 1.$$

The monomial power set of degree m is denoted as

$$\mathbb{N}_m^n := \{\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n) \in \mathbb{N}^n : |\alpha| \leq m\}.$$

The symmetric tensor $\mathcal{F} \in \mathrm{S}^m(\mathbb{C}^{n+1})$ can be labelled by the monomial power set $\mathbb{N}_m^n$, i.e.,

$$\mathcal{F}_\alpha = \mathcal{F}_{x^\alpha} = \mathcal{F}_{i_1 \ldots i_m}$$

where $x^\alpha = x_0^{m-|\alpha|} x^\alpha = x_{i_1} \ldots x_{i_m}$.

For a finite set $\mathscr{B} \subset \mathbb{C}[x]$ of monomials and a vector $v \in \mathbb{C}^n$, we denote the vector of monomials in $\mathscr{B}$ evaluated on v as

$$[v]_{\mathscr{B}} := (f(v))_{f \in \mathscr{B}}.$$

Let $\mathbb{C}[x]_m$ be the space of all polynomials in x with complex coefficients and degrees no more than m. For a polynomial $p \in \mathbb{C}[x]_m$ and a symmetric tensor $\mathcal{F} \in \mathrm{S}^m(\mathbb{C}^{n+1})$, we define the bilinear product (note that $x_0 = 1$)

$$\langle p, \mathcal{F} \rangle = \sum_{\alpha \in \mathbb{N}_m^n} p_\alpha \mathcal{F}_\alpha \quad \text{for} \quad p = \sum_{\alpha \in \mathbb{N}_m^n} p_\alpha x^\alpha, \tag{1.1}$$

where p_α's are coefficients of p. A polynomial $g \in \mathbb{C}[x]_m$ is called a *generating polynomial* for a symmetric tensor $\mathcal{F} \in \mathrm{S}^m(\mathbb{C}^{n+1})$ if

$$\langle g \cdot x^{\beta},\ \mathcal{F} \rangle = 0 \quad \forall \beta \in \mathbb{N}^n_{m-\deg(g)}, \tag{1.2}$$

where $\deg(g)$ denotes the degree of g in x.

For a cubic polynomial $p \in \mathbb{C}[x]_3$ and $\mathcal{F} \in \mathrm{S}^3(\mathbb{C}^{n+1})$, we have the bilinear product (note that $x_0 = 1$)

$$\langle p, \mathcal{F} \rangle = \sum_{0 \le i_1, i_2, i_3 \le n} p_{i_1 i_2 i_3} \mathcal{F}_{i_1 i_2 i_3} \quad \text{for} \quad p = \sum_{0 \le i_1, i_2, i_3 \le n} p_{i_1 i_2 i_3} x_{i_1} x_{i_2} x_{i_3}, \tag{1.3}$$

where $p_{i_1 i_2 i_3}$ are coefficients of p.

Example 1.2.1. *Suppose there is a tensor $\mathcal{F} \in S^3(\mathbb{C}^3)$ that can be represented as*

$$\left[\begin{array}{ccc|ccc|ccc} 8 & -7 & 19 & -7 & 11 & 12 & -19 & 12 & 29 \\ -7 & 11 & 12 & 11 & -1 & -22 & 12 & -22 & -34 \\ -19 & 12 & 29 & 12 & -22 & -34 & 29 & -34 & -73 \end{array}\right]$$

The following polynomials are generating polynomials for the tensor $\mathcal{F}$

$$g_1 = 1 + 1.2x_1 - 0.6x_2 - x_1^2,$$

$$g_2 = 2 - 1.6x_1 + 0.8x_2 - x_1x_2,$$

$$g_3 = 4 - 1.2x_1 + 0.6x_2 - x_2^2.$$

One can verify it by checking the definition in (1.2).

These three polynomials have common zeros

$$(1, 2),\ (2, -1),\ (-1, 2),$$

which can be used to construct a symmetric rank decomposition of $\mathcal{F}$ as

$$\mathcal{F} = -(1,1,2)^{\otimes 3} + (1,2,-1)^{\otimes 3} + 2(1,-1,-2)^{\otimes 3}.$$

Using generating polynomials of a rank r tensor, we can show that it only uses its first r entries and a set of generating polynomials to represent the whole tensor as in the work [43].

For a given rank r, denote the index sets

$$\mathbb{B}_0 \coloneqq \{\underbrace{1, x_1, \dots, x_n, x_1^2, x_1x_2, \dots}_{\text{first r monomials}}\},$$
$$\mathbb{B}_1 \coloneqq (\mathbb{B}_0 \cup x_1\mathbb{B}_0 \cup \cdots \cup x_n\mathbb{B}_0)\backslash\mathbb{B}_0.$$

Then for $\alpha \in \mathbb{B}_1$ and $G \in \mathbb{C}^{\mathbb{B}_0 \times \mathbb{B}_1}$, define the polynomials

$$\varphi[G,\alpha] \coloneqq \sum_{\beta \in \mathbb{B}_0} G(\beta,\alpha)x^\beta - x^\alpha.$$

Proposition 1.2.2. *For a tensor $\mathcal{F} \in S^m(\mathbb{C}^{n+1})$, if $\varphi[G,\alpha]$ is a generating polynomial for a matrix $G \in \mathbb{C}^{\mathbb{B}_0 \times \mathbb{B}_1}$ and $\forall \alpha \in \mathbb{B}_1$, then for $\forall \alpha \in \mathbb{B}_1$ and $\forall \gamma \in \mathbb{N}^n$ with $|\gamma| + |\alpha| \leq m$, we have*

$$\mathcal{F}_{\alpha+\gamma} = \sum_{\beta \in \mathbb{B}_0} G(\beta,\alpha)\mathcal{F}_{\beta+\gamma}.$$

We refer to [43] for more details of generating polynomials. The generating polynomials are powerful tools to compute low-rank tensor decompositions and approximations [64, 45, 43]. More work about tensor optimization can be found in [19, 20, 46, 21, 22, 48, 47].

1.3 Gaussian Mixture Models

A Gaussian mixture model consists of several component Gaussian distributions. For given samples of a Gaussian mixture model, people often need to estimate parameters for each component Gaussian distribution [27, 35]. Consider a Gaussian mixture model with r components. For each $i \in [r] := \{1, \ldots, r\}$, let ω_i be the positive probability for the ith component Gaussian to appear in the mixture model. We have each $\omega_i > 0$ and $\sum_{i=1}^{r} \omega_i = 1$. Suppose the ith Gaussian distribution is $\mathcal{N}(\mu_i, \Sigma_i)$, where $\mu_i \in \mathbb{R}^d$ is the expectation (or mean) and $\Sigma_i \in \mathbb{R}^{d\times d}$ is the covariance matrix. Let $y \in \mathbb{R}^d$ be the random vector for the Gaussian mixture model and let $y_1, \ldots, y_N$ be identically independent distributed (i.i.d) samples from the mixture model. Each y_j is sampled from one of the r component Gaussian distributions, associated with a label $Z_j \in [r]$ indicating the component that it is sampled from. The probability that a sample comes from the ith component is ω_i. When people observe only samples without labels, the Z_j's are called latent variables. The density function for the random variable y is

$$f(y) := \sum_{i=1}^{r} \omega_i \frac{1}{\sqrt{(2\pi)^d \det \Sigma_i}} \exp\Big\{ -\frac{1}{2}(y-\mu_i)^T \Sigma_i^{-1} (y-\mu_i) \Big\},$$

where μ_i is the mean and Σ_i is the covariance matrix for the ith component.

Learning a Gaussian mixture model is to estimate the parameters $\omega_i, \mu_i, \Sigma_i$ for each $i \in [r]$, from given samples of y. The number of parameters in a covariance matrix grows quadratically with respect to the dimension. Due to the curse of dimensionality, the computation becomes very expensive for large d [38]. Hence, diagonal covariance matrices are preferable in applications. In this paper, we focus on learning Gaussian mixture models with diagonal covariance matrices, i.e.

$$\Sigma_i = \operatorname{diag}\big(\sigma_{i1}^2, \ldots, \sigma_{id}^2\big), \quad i = 1, \ldots, r.$$

1.3.1 Third Order Moment Structure

Let $M_3 := \mathbb{E}(y \otimes y \otimes y)$ be the third order tensor of moments for y. One can write that $y = \eta(z) + \zeta(z)$, where z is a discrete random variable such that $\text{Prob}(z = i) = \omega_i$, $\eta(i) = \mu_i \in \mathbb{R}^d$ and $\zeta(i)$ is the random variable ζ_i obeying the Gaussian distribution $\mathcal{N}(0, \Sigma_i)$. Assume all Σ_i are diagonal, then

$$M_3 = \sum_{i=1}^{r} \omega_i \mathbb{E}[(\eta(i) + \zeta_i)^{\otimes 3}] = \sum_{i=1}^{r} \omega_i \Big(\mu_i \otimes \mu_i \otimes \mu_i + \mathbb{E}[\mu_i \otimes \zeta_i \otimes \zeta_i] + \mathbb{E}[\zeta_i \otimes \mu_i \otimes \zeta_i] + \mathbb{E}[\zeta_i \otimes \zeta_i \otimes \mu_i] \Big). \quad (1.4)$$

The second equality holds because ζ_i has zero mean and

$$\mathbb{E}[\zeta_i \otimes \zeta_i \otimes \zeta_i] = \mathbb{E}[\mu_i \otimes \mu_i \otimes \zeta_i] = \mathbb{E}[\zeta_i \otimes \mu_i \otimes \mu_i] = \mathbb{E}[\mu_i \otimes \zeta_i \otimes \mu_i] = 0.$$

The random variable ζ_i has diagonal covariance matrix, so $\mathbb{E}[(\zeta_i)_j (\zeta_i)_l] = 0$ for $j \neq l$. Therefore,

$$\sum_{i=1}^{r} \omega_i \mathbb{E}[\mu_i \otimes \zeta_i \otimes \zeta_i] = \sum_{i=1}^{r} \sum_{j=1}^{d} \omega_i \sigma_{ij}^2 \mu_i \otimes e_j \otimes e_j = \sum_{j=1}^{d} a_j \otimes e_j \otimes e_j,$$

where the vectors a_j are given by

$$a_j := \sum_{i=1}^{r} \omega_i \sigma_{ij}^2 \mu_i, \quad j = 1, \ldots, d.$$

Similarly, we have

$$\sum_{i=1}^{r} \omega_i \mathbb{E}[\zeta_i \otimes \mu_i \otimes \zeta_i] = \sum_{j=1}^{d} e_j \otimes a_j \otimes e_j, \ \sum_{i=1}^{r} \omega_i \mathbb{E}[\zeta_i \otimes \zeta_i \otimes \mu_i] = \sum_{j=1}^{d} e_j \otimes e_j \otimes a_j.$$

Therefore, we can express M_3 in terms of $\omega_i, \mu_i, \Sigma_i$ as

$$M_3 = \sum_{i=1}^{r} \omega_i \mu_i \otimes \mu_i \otimes \mu_i + \sum_{j=1}^{d} \Big(a_j \otimes e_j \otimes e_j + e_j \otimes a_j \otimes e_j + e_j \otimes e_j \otimes a_j \Big). \tag{1.5}$$

We are particularly interested in the following third order symmetric tensor

$$\mathcal{F} := \sum_{i=1}^{r} \omega_i \mu_i \otimes \mu_i \otimes \mu_i. \tag{1.6}$$

When the labels i_1, i_2, i_3 are distinct from each other, we have

$$(M_3)_{i_1 i_2 i_3} = (\mathcal{F})_{i_1 i_2 i_3} \quad \text{for} \quad i_1 \neq i_2 \neq i_3 \neq i_1.$$

Denote the label set

$$\Omega = \{(i_1, i_2, i_3) : i_1 \neq i_2 \neq i_3 \neq i_1,\ i_1, i_2, i_3 \text{ are labels for } M_3\}. \tag{1.7}$$

The tensor M_3 can be estimated from the samplings for y, so the entries $\mathcal{F}_{i_1 i_2 i_3}$ with $(i_1, i_2, i_3) \in \Omega$ can also be obtained from the estimation of M_3. To recover the parameters ω_i, μ_i, we first find the tensor decomposition for $\mathcal{F}$, from the partially given entries $\mathcal{F}_{i_1 i_2 i_3}$ with $(i_1, i_2, i_3) \in \Omega$. Once the parameters ω_i, μ_i are known, we can determine Σ_i from the expressions of a_j as in (1.3.1).

1.3.2 Higher Order Moment Structure

The higher-order moments can be expressed by means and covariance matrices as in the work [26]. Let $z = (z_1, \cdots, z_t)$ be a multivariate Gaussian random vector with mean μ and covariance Σ, then

$$\mathbb{E}[z_1 \cdots z_t] = \sum_{\substack{\lambda \in P_t \\ \lambda = \lambda_p \cup \lambda_s}} \prod_{(u,v) \in \lambda_p} \Sigma_{u,v} \prod_{c \in \lambda_s} \mu_c, \tag{1.8}$$

where P_t contains all distinct ways of partitioning $z_1, \cdots, z_t$ into two parts, one part λ_p represents p pairs of (u, v), and another part λ_s consists of s singletons of (c), where $p \geq 0$, $s \geq 0$ and $2p + s = t$.

We denote the label set Ω_m for mth order tensor M_m

$$\Omega_m = \{(i_1, \ldots, i_m) : i_1, \ldots, i_m \text{ are distinct from each other}\}. \tag{1.9}$$

Let $z_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ be the random vector for the ith component of the diagonal Gaussian mixture model. For $(i_1, \ldots, i_m) \in \Omega_m$, the expression (1.8) implies that

$$\begin{aligned}
(M_m)_{i_1 \ldots i_m} &= \sum_{i=1}^{r} \omega_i \left(\mathbb{E}[z_i^{\otimes m}] \right)_{i_1 \ldots i_m} \\
&= \sum_{i=1}^{r} \omega_i \mathbb{E}[(z_i)_{i_1} \cdots (z_i)_{i_m}] \\
&= \sum_{i=1}^{r} \omega_i \sum_{\substack{\lambda \in P_m \\ \lambda = \lambda_p \cup \lambda_s}} \prod_{(u,v) \in \lambda_p} (\Sigma_i)_{u,v} \prod_{c \in \lambda_s} (\mu_i)_c \\
&= \sum_{i=1}^{r} \omega_i (\mu_i)_{i_1} \cdots (\mu_i)_{i_m} \\
&= (\mathcal{F}_m)_{i_1 \ldots i_m},
\end{aligned}$$

where P_m contains all distinct ways of partitioning $\{i_1, \ldots, i_m\}$ into two parts and λ_p, λ_s are similarly defined as in (1.8). When $\lambda_p \neq \emptyset$, we have $(\Sigma_i)_{u,v} = 0$ for diagonal covariance matrices. Thus, we only need to consider $\lambda_p = \emptyset$ and $\lambda_s = \{i_1, \ldots, i_m\}$. It demonstrates the above equations. We conclude that the moment tensors for diagonal Gaussian mixtures satisfy

$$(M_m)_{i_1 \dots i_m} = (\mathcal{F}_m)_{i_1 \dots i_m} \tag{1.10}$$

where $\mathcal{F}_m = \sum_{i=1}^{r} \omega_i \mu_i^{\otimes m}$ and $(i_1, \dots, i_m) \in \Omega_m$.

Chapter 2

Learning Diagonal Gaussian Mixture Models Using Third Order Moment

2.1 Incomplete Tensor Decomposition

The moment structure of the third order moment for a diagonal Gaussian mixture model observed in (1.5) leads to the incomplete tensor decomposition problem. For a third order symmetric tensor $\mathcal{F}$ whose partial entries $\mathcal{F}_{i_1 i_2 i_3}$ with $(i_1, i_2, i_3) \in \Omega$ are known, we are looking for vectors $p_1, \dots, p_r$ such that

$$\mathcal{F}_{i_1 i_2 i_3} = \left(p_1^{\otimes 3} + \cdots + p_r^{\otimes 3}\right)_{i_1 i_2 i_3}, \quad \text{for all} \quad (i_1, i_2, i_3) \in \Omega. \tag{2.1}$$

The above is called an incomplete tensor decomposition for $\mathcal{F}$.

This section discusses how to compute an incomplete tensor decomposition for a symmetric tensor $\mathcal{F} \in \mathrm{S}^3(\mathbb{C}^d)$ when only its subtensor $\mathcal{F}_\Omega$ is given, for the label set Ω in (1.7). For convenience of notation, the labels for $\mathcal{F}$ begin with zeros while a vector $u \in \mathbb{C}^d$ is still labelled as $u := (u_1, \dots, u_d)$. We set

$$n := d - 1, \quad x = (x_1, \dots, x_n), \quad x_0 := 1.$$

For a given rank r, denote the monomial sets

$$\mathscr{B}_0 \coloneqq \{x_1, \cdots, x_r\}, \quad \mathscr{B}_1 = \{x_i x_j : i \in [r],\ j \in [r+1, n]\}. \tag{2.2}$$

For a monomial power $\alpha \in \mathbb{N}^n$, by writing $\alpha \in \mathscr{B}_1$, we mean that $x^\alpha \in \mathscr{B}_1$. For each $\alpha \in \mathscr{B}_1$, one can write $\alpha = e_i + e_j$ with $i \in [r],\ j \in [r+1, n]$. Let $\mathbb{C}^{[r] \times \mathscr{B}_1}$ denote the space of matrices labelled by the pair $(k, \alpha) \in [r] \times \mathscr{B}_1$. For each $\alpha = e_i + e_j \in \mathscr{B}_1$ and $G \in \mathbb{C}^{[r] \times \mathscr{B}_1}$, denote the quadratic polynomial in x

$$\varphi_{ij}[G](x) \coloneqq \sum_{k=1}^{r} G(k, e_i + e_j) x_k - x_i x_j. \tag{2.3}$$

Suppose r is the symmetric rank of $\mathcal{F}$. A matrix $G \in \mathbb{C}^{[r] \times \mathscr{B}_1}$ is called a *generating matrix* of $\mathcal{F}$ if each $\varphi_{ij}[G](x)$, with $\alpha = e_i + e_j \in \mathscr{B}_1$, is a generating polynomial of $\mathcal{F}$. Equivalently, G is a generating matrix of $\mathcal{F}$ if and only if

$$\langle x_t \varphi_{ij}[G](x), \mathcal{F} \rangle = \sum_{k=1}^{r} G(k, e_i + e_j) \mathcal{F}_{0kt} - \mathcal{F}_{ijt} \ = \ 0, \quad t = 0, 1, \ldots, n, \tag{2.4}$$

for all $i \in [r],\ j \in [r+1, n]$. The notion *generating matrix* is motivated from that the entire tensor $\mathcal{F}$ can be recursively determined by G and its first r entries (see [43]). The existence and uniqueness of the generating matrix G is shown as follows.

Theorem 2.1.1. *Suppose $\mathcal{F}$ has the decomposition*

$$\mathcal{F} = \lambda_1 \begin{bmatrix} 1 \\ u_1 \end{bmatrix}^{\otimes 3} + \cdots + \lambda_r \begin{bmatrix} 1 \\ u_r \end{bmatrix}^{\otimes 3}, \tag{2.5}$$

for vectors $u_i \in \mathbb{C}^n$ and scalars $0 \neq \lambda_i \in \mathbb{C}$. If the subvectors $(u_1)_{1:r}, \ldots, (u_r)_{1:r}$ are linearly independent, then there exists a unique generating matrix $G \in \mathbb{C}^{[r] \times \mathscr{B}_1}$ satisfying (2.4) *for the tensor $\mathcal{F}$.*

Proof. We first prove the existence. For each $i = 1, \dots, r$, denote the vectors $v_i = (u_i)_{1:r}$. Under the given assumption, $V := [v_1 \, \dots \, v_r]$ is an invertible matrix. For each $l = r+1, \dots, n$, let

$$N_l := V \cdot \operatorname{diag}\big((u_1)_l, \dots, (u_r)_l\big) \cdot V^{-1}. \tag{2.6}$$

Then $N_l v_i = (u_i)_l v_i$ for $i = 1, \dots, r$, i.e., N_l has eigenvalues $(u_1)_l, \dots, (u_r)_l$ with corresponding eigenvectors $(u_1)_{1:r}, \dots, (u_r)_{1:r}$. We select $G \in \mathbb{C}^{[r] \times \mathscr{B}_1}$ to be the matrix such that

$$N_l = \begin{bmatrix} G(1, e_1 + e_l) & \cdots & G(r, e_1 + e_l) \\ \vdots & \ddots & \vdots \\ G(1, e_r + e_l) & \cdots & G(r, e_r + e_l) \end{bmatrix}, \, l = r+1, \dots, n. \tag{2.7}$$

For each $s = 1, \dots, r$ and $\alpha = e_i + e_j \in \mathbb{B}_1$ with $i \in [r], \; j \in [r+1, n]$,

$$\varphi_{ij}[G](u_s) = \sum_{k=1}^{r} G(k, e_i + e_j)(u_s)_k - (u_s)_i (u_s)_j \; = \; 0.$$

For each $t = 1, \dots, n$, it holds that

$$\begin{aligned}
\langle x_t \varphi_{ij}[G](x), \mathcal{F} \rangle &= \left\langle \sum_{k=1}^{r} G(k, e_i + e_j) x_t x_k - x_t x_i x_j, \mathcal{F} \right\rangle \\
&= \left\langle \sum_{k=1}^{r} G(k, e_i + e_j) x_t x_k - x_t x_i x_j, \sum_{s=1}^{r} \lambda_s \begin{bmatrix} 1 \\ u_s \end{bmatrix}^{\otimes 3} \right\rangle \\
&= \sum_{k=1}^{r} G(k, e_i + e_j) \sum_{s=1}^{r} \lambda_s (u_s)_t (u_s)_k - \sum_{s=1}^{r} \lambda_s (u_s)_t (u_s)_i (u_s)_j \\
&= \sum_{s=1}^{r} \lambda_s (u_s)_t \left(\sum_{k=1}^{r} G(k, e_i + e_j)(u_s)_k - (u_s)_i (u_s)_j \right) \\
&= 0.
\end{aligned}$$

When $t = 0$, we can similarly get

$$\begin{aligned}\langle \varphi_{ij}[G](x), \mathcal{F}\rangle &= \left\langle \sum_{k=1}^{r} G(k, e_i + e_j)x_k - x_i x_j, \mathcal{F} \right\rangle \\ &= \sum_{s=1}^{r} \lambda_s \left(\sum_{k=1}^{r} G(k, e_i + e_j)(u_s)_k - (u_s)_i (u_s)_j \right) \\ &= 0.\end{aligned}$$

Therefore, the matrix G satisfies (2.4) and it is a generating matrix for $\mathcal{F}$.

Second, we prove the uniqueness of such G. For each $\alpha = e_i + e_j \in \mathscr{B}_1$, let

$$F := \begin{bmatrix} \mathcal{F}_{011} & \cdots & \mathcal{F}_{0r1} \\ \vdots & \ddots & \vdots \\ \mathcal{F}_{01n} & \cdots & \mathcal{F}_{0rn} \end{bmatrix}, \ g_{ij} := \begin{bmatrix} \mathcal{F}_{1ij} \\ \vdots \\ \mathcal{F}_{nij} \end{bmatrix}.$$

Since G satisfies (2.4), we have $F \cdot G(:, e_i + e_j) = g_{ij}$. The decomposition (2.5) implies that

$$F = \begin{bmatrix} u_1 & \cdots & u_r \end{bmatrix} \cdot \operatorname{diag}(\lambda_1, \ldots, \lambda_r) \cdot \begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix}^T.$$

The sets $\{v_1, \ldots, v_r\}$ and $\{u_1, \ldots, u_r\}$ are both linearly independent. Since each $\lambda_i \neq 0$, the matrix F has full column rank. Hence, the generating matrix G satisfying $F \cdot G(:, e_i + e_j) = g_{ij}$ for all $i \in [r], j \in [r+1, n]$ is unique. □

The following is an example of generating matrices.

Example 2.1.2. *Consider the tensor $\mathcal{F} \in \mathrm{S}^3(\mathbb{C}^6)$ that is given as*

$$\mathcal{F} = 0.4 \cdot (1, 1, 1, 1, 1, 1)^{\otimes 3} + 0.6 \cdot (1, -1, 2, -1, 2, 3)^{\otimes 3}.$$

The rank $r = 2$, $\mathscr{B}_0 = \{x_1, x_2\}$ and $\mathscr{B}_1 = \{x_1x_3, x_1x_4, x_1x_5, x_2x_3, x_2x_4, x_2x_5\}$. We have *the vectors*

$$u_1 = (1,1,1,1,1), \quad u_2 = (-1,2,-1,2,3), \quad v_1 = (1,1), \quad v_2 = (-1,2).$$

The matrices N_3, N_4, N_5 *as in (2.6) are*

$$N_3 = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} 1/3 & 2/3 \\ 4/3 & -1/3 \end{bmatrix},$$
$$N_4 = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} 4/3 & -1/3 \\ -2/3 & 5/3 \end{bmatrix},$$
$$N_5 = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} 5/3 & -2/3 \\ -4/3 & 7/3 \end{bmatrix}.$$

The entries of the generating matrix G *are listed as below:*

$k\backslash(i,j)$	$(1,3)$	$(1,4)$	$(1,5)$	$(2,3)$	$(2,4)$	$(2,5)$
1	$1/3$	$4/3$	$5/3$	$4/3$	$-2/3$	$-4/3$
2	$2/3$	$-1/3$	$-2/3$	$-1/3$	$5/3$	$7/3$

(2.8)

The generating polynomials in (2.3) are

$$\varphi_{13}[G](x) = \frac{1}{3}x_1 + \frac{2}{3}x_2 - x_1x_3, \quad \varphi_{23}[G](x) = \frac{4}{3}x_1 - \frac{1}{3}x_2 - x_2x_3,$$
$$\varphi_{14}[G](x) = \frac{4}{3}x_1 - \frac{1}{3}x_2 - x_1x_4, \quad \varphi_{24}[G](x) = -\frac{2}{3}x_1 + \frac{5}{3}x_2 - x_2x_4,$$
$$\varphi_{15}[G](x) = \frac{5}{3}x_1 - \frac{2}{3}x_2 - x_1x_5, \quad \varphi_{25}[G](x) = -\frac{4}{3}x_1 + \frac{7}{3}x_2 - x_2x_5.$$

Above generating polynomials can be written in the following form

$$\begin{bmatrix} \varphi_{1j}[G](x) \\ \varphi_{2j}[G](x) \end{bmatrix} = N_j \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - x_j \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \textit{ for } j = 3, 4, 5.$$

For x to be a common zero of $\varphi_{1j}[G](x)$ and $\varphi_{2j}[G](x)$, it requires that (x_1, x_2) is an eigenvector of N_j with the corresponding eigenvalue x_j.

We show how to find an incomplete tensor decomposition (2.5) for $\mathcal{F}$ when only its subtensor $\mathcal{F}_\Omega$ is given, where the label set Ω is as in (1.7). Suppose that there exists the decomposition (2.5) for $\mathcal{F}$, for vectors $u_i \in \mathbb{C}^n$ and nonzero scalars $\lambda_i \in \mathbb{C}$. Assume the subvectors $(u_1)_{1:r}, \ldots, (u_r)_{1:r}$ are linearly independent, so there is a unique generating matrix G for $\mathcal{F}$, by Theorem 2.1.1.

For each $\alpha = e_i + e_j \in \mathscr{B}_1$ with $i \in [r], j \in [r+1, n]$ and for each

$$l = r+1, \ldots, j-1, j+1, \ldots, n,$$

the generating matrix G satisfies the equations

$$\left\langle x_l \left(\sum_{k=1}^{r} G(k, e_i + e_j) x_k - x_i x_j \right), \mathcal{F} \right\rangle = \sum_{k=1}^{r} G(k, e_i + e_j) \mathcal{F}_{0kl} - \mathcal{F}_{ijl} = 0. \tag{2.9}$$

Let the matrix $A_{ij}[\mathcal{F}] \in \mathbb{C}^{(n-r-1)\times r}$ and the vector $b_{ij}[\mathcal{F}] \in \mathbb{C}^{n-r-1}$ be such that

$$A_{ij}[\mathcal{F}] := \begin{bmatrix} \mathcal{F}_{0,1,r+1} & \cdots & \mathcal{F}_{0,r,r+1} \\ \vdots & \ddots & \vdots \\ \mathcal{F}_{0,1,j-1} & \cdots & \mathcal{F}_{0,r,j-1} \\ \mathcal{F}_{0,1,j+1} & \cdots & \mathcal{F}_{0,r,j+1} \\ \vdots & \ddots & \vdots \\ \mathcal{F}_{0,1,n} & \cdots & \mathcal{F}_{0,r,n} \end{bmatrix}, \quad b_{ij}[\mathcal{F}] := \begin{bmatrix} \mathcal{F}_{i,j,r+1} \\ \vdots \\ \mathcal{F}_{i,j,j-1} \\ \mathcal{F}_{i,j,j+1} \\ \vdots \\ \mathcal{F}_{i,j,n} \end{bmatrix}. \tag{2.10}$$

To distinguish changes in the labels of tensor entries of $\mathcal{F}$, the commas are inserted to separate labeling numbers.

The equations in (2.9) can be equivalently written as

$$A_{ij}[\mathcal{F}] \cdot G(:, e_i + e_j) \;=\; b_{ij}[\mathcal{F}]. \tag{2.11}$$

If the rank $r \le \frac{d}{2} - 1$, then $n - r - 1 = d - r - 2 \ge r$. Thus, the number of rows is not less than the number of columns for matrices $A_{ij}[\mathcal{F}]$. If $A_{ij}[\mathcal{F}]$ has linearly independent columns, then (2.11) uniquely determines $G(:, \alpha)$. For such a case, the matrix G can be fully determined by the linear system (2.11). Let $N_{r+1}(G), \ldots, N_m(G) \in \mathbb{C}^{r \times r}$ be the matrices given as

$$N_l(G) = \begin{bmatrix} G(1, e_1 + e_l) & \cdots & G(r, e_1 + e_l) \\ \vdots & \ddots & \vdots \\ G(1, e_r + e_l) & \cdots & G(r, e_r + e_l) \end{bmatrix}, \, l = r+1, \ldots, n. \tag{2.12}$$

As in the proof of Theorem 2.1.1, one can see that

$$N_l(G) \begin{bmatrix} (u_i)_1 \\ \vdots \\ (u_i)_r \end{bmatrix} = (u_i)_l \cdot \begin{bmatrix} (u_i)_1 \\ \vdots \\ (u_i)_r \end{bmatrix}, \, l = r+1, \ldots, n. \tag{2.13}$$

The above is equivalent to the equations

$$N_l(G) v_i \;=\; (w_i)_{l-r} \cdot v_i, \quad l = r+1, \ldots, n,$$

for the vectors $(i = 1, \ldots, r)$

$$v_i \;:=\; (u_i)_{1:r}, \quad w_i \;:=\; (u_i)_{r+1:n}. \tag{2.14}$$

Each v_i is a common eigenvector of the matrices $N_{r+1}(G), \ldots, N_n(G)$ and $(w_i)_{l-r}$ is the associated eigenvalue of $N_l(G)$. These matrices may or may not have repeated eigenvalues. Therefore, we select a generic vector $\xi := (\xi_{r+1}, \cdots, \xi_n)$ and let

$$N(\xi) := \xi_{r+1} N_{r+1} + \cdots + \xi_n N_n. \tag{2.15}$$

The eigenvalues of $N(\xi)$ are $\xi^T w_1, \ldots, \xi^T w_r$. When $w_1, \ldots, w_r$ are distinct from each other and ξ is generic, the matrix $N(\xi)$ does not have a repeated eigenvalue and hence it has unique eigenvectors $v_1, \ldots, v_r$, up to scaling. Let $\tilde{v}_1, \ldots, \tilde{v}_r$ be unit length eigenvectors of $N(\xi)$. They are also common eigenvectors of $N_{r+1}(G), \ldots, N_n(G)$. For each $i = 1, \ldots, r$, let $\tilde{w}_i$ be the vector such that its jth entry $(\tilde{w}_i)_j$ is the eigenvalue of $N_{j+r}(G)$, associated to the eigenvector $\tilde{v}_i$, or equivalently,

$$\tilde{w}_i = (\tilde{v}_i^H N_{r+1}(G)\tilde{v}_i, \cdots, \tilde{v}_i^H N_n(G)\tilde{v}_i) \quad i = 1, \ldots, r. \tag{2.16}$$

Up to a permutation of $(\tilde{v}_1, \ldots, \tilde{v}_r)$, there exist scalars γ_i such that

$$v_i = \gamma_i \tilde{v}_i, \quad w_i = \tilde{w}_i. \tag{2.17}$$

The tensor decomposition of $\mathcal{F}$ can also be written as

$$\mathcal{F} = \lambda_1 \begin{bmatrix} 1 \\ \gamma_1 \tilde{v}_1 \\ \tilde{w}_1 \end{bmatrix}^{\otimes 3} + \cdots + \lambda_r \begin{bmatrix} 1 \\ \gamma_r \tilde{v}_r \\ \tilde{w}_r \end{bmatrix}^{\otimes 3}.$$

The scalars $\lambda_1, \cdots, \lambda_r$ and $\gamma_1, \cdots, \gamma_r$ satisfy the linear equations

$$\begin{aligned} \lambda_1 \gamma_1 \tilde{v}_1 \otimes \tilde{w}_1 + \cdots + \lambda_r \gamma_r \tilde{v}_r \otimes \tilde{w}_r &= \mathcal{F}_{[0,1:r,r+1:n]}, \\ \lambda_1 \gamma_1^2 \tilde{v}_1 \otimes \tilde{v}_1 \otimes \tilde{w}_1 + \cdots + \lambda_r \gamma_r^2 \tilde{v}_r \otimes \tilde{v}_r \otimes \tilde{w}_r &= \mathcal{F}_{[1:r,1:r,r+1:n]}. \end{aligned}$$

Denote the label sets

$$\boxed{\begin{array}{l} J_1 := \{(0, i_1, i_2) : i_1 \in [r],\, i_2 \in [r+1, n]\}, \\ J_2 := \{(i_1, i_2, i_3) : i_1 \neq i_2,\, i_1, i_2 \in [r], i_3 \in [r+1, n]\}. \end{array}} \tag{2.18}$$

To determine the scalars λ_i, γ_i, we can solve the linear least squares

$$\min_{(\beta_1,\ldots,\beta_r)} \left\| \mathcal{F}_{J_1} - \sum_{i=1}^{r} \beta_i \cdot \tilde{v}_i \otimes \tilde{w}_i \right\|^2, \tag{2.19}$$

$$\min_{(\theta_1,\ldots,\theta_r)} \left\| \mathcal{F}_{J_2} - \sum_{k=1}^{r} \theta_k \cdot (\tilde{v}_k \otimes \tilde{v}_k \otimes \tilde{w}_i)_{J_2} \right\|^2. \tag{2.20}$$

Let $(\beta_1^*, \ldots, \beta_r^*)$, $(\theta_1^*, \ldots, \theta_r^*)$ be minimizers of (2.19) and (2.20) respectively. Then, for each $i = 1, \ldots, r$, let

$$\lambda_i := (\beta_i^*)^2/\theta_i^*, \quad \gamma_i := \theta_i^*/\beta_i^*. \tag{2.21}$$

For the vectors $(i = 1, \ldots, r)$

$$p_i := \sqrt[3]{\lambda_i}(1, \gamma_i \tilde{v}_i, \tilde{w}_i),$$

the sum $p_1^{\otimes 3} + \cdots + p_r^{\otimes 3}$ is a tensor decomposition for $\mathcal{F}$. This is justified in the following theorem.

Theorem 2.1.3. *Suppose the tensor $\mathcal{F}$ has the decomposition as in (2.5). Assume that the vectors $v_1, \ldots, v_r$ are linearly independent and the vectors $w_1, \ldots, w_r$ are distinct from each other, where $v_1, \ldots, v_r, w_1, \ldots, w_r$ are defined as in (2.14). Let ξ be a generically chosen coefficient vector and let $p_1, \ldots, p_r$ be the vectors produced as above. Then, the tensor decomposition $\mathcal{F} = p_1^{\otimes 3} + \cdots + p_r^{\otimes 3}$ is unique.*

Proof. Since $v_1, \ldots, v_r$ are linearly independent, the tensor decomposition (2.5) is unique, up to scalings and permutations. By Theorem 2.1.1, there is a unique generating matrix G for $\mathcal{F}$ satisfying (2.4). Under the given assumptions, the equation (2.11) uniquely determines G. Note that $\xi^T w_1, \dots, \xi^T w_r$ are the eigenvalues of $N(\xi)$ and $v_1, \dots, v_r$ are the corresponding eigenvectors. When ξ is generically chosen, the values of $\xi^T w_1, \dots, \xi^T w_r$ are distinct eigenvalues of $N(\xi)$. So $N(\xi)$ has unique eigenvalue decompositions, and hence (2.17) must hold, up to a permutation of $(v_1, \dots, v_r)$. Since the coefficient matrices have full column ranks, the linear least squares problems have unique optimal solutions. Up to a permutation of $p_1, \dots, p_r$, it holds that $p_i = \sqrt[3]{\lambda_i} \begin{bmatrix} 1 \\ u_i \end{bmatrix}$. Then, the conclusion follows readily. □

The following is the algorithm for computing an incomplete tensor decomposition for $\mathcal{F}$ when only its subtensor $\mathcal{F}_\Omega$ is given.

Algorithm 2.1.4. *(Incomplete symmetric tensor decompositions.)*

Input: A third order symmetric subtensor $\mathcal{F}_\Omega$ and a rank $r = rank_S(\mathcal{F}) \leq \frac{d}{2} - 1$.

1. *Determine the matrix G by solving (2.11) for each $\alpha = e_i + e_j \in \mathbb{B}_1$.*

2. *Let $N(\xi)$ be the matrix as in (2.15), for a randomly selected vector ξ. Compute the unit length eigenvectors $\tilde{v}_1, \dots, \tilde{v}_r$ of $N(\xi)$ and choose $\tilde{w}_i$ as in (2.16).*

3. *Solve the linear least squares (2.19) and (2.20) to get the coefficients λ_i, γ_i as in (2.21).*

4. *For each $i = 1, \dots, r$, let $p_i := \sqrt[3]{\lambda_i}(1, \gamma_i \tilde{v}_i, \tilde{w}_i)$.*

Output: The tensor decomposition $\mathcal{F} = (p_1)^{\otimes 3} + \cdots + (p_r)^{\otimes 3}$.

The following is an example of applying Algorithm 2.1.4.

Example 2.1.5. *Consider the same tensor $\mathcal{F}$ as in Example 2.1.2. The monomial sets*

$\mathscr{B}_0$, $\mathscr{B}_1$ are the same. The matrices $A_{ij}[\mathcal{F}]$ and vectors $b_{ij}[\mathcal{F}]$ are

$$A_{13}[\mathcal{F}] = A_{23}[\mathcal{F}] = \begin{bmatrix} -0.8 & 2.8 \\ -1.4 & 4 \end{bmatrix}, \quad b_{13}[\mathcal{F}] = \begin{bmatrix} 1.6 \\ 2.2 \end{bmatrix}, b_{23}[\mathcal{F}] = \begin{bmatrix} -2 \\ -3.2 \end{bmatrix},$$

$$A_{14}[\mathcal{F}] = A_{24}[\mathcal{F}] = \begin{bmatrix} 1 & -0.8 \\ -1.4 & 4 \end{bmatrix}, \quad b_{14}[\mathcal{F}] = \begin{bmatrix} 1.6 \\ -3.2 \end{bmatrix}, b_{24}[\mathcal{F}] = \begin{bmatrix} -2 \\ 7.6 \end{bmatrix},$$

$$A_{15}[\mathcal{F}] = A_{25}[\mathcal{F}] = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 2.8 \end{bmatrix}, \quad b_{15}[\mathcal{F}] = \begin{bmatrix} 2.2 \\ -3.2 \end{bmatrix}, b_{25}[\mathcal{F}] = \begin{bmatrix} -3.2 \\ 7.6 \end{bmatrix}.$$

Solve (2.11) to obtain G, which is same as in (2.8). The matrices $N_3(G), N_4(G), N_5(G)$ are

$$N_3(G) = \begin{bmatrix} 1/3 & 2/3 \\ 4/3 & -1/3 \end{bmatrix}, \ N_4(G) = \begin{bmatrix} 4/3 & -1/3 \\ -2/3 & 5/3 \end{bmatrix}, \ N_5(G) = \begin{bmatrix} 5/3 & -2/3 \\ -4/3 & 7/3 \end{bmatrix}.$$

Choose a generic ξ, say, $\xi = (3, 4, 5)$, then

$$N(\xi) = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{5} \\ 1/\sqrt{2} & 2/\sqrt{5} \end{bmatrix} \begin{bmatrix} 12 & 0 \\ 0 & 20 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{5} \\ 1/\sqrt{2} & 2/\sqrt{5} \end{bmatrix}^{-1}.$$

The unit length eigenvectors are

$$\tilde{v}_1 = (1/\sqrt{2}, 1/\sqrt{2}), \quad \tilde{v}_2 = (-1/\sqrt{5}, 2/\sqrt{5}).$$

As in (2.16), we get the vectors

$$w_1 = (1, 1, 1), \ w_2 = (-1, 2, 3).$$

Solving (2.19) and (2.20), we get the scalars

$$\gamma_1 = \sqrt{2}, \quad \gamma_2 = \sqrt{5}, \quad \lambda_1 = 0.4, \quad \lambda_2 = 0.6.$$

This produces the decomposition $\mathcal{F} = \lambda_1 u_1^{\otimes 3} + \lambda_2 u_2^{\otimes 3}$ *for the vectors*

$$u_1 = (1, \gamma_1 v_1, w_1) = (1,1,1,1,1,1), \quad u_2 = (1, \gamma_2 v_2, w_2) = (1,-1,2,-1,2,3).$$

Remark 2.1.6. *Algorithm 2.1.4 requires the value of* r*. This is generally a hard question. In computational practice, one can estimate the value of* r *as follows. Let* $Flat(\mathcal{F}) \in \mathbb{C}^{(n+1)\times(n+1)^2}$ *be the flattening matrix, labelled by* $(i,(j,k))$ *such that*

$$Flat(\mathcal{F})_{i,(j,k)} \; = \; \mathcal{F}_{ijk}$$

for all $i,j,k = 0,1,\ldots,n$*. The rank of* $Flat(\mathcal{F})$ *equals the rank of* $\mathcal{F}$ *when the vectors* $p_1,\ldots,p_r$ *are linearly independent. The rank of* $Flat(\mathcal{F})$ *is not available since only the subtensor* $(\mathcal{F})_\Omega$ *is known. However, we can calculate the ranks of submatrices of* $(\mathcal{F})_\Omega$ *whose entries are known. If the tensor* $\mathcal{F}$ *as in (2.5) is such that both the sets* $\{v_1,\ldots,v_r\}$ *and* $\{w_1,\ldots,w_r\}$ *are linearly independent, one can see that* $\sum_{i=1}^r \lambda_i v_i w_i^T$ *is a known submatrix of* $Flat(\mathcal{F})$ *whose rank is* r*. This is generally the case if* $r \leq \frac{d}{2} - 1$*, since* v_i *has the length* r *and* w_i *has length* $d - 1 - r \geq r$*. Therefore, the known submatrices of* $Flat(\mathcal{F})$ *are generally sufficient to estimate* $rank_S(\mathcal{F})$*. For instance, we consider the case* $\mathcal{F} \in S^3(\mathbb{C}^7)$*.*

The flattening matrix Flat($\mathcal{F}$) is

$$\begin{bmatrix} * & * & * & * & * & * & * \\ * & * & \mathcal{F}_{120} & \mathcal{F}_{130} & \mathcal{F}_{140} & \mathcal{F}_{150} & \mathcal{F}_{160} \\ * & \mathcal{F}_{210} & * & \mathcal{F}_{230} & \mathcal{F}_{240} & \mathcal{F}_{250} & \mathcal{F}_{260} \\ * & \mathcal{F}_{310} & \mathcal{F}_{320} & * & \mathcal{F}_{340} & \mathcal{F}_{350} & \mathcal{F}_{360} \\ * & \mathcal{F}_{410} & \mathcal{F}_{420} & \mathcal{F}_{430} & * & \mathcal{F}_{450} & \mathcal{F}_{460} \\ * & \mathcal{F}_{510} & \mathcal{F}_{520} & \mathcal{F}_{530} & \mathcal{F}_{540} & * & \mathcal{F}_{560} \\ * & \mathcal{F}_{610} & \mathcal{F}_{620} & \mathcal{F}_{630} & \mathcal{F}_{640} & \mathcal{F}_{650} & * \end{bmatrix}, \tag{2.22}$$

where each $*$ *means that entry is not given. The largest submatrices with known entries are*

$$\begin{bmatrix} \mathcal{F}_{410} & \mathcal{F}_{420} & \mathcal{F}_{430} \\ \mathcal{F}_{510} & \mathcal{F}_{520} & \mathcal{F}_{530} \\ \mathcal{F}_{610} & \mathcal{F}_{620} & \mathcal{F}_{630} \end{bmatrix}, \quad \begin{bmatrix} \mathcal{F}_{140} & \mathcal{F}_{150} & \mathcal{F}_{160} \\ \mathcal{F}_{240} & \mathcal{F}_{250} & \mathcal{F}_{260} \\ \mathcal{F}_{340} & \mathcal{F}_{350} & \mathcal{F}_{360} \end{bmatrix}.$$

The rank of above matrices generally equals $\text{rank}_S(\mathcal{F})$ *if* $r \leq \frac{d}{2} - 1 = 2.5$.

2.2 Tensor Approximations and Stability Analysis

In some applications, we do not have the subtensor $\mathcal{F}_\Omega$ exactly but only have an approximation $\widehat{\mathcal{F}}_\Omega$ for it. The Algorithm 2.1.4 can still provide a good rank-r approximation for $\mathcal{F}$ when it is applied to $\widehat{\mathcal{F}}_\Omega$. We define the matrix $A_{ij}[\widehat{\mathcal{F}}]$ and the vector $b_{ij}[\widehat{\mathcal{F}}]$ in the same way as in (2.10), for each $\alpha = e_i + e_j \in \mathscr{B}_1$. The generating matrix G for $\mathcal{F}$ can be approximated by solving the linear least squares

$$\min_{g \in \mathbb{C}^r} \quad \|A_{ij}[\widehat{\mathcal{F}}] \cdot g - b_{ij}[\widehat{\mathcal{F}}]\|^2, \tag{2.23}$$

for each $\alpha = e_i + e_j \in \mathbb{B}_1$. Let $\widehat{G}(:, e_i + e_j)$ be the optimizer of the above and $\widehat{G}$ be the matrix consisting of all such $\widehat{G}(:, e_i + e_j)$. Then $\widehat{G}$ is an approximation for G. For each $l = r+1, \ldots, n$, define the matrix $N_l(\widehat{G})$ similarly as in (2.12). Choose a generic vector $\xi = (\xi_{r+1}, \ldots, \xi_n)$ and let

$$\widehat{N}(\xi) := \xi_{r+1} N_{r+1}(\widehat{G}) + \cdots + \xi_n N_n(\widehat{G}). \tag{2.24}$$

The matrix $\widehat{N}(\xi)$ is an approximation for $N(\xi)$. Let $\hat{v}_1, \ldots, \hat{v}_r$ be unit length eigenvectors of $\widehat{N}(\xi)$. For $k = 1, \ldots, r$, let

$$\hat{w}_k := \big((\hat{v}_k)^H N_{r+1}(\widehat{G}) \hat{v}_k, \ldots, (\hat{v}_k)^H N_n(\widehat{G}) \hat{v}_k\big). \tag{2.25}$$

For the label sets J_1, J_2 as in (2.18), the subtensors $\widehat{\mathcal{F}}_{J_1}, \widehat{\mathcal{F}}_{J_2}$ are similarly defined like $\mathcal{F}_{J_1}, \mathcal{F}_{J_2}$. Consider the following linear least square problems

$$\min_{(\beta_1, \ldots, \beta_r)} \left\| \widehat{\mathcal{F}}_{J_1} - \sum_{k=1}^{r} \beta_k \cdot \hat{v}_k \otimes \hat{w}_k \right\|^2, \tag{2.26}$$

$$\min_{(\theta_1, \ldots, \theta_r)} \left\| \widehat{\mathcal{F}}_{J_2} - \sum_{k=1}^{r} \theta_i \cdot (\hat{v}_k \otimes \hat{v}_k \otimes \hat{w}_k)_{J_2} \right\|^2. \tag{2.27}$$

Let $(\hat{\beta}_1, \ldots, \hat{\beta}_r)$ and $(\hat{\theta}_1, \ldots, \hat{\theta}_r)$ be their optimizers respectively. For each $k = 1, \ldots, r$, let

$$\hat{\lambda}_k := (\hat{\beta}_k)^2 / \hat{\theta}_k, \quad \hat{\gamma}_k := \hat{\theta}_k / \hat{\beta}_k. \tag{2.28}$$

This results in the tensor approximation

$$\mathcal{F} \approx (\hat{p}_1)^{\otimes 3} + \cdots + (\hat{p}_r)^{\otimes 3},$$

for the vectors $\hat{p}_k := \sqrt[3]{\hat{\lambda}_k}(1, \hat{\gamma}_k \hat{v}_k, \hat{w}_k)$. The above may not give an optimal tensor approximation. To get an improved one, we can use $\hat{p}_1, \dots, \hat{p}_r$ as starting points to solve the following nonlinear optimization

$$\min_{(q_1,\dots,q_r)} \left\| \left(\sum_{k=1}^{r} (q_k)^{\otimes 3} - \widehat{\mathcal{F}} \right)_{\Omega} \right\|^2. \tag{2.29}$$

The minimizer of the optimization (2.29) is denoted as $(p_1^*, \dots, p_r^*)$.

Summarizing the above, we have the following algorithm for computing a tensor approximation.

Algorithm 2.2.1. *(Incomplete symmetric tensor approximations.)*

Input: A third order symmetric subtensor $\widehat{\mathcal{F}}_\Omega$ *and a rank* $r \le \frac{d}{2} - 1$.

1. *Find the matrix* $\widehat{G}$ *by solving (2.23) for each* $\alpha = e_i + e_j \in \mathbb{B}_1$.
2. *Choose a generic vector and let* $\widehat{N}(\xi)$ *be the matrix as in (2.24).*

   Compute unit length eigenvectors $\hat{v}_1, \dots, \hat{v}_r$ *for* $\widehat{N}(\xi)$ *and define* $\hat{w}_i$ *in (2.25).*
3. *Solve the linear least squares (2.26), (2.27) to get the coefficients* $\hat{\lambda}_i, \hat{\gamma}_i$.
4. *For each* $i = 1, \dots, r$, *let* $\hat{p}_i := \sqrt[3]{\hat{\lambda}_i}(1, \hat{\gamma}_i\hat{v}_i, \hat{w}_i)$. *Then* $(\hat{p}_1)^{\otimes 3} + \cdots + (\hat{p}_r)^{\otimes 3}$ *is a tensor approximation for* $\widehat{\mathcal{F}}$.
5. *Use* $\hat{p}_1, \dots, \hat{p}_r$ *as starting points to solve the nonlinear optimization (2.29) for an optimizer* $(p_1^*, \dots, p_r^*)$.

Output: The tensor approximation $(p_1^*)^{\otimes 3} + \cdots + (p_r^*)^{\otimes 3}$ *for* $\widehat{\mathcal{F}}$.

When $\widehat{\mathcal{F}}$ is close to $\mathcal{F}$, Algorithm 2.2.1 also produces a good rank-r tensor approximation for $\mathcal{F}$. This is shown in the following.

Theorem 2.2.2. *Suppose the tensor* $\mathcal{F} = (p_1)^{\otimes 3} + \cdots + (p_r)^{\otimes 3}$, *with* $r \le \frac{d}{2} - 1$, *satisfies the following conditions:*

(i) The leading entry of each p_i is nonzero;

(ii) the subvectors $(p_1)_{2:r+1}, \ldots, (p_r)_{2:r+1}$ are linearly independent;

(iii) the subvectors $(p_1)_{[r+2:j,j+2:d]}, \ldots, (p_r)_{[r+2:j,j+2:d]}$ are linearly independent for each $j \in [r+1, n]$;

(iv) the eigenvalues of the matrix $N(\xi)$ in (2.15) are distinct from each other.

Let $\hat{p}_i, p_i^*$ be the vectors produced by Algorithm 2.2.1. If the distance $\epsilon := \|(\widehat{\mathcal{F}} - \mathcal{F})_\Omega\|$ is small enough, then there exist scalars $\hat{\tau}_i, \tau_i^*$ such that

$$(\hat{\tau}_i)^3 = (\tau_i^*)^3 = 1, \quad \|\hat{\tau}_i \hat{p}_i - p_i\| = O(\epsilon), \quad \|\tau_i^* p_i^* - p_i\| = O(\epsilon),$$

up to a permutation of $(p_1, \ldots, p_r)$, where the constants inside $O(\cdot)$ only depend on $\mathcal{F}$ and the choice of ξ in Algorithm 2.2.1.

Proof. The conditions (i)-(ii), by Theorem 2.1.1, imply that there is a unique generating matrix G for $\mathcal{F}$. The matrix G can be approximated by solving the linear least square problems (2.23). Note that

$$\|A_{ij}[\widehat{\mathcal{F}}] - A_{ij}[\mathcal{F}]\| \le \epsilon, \quad \|b_{ij}[\widehat{\mathcal{F}}] - b_{ij}[\mathcal{F}]\| \le \epsilon,$$

for all $\alpha = e_i + e_j \in \mathscr{B}_1$. The matrix $A_{ij}[\mathcal{F}]$ can be written as

$$A_{ij}[\mathcal{F}] = [(p_1)_{[r+2:j,j+2:d]}, \ldots, (p_r)_{[r+2:j,j+2:d]}] \cdot [(p_1)_{2:r+1}, \ldots, (p_r)_{2:r+1}]^T.$$

By the conditions (ii)-(iii), the matrix $A_{ij}[\mathcal{F}]$ has full column rank for each $j \in [r+1, n]$ and hence the matrix $A_{ij}[\widehat{\mathcal{F}}]$ has full column rank when ϵ is small enough. Therefore, the linear least problems (2.23) have unique solutions and the solution $\widehat{G}$ satisfies that

$$\|\widehat{G} - G\| = O(\epsilon),$$

where $O(\epsilon)$ depends on $\mathcal{F}$ (see [14, Theorem 3.4]). For each $j = r+1, \ldots, n$, $N_j(\widehat{G})$ is part of the generating matrix $\widehat{G}$, so

$$\|N_j(\widehat{G}) - N_j(G)\| \le \|\widehat{G} - G\| = O(\epsilon), \quad j = r+1, \ldots, n.$$

This implies that $\|\widehat{N}(\xi) - N(\xi)\| = O(\epsilon)$. When ϵ is small enough, the matrix $\widehat{N}(\xi)$ does not have repeated eigenvalues, due to the condition (iv). Thus, the matrix $N(\xi)$ has a set of unit length eigenvectors $\tilde{v}_1, \ldots, \tilde{v}_r$ with eigenvalues $\tilde{w}_1, \ldots, \tilde{w}_r$ respectively, such that

$$\|\hat{v}_i - \tilde{v}_i\| = O(\epsilon), \quad \|\hat{w}_i - \tilde{w}_i\| = O(\epsilon).$$

This follows from Proposition 4.2.1 in [8]. The constants inside the above $O(\cdot)$ depend only on $\mathcal{F}$ and ξ. The $\tilde{w}_1, \ldots, \tilde{w}_r$ are scalar multiples of linearly independent vectors $(p_1)_{r+2:d}, \ldots, (p_r)_{r+2:d}$ respectively, so $\tilde{w}_1, \ldots, \tilde{w}_r$ are linearly independent. When ϵ is small, $\hat{w}_1, \ldots, \hat{w}_r$ are linearly independent as well. The scalars $\hat{\lambda}_i\hat{\gamma}_i$ and $\hat{\lambda}_i(\hat{\gamma}_i)^2$ are optimizers for the linear least square problems (2.26) and (2.27). By Theorem 3.4 in [14], we have

$$\|\hat{\lambda}_i\hat{\gamma}_i - \lambda_i\gamma_i\| = O(\epsilon), \; \|\hat{\lambda}_i(\hat{\gamma}_i)^2 - \lambda_i\gamma_i^2\| = O(\epsilon).$$

The vector p_i can be written as $p_i = \sqrt[3]{\lambda_i}(1, \gamma_i\tilde{v}_i, \tilde{w}_i)$, so we must have $\lambda_i, \gamma_i \neq 0$ due to the condition (ii). Thus, it holds that

$$\|\hat{\lambda}_i - \lambda_i\| = O(\epsilon), \; \|\hat{\gamma}_i - \gamma_i\| = O(\epsilon),$$

where constants inside $O(\cdot)$ depend only on $\mathcal{F}$ and ξ. For the vectors $\tilde{p}_i := \sqrt[3]{\lambda_i}(1, \gamma_i\tilde{v}_i, \tilde{w}_i)$, we have $\mathcal{F} = \sum_{i=1}^r \tilde{p}_i^{\otimes 3}$, by Theorem 2.1.3. Since $p_1, \ldots, p_r$ are linearly independent by the assumption, the rank decomposition of $\mathcal{F}$ is unique up to scaling and permutation. There exist scalars $\hat{\tau}_i$ such that $(\hat{\tau}_i)^3 = 1$ and $\hat{\tau}_i\tilde{p}_i = p_i$, up to a permutation of $p_1, \ldots, p_r$. For $\hat{p}_i = \sqrt[3]{\hat{\lambda}_i}(1, \hat{\gamma}_i\hat{v}_i, \hat{w}_i)$, we have $\|\hat{\tau}_i\hat{p}_i - p_i\| = O(\epsilon)$, where the constants in $O(\cdot)$ only depend on $\mathcal{F}$ and ξ.

Since $\|\hat{\tau}_i\hat{p}_i - p_i\| = O(\epsilon)$, we have $\|(\sum_{i=1}^r (\hat{p}_i)^{\otimes 3} - \mathcal{F})_\Omega\| = O(\epsilon)$. The $(p_1^*, \ldots, p_r^*)$ is a minimizer of (2.29), so

$$\left\| \left(\sum_{i=1}^r (p_i^*)^{\otimes 3} - \widehat{\mathcal{F}} \right)_\Omega \right\| \leq \left\| \left(\sum_{i=1}^r (\hat{p}_i)^{\otimes 3} - \widehat{\mathcal{F}} \right)_\Omega \right\| = O(\epsilon).$$

For the tensor $\mathcal{F}^* := \sum_{i=1}^r (p_i^*)^{\otimes 3}$, we get

$$\|(\mathcal{F}^* - \mathcal{F})_\Omega\| \leq \|(\mathcal{F}^* - \widehat{\mathcal{F}})_\Omega\| + \|(\widehat{\mathcal{F}} - \mathcal{F})_\Omega\| = O(\epsilon).$$

When Algorithm 2.2.1 is applied to $(\mathcal{F}^*)_\Omega$, the Step 4 will give the exact decomposition $\mathcal{F}^* = \sum_{i=1}^r (p_i^*)^{\otimes 3}$. By repeating the previous argument, we can similarly show that $\|p_i - \tau_i^* p_i^*\| = O(\epsilon)$ for some τ_i^* such that $(\tau_i^*)^3 = 1$, where the constants in $O(\cdot)$ only depend on $\mathcal{F}$ and ξ. □

Remark 2.2.3. *For the special case that $\epsilon = 0$, Algorithm 2.2.1 is the same as Algorithm 2.1.4, which produces the exact rank decomposition for $\mathcal{F}$. The conditions in Theorem 2.2.2 are satisfied for generic vectors $p_1, \ldots, p_r$, since $r \leq \frac{d}{2} - 1$. The constant in $O(\cdot)$ is not explicitly given in the proof. It is related to the condition number $\kappa(\mathcal{F})$ for tensor decomposition. It was shown by Breiding and Vannieuwenhoven [5] that*

$$\sqrt{\sum_{i=1}^r \|p_i^{\otimes 3} - \hat{p}_i^{\otimes 3}\|^2} \leq \kappa(\mathcal{F})\|\mathcal{F} - \hat{\mathcal{F}}\| + c\epsilon^2$$

for some constant c. The continuity of $\hat{G}$ in $\hat{\mathcal{F}}$ is implicitly impled by the proof. Eigenvalues and unit eigenvectors of $\widehat{N}(\xi)$ are continuous in $\hat{G}$. Furthermore, $\hat{\lambda}_i, \hat{\gamma}_i$ are continuous in the eigenvalues and unit eigenvectors. All these functions are locally Lipschitz continuous. The $\hat{p}_i$ is Lipschitz continuous with respect to $\hat{\mathcal{F}}$, in a neighborhood of $\mathcal{F}$, which also implies an error bound for $\hat{p}_i$. The tensors $(p_i^*)^{\otimes 3}$ are also locally Lipschitz continuous in $\widehat{\mathcal{F}}$ illustrated by [6]. This also gives error bounds for decomposing vectors p_i^*. We refer to [5, 6] for more details about condition numbers of tensor decompositions.

Example 2.2.4. *We consider the same tensor $\mathcal{F}$ as in Example 2.1.2. The subtensor $(\mathcal{F})_\Omega$ is perturbed to $(\widehat{\mathcal{F}})_\Omega$. The perturbation is randomly generated from the Gaussian distribution $\mathcal{N}(0, 0.01)$. For neatness of the paper, we do not display $(\widehat{\mathcal{F}})_\Omega$ here. We use Algorithm 2.2.1 to compute the incomplete tensor approximation. The matrices $A_{ij}[\widehat{\mathcal{F}}]$ and vectors $b_{ij}[\widehat{\mathcal{F}}]$ are given as follows:*

$$A_{13}[\widehat{\mathcal{F}}] = A_{23}[\widehat{\mathcal{F}}] = \begin{bmatrix} -0.8135 & 2.7988 \\ -1.3697 & 4.0149 \end{bmatrix}, \quad b_{13}[\widehat{\mathcal{F}}] = \begin{bmatrix} 1.5980 \\ 2.1879 \end{bmatrix}, \quad b_{23}[\widehat{\mathcal{F}}] = \begin{bmatrix} -2.0047 \\ -3.2027 \end{bmatrix},$$

$$A_{14}[\widehat{\mathcal{F}}] = A_{24}[\widehat{\mathcal{F}}] = \begin{bmatrix} 1.0277 & -0.8020 \\ -1.3697 & 4.0149 \end{bmatrix}, \quad b_{14}[\widehat{\mathcal{F}}] = \begin{bmatrix} 1.5920 \\ -3.2013 \end{bmatrix}, \quad b_{24}[\widehat{\mathcal{F}}] = \begin{bmatrix} -2.0059 \\ 7.5915 \end{bmatrix},$$

$$A_{15}[\widehat{\mathcal{F}}] = A_{25}[\widehat{\mathcal{F}}] = \begin{bmatrix} 1.0277 & -0.8020 \\ -0.8135 & 2.7988 \end{bmatrix}, \quad b_{15}[\widehat{\mathcal{F}}] = \begin{bmatrix} 2.1993 \\ -3.2020 \end{bmatrix}, \quad b_{25}[\widehat{\mathcal{F}}] = \begin{bmatrix} -3.1917 \\ 7.6153 \end{bmatrix}.$$

The linear least square problems (2.23) are solved to obtain $\widehat{G}$ and $N_3(\widehat{G}), N_4(\widehat{G}), N_5(\widehat{G})$, *which are*

$$N_3(\widehat{G}) = \begin{bmatrix} 0.5156 & 0.7208 \\ 1.6132 & -0.2474 \end{bmatrix}, \ N_4(\widehat{G}) = \begin{bmatrix} 1.2631 & -0.3665 \\ -0.6489 & 1.6695 \end{bmatrix},$$
$$N_5(\widehat{G}) = \begin{bmatrix} 1.6131 & -0.6752 \\ -1.2704 & 2.3517 \end{bmatrix}.$$

For $\xi = (3, 4, 5)$, *the eigendecomposition of the matrix* $\widehat{N}(\xi)$ *in (2.24) is*

$$\widehat{N}(\xi) = \begin{bmatrix} -0.7078 & 0.4470 \\ -0.7064 & -0.8945 \end{bmatrix} \begin{bmatrix} 12.0343 & 0 \\ 0 & 20.0786 \end{bmatrix} \begin{bmatrix} -0.7524 & 0.4499 \\ -0.6588 & -0.8931 \end{bmatrix}^{-1}.$$

It has eigenvectors $\hat{v}_1 = (-0.7078, -0.7064), \hat{v}_2 = (0.4470, -0.8945)$. *The vectors* $\hat{w}_1, \hat{w}_2$ *obtained as in (2.25) are*

$$\hat{w}_1 = (1.2021, 0.9918, 0.9899),\ \hat{w}_2 = (-1.0389, 2.0145, 3.0016).$$

By solving (2.26) and (2.27), we got the scalars

$$\hat{\gamma}_1 = -1.1990,\ \hat{\gamma}_2 = -2.1458, \qquad \hat{\lambda}_1 = 0.4521,\ \hat{\lambda}_2 = 0.6232.$$

Finally, we got the decomposition $\hat{\lambda}_1 \hat{u}_1^{\otimes 3} + \hat{\lambda}_2 \hat{u}_2^{\otimes 3}$ *with*

$$\hat{u}_1 = (1, \hat{\gamma}_1 \hat{v}_1, \hat{w}_1) = (1, 0.8477, 0.8479, 1.2021, 0.9918, 0.9899),$$
$$\hat{u}_2 = (1, \hat{\gamma}_2 \hat{v}_2, \hat{w}_2) = (1, -0.9776, 1.9102, -1.0389, 2.0145, 3.0016).$$

They are pretty close to the decomposition of $\mathcal{F}$.

2.3 Learning Diagonal Gaussian Mixtures

We use the incomplete tensor decomposition or approximation method to learn parameters for Gaussian mixture models. The Algorithms 2.1.4 and 2.2.1 can be applied to do that.

Let y be the random variable of dimension d for a Gaussian mixture model, with r components of Gaussian distribution parameters $(\omega_i, \mu_i, \Sigma_i)$, $i = 1, \dots, r$. We consider the case that $r \leq \frac{d}{2} - 1$. Let $y_1, \dots, y_N$ be samples drawn from the Gaussian mixture model. The sample average

$$\widehat{M_1} := \frac{1}{N}(y_1 + \cdots + y_N)$$

is an estimation for the mean $M_1 := \mathbb{E}[y] = \omega_1\mu_1 + \cdots + \omega_r\mu_r$. The symmetric tensor

$$\widehat{M_3} := \frac{1}{N}(y_1^{\otimes 3} + \cdots + y_N^{\otimes 3})$$

is an estimation for the third order moment tensor $M_3 := \mathbb{E}[y^{\otimes 3}]$. Recall that $\mathcal{F} = \sum_{i=1}^r \omega_i \mu_i^{\otimes 3}$. When all the covariance matrices Σ_i are diagonal, we have shown in (1.5) that

$$M_3 = \mathcal{F} + \sum_{j=1}^{d}(a_j \otimes e_j \otimes e_j + e_j \otimes a_j \otimes e_j + e_j \otimes e_j \otimes a_j).$$

If the labels i_1, i_2, i_3 are distinct from each other, $(M_3)_{i_1 i_2 i_3} = (\mathcal{F})_{i_1 i_2 i_3}$. Recall the label set Ω in (1.7). It holds that

$$(M_3)_\Omega = (\mathcal{F})_\Omega.$$

Note that $(\widehat{M_3})_\Omega$ is only an approximation for $(M_3)_\Omega$ and $(\mathcal{F})_\Omega$, due to sampling errors. If the rank $r \leq \frac{d}{2} - 1$, we can apply Algorithm 2.2.1 with the input $(\widehat{M_3})_\Omega$, to compute a rank-r tensor approximation for $\mathcal{F}$. Suppose the tensor approximation produced by Algorithm 2.2.1 is

$$\mathcal{F} \approx (p_1^*)^{\otimes 3} + \cdots + (p_r^*)^{\otimes 3}.$$

The computed $p_1^*, \ldots, p_r^*$ may not be real vectors, even if $\mathcal{F}$ is real. When the error $\epsilon := \|(\mathcal{F} - \widehat{M_3})_\Omega\|$ is small, by Theorem 2.2.2, we know

$$\|\tau_i^* p_i^* - \sqrt[3]{\omega_i}\mu_i\| \; = \; O(\epsilon)$$

where $(\tau_i^*)^3 = 1$. In computation, we can choose τ_i^* such that $(\tau_i^*)^3 = 1$ and the imaginary part vector $\mathrm{Im}(\tau_i^* p_i^*)$ has the smallest norm. It can be done by checking the imaginary part of $\tau_i^* p_i^*$ one by one for

$$\tau_i^* = 1, \; -\frac{1}{2} + \frac{\sqrt{-3}}{2}, \; -\frac{1}{2} - \frac{\sqrt{-3}}{2}.$$

Then we get the real vector

$$\hat{q}_i \; := \; \mathrm{Re}(\tau_i^* p_i^*).$$

It is expected that $\hat{q}_i \approx \sqrt[3]{\omega_i}\mu_i$. Since

$$M_1 = \omega_1\mu_1 + \cdots + \omega_r\mu_r \approx \omega_1^{2/3}\hat{q}_1 + \cdots + \omega_r^{2/3}\hat{q}_r,$$

the scalars $\omega_1^{2/3}, \ldots, \omega_r^{2/3}$ can be obtained by solving the linear least squares

$$\min_{(\beta_1,\ldots,\beta_r)\in\mathbb{R}_+^r} \left\| \widehat{M_1} - \sum_{i=1}^r \beta_i \hat{q}_i \right\|^2. \tag{2.30}$$

Let $(\beta_1^*, \ldots, \beta_r^*)$ be an optimizer for the above, then $\hat{\omega}_i := (\beta_i^*)^{3/2}$ is a good approximation for ω_i and the vector

$$\hat{\mu}_i \; := \; \hat{q}_i / \sqrt[3]{\hat{\omega}_i}$$

is a good approximation for μ_i. We may use

$$\hat{\mu}_i, \quad \big(\sum_{j=1}^{r} \hat{\omega}_j\big)^{-1} \hat{\omega}_i, \quad i = 1, \dots, r$$

as starting points to solve the nonlinear optimization

$$\begin{cases} \min\limits_{(\omega_1,\dots,\omega_r,\mu_1,\dots,\mu_r)} & \| \sum_{i=1}^{r} \omega_i \mu_i - \widehat{M}_1 \|^2 + \| \sum_{i=1}^{r} \omega_i (\mu_i^{\otimes 3})_\Omega - (\widehat{M}_3)_\Omega \|^2 \\ \text{subject to} & \omega_1 + \cdots + \omega_r = 1,\, \omega_1, \dots, \omega_r \geq 0, \end{cases} \tag{2.31}$$

for getting improved approximations. Suppose an optimizer of the above is

$$(\omega_1^*, \dots, \omega_r^*, \mu_1^*, \dots, \mu_r^*).$$

Now we discuss how to estimate the diagonal covariance matrices Σ_i. Let

$$\mathcal{A} := M_3 - \mathcal{F}, \quad \widehat{\mathcal{A}} := \widehat{M}_3 - (\hat{q}_1)^{\otimes 3} - \cdots - (\hat{q}_r)^{\otimes 3}. \tag{2.32}$$

By (1.5), we know that

$$\mathcal{A} = \sum_{j=1}^{d} (a_j \otimes e_j \otimes e_j + e_j \otimes a_j \otimes e_j + e_j \otimes e_j \otimes a_j), \tag{2.33}$$

where $a_j = \sum\limits_{i=1}^{r} \omega_i \sigma_{ij}^2 \mu_i$ for $j = 1, \cdots, d$. The equation (2.33) implies that

$$(a_j)_j = \frac{1}{3} \mathcal{A}_{jjj}, \quad (a_j)_i = \mathcal{A}_{jij}, \tag{2.34}$$

for $i, j = 1, \cdots, d$ and $i \neq j$. So we choose vectors $\hat{a}_j \in \mathbb{R}^d$ such that

$$(\hat{a}_j)_j = \frac{1}{3} \widehat{\mathcal{A}}_{jjj}, \quad (\hat{a}_j)_i = \widehat{\mathcal{A}}_{jij} \quad \text{for} \quad i \neq j. \tag{2.35}$$

Since $\hat{a}_j \approx \sum_{i=1}^{r} \omega_i \sigma_{ij}^2 \mu_i$, the covariance matrices $\Sigma_i = \mathrm{diag}(\sigma_{i1}^2, \ldots, \sigma_{id}^2)$ can be estimated by solving the nonnegative linear least squares ($j = 1, \ldots, d$)

$$\begin{cases} \min\limits_{(\beta_{1j},\ldots,\beta_{rj})} & \left\| \hat{a}_j - \sum\limits_{i=1}^{r} \omega_i^* \mu_i^* \beta_{ij} \right\|^2 \\ \text{subject to} & \beta_{1j} \geq 0, \ldots, \beta_{rj} \geq 0. \end{cases} \tag{2.36}$$

For each j, let $(\beta_{1j}^*, \ldots, \beta_{rj}^*)$ be the optimizer for the above. When $(\widehat{M_3})_\Omega$ is close to $(M_3)_\Omega$, it is expected that β_{ij}^* is close to $(\sigma_{ij})^2$. Therefore, we can estimate the covariance matrices Σ_i as follows

$$\Sigma_i^* := \mathrm{diag}(\beta_{i1}^*, \ldots, \beta_{id}^*), \quad (\sigma_{ij}^*)^2 := \beta_{ij}^*. \tag{2.37}$$

The following is the algorithm for learning Gaussian mixture models.

Algorithm 2.3.1. *(Learning diagonal Gaussian mixture models.)*

Input: Samples $\{y_1, \ldots, y_N\} \subseteq \mathbb{R}^d$ *drawn from a Gaussian mixture model and the number* r *of component Gaussian distributions.*

Step 1. Compute the sample averages $\widehat{M_1} := \frac{1}{N}\sum_{i=1}^{N} y_i$ *and* $\widehat{M_3} := \frac{1}{N}\sum\limits_{i=1}^{N} y_i^{\otimes 3}$.

Step 2. Apply Algorithm 2.2.1 to the subtensor $(\widehat{\mathcal{F}})_\Omega := (\widehat{M_3})_\Omega$. *Let* $(p_1^*)^{\otimes 3} + \cdots + (p_r^*)^{\otimes 3}$ *be the obtained rank-*r *tensor approximation for* $\widehat{\mathcal{F}}$. *For each* $i = 1, \ldots, r$, *let* $\hat{q}_i := Re(\tau_i p_i^*)$ *where* τ_i *is the cube root of* 1 *that minimizes the imaginary part vector norm* $\|Im(\tau_i p_i^*)\|$.

Step 3. Solve (2.30) to get $\hat{\omega}_1, \ldots, \hat{\omega}_r$ *and* $\hat{\mu}_i = q_i/\sqrt[3]{\hat{\omega}_i}, i = 1, \ldots, r$.

Step 4. Use the above $\hat{\omega}_i$, $\hat{q}_i$ *as initial points to solve the nonlinear optimization (2.31) for the optimal* $\omega_i^*, \mu_i^*, i = 1, \ldots, r$.

Step 5. Get vectors $\hat{a}_1, \ldots, \hat{a}_d$ *as in (2.35). Solve the optimization (2.36) to get optimizers* β_{ij}^* *and then choose* Σ_i^* *as in (2.37).*

Output: Component Gaussian distribution parameters $(\mu_i^*, \Sigma_i^*, \omega_i^*), i = 1, \dots, r.$

The sample averages $\widehat{M}_1, \widehat{M}_3$ can typically be used as good estimates for the true moments M_1, M_3. When the value of r is not known, it can be determined as in Remark 2.1.6. The performance of Algorithm 2.3.1 is analyzed as follows.

Theorem 2.3.2. *Consider the d-dimensional diagonal Gaussian mixture model with parameters* $\{(\omega_i, \mu_i, \Sigma_i) : i \in [r]\}$ *and* $r \le \frac{d}{2} - 1$. *Let* $\{(\omega_i^*, \mu_i^*, \Sigma_i^*) : i \in [r]\}$ *be produced by Algorithm 2.3.1. If the distance* $\epsilon := \max(\|M_3 - \widehat{M}_3\|, \|M_1 - \widehat{M}_1\|)$ *is small enough and the tensor* $\mathcal{F} = \sum_{i=1}^r \omega_i \mu_i^{\otimes 3}$ *satisfies conditions of Theorem 2.2.2, then*

$$\|\mu_i - \mu_i^*\| = O(\epsilon), \|\omega_i - \omega_i^*\| = O(\epsilon), \|\Sigma_i - \Sigma_i^*\| = O(\epsilon),$$

where the above constants inside $O(\cdot)$ *only depend on parameters* $\{(\omega_i, \mu_i, \Sigma_i) : i \in [r]\}$ *and the choice of* ξ *in Algorithm 2.3.1.*

Proof. For the vectors $p_i := \sqrt[3]{\omega_i}\mu_i$, we have $\mathcal{F} = \sum_{i=1}^r p_i^{\otimes 3}$. Since

$$\|(\mathcal{F} - \widehat{\mathcal{F}})_\Omega\| = \|(M_3 - \widehat{M}_3)_\Omega\| \le \epsilon$$

and $\mathcal{F}$ satisfies conditions of Theorem 2.2.2, we know $\|\tau_i^* p_i^* - p_i\| = O(\epsilon)$ for some $(\tau_i^*)^3 = 1$, by Theorem 2.2.2. The constants inside $O(\epsilon)$ depend on parameters of the Gaussian model and ξ. Then, we have $\|\mathrm{Im}(\tau_i^* p_i^*)\| = O(\epsilon)$ since the vectors p_i are real. When ϵ is small enough, such τ_i^* is the τ in Step 2 of Algorithm 2.3.1 that minimizes $\|\mathrm{Im}(\tau_i p_i^*)\|$, so we have

$$\|\hat{q}_i - p_i\| \le \|\tau_i p_i^* - p_i\| = O(\epsilon)$$

where $\hat{q}_i = \mathrm{Re}(\tau_i p_i^*)$ is from the Step 2. The vectors $\hat{q}_1, \dots, \hat{q}_r$ are linearly independent when ϵ is small. Thus, the problem (2.30) has a unique solution and the weights $\hat{\omega}_i$ can be found by solving (2.30). Since $\|M_1 - \widehat{M}_1\| \le \epsilon$ and $\|\hat{q}_i - p_i\| = O(\epsilon)$, we have

$\|\omega_i - \hat{\omega}_i\| = O(\epsilon)$ (see [14, Theorem 3.4]). The mean vectors $\hat{\mu}_i$ are obtained by $\hat{\mu}_i = \hat{q}_i/\sqrt[3]{\hat{\omega}_i}$, so the approximation error is

$$\|\mu_i - \hat{\mu}_i\| = \|p_i/\sqrt[3]{\omega_i} - \hat{q}_i/\sqrt[3]{\hat{\omega}_i}\| = O(\epsilon).$$

The constants inside the above $O(\epsilon)$ depend on parameters of the Gaussian mixture model and ξ.

The problem (2.31) is solved to obtain ω_i^* and μ_i^*, so

$$\left\| \widehat{M_1} - \sum_{i=3}^{r} \omega_i^* \mu_i^* \right\| + \left\| \widehat{\mathcal{F}} - \sum_{i=1}^{r} \omega_i^* (\mu_i^*)^{\otimes 3} \right\| = O(\epsilon).$$

Let $\mathcal{F}^* := \sum_{i=1}^{r} \omega_i^* (\mu_i^*)^{\otimes 3} = \sum_{i=1}^{r} (\sqrt[3]{\omega_i^*} \mu_i^*)^{\otimes 3}$, then

$$\|\mathcal{F} - \mathcal{F}^*\| \leq \|\mathcal{F} - \hat{\mathcal{F}}\| + \|\hat{\mathcal{F}} - \mathcal{F}^*\| = O(\epsilon).$$

Theorem 2.2.2 implies $\|p_i - \sqrt[3]{\omega_i^*} \mu_i^*\| = O(\epsilon)$. In addition, we have

$$\left\| \widehat{M_1} - \sum_{i=1}^{r} \omega_i^* \mu_i^* \right\| = \left\| \widehat{M_1} - \sum_{i=1}^{r} (\omega_i^*)^{2/3} \sqrt[3]{\omega_i^*} \mu_i^* \right\| = O(\epsilon).$$

The first order moment is $M_1 = \sum_{i=1}^{r} (\omega_i)^{2/3} p_i$. Since $\|M_1 - \hat{M}_1\| = O(\epsilon)$ and $\|p_i - \sqrt[3]{\omega_i^*} \mu_i^*\| = O(\epsilon)$, it holds that $\|\omega_i^{2/3} - (\omega_i^*)^{2/3}\| = O(\epsilon)$ by [14, Theorem 3.4]. This implies that $\|\omega_i - \omega_i^*\| = O(\epsilon)$, so

$$\|\mu_i - \mu_i^*\| = \|p_i/\sqrt[3]{\omega_i} - (\sqrt[3]{\omega_i^*} \mu_i^*)/\sqrt[3]{\omega_i^*}\| = O(\epsilon).$$

The constants inside the above $O(\cdot)$ only depend on parameters $\{(\omega_i, \mu_i, \Sigma_i) : i \in [r]\}$ and ξ.

The covariance matrices Σ_i are recovered by solving the linear least squares (2.36).

In the least square problems, it holds that $\|\omega_i\mu_i - \omega_i^*\mu_i^*\| = O(\epsilon)$ and

$$\|\mathcal{A} - \widehat{\mathcal{A}}\| \leq \|M_3 - \widehat{M_3}\| + \|\mathcal{F} - \sum_{i=1}^{r} \hat{q}_i^{\otimes 3}\| = O(\epsilon),$$

where tensors $\mathcal{A}, \widehat{\mathcal{A}}$ are defined in (2.32). When the error ϵ is small, vectors $\omega_i^*\mu_1^*, \ldots, \omega_i^*\mu_r^*$ are linearly independent and hence (2.36) has a unique solution for each j. By [14, Theorem 3.4], we have

$$\|(\sigma_{ij})^2 - (\sigma_{ij}^*)^2\| = O(\epsilon).$$

It implies that $\|\Sigma_i - \Sigma_i^*\| = O(\epsilon)$, where the constants inside $O(\cdot)$ only depend on parameters $\{(\omega_i, \mu_i, \Sigma_i) : i \in [r]\}$ and ξ. □

2.4 Numerical Simulations

First, we show the performance of Algorithm 2.2.1 for computing incomplete symmetric tensor approximations. For a range of dimension d and rank r, we get the tensor $\mathcal{F} = (p_1)^{\otimes 3} + \cdots + (p_r)^{\otimes 3}$, where each p_i is randomly generated according to the Gaussian distribution in `MATLAB`. Then, we apply the perturbation $(\widehat{\mathcal{F}})_\Omega = (\mathcal{F})_\Omega + \mathcal{E}_\Omega$, where $\mathcal{E}$ is a randomly generated tensor, also according to the Gaussian distribution in `MATLAB`, with the norm $\|\mathcal{E}_\omega\|_\Omega = \epsilon$. After that, Algorithm 2.2.1 is applied to the subtensor $(\widehat{\mathcal{F}})_\Omega$ to find the rank-r tensor approximation. The approximation quality is measured by the absolute error and the relative error

$$\text{abs-error} := \|(\mathcal{F}^* - \mathcal{F})_\Omega\|, \quad \text{rel-error} := \frac{\|(\mathcal{F}^* - \widehat{\mathcal{F}})_\Omega\|}{\|(\mathcal{F} - \widehat{\mathcal{F}})_\Omega\|},$$

where $\mathcal{F}^*$ is the output of Algorithm 2.2.1. For each case of (d, r, ϵ), we generate 100 random instances. The min, average, and max relative errors for each dimension d and rank r are reported in the Table 2.1. The results show that Algorithm 2.2.1 performs very well for computing tensor approximations.

Table 2.1. The performance of Algorithm 2.2.1

			rel-error			abs-error			
d	r	ϵ	min	average	max	min	average	max	time
20	3	0.1	0.9610	0.9731	0.9835	0.0141	0.0268	0.0556	0.2687
	5	0.01	0.9634	0.9700	0.9742	0.0019	0.0032	0.0068	0.2392
	7	0.001	0.9148	0.9373	0.9525	$2.3 \cdot 10^{-4}$	$3.8 \cdot 10^{-4}$	$6.6 \cdot 10^{-4}$	0.2638
30	4	0.1	0.9816	0.9854	0.9890	0.0094	0.0174	0.0533	0.4386
	8	0.01	0.9634	0.9700	0.9742	0.0015	0.0024	0.0060	0.7957
	11	0.001	0.9501	0.9587	0.9667	$1.8 \cdot 10^{-4}$	$3.0 \cdot 10^{-4}$	$5.7 \cdot 10^{-4}$	0.8954
40	6	0.1	0.9853	0.9877	0.9904	0.0099	0.0146	0.0359	1.7779
	10	0.01	0.9761	0.9795	0.9820	0.0013	0.0020	0.0045	2.6454
	15	0.001	0.9653	0.9690	0.9734	$1.7 \cdot 10^{-4}$	$2.6 \cdot 10^{-4}$	$4.8 \cdot 10^{-4}$	3.6785
50	7	0.1	0.9887	0.9911	0.9925	0.0081	0.0128	0.0294	4.9774
	13	0.01	0.9812	0.9831	0.9854	0.0011	0.0018	0.0045	8.7655
	18	0.001	0.9739	0.9767	0.9792	$1.5 \cdot 10^{-4}$	$2.2 \cdot 10^{-4}$	$4.1 \cdot 10^{-4}$	11.6248

Second, we explore the performance of Algorithm 2.3.1 for learning diagonal Gaussian mixture models. We compare it with the classical EM algorithm, for which the MATLAB function `fitgmdist` is used (`MaxIter` is set to be 100 and `RegularizationValue` is set to be 0.0001). The dimensions $d = 20, 30, 40, 50, 60$ are tested. Three values of r are tested for each case of d. We generate 100 random instances of $\{(\omega_i, \mu_i, \Sigma_i) : i = 1, \cdots, r\}$ for $d \in \{20, 30, 40\}$, and 20 random instances for $d \in \{50, 60\}$, because of the relatively more computational time for the latter case. For each instance, 10000 samples are generated. To generate the weights $\omega_1, \ldots, \omega_r$, we first use the MATLAB function `randi` to generate a random 10000−dimensional integer vector of entries from $[r]$, then the occurring frequency of i in $[r]$ is used as the weight ω_i. For each diagonal covariance matrix Σ_i, its diagonal vector is set to be the square of a random vector generated by the MATLAB function `randn`. Each sample is generated from one of r component Gaussian distributions, so they are naturally separated into r groups. Algorithm 2.3.1 and the EM algorithm are applied to fit the Gaussian mixture model to the 10000 samples for each instance. For each sample, we calculate the likelihood of the sample to each component Gaussian distribution in the estimated Gaussian mixture model. A sample is classified to the ith group if its likelihood for the ith component is maximum. The classification accuracy is the rate that samples are classified to the correct group. In Table 2.2, for each pair (d, r), we report the accuracy of Algorithm 2.3.1 in the first row and the accuracy of the EM algorithm in the second row. As one can see, Algorithm 2.3.1 performs better than EM algorithm, and its accuracy isn't affected when the dimensions and ranks increase. Indeed, as the difference between the dimension d and the rank r increases, Algorithm 2.3.1 becomes more and more accurate. This is opposite to the EM algorithm. The reason is that the difference between the number of rows and the number of columns of $A_{ij}[\mathcal{F}]$ in (2.10) increases as $d - r$ becomes bigger, which makes Algorithm 2.3.1 more robust.

Last, we apply Algorithm 2.3.1 to do texture classifications. We select 8 textured images of 512×512 pixels from the VisTex database. We use the MATLAB function `rgb2gray` to convert them into grayscale version since we only need their structure and texture information. Each image is divided into subimages of 32×32 pixels. We perform the discrete cosine transformation(DCT) on each block of size 16×16 pixels with overlap of 8 pixels. Each component of 'Wavelet-like' DCT feature is the sum of the absolute value of the DCT coefficients in the corresponding sub-block. So the dimension d of the feature vector extracted from each subimage is 13. We use blocks extracted from the first 160 subimages for training and those from the rest 96 subimages for testing. We refer to [50] for more details. For each image, we apply Algorithm 2.3.1 and the EM algorithm to fit a Gaussian mixture model to the image. We choose the number of components r according to Remark 2.1.6. To classify the test data, we follow the Bayes decision rule that assigns each block to the texture which maximizes the posteriori probability, where we assume a uniform prior over all classes [17]. The classification accuracy is the rate that a

Table 2.2. Comparison between Algorithm 2.3.1 and EM for simulations

		accuracy		time	
d	r	Algorithm 2.3.1	EM	Algorithm 2.3.1	EM
20	3	0.9861	0.9763	0.8745	0.1649
	5	0.9740	0.9400	2.3476	0.3852
	7	0.9659	0.9252	3.4352	0.6777
30	4	0.9965	0.9684	4.5266	0.2959
	8	0.9923	0.9277	8.5494	0.8525
	11	0.9895	0.9219	17.2091	1.4106
40	6	0.9990	0.9117	18.9160	0.6273
	10	0.9981	0.8931	28.4161	1.2617
	15	0.9971	0.9111	69.8013	2.0627
50	7	0.9997	0.8997	40.6810	0.8314
	13	0.9995	0.9073	104.7927	1.7867
	18	0.9993	0.9038	163.2711	2.6862
60	8	0.9999	0.8874	93.9836	1.1266
	15	0.9998	0.8632	234.0331	2.6435
	22	0.9995	0.8929	497.9371	3.5527

subimage is correctly classified, which is shown in Table 2.3. Algorithm 2.3.1 outperforms the classical EM algorithm for the accuracy rates for six of the images.

Table 2.3. Classification results on 8 textures

Accuracy	Algorithm 2.3.1	EM
Bark.0000	0.5376	0.8413
Bark.0009	0.5107	0.7150
Flowers.0001	0.8137	0.6315
Tile.0000	0.8219	0.7239
Paintings.11.0001	0.8047	0.7350
Grass.0001	0.9841	0.9068
Brick.0004	0.9406	0.8854
Fabric.0013	0.9220	0.9048

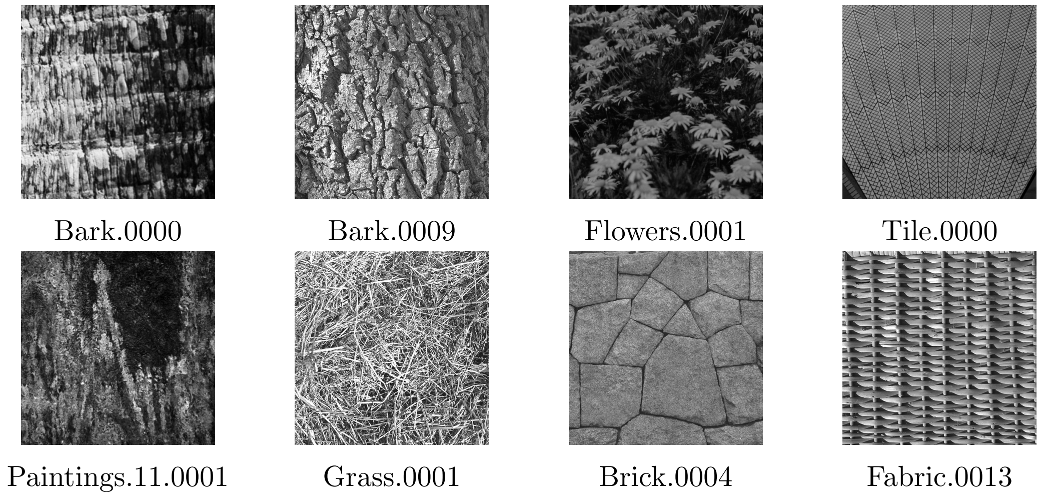

Figure 2.1. Textures from VisTex

Acknowledgement. The Chapter 2, in full, has been published in *Vietnam Journal of Mathematics* 2021 [64]. The dissertation author coauthored this paper with Nie, Jiawang and Yang, Zi.

Chapter 3

Learning Diagonal Gaussian Mixture Models Using Higher Order Moment

Previous work performs the incomplete tensor decomposition to learn diagonal Gaussian mixture models using partially given entries of the moment tensor when $r \le \frac{d}{2} - 1$. This result uses the first and third-order moments to recover unknown model parameters. However, the third order moment M_3 is insufficient when $r > \frac{d}{2} - 1$. In the following sections, we propose to utilize higher-order moments to learn Gaussian mixture models with more components.

3.1 Incomplete Tensor Decomposition of Higher Order

In this section, we discuss how to solve the incomplete symmetric tensor decomposition problem arising from learning Gaussian mixtures with parameters $\{\omega_i, \mu_i, \Sigma_i\}_{i=1}^r$. Let $\mathcal{F}_m \in S^m(\mathbb{C}^d)$ be the symmetric tensor. In the following, we discuss how to obtain the decomposition of $\mathcal{F}_m$ given entries $(\mathcal{F}_m)_{\Omega_m}$.

For convenience, we denote $n \coloneqq d - 1$. Suppose that $\mathcal{F}_m$ has the decomposition

$$\mathcal{F}_m = \omega_1 \mu_1^{\otimes m} + \cdots \omega_r \mu_r^{\otimes m}, \tag{3.1}$$

where $\mu_i = ((\mu_i)_0, (\mu_i)_1, \ldots, (\mu_i)_n) \in \mathbb{C}^{n+1}$. When the leading entry of each μ_i is nonzero, we can write the decomposition (3.1) as

$$\mathcal{F}_m = \lambda_1 \begin{bmatrix} 1 \\ u_1 \end{bmatrix}^{\otimes m} + \cdots + \lambda_r \begin{bmatrix} 1 \\ u_r \end{bmatrix}^{\otimes m}, \tag{3.2}$$

where $\lambda_i = \omega_i((\mu_1)_0)^m$, and $u_i = ((u_i)_1, \ldots, (u_i)_n) = (\mu_i)_{1:n}/(\mu_i)_0 \in \mathbb{C}^n$.

Let $1 \leq p \leq m-2$ and $p \leq k \leq n-m-p$ be numbers such that

$$\binom{k}{p} \geq r \quad \text{and} \quad \binom{n-k-1}{m-p-1} \geq r.$$

Define the set

$$\mathscr{B}_0 \subseteq \{x_{i_1} \cdots x_{i_p} : 1 \leq i_1 < \cdots < i_p \leq k\} \tag{3.3}$$

such that $\mathscr{B}_0$ consists of the first r monomials in the graded lexicographic order. Correspondingly, the set $\mathscr{B}_1$ is defined as

$$\mathscr{B}_1 := \{x_{j_1} \cdots x_{j_{p+1}} : 1 \leq j_1 < \cdots < j_p \leq k < j_{p+1} \leq n\}. \tag{3.4}$$

For convenience, we say $\alpha \in \mathbb{N}^n$ is in $\mathscr{B}_0$ (resp. $\mathscr{B}_1$) if $x^\alpha \in \mathscr{B}_0$ (resp. $\mathscr{B}_1$). Let $\alpha = e_{j_1} + \cdots + e_{j_p} + e_{j_{p+1}} \in \mathscr{B}_1$ and $G \in \mathbb{C}^{\mathscr{B}_0 \times \mathscr{B}_1}$ be a matrix labelled by monomials in $\mathscr{B}_0$ and $\mathscr{B}_1$. We consider the polynomial

$$\varphi_{j_1 \cdots j_p j_{p+1}}[G](x) := \sum_{(i_1, \ldots, i_p) \in \mathscr{B}_0} G(\sum_{t=1}^{p} e_{i_t}, \sum_{t=1}^{p+1} e_{j_t}) x_{i_1} \cdots x_{i_p} - x_{j_1} \cdots x_{j_p} x_{j_{p+1}}.$$

Recall that $\varphi_{j_1 \cdots j_{p+1}}[G](x)$ is a generating polynomial for $\mathcal{F}_m$ if it satisfies (1.2), i.e.

$$\langle \varphi_{j_1 \cdots j_p j_{p+1}}[G](x) \cdot x^\beta, \ \mathcal{F}_m \rangle = 0 \quad \forall \beta \in \mathbb{N}^n_{m-p-1}.$$

The matrix G is called a generating matrix if $\varphi_{j_1\cdots j_{p+1}}[G](x)$ is a generating polynomial. If the matrix G is a generating matrix of $\mathcal{F}_m$, it should satisfy the equations

$$\langle x_{s_1}\cdots x_{s_{m-p-1}}\varphi_{j_1\cdots j_{p+1}}[G](x),\, \mathcal{F}_m\rangle = 0 \tag{3.5}$$

for each $\alpha = e_{j_1}+\cdots+e_{j_p}+e_{j_{p+1}} \in \mathscr{B}_1$ and each tuple $(s_1,\ldots,s_{m-p-1}) \in \mathcal{O}_\alpha$, where

$$\mathcal{O}_\alpha \coloneqq \left\{ (s_1,\ldots,s_{m-p-1}) : \begin{array}{c} k+1 \le s_1 < \ldots < s_{m-p-1} \le n, \\ s_1 \ne j_{p+1},\ldots,s_{m-p-1} \ne j_{p+1} \end{array} \right\}.$$

Define the matrix $A[\alpha,\mathcal{F}_m]$ and the vector $b[\alpha,\mathcal{F}_m]$ be such that

$$\begin{cases} A[\alpha,\mathcal{F}_m]_{\gamma,\beta} \coloneqq (\mathcal{F}_m)_{\beta+\gamma}, & \forall(\gamma,\beta) \in \mathcal{O}_\alpha \times \mathscr{B}_0 \\ b[\alpha,\mathcal{F}_m]_\gamma \coloneqq (\mathcal{F}_m)_{\alpha+\gamma}, & \forall\gamma \in \mathcal{O}_\alpha. \end{cases} \tag{3.6}$$

The dimension of $A[\alpha,\mathcal{F}_m]$ is $\binom{n-k-1}{m-p-1} \times r$ and the equations in (3.5) can be equivalently written as

$$A[\alpha,\mathcal{F}_m]\cdot G(:,\alpha) = b[\alpha,\mathcal{F}_m]. \tag{3.7}$$

Lemma 3.1.1 proves that the matrix $A[\alpha,\mathcal{F}_m]$ in (3.6) has full column rank under some genericity conditions.

Lemma 3.1.1. *Suppose that $\binom{k}{p} \ge r$ and $\binom{n-k-1}{m-p-1} \ge r$. Let $\mathcal{F}_m$ be the tensor with the decomposition* (3.2). *If vectors $\{[u_i]_{\mathscr{B}_0}\}_{i=1}^r$ and $\{[u_i]_{\mathcal{O}_\alpha}\}_{i=1}^r$ are both linearly independent, then the matrix $A[\alpha,\mathcal{F}_m]$ as in* (3.6) *has full column rank.*

Proof. The matrix $A[\alpha,\mathcal{F}_m]$ can be written as

$$A[\alpha,\mathcal{F}_m] = \sum_{i=1}^r \lambda_i [u_i]_{\mathcal{O}_\alpha}[u_i]_{\mathscr{B}_0}^T.$$

Therefore, $A[\alpha,\mathcal{F}_m]$ has full column rank. □

Remark 3.1.2. *A successful construction of $\mathscr{B}_0$ requires that $\binom{k}{p} \geq r$. The vectors $\{[u_i]_{\mathscr{B}_0}\}_{i=1}^r$ and $\{[u_i]_{\mathcal{O}_\alpha}\}_{i=1}^r$ have dimensions r and $\binom{n-k-1}{m-p-1}$ respectively. Thus, when*

$$\binom{k}{p} \geq r \quad \textit{and} \quad \binom{n-k-1}{m-p-1} \geq r,$$

the vectors $\{[u_i]_{\mathscr{B}_0}\}_{i=1}^r$ and $\{[u_i]_{\mathcal{O}_\alpha}\}_{i=1}^r$ are both linearly independent for generic vectors $u_1, \ldots, u_r$ in real or complex field.

Under the condition of Lemma 3.1.1, we can prove there exists a unique generating matrix G for $\mathcal{F}_m$.

Theorem 3.1.3. *Let $\mathcal{F}_m$ be the tensor in* (3.2)*. Suppose that conditions of Lemma 3.1.1 hold, then there exists a unique generating matrix G for the tensor $\mathcal{F}_m$.*

Proof. We first prove the existence of G. For $k+1 \leq j \leq n$, we denote

$$d_j = ((u_1)_j, \ldots, (u_r)_j).$$

Under the assumption of Lemma 3.1.1, we can define

$$N_j = ([u_1]_{\mathscr{B}_0}, \ldots, [u_r]_{\mathscr{B}_0})\mathrm{diag}(d_j)([u_1]_{\mathscr{B}_0}, \ldots, [u_r]_{\mathscr{B}_0})^{-1}.$$

The matrix G is constructed as

$$G(\beta, \nu + e_j) = (N_j)_{\nu,\beta}$$

for $j = k+1, \ldots, n$ and $\nu, \beta \in \mathscr{B}_0$. For every $\alpha = \nu + e_j \in \mathscr{B}_1$, it holds that

$$\sum_{\beta \in \mathscr{B}_0} G(\beta, \alpha) u_i^\beta - u_i^\alpha = (N_j)_{\nu,:}[u_i]_{\mathscr{B}_0} - (u_i)_j u_i^\nu = 0$$

Thus, for every $\gamma \in \mathbb{N}^n_{m-p-1}$, it holds that

$$\langle x^\gamma \varphi_\alpha[G](x), \mathcal{F}_m \rangle = \sum_{i=1}^{r} \lambda_i u_i^\gamma \sum_{\theta \in \mathscr{B}_0} (G(\theta, \alpha) u_i^\theta - u_i^\alpha) = 0.$$

It proves that the matrix G is a generating matrix for $\mathcal{F}_m$.

Next, we show the uniqueness. The matrix $A[\alpha, \mathcal{F}_m]$ has full column rank by Lemma 3.1.1, so the generating matrix G is uniquely determined by linear systems in (3.7). It proves the uniqueness of G. □

By Theorem 3.1.3 and Lemma 3.1.1, the generating matrix G can be uniquely determined by solving the linear system (3.7). Let $N_{k+1}(G), \dots, N_n(G) \in \mathbb{C}^{r \times r}$ be the matrices given as ($\nu, \beta \in \mathscr{B}_0$):

$$N_l(G)_{\nu,\beta} = G(\beta,\ \nu + e_l) \quad \text{for } l = k+1, \dots, n. \tag{3.8}$$

Then we have

$$N_l(G)[v_i]_{\mathscr{B}_0} = (w_i)_{l-k}[v_i]_{\mathscr{B}_0} \quad \text{for } l = k+1, \dots, n.$$

for the vectors ($i = 1, \dots, r$)

$$v_i := ((v_i)_1, \dots, (v_i)_k) = (u_i)_{1:k},$$
$$w_i := ((w_i)_1, \dots, (w_i)_{n-k}) = (u_i)_{k+1:n}.$$

We select a generic vector $\xi := (\xi_{k+1}, \dots, \xi_n)$ and let

$$N(\xi) \ := \ \xi_{k+1} N_{k+1} + \cdots + \xi_n N_n. \tag{3.9}$$

Let $\tilde{v}_1, \dots, \tilde{v}_r$ be unit length eigenvectors of $N(\xi)$, which are also common eigenvectors of $N_{k+1}(G), \dots, N_n(G)$. For each $i = 1, \dots, r$, let $\tilde{w}_i$ be the vector such that its jth entry

$(\tilde{w}_i)_j$ is the eigenvalue of $N_{k+j}(G)$, associated to the eigenvector $\tilde{v}_i$. Equivalently,

$$\tilde{w}_i := (\tilde{v}_i^H N_{k+1}(G)\tilde{v}_i, \cdots, \tilde{v}_i^H N_n(G)\tilde{v}_i) \quad i = 1, \ldots, r. \tag{3.10}$$

Up to a permutation of $(\tilde{v}_1, \ldots, \tilde{v}_r)$, we have

$$w_i = \tilde{w}_i.$$

We denote the sets

$$\boxed{\begin{aligned} &J_1 := \{x_{i_1} \cdots x_{i_p} : 1 \leq i_1 < \cdots < i_p \leq k\}, \\ &J_1^{-j} := J_1 \cap \{x_{i_1} \cdots x_{i_p} : i_1, \ldots, i_p \neq j\}, \\ &J_2 := \{(x_{i_1} \cdots x_{i_{m-p-1}} : k+1 \leq i_1 < \cdots < i_{m-p-1} \leq n\}, \\ &J_3 := \{x_{i_1-k} \cdots x_{i_{m-p-1}-k} : (i_1, \ldots, i_{m-p-1}) \in J_2\}. \end{aligned}} \tag{3.11}$$

The tensors $\lambda_1 v_1^{\otimes p}, \ldots, \lambda_r v_r^{\otimes p}$ satisfy the linear equation

$$\sum_{i=1}^{r} \lambda_i v_i^{\otimes p} \otimes \tilde{w}_i^{\otimes(m-p-1)} = (\mathcal{F}_m)_{0,[1:k]^p,[k+1:n]^{(m-p-1)}}.$$

Thus, $\lambda_1[v_1]_{J_1}, \ldots, \lambda_r[v_r]_{J_1}$ can be obtained by the linear equation

$$\min_{(\gamma_1,\ldots,\gamma_r)} \left\| (\mathcal{F}_m)_{J_1 \cdot J_2} - \sum_{i=1}^{r} \gamma_i \otimes [\tilde{w}_i]_{J_3} \right\|^2. \tag{3.12}$$

We denote the minimizer of (3.12) by $(\tilde{\gamma}_1, \ldots, \tilde{\gamma}_r)$.

The vectors $v_1, \ldots, v_r$ satisfy the linear equation

$$\sum_{i=1}^{r} v_i \otimes \lambda_i v_i^{\otimes p} \otimes \tilde{w}_i^{\otimes(m-p-1)} = (\mathcal{F}_m)_{[1:k]^{p+1} \times [k+1:n]^{(m-p-1)}}.$$

For each $j \in [k]$, we solve the linear least square problem

$$\min_{(v_1,\dots,v_r)} \left\| (\mathcal{F}_m)_{x_j \cdot J_1^{-j} \cdot J_2} - \sum_{i=1}^{r} (v_i)_j \cdot \tilde{\gamma}_i \otimes [\tilde{w}_i]_{J_3} \right\|^2 . \tag{3.13}$$

We denote the minimizer of (3.13) as $(\tilde{v}_1, \dots, \tilde{v}_r)$.

The scalars $\lambda_1, \dots, \lambda_r$ in (3.2) satisfy the linear equation

$$\lambda_1 \begin{bmatrix} 1 \\ \tilde{u}_1 \end{bmatrix}^{\otimes m} + \cdots + \lambda_r \begin{bmatrix} 1 \\ \tilde{u}_r \end{bmatrix}^{\otimes m} = \mathcal{F}_m, \tag{3.14}$$

where $\tilde{u}_i = (\tilde{v}_i, \tilde{w}_i)$ for $i = 1, \dots, r$. They can be solved by the following linear least square problem

$$\min_{(\lambda_1,\dots,\lambda_r)} \left\| (\mathcal{F})_{\Omega_m} - \sum_{i=1}^{r} \lambda_i \cdot \left(\begin{bmatrix} 1 \\ \tilde{u}_i \end{bmatrix}^{\otimes m} \right)_{\Omega_m} \right\|^2 . \tag{3.15}$$

Let $(\tilde{\lambda}_1, \dots, \tilde{\lambda}_r)$ be the minimizer of (3.15).

Concluding everything above, we obtain the decomposition of $\mathcal{F}_m$

$$\mathcal{F}_m = q_1^{\otimes m} + \cdots + q_r^{\otimes m},$$

where $q_i \coloneqq (\tilde{\lambda})^{1/m}(1, \tilde{v}_i, \tilde{w}_i)$, for $i = 1, \dots, r$. All steps to obtain the decomposition are summarized in Algorithm 3.1.4

Algorithm 3.1.4. *(Incomplete symmetric tensor decompositions.)*

Input: Rank r, dimension d, constant p and subtensor $(\mathcal{F}_m)_{\Omega_m}$ in (3.2).

Step 1. Determine the matrix G by solving (3.7) for each $\alpha = e_{j_1} + \cdots + e_{j_{p+1}} \in \mathscr{B}_1$.

Step 2. Let $N(\xi)$ be the matrix as in (3.9), for a randomly selected vector ξ. Compute the vectors $\tilde{w}_i$ as in (3.10).

Step 3. Solve the linear least squares (3.12), (3.13) and (3.15) to get the scalars $\tilde{\lambda}_i$ *and vectors* $\tilde{v}_i$.

Output: The tensor decomposition $\mathcal{F}_m = q_1^{\otimes m} + \cdots + q_r^{\otimes m}$, *for* $q_i = (\tilde{\lambda})^{1/m}(1, \tilde{v}_i, \tilde{w}_i)$.

Theorem 3.1.5. *Let* $\mathcal{F}_m$ *be the tensor in* (3.2). *If* $\mathcal{F}_m$ *satisfies conditions of Lemma 3.1.1 and the matrix* $N(\xi)$ *in* (3.9) *has distinct eigenvalues, then Algorithm 3.1.4 finds the unique rank-*r *decomposition of* $\mathcal{F}$.

Proof. Under the assumptions of Lemma 3.1.1, the tensor $\mathcal{F}_m$ has a unique generating matrix by Theorem 3.1.3 and the generating matrix G is uniquely determined by solving (3.7). The matrix $N(\xi)$ in (3.9) has distinct eigenvalues, so the vectors $\tilde{w}_i$ are determined by (3.10). Lemma 3.1.1 assumes $\{[u_i]_{\mathcal{O}_\alpha}\}_{i=1}^r$ are linearly independent, it implies that $\{[\tilde{w}_i]_{J_3}\}_{i=1}^r$ are also linearly independent. Thus, the systems (3.12) and (3.13) both have unique solutions. By the uniqueness of every step in the Algorithm 3.1.4, we conclude that Algorithm 3.1.4 finds the unique rank-r decomposition of $\mathcal{F}_m$. □

Algorithm 3.1.4 requires the tensor $\mathcal{F}_m$ to satisfy the condition of Lemma 3.1.1. Thus, the rank r should satisfy

$$r \leq \min\left\{ \binom{k}{p}, \binom{n-k-1}{m-p-1} \right\}.$$

In the following, we will find the largest rank that Algorithm 3.1.4 can compute for the given order m.

Lemma 3.1.6. *If* $n \geq \max\{2m-1, \frac{m^2}{4} - 1\}$, *then*

$$\begin{aligned} &\max\left(\binom{k^*}{p^*}, \binom{n-k^*-2}{m-p^*-1} \right) \\ &= \max_{p \in \mathbb{N}\cap[1,m-2]} \max_{k \in \mathbb{N}\cap[p,n-m+p]} \left(\min\left(\binom{k}{p}, \binom{n-k-1}{m-p-1} \right) \right), \end{aligned} \tag{3.16}$$

where $p^* = \lfloor \frac{m-1}{2} \rfloor$ *and* k^* *is largest* k *such that* $\binom{k}{p^*} \leq \binom{n-k-1}{m-p^*-1}$.

Proof. For a fixed $p \in [1, \frac{m-1}{2}] \cap \mathbb{N}$, it holds that $\binom{k}{p}$ is increasing in k and $\binom{n-k-1}{m-p-1}$ is decreasing in k. For the fixed p, let k_p be the largest k such that

$$\binom{k}{p} \leq \binom{n-k-1}{m-p-1}.$$

It holds that

$$r_p := \max_{k \in \mathbb{N} \cap [p, n-m+p]} \left(\min \left(\binom{k}{p}, \binom{n-k-1}{m-p-1} \right) \right) = \max \left(\binom{k_p}{p}, \binom{n-k_p-2}{m-p-1} \right).$$

For $p \in (\frac{m-1}{2}, m-2] \cap \mathbb{N}$ and $k \in [p, n-m+p]$, let $p' = m-p-1$ and $k' = n-k-1$. We can verify that $p' \in [1, \frac{m-1}{2}]$, $k' \in [p', n-m+p']$, and

$$\min \left(\binom{k}{p}, \binom{n-k-1}{m-p-1} \right) = \min \left(\binom{k'}{p'}, \binom{n-k'-1}{m-p'-1} \right).$$

Therefore, it holds that $\max_{p \in \mathbb{N} \cap [1, m-2]} r_p = \max_{p \in \mathbb{N} \cap [1, p^*]} r_p$. Next, we will prove $\max_p r_p = r_{p*}$ by showing $r_p \geq r_{p-1}$ for $p \in \mathbb{N} \cap [2, p^*]$.

When $p \leq \frac{m-1}{2}$ and $n \geq 2m-1$, we have $p \leq m-p-1 \leq n-1-\lfloor \frac{n-1}{2} \rfloor - p$. Hence,

$$\binom{\lfloor \frac{n-1}{2} \rfloor}{p} \leq \binom{n-1-\lfloor \frac{n-1}{2} \rfloor}{p} = \binom{n-1-\lfloor \frac{n-1}{2} \rfloor}{n-1-\lfloor \frac{n-1}{2} \rfloor - p} \leq \binom{n-1-\lfloor \frac{n-1}{2} \rfloor}{m-p-1}.$$

The above equation implies $k_p \geq \lfloor \frac{n-1}{2} \rfloor \geq \frac{n-2}{2}$.

If $k_{p-1} < k_p$, then it holds that

$$r_{p-1} \leq \binom{k_{p-1}+1}{p-1} \leq \binom{k_p}{p-1} = \binom{k_p}{p} \frac{p}{k_p - p + 1} \leq \binom{k_p}{p} \frac{m-1}{n-m} \leq \binom{k_p}{p} \leq r_p.$$

In the following proof, we show $r_{p-1} \leq r_p$ if $k_{p-1} \geq k_p$.

Case 1: $\binom{k_p}{p} > \binom{n-k_p-2}{m-p-1}$. In this case, $r_p = \binom{k_p}{p}$. It holds that

$$\binom{k'_p - C}{p'+1} = \binom{k'_p}{p'}\frac{k'_p - p'}{p'+1}\Pi_{i=1}^{C}\frac{k'_p - i - p'}{k'_p - i + 1},$$

$$\binom{k_p + C}{p-1} = \binom{k_p}{p}\frac{p}{k_p - p + 1}\Pi_{i=1}^{C}\frac{k_p + i}{k_p - p + 1 + i},$$

for $C \geq 0, p' = m - p - 1, k'_p = n - k_p - 1$. By direct computation, we have

$$\frac{k'_p - i - p'}{k'_p - i + 1}\frac{k_p + i}{k_p - p + 1 + i} \leq 1 \Leftrightarrow k_p m - np + n - k_p + im - i \geq 0.$$

The inequalities $n - m + p \geq k_p \geq \frac{n-2}{2}, p \leq \frac{m-1}{2}, n \geq 2m - 1, i \geq 0$ imply

$$k_p m - np + n - k_p + im - i \geq n - m + i(m-1) + 1 \geq 0.$$

It proves

$$\frac{k'_p - i - p'}{k'_p - i + 1}\frac{k_p + i}{k_p - p + 1 + i} \leq 1, \text{ for } i \geq 0. \tag{3.17}$$

If $p = \frac{m-1}{2}$, then $p = m - p - 1 = p'$ and $k_p = \lfloor \frac{n-1}{2} \rfloor$. If n is even, then $k_p = \frac{n-2}{2}$ and $\binom{k_p}{p} = \binom{n-k_p-2}{m-p-1}$, which does not satisfy the assumption of Case 1. If n is odd, then $k_p = \frac{n-1}{2}$ and $k_p = k'_p$. Thus, we have $\binom{k'_p}{p'} = \binom{k_p}{p}$ and $\frac{k'_p - p'}{p'+1}\frac{p}{k_p - p + 1} = \frac{k_p - p}{p+1}\frac{p}{k_p - p + 1} < 1$. These inequalities and (3.17) imply that

$$\binom{k_p + C}{p-1}\binom{k'_p - C}{p'+1} < \binom{k_p}{p}^2 \Rightarrow \min\left(\binom{k_p + C}{p-1}, \binom{k'_p - C}{p'+1}\right) < r_p.$$

Then, we consider $p \leq \frac{m-2}{2}$. Under the assumption of Case 1, we have

$$\begin{aligned}\binom{k'_p - C}{p'+1} &= \binom{k'_p}{p'}\frac{k'_p - p'}{p'+1}\Pi_{i=1}^{C}\frac{k'_p - i - p'}{k'_p - i + 1} \\ &= \binom{k'_p - 1}{p'}\frac{k'_p}{k'_p - p'}\frac{k'_p - p'}{p'+1}\Pi_{i=1}^{C}\frac{k'_p - i - p'}{k'_p - i + 1} \\ &< \binom{k_p}{p}\frac{k'_p}{p'+1}\Pi_{i=1}^{C}\frac{k'_p - i - p'}{k'_p - i + 1}.\end{aligned}$$

It holds that $\frac{k'_p}{p'+1}\frac{p}{k_p-p+1} \leq 1$ if and only if $k_p m + m + (p - m - n)p \geq 0$. We observe that $k_p m + m + (p - m - n)p$ is increasing in k_p and decreasing in p. When $p \leq \frac{m-2}{2}$, we have

$$k_p m + m + (p - m - n)p \geq \frac{n-2}{2}m + m + (\frac{m-2}{2} - m - n)\frac{m-2}{2} = n + 1 - \frac{m^2}{4}.$$

As a result, $\frac{k'_p}{p'+1}\frac{p}{k_p-p+1} \leq 1$ when $n \geq \frac{m^2}{4} - 1$. This inequality and (3.17) imply

$$\binom{k_p + C}{p-1}\binom{k'_p - C}{p'+1} < \binom{k_p}{p}^2 \Rightarrow \min\left(\binom{k_p + C}{p-1}, \binom{k'_p - C}{p'+1}\right) < r_p.$$

For all choices of p, the above inequality holds. Under the assumption that $k_{p-1} \geq k_p$, we have $r_{p-1} = \min\left(\binom{k_p+C}{p-1}, \binom{k'_p-C}{p'+1}\right)$ for some $C \geq 0$. It proves that $r_p > r_{p-1}$ when $n \geq \max\{2m-1, \frac{m^2}{4} - 1\}$ under the assumption of Case 1.

Case 2: $\binom{k_p}{p} \leq \binom{n-k_p-2}{m-p-1}$. In this case, $r_p = \binom{n-k_p-2}{m-p-1}$. Similar to Case 1, we can show

$$\binom{k_p + 1 + C}{p-1} < \binom{k'_p - 1}{p'}\frac{k_p+1}{k_p+1-p}\frac{p}{k_p-p+2}\Pi_{i=1}^{C}\frac{k_p+1+i}{k_p-p+2+i}$$

$$\binom{k'_p - 1 - C}{p'+1} < \binom{k'_p - 1}{p'}\frac{k'_p - p' - 1}{p'+1}\Pi_{i=1}^{C}\frac{k'_p - p' - 1 - i}{k'_p - i}$$

and $\frac{k_p+1+i}{k_p-p+2+i}\frac{k'_p-p'-1-i}{k'_p-i} \leq 1$ for $i \geq -1$. We observe that $\frac{k_p+1}{k_p+1-p}\frac{p}{k_p-p+2}\frac{k'_p-p'-1}{p'+1}$ is decreasing in k_p and increasing in p. Recall that $k \geq \frac{n-2}{2}, p \leq \frac{m-1}{2}$, so we can show

$$\frac{k_p+1}{k_p+1-p}\frac{p}{k_p-p+2}\frac{k_p'-p'-1}{p'+1} \leq 1 \Leftarrow 4n^2 - \beta n + \gamma \geq 0, \tag{3.18}$$

where $\beta = m^2 - 2m - 8$ and $\gamma = m^3 - 4m^2 - 4m + 16$. The inequality on the right of (3.18) is quadratic in n, so it holds when $\beta^2 - 16\gamma \leq 0$ or $n \geq \frac{\beta+\sqrt{\beta^2-16\gamma}}{8}$. The assumption $n \geq \max\{2m-1, \frac{m^2}{4} - 1\}$ implies that

$$n \geq \frac{m^2}{4} - 1 \geq \frac{\beta}{4} \geq \frac{\beta + \sqrt{\beta^2 - 16\gamma}}{8}.$$

Therefore, the inequality (3.18) holds. Similar to Case 1, it concludes the proof of $r_p > r_{p-1}$ for Case 2.

Summarizing everything above, we prove that (3.16) holds for $n \geq \max\{2m - 1, \frac{m^2}{4} - 1\}$. □

The following Theorem 3.1.7 provides the largest rank that Algorithm 3.1.4 can compute based on the result of Lemma 3.1.1.

Theorem 3.1.7. *Let $\mathcal{F}_m \in S^m(\mathbb{C}^{n+1})$ be the tensor as in (3.2). When $n \geq \max(2m - 1, \frac{m^2}{4} - 1)$, the largest rank r of $\mathcal{F}_m$ that Algorithm 3.1.4 can calculate is*

$$r_{max} = \max(\binom{k^*}{p^*}, \binom{n-2-k^*}{m-1-p^*}), \tag{3.19}$$

where $p^* = \lfloor\frac{m-1}{2}\rfloor$ and k^* is largest integer k such that $\binom{k}{p^*} \leq \binom{n-k-1}{m-p^*-1}$.

Proof. By Theorem 3.1.5, Algorithm 3.1.4 requires $\binom{k}{p} \geq r$ and $\binom{n-k-1}{m-p-1} \geq r$ to find a rank-r decomposition of tensor $\mathcal{F}_m$. For the given tensor $\mathcal{F}_m$ with dimension $n+1$ and order m, the largest computable rank of Algorithm 3.1.4 is

$$r_{\max} = \max_{k,p}\left(\min\left(\binom{k}{p}, \binom{n-k-1}{m-p-1}\right)\right), \tag{3.20}$$

where $p \in [1,\, m-2]$ and $k \in [p+1,\, n-m+p-1]$. Therefore, (3.19) is a direct result of Lemma 3.1.6. □

Remark 3.1.8. *The k^* in Theorem 3.1.7 can be obtained by solving*

$$\binom{k}{p^*} = \binom{n-k-1}{m-1-p^*}, \tag{3.21}$$

where the above binomial coefficients are generalized to binomial series for real number k. Let $\tilde{k} \in \mathbb{R}$ be the solution to (3.21), then $k^* = \lfloor \tilde{k} \rfloor$. Especially, when m is odd, we have $p^* = m-1-p^* = \frac{m-1}{2}$, $k^* = \lfloor \frac{n-1}{2} \rfloor$, and the corresponding largest rank is

$$r_{max} = \binom{\lfloor \frac{n-1}{2} \rfloor}{\frac{m-1}{2}}.$$

There is no uniform formula for the largest ranks when m is even. The largest ranks for some small orders are summarized in Table 3.1.

Table 3.1. The largest rank r that Algorithm 3.1.4 can compute.

m	the largest r
3	$\lfloor \frac{n-1}{2} \rfloor$
4	$\lfloor \frac{2n-1-\sqrt{8n-7}}{2} \rfloor$
5	$\binom{\lfloor \frac{n-1}{2} \rfloor}{2}$
6	$\max(\binom{\lfloor \tilde{k} \rfloor}{2}, \binom{n-\lfloor \tilde{k} \rfloor-2}{3})$, where $\Delta = \frac{9}{4}n^4 - \frac{47}{2}n^3 + \frac{353}{4}n^2 - \frac{412}{3}n + \frac{1889}{27}$ and $\tilde{k} = \sqrt[3]{-\frac{3}{2}(n-3)(n-4) + \sqrt{\Delta}} + \sqrt[3]{-\frac{3}{2}(n-3)(n-4) - \sqrt{\Delta}} + n - 3$
7	$\binom{\lfloor \frac{n-1}{2} \rfloor}{3}$

3.2 Incomplete Tensor Approximations and Error Analysis

When learning Gaussian mixture models, the subtensor $(\mathcal{F}_m)_{\Omega_m}$ is estimated from samples and is not exactly given. In such case, Algorithm 3.1.4 can still find a good low-rank approximation of $\mathcal{F}_m$. In this section, we discuss how to obtain a good tensor approximation of $\mathcal{F}_m$ and provide an error analysis for the approximation.

Let $\widehat{\mathcal{F}}_m$ be approximations of $\mathcal{F}_m$. Given the subtensor $(\widehat{\mathcal{F}}_m)_{\Omega_m}$, we can find a low-rank approximation of $\mathcal{F}_m$ following Algorithm 3.1.4. We define the matrix $A[\alpha, \widehat{\mathcal{F}}_m]$ and the vector $b[\alpha, \widehat{\mathcal{F}}_m]$ in the same way as in (3.6), for each $\alpha \in \mathscr{B}_1$. Then we have the following linear least square problem

$$\min_{g_\alpha \in \mathbb{C}^{\mathscr{B}_0}} \quad \left\| A[\alpha, \widehat{\mathcal{F}}_m] \cdot g_\alpha - b[\alpha, \widehat{\mathcal{F}}_m] \right\|^2 . \tag{3.22}$$

For each $\alpha \in \mathscr{B}_1$, we solve (3.22) to get $\widehat{G}[:, \alpha]$ which is an approximation of $G[:, \alpha]$. Combining all $\widehat{G}[:, \alpha]$'s, we get $\widehat{G} \in \mathbb{C}^{\mathscr{B}_0 \times \mathscr{B}_1}$ approximating the generating matrix G. Similar to (3.8), for $l = k+1, \ldots, n$, we define $N_l(\widehat{G})$ as an approximation of $N_l(G)$ and let

$$\widehat{N}(\xi) \coloneqq \xi_{k+1} N_{k+1}(\widehat{G}) + \cdots + \xi_n N_n(\widehat{G}), \tag{3.23}$$

where $\xi = (\xi_{k+1}, \ldots, \xi_n)$ is a generic vector. Let $\hat{v}_1, \ldots, \hat{v}_r$ be the unit length eigenvectors of $\widehat{N}(\xi)$ and

$$\hat{w}_i \coloneqq (\hat{v}_i^H N_{k+1}(\widehat{G}) \hat{v}_i, \cdots, \hat{v}_i^H N_n(\widehat{G}) \hat{v}_i) \quad i = 1, \ldots, r. \tag{3.24}$$

For the sets J_1, J_1^{-j}, J_2, J_3 defined in (3.11), we solve the linear least square problem

$$\min_{(\gamma_1, \ldots, \gamma_r)} \left\| (\widehat{\mathcal{F}}_m)_{J_1 \cdot J_2} - \sum_{i=1}^{r} \gamma_i \otimes [\hat{w}_i]_{J_3} \right\|^2 . \tag{3.25}$$

Let $(\hat{\gamma}_1, \ldots, \hat{\gamma}_r)$ be the minimizer of the above problem. Then, we consider the following linear least square problem

$$\min_{(v_1,\ldots,v_r)} \left\| (\widehat{\mathcal{F}}_m)_{x_j \cdot J_1^{-j} \cdot J_2} - \sum_{i=1}^{r} (v_i)_j \cdot \hat{\gamma}_i \otimes [\hat{w}_i]_{J_3} \right\|^2. \tag{3.26}$$

We obtain $(\hat{v}_1, \ldots, \hat{v}_r)$ by solving the above problem for $j = 1, \ldots, k$. Let $\hat{u} = (\hat{v}, \hat{w})$. Then, we have the following linear least square problem

$$\min_{(\lambda_1,\ldots,\lambda_r)} \left\| (\widehat{\mathcal{F}}_m)_{\Omega_m} - \sum_{i=1}^{r} \lambda_i \cdot \left(\begin{bmatrix} 1 \\ \hat{u}_r \end{bmatrix}^{\otimes m} \right)_{\Omega_m} \right\|^2. \tag{3.27}$$

Denote the minimizer of (3.27) as $(\hat{\lambda}_1, \ldots, \hat{\lambda}_r)$. For $i = 1, \ldots, r$, let

$$\hat{q}_i \coloneqq (\hat{\lambda}_i)^{1/m}(1, \hat{v}_i, \hat{w}_i).$$

Now, we obtain the approximation of the tensor $\mathcal{F}_m$

$$\mathcal{F}_m \approx (\hat{q}_1)^{\otimes m} + \cdots + (\hat{q}_r)^{\otimes m}.$$

This result may not be optimal due to sample errors. We can get a more accurate approximation by using $(\hat{q}_1, \ldots, \hat{q}_r)$ as starting points to solve the nonlinear optimization

$$\min_{(q_1,\ldots,q_r)} \left\| (\widehat{\mathcal{F}}_m)_{\Omega_m} - \sum_{i=1}^{r} (q_i^{\otimes m})_{\Omega_m} \right\|^2. \tag{3.28}$$

We denote the minimizer of the optimization (3.28) as $(q_1^*, \ldots, q_r^*)$.

We summarize the above calculations as a tensor approximation algorithm in Algorithm 3.2.1.

Algorithm 3.2.1. *(Incomplete symmetric tensor approximation.)*

Input: The rank r, the dimension d, the constant p, and the subtensor $(\widehat{\mathcal{F}}_m)_{\Omega_m}$ as in

(3.30).

Step 1. Determine the generating matrix $\widehat{G}$ *by solving (3.22) for each* $\alpha \in \mathscr{B}_1$.

Step 2. Choose a generic vector ξ *and define* $\widehat{N}(\xi)$ *as in* (3.23). *Calculate unit length eigenvectors of* $\widehat{N}(\xi)$ *and corresponding eigenvalues of each* $N_i(\widehat{G})$ *to define* $\hat{w}_i$ *as in* (3.24).

Step 3. Solve (3.25), (3.26) *and* (3.27) *to obtain the coefficients* $\hat{\lambda}_i$ *and vectors* $\hat{v}_i$.

Step 4. Let $\hat{q}_i := (\hat{\lambda}_i)^{1/m}(1, \hat{v}_i, \hat{w}_i)$ *for* $i = 1, \dots, r$. *Use* $\hat{q}_1, \dots, \hat{q}_r$ *as start points to solve the nonlinear optimization* (3.28) *and get an optimizer* $(q_1^*, \dots, q_r^*)$.

Output: The incomplete tensor approximation $(q_1^*)^{\otimes m} + \cdots + (q_r^*)^{\otimes m}$ *for* $\widehat{\mathcal{F}}_m$.

We can show that Algorithm 3.2.1 provides a good rank-r approximation when the input subtensor $(\widehat{\mathcal{F}}_m)_{\Omega_m}$ is close to exact tensors $\mathcal{F}_m$.

Theorem 3.2.2. *Let* $\mathcal{F}_m = \omega_1(\mu_1)^{\otimes m} + \cdots + \omega_r(\mu_r)^{\otimes m}$ *as in* (3.1) *and constants* k, p *be such that* $\min\left(\binom{k}{p}, \binom{n-k-1}{m-p-1}\right) \geq r$. *We assume the following conditions:*

(i) the scalars ω_i *and the leading entry of each* μ_i *are nonzero;*

(ii) the vectors $\{[(\mu_i)_{1:n}]_{\mathscr{B}_0}\}_{i=1}^r$ *are linearly independent;*

(iii) the vectors $\{[(\mu_i)_{1:n}]_{\mathcal{O}_\alpha}\}_{i=1}^r$ *are linearly independent for all* $\alpha \in \mathscr{B}_1$*;*

(iv) the eigenvalues of the matrix $N(\xi)$ *in (3.9) are distinct from each other.*

Let $q_i = (\omega_i)^{1/m}\mu_i$ *and* q_i^* *be the output vectors of Algorithm 3.2.1. If the distance* $\epsilon := \|(\widehat{\mathcal{F}}_m - \mathcal{F}_m)_{\Omega_m}\|$ *is small enough, then there exist scalars* $\tilde{\eta}_i, \eta_i^*$ *such that*

$$(\hat{\eta}_i)^m = (\eta_i^*)^m = 1, \quad \|\hat{\eta}_i \hat{q}_i - q_i\| = O(\epsilon), \quad \|\eta_i^* q_i^* - q_i\| = O(\epsilon),$$

up to a permutation of $(q_1, \ldots, q_r)$*, where the constants inside* $O(\cdot)$ *only depend on* $\mathcal{F}_m$ *and the choice of* ξ *in Algorithm 3.2.1.*

Proof. The vectors $(1, u_1)$, …, $(1, u_r)$ in (3.2) are scalar multiples of μ_1, …, μ_r respectively. By Conditions (ii) and (iii), the vectors $\{[u_i]_{\mathscr{B}_0}\}_{i=1}^r$ and $\{[u_i]_{\mathcal{O}_\alpha}\}_{i=1}^r$ are both linearly independent, which satisfies the condition of Lemma 3.1.1. Thus Conditions (i)-(iii) imply that there exists a unique generating matrix G for $\mathcal{F}_m$ by Theorem 3.1.3 and it can be calculated by (2.11). By Lemma 3.1.1, the matrix $A[\alpha, \mathcal{F}_m]$ has full column rank. It holds that

$$\|A[\alpha, \mathcal{F}_m] - A[\alpha, \widehat{\mathcal{F}}_m]\| \leq \epsilon, \ \|b[\alpha, \mathcal{F}_m] - b[\alpha, \widehat{\mathcal{F}}_m]\| \leq \epsilon, \tag{3.29}$$

for $\alpha \in \mathscr{B}_1$. When ϵ is small enough, the matrix $A[\alpha, \widehat{\mathcal{F}}_m]$ also has full column rank. Then the linear least square problems (3.22) have unique solutions and the collection of solutions $\widehat{G}$ satisfies that

$$\|G - \widehat{G}\| = O(\epsilon),$$

where $O(\epsilon)$ depends on $\mathcal{F}_m$ (see [14, Theorem 3.4]). Since $N_l(\widehat{G})$ is part of the generating matrix $\widehat{G}$ for each $l = k+1, \ldots, n$, we have

$$\|N_l(\widehat{G}) - N_l(G)\| \leq \|\widehat{G} - G\| = O(\epsilon), \quad l = k+1, \ldots, n,$$

which implies that $\|\widehat{N}(\xi) - N(\xi)\| = O(\epsilon)$. By condition (iv) we know that the matrix $\widehat{N}(\xi)$ has distinct eigenvalues $\hat{w}_1, \ldots, \hat{w}_r$ if ϵ is small enough. So the matrix $N(\xi)$ has a set of eigenvalues $\tilde{w}_i$ such that

$$\|\hat{w}_i - \tilde{w}_i\| = O(\epsilon).$$

This follows from Proposition 4.2.1 in [8]. The constants inside the above $O(\cdot)$ depend only on $\mathcal{F}_m$ and ξ. The vectors $\tilde{w}_1, \ldots, \tilde{w}_r$ are multiples of the vectors $(\mu_1)_{k+1:n}, \ldots, (\mu_r)_{k+1:n}$ respectively. Thus, we conclude that $[\tilde{w}_1]_{J_3}, \ldots, [\tilde{w}_r]_{J_3}$ are linearly independent by condition (iii). When ϵ is small, the vectors $[\hat{w}_1]_{J_3}, \ldots, [\hat{w}_r]_{J_3}$ are also linearly independent. For optimizers $\hat{\gamma}_i$, $\hat{v}_i$, $\hat{\lambda}_i$ of linear least square problems (3.25), (3.26) and (3.27), by [14, Theorem 3.4], we have

$$\|\hat{\gamma}_i - \gamma_i\| = O(\epsilon),\ \|\hat{v}_i - v_i\| = O(\epsilon),\ \|\hat{\lambda}_i - \lambda_i\| = O(\epsilon),$$

where constants inside $O(\cdot)$ depend on $\mathcal{F}_m$ and ξ. By Theorem 3.1.5, we have $\mathcal{F}_m = \sum_{i=1}^r \tilde{q}_i^{\otimes m}$ where $\tilde{q}_i = (\tilde{\lambda})^{1/m}(1, \tilde{v}_i, \tilde{w}_i)$.The rank-$r$ decomposition of $\mathcal{F}_m$ is unique up to scaling and permutation by Theorem 3.1.5. Thus, there exist scalars $\hat{\eta}_i$ such that $(\hat{\eta}_i)^m = 1$ and $\hat{\eta}_i\tilde{q}_i = q_i$, up to a permutation of $q_1, \ldots, q_r$. Then for $\hat{q}_i = (\hat{\lambda})^{1/m}(1, \hat{v}_i, \hat{w}_i)$, we have $\|\hat{\eta}_i\hat{q}_i - q_i\| = O(\epsilon)$ where constants inside $O(\cdot)$ depend on $\mathcal{F}_m$ and ξ.

Since $\|\hat{\eta}_i\hat{q}_i - q_i\| = O(\epsilon)$, we have $\|\mathcal{F}_m - (\sum_{i=1}^r (\hat{q}_i)^{\otimes m})_{\Omega_m}\| = O(\epsilon)$. For the minimizer $(q_1^*, \ldots, q_r^*)$ of (3.28), it holds that

$$\left\|\left(\widehat{\mathcal{F}}_m - \sum_{i=1}^r (q_i^*)^{\otimes m}\right)_{\Omega_m}\right\| \le \left\|\left(\widehat{\mathcal{F}}_m - \sum_{i=1}^r (\hat{q}_i)^{\otimes m}\right)_{\Omega_m}\right\| = O(\epsilon).$$

For the tensor $\mathcal{F}_m^* := \sum_{i=1}^r (q_i^*)^{\otimes m}$, we have

$$\|(\mathcal{F}_m^* - \mathcal{F}_m)_{\Omega_m}\| \le \|(\mathcal{F}_m^* - \widehat{\mathcal{F}}_m)_{\Omega_m}\| + \|(\widehat{\mathcal{F}}_m - \mathcal{F}_m)_{\Omega_m}\| = O(\epsilon)$$

If we apply Algorithm 3.2.1 to $(\mathcal{F}_m^*)_{\Omega_m}$, we will get the exact decomposition $\mathcal{F}_m^* = \sum_{i=1}^r (q_i^*)^{\otimes m}$. By repeating the above argument, similarly we can obtain that $\|\eta_i^* q_i^* - q_i\| = O(\epsilon)$ for some η_i^* such that $(\eta_i^*)^m = 1$, where the constants in $O(\cdot)$ only depend on $\mathcal{F}_m$ and ξ. □

3.3 Learning General Diagonal Gaussian Mixture

Let y be the random variable of a diagonal Gaussian mixture model and $y_1, \dots, y_N$ be i.i.d. samples drawn from the model. The moment tensors $M_m := \mathbb{E}[y^{\otimes m}]$ can be estimated as follows

$$\widehat{M}_m := \frac{1}{N}(y_1^{\otimes m} + \cdots + y_N^{\otimes m}).$$

Recall that $\mathcal{F}_m = \sum_{i=1}^r \omega_i \mu_i^{\otimes m}$. By Corollary 1.8 and (1.10), we have

$$(M_m)_{\Omega_m} = (\mathcal{F}_m)_{\Omega_m},$$

where Ω_m is the index set defined in (1.9). Let $\widehat{\mathcal{F}}_m$ be such that

$$(\widehat{\mathcal{F}}_m)_{\Omega_m} := (\widehat{M}_m)_{\Omega_m}. \tag{3.30}$$

We can apply Algorithm 3.2.1 to find the low-rank approximation of $\widehat{\mathcal{F}}_m$. Let $\widehat{\mathcal{F}}_m \approx \sum_{i=1}^r (q_i^*)^{\otimes m}$ be the tensor approximation generated by Algorithm 3.2.1. By Theorem 3.2.2, when $\epsilon = \|(\widehat{M}_m)_{\Omega_m} - (M_m)_{\Omega_m}\|$ is small, there exists $\eta_i \in \mathbb{C}$ such that $\eta_i^m = 1$ and $\|\eta_i q_i^* - (\omega_i)^{1/m}\mu_i\| = O(\epsilon)$. The η_i appears here because the vector q_i^* can be complex even though $\widehat{\mathcal{F}}_m$ is a real tensor. But ω_i, μ_i are both real in Gaussian mixture models. In practice, we can choose the η_i from all mth roots of 1 that minimizes $\|\mathrm{Im}(\eta_i q_i^*)\|$. Let

$$\check{q}_i := (\eta_i q_i^*). \tag{3.31}$$

We expect that $\check{q}_i \approx (\omega_i)^{1/m}\mu_i$. Then, we consider the tensor

$$\mathcal{F}_t = \omega_1 \mu_1^{\otimes t} + \cdots + \omega_r \mu_r^{\otimes t} \approx (\omega_1)^{\frac{m-t}{m}} (\check{q}_1)^{\otimes t} + \cdots + (\omega_r)^{\frac{m-t}{m}} (\check{q}_r)^{\otimes t},$$

where t is the smallest number such that $\binom{d}{t} \geq r$. It holds that $(\mathcal{F}_t)_{\Omega_t} = (M_t)_{\Omega_t} \approx (\widehat{M}_t)_{\Omega_t}$, so we obtain the scalars $(\omega_i)^{\frac{m-t}{m}}$ by solving the linear least square problem

$$\min_{(\beta_1,\ldots,\beta_r)\in\mathbb{R}^r_+} \left\| (\widehat{M}_t)_{\Omega_t} - \sum_{i=1}^r \beta_i \left((\check{q}_i)^{\otimes t}\right)_{\Omega_t} \right\|^2. \tag{3.32}$$

Let the optimizer of (3.32) be $(\beta_1^*, \ldots, \beta_r^*)$, then

$$\hat{\omega}_i = (\beta^*)^{\frac{m}{m-t}} \quad \text{and} \quad \hat{\mu}_i = q_i^*/(\beta_i^*)^{\frac{1}{m-t}} \tag{3.33}$$

should be reasonable approximations of ω_i and μ_i respectively.

To obtain more accurate results, we can use $(\hat{\omega}_1, \ldots, \hat{\omega}_r, \hat{\mu}_1, \ldots, \hat{\mu}_r)$ as starting points to solve the following nonlinear optimization

$$\begin{cases} \min\limits_{\substack{\omega_1,\ldots,\omega_r,\\ \mu_1,\ldots,\mu_r}} & \|(\widehat{M}_m)_{\Omega_m} - \sum\limits_{i=1}^r \omega_i(\mu_i^{\otimes m})_{\Omega_m}\|^2 + \|(\widehat{M}_t)_{\Omega_t} - \sum\limits_{i=1}^r \omega_i(\mu_i^{\otimes t})_{\Omega_t}\|^2 \\ \text{subject to} & \omega_1 + \cdots + \omega_r = 1,\, \omega_1, \ldots, \omega_r \geq 0, \end{cases} \tag{3.34}$$

and obtain the optimizer $(\omega_1^*, \ldots, \omega_r^*, \mu_1^*, \ldots, \mu_r^*)$.

Next, we will show how to calculate the diagonal covariance matrices. We define a label set

$$L_j = \{(j, j, i_1, \ldots, i_{m-2}) : 1 \leq i_1 < \cdots < i_{m-2} \leq d, \text{and } i_1 \neq j, \ldots, i_{m-2} \neq j\}.$$

For $(j, j, i_1, \ldots, i_{m-2}) \in L_j$, we have

$$(M_m)_{j,j,i_1,\ldots,i_{m-2}} = \sum_{i=1}^r \omega_i \left((\mu_i)_j(\mu_i)_j(\mu_i)_{i_1}\cdots(\mu_i)_{i_{m-2}} + \Sigma_{jj}^{(i)}(\mu_i)_{i_1}\cdots(\mu_i)_{i_{m-2}}\right).$$

The above equation is a direct result of (1.8) since all covariance matrices are diagonal.

Let

$$\mathcal{A} \coloneqq M_m - \mathcal{F}_m, \quad \widehat{\mathcal{A}} \coloneqq \widehat{M}_m - (\check{q}_1)^{\otimes m} - \cdots - (\check{q}_r)^{\otimes m}. \tag{3.35}$$

To get the estimation of covariance matrices $\Sigma_i = \mathrm{diag}(\sigma_{i1}^2, \ldots, \sigma_{id}^2)$, we solve the nonnegative linear least square problems ($j = 1, \ldots, d$)

$$\begin{cases} \min\limits_{(\theta_{1j},\ldots,\theta_{rj})} & \left\| \left(\widehat{\mathcal{A}}\right)_{L_j} - \sum\limits_{i=1}^{r} \theta_{ij}\omega_i^* \left((\mu_i^*)^{\otimes m-2}\right)_{\hat{L}_j} \right\|^2 \\ \text{subject to} & \theta_{1j} \geq 0, \ldots, \theta_{rj} \geq 0 \end{cases} \tag{3.36}$$

where $\hat{L}_j = \{(i_1, \ldots, i_{m-2}) : (j, j, i_1, \ldots, i_{m-2}) \in L_j\}$. The vector $((\mu_i^*)^{\otimes m-2})_{\hat{L}_j}$ has length $\binom{n}{m-2} \geq \binom{k}{p} \geq r$, where k, p are constants in Algorithm 3.2.1. Therefore, $((\mu_1^*)^{\otimes m-2})_{\hat{L}_j}, \ldots, ((\mu_r^*)^{\otimes m-2})_{\hat{L}_j}$ are generically linearly independent and hence (3.36) has a unique optimizer. Suppose the optimizer is $(\theta_{1j}^*, \ldots, \theta_{rj}^*)$. The covariance matrix $\hat{\Sigma}_i$ can be approximated as

$$\Sigma_i^* \coloneqq \{\mathrm{diag}((\theta_{i1}^*, \ldots, \theta_{id}^*))\}, \; \sigma_{ij}^* \coloneqq \sqrt{\theta_{ij}^*}. \tag{3.37}$$

The following is the complete algorithm to recover the unknown parameters $\{\mu_i, \Sigma_i, \Omega_i\}_{i=1}^r$.

Algorithm 3.3.1. *(Learning diagonal Gaussian mixture models.)*

Input: The mth order sample moment tensor $\widehat{M}_m$, the tth order sample moment tensor $\widehat{M}_t$, and the number of components r.

Step 1. Apply Algorithm 3.2.1 to subtensor $(\widehat{\mathcal{F}}_m)_{\Omega_m}$ defined in (3.30). *Let $(q_1^*)^{\otimes m} + \cdots + (q_r^*)^{\otimes m}$ be the output incomplete tensor approximation for $\widehat{\mathcal{F}}_m$.*

Step 2. For $i = 1, \ldots, r$, we choose η_i such that $\eta_i^m = 1$ and it minimizes $\|Im(\eta_i q_i^*)\|$. Let $\check{q}_i = Re(\eta_i q_i^*)$ as in (3.31).

Step 3. Solve (3.32) to get the optimizer $(\beta_1^*, \ldots, \beta_r^*)$ and compute $\hat{\omega}_i$, $\hat{\mu}_i$ as in (3.33) *for $i = 1, \ldots, r$.*

Step 4. Use $(\hat{\omega}_1, \ldots, \hat{\omega}_r, \hat{\mu}_1, \ldots, \hat{\mu}_r)$ *as starting points to solve* (3.34) *to obtain the optimizer* $(\omega_1^*, \ldots, \omega_r^*, \mu_1^*, \ldots, \mu_r^*)$.

Step 5. Solve the optimization (3.36) to get optimizers θ_{ij}^* *and then compute* Σ_i^* *as in (3.37).*

Output: Mixture Gaussian parameters $(\omega_i^*, \mu_i^*, \Sigma_i^*), i = 1, \ldots, r$.

When the sampled moment tensors are close to the accurate moment tensors, the parameters generated by Algorithm 3.3.1 are close to the true model parameters. The analysis is shown in the following theorem.

Theorem 3.3.2. *Given a d-dimensional diagonal Gaussian mixture model with parameters* $\{(\omega_i, \mu_i, \Sigma_i) : i \in [r]\}$ *and* r *no greater than the* $r_{\max}$ *in* (3.19). *Let* $\{(\omega_i^*, \mu_i^*, \Sigma_i^*) : i \in [r]\}$ *be the output of Algorithm 3.3.1. If the distance* $\epsilon := \max(\|\widehat{M}_m - M_m\|, \|\widehat{M}_t - M_t\|)$ *is small enough,* $(\mu_1^{\otimes t})_{\Omega_t}, \ldots, (\mu_r^{\otimes t})_{\Omega_t}$ *are linearly independent, and the tensor* $\mathcal{F}_m = \sum_{i=1}^r \omega_i \mu_i^{\otimes m}$ *satisfies the conditions of Theorem 3.2.2, then*

$$\|\mu_i^* - \mu_i\| = O(\epsilon), \|\omega_i^* - \omega_i\| = O(\epsilon), \|\Sigma_i^* - \Sigma_i\| = O(\epsilon),$$

where the constants inside $O(\cdot)$ *depend on parameters* $\{(\omega_i, \mu_i, \Sigma_i) : i \in [r]\}$ *and the choice of* ξ *in Algorithm 3.3.1.*

Proof. We have

$$\|(\widehat{\mathcal{F}}_m - \mathcal{F}_m)_{\Omega_m}\| = \|(\widehat{M}_m - M_m)_{\Omega_m}\| \leq \epsilon,$$
$$\|(\widehat{\mathcal{F}}_t - \mathcal{F}_t)_{\Omega_t}\| = \|(\widehat{M}_t - M_t)_{\Omega_t}\| \leq \epsilon.$$

and $\mathcal{F}_m$, $\mathcal{F}_t$ satisfy conditions of Theorem 3.2.2. They imply that $\|\eta_i^* q_i^* - q_i\| = O(\epsilon)$ for some $(\eta_i^*)^m = 1$ by Theorem 3.2.2. The constants inside $O(\epsilon)$ depend on the parameters of the Gaussian model and vector ξ. Since vectors q_i are real, we have $\|\mathrm{Im}(\eta_i^* q_i^*)\| = O(\epsilon)$. When ϵ is small enough, such η_i^* minimizes $\|\mathrm{Im}(\eta_i^* q_i^*)\|$ and we have

$$\|\mathrm{Re}(\eta_i^* q_i^*) - q_i\| \leq \|\eta_i^* q_i^* - q_i\| = O(\epsilon).$$

Let $\check{q}_i := \mathrm{Re}(\eta_i^* q_i^*)$. When ϵ is small, vectors $(\check{q}_1^{\otimes t})_{\Omega_t}, \ldots, (\check{q}_r^{\otimes t})_{\Omega_t}$ are linearly independent since $(\mu_1^{\otimes t})_{\Omega_t}, \ldots, (\mu_r^{\otimes t})_{\Omega_t}$ are linearly independent by our assumption. It implies that the problem (3.32) has a unique solution. The weights $\hat{\omega}_i$ and mean vectors $\hat{\mu}_i$ can be calculated by (3.33). Since $\|(\widehat{M}_t - M_t)_{\Omega_t}\| \leq \epsilon$ and $\|\check{q}_i - q_i\| = O(\epsilon)$, we have $\|\omega_i - \hat{\omega}_i\| = O(\epsilon)$ (see [14, Theorem 3.4]). The approximation error for the mean vectors is

$$\|\hat{\mu}_i - \mu_i\| = \|\check{q}_i/(\hat{\omega}_i)^{1/m} - q_i/(\omega_i)^{1/m}\| = O(\epsilon).$$

The constants inside $O(\epsilon)$ depend on parameters of the Gaussian mixture model and ξ.

We obtain optimizers ω_i and μ_i by solving the problem (3.34), so it holds

$$\left\| (\widehat{M}_m)_{\Omega_m} - \sum_{i=1}^{r} \omega_i^* \left((\mu_i^*)^{\otimes m} \right)_{\Omega_m} \right\| = O(\epsilon).$$

Let $\mathcal{F}_m^* := \sum_{i=1}^r \omega_i^* (\mu_i^*)^{\otimes m}$ and $\mathcal{F}_t^* := \sum_{i=1}^r \omega_i^* (\mu_i^*)^{\otimes t}$, then

$$\|(\mathcal{F}_m^* - \mathcal{F}_m)_{\Omega_m}\| \leq \|(\widehat{\mathcal{F}}_m - \mathcal{F}_m)_{\Omega_m}\| + \|(\widehat{\mathcal{F}}_m - \mathcal{F}_m^*)_{\Omega_m}\| = O(\epsilon).$$
$$\|(\mathcal{F}_t^* - \mathcal{F}_t)_{\Omega_t}\| \leq \|(\widehat{\mathcal{F}}_t - \mathcal{F}_t)_{\Omega_t}\| + \|(\widehat{\mathcal{F}}_t - \mathcal{F}_t^*)_{\Omega_t}\| = O(\epsilon).$$

By Theorem 3.2.2, we have $\|(\omega_i^*)^{1/m} \mu_i^* - q_i\| = O(\epsilon)$. Since we are optimizing (3.34), it also holds that

$$\left\| (\widehat{M}_t)_{\Omega_t} - \sum_{i=1}^{r} \omega_i \left((\mu_i^*)^{\otimes t} \right)_{\Omega_t} \right\| = \left\| (\widehat{M}_t)_{\Omega_t} - \sum_{i=1}^{r} (\omega_i^*)^{\frac{m-t}{m}} \left(((\omega_i^*)^{1/m} \mu_i^*)^{\otimes t} \right)_{\Omega_t} \right\| = O(\epsilon).$$

Combining the above with $\|(\widehat{M}_t - M_t)_{\Omega_t}\| = O(\epsilon)$, we get $\|(\omega_i^*)^{\frac{m-t}{m}} - \omega_i^{\frac{m-t}{m}}\| = O(\epsilon)$ by [14, Theorem 3.4] and hence $\|\omega_i^* - \omega_i\| = O(\epsilon)$. For mean vectors μ_i we have

$$\|\mu_i^* - \mu_i\| = \|((\omega_i^*)^{1/m}\mu_i^*)/(\omega_i^*)^{1/m} - q_i/(\omega_i)^{1/m}\| = O(\epsilon).$$

The constants inside the above $O(\cdot)$ only depend on parameters $\{(\omega_i, \mu_i, \Sigma_i) : i \in [r]\}$ and ξ.

We obtain the covariance matrices Σ_i by solving (3.36). It holds that

$$\|\omega_i^*(\mu_i^*)^{\otimes(m-2)} - \omega_i\mu_i^{\otimes(m-2)}\| = O(\epsilon),$$
$$\|\widehat{\mathcal{A}} - \mathcal{A}\| \leq \|\widehat{M}_m - M_m\| + \|\sum_{i=1}^{r}(q_i^*)^m - \mathcal{F}_m\| \leq O(\epsilon),$$

where $\widehat{\mathcal{A}}$ and $\mathcal{A}$ are defined in (3.35). The tensor $\mathcal{F}_m$ satisfies the condition of Theorem 3.2.2, so the tensors $\mu_1^{\otimes(m-2)}, \ldots, \mu_r^{\otimes(m-2)}$ are linearly independent. It implies that $\{\omega_i^*(\mu_i^*)^{\otimes(m-2)}\}_{i=1}^r$ are linearly independent when ϵ is small. Therefore, (3.36) has a unique solution for each j. By [14, Theorem 3.4], we have

$$\|(\sigma_{ij}^*)^2 - (\sigma_{ij})^2\| = O(\epsilon).$$

It implies that $\|\Sigma_i^* - \Sigma_i\| = O(\epsilon)$, where the constants inside $O(\cdot)$ only depend on parameters $\{(\omega_i, \mu_i, \Sigma_i) : i \in [r]\}$ and ξ. □

Remark 3.3.3. *Given the dimension d and the highest order of moment m, the largest number of components in the Gaussian mixture model that Algorithm 3.3.1 can learn is the same as the largest rank r_{max} as in Theorem 3.1.7, i.e.,*

$$r_{max} = \max(\binom{k^*}{p^*}, \binom{d-3-k^*}{m-1-p^*}),$$

where $p^* = \lfloor \frac{m-1}{2} \rfloor$ and k^* is largest integer k such that

$$\binom{k}{p^*} \le \binom{d-k-2}{m-p^*-1}.$$

Given a d-dimensional Gaussian mixture model with r components, we can use Theorem 3.1.7 to obtain the smallest order m required for the Algorithm 3.3.1 and then apply Algorithm 3.3.1 to learn the Gaussian mixture model using the mth order moment.

3.4 Numerical Experiments

First, we present numerical experiments for Algorithm 3.1.4. We construct

$$\mathcal{F}_m = \sum_{i=1}^{r} q_i^{\otimes m} \in \mathrm{S}^m(\mathbb{R}^d) \tag{3.38}$$

by randomly generating each $q_i \in \mathbb{R}^d$ in Gaussian distribution by the `randn` function in `MATLAB`. Then we apply Algorithm 3.1.4 to the subtensor $(\mathcal{F}_m)_{\Omega_m}$ to calculate the rank-r tensor decomposition. The relative error of tensors and components are used to measure the decomposition result

$$\text{decomp-err}_m := \frac{\|(\mathcal{F}_m - \widetilde{\mathcal{F}}_m)_{\Omega_m}\|}{\|(\mathcal{F}_m)_{\Omega_m}\|}, \quad \text{vec-err-max} := \max_i \frac{\|q_i - \tilde{q}_i\|}{\|q_i\|},$$

where $\widetilde{\mathcal{F}}_m$, $\tilde{q}_i$ are output of Algorithm 3.1.4. We choose the values of d, m as

$$d = 15,\ 25,\ 30,\ 40, \quad m = 3,\ 4,\ 5,\ 6,\ 7,$$

and r as largest computable rank in Theorem 3.1.7 given d and m. For each (d, m, r), we generate 100 random instances, except for the case $(40, 7, 969)$ where 20 instances are generated due to the long computation time. The min, average, and max relative errors of tensors for each dimension d, order m, and the average relative errors of component vectors are shown in Table 3.2. The results show that Algorithm 3.1.4 finds the correct decomposition of randomly generated tensors.

Table 3.2. The performance of Algorithm 3.1.4

d	m	r	decomp-err$_m$ min	decomp-err$_m$ average	decomp-err$_m$ max	vec-err-max
15	3	6	$1.7 \cdot 10^{-15}$	$3.1 \cdot 10^{-12}$	$1.7 \cdot 10^{-10}$	$1.1 \cdot 10^{-11}$
	4	8	$4.0 \cdot 10^{-15}$	$7.8 \cdot 10^{-10}$	$7.7 \cdot 10^{-8}$	$1.2 \cdot 10^{-10}$
	5	15	$1.9 \cdot 10^{-14}$	$2.5 \cdot 10^{-11}$	$8.7 \cdot 10^{-10}$	$9.1 \cdot 10^{-11}$
	6	20	$5.2 \cdot 10^{-13}$	$2.3 \cdot 10^{-10}$	$1.2 \cdot 10^{-8}$	$9.5 \cdot 10^{-10}$
	7	20	$7.4 \cdot 10^{-14}$	$1.7 \cdot 10^{-10}$	$1.3 \cdot 10^{-8}$	$3.4 \cdot 10^{-10}$
25	3	11	$9.3 \cdot 10^{-15}$	$7.3 \cdot 10^{-12}$	$6.3 \cdot 10^{-10}$	$1.3 \cdot 10^{-11}$
	4	16	$6.1 \cdot 10^{-14}$	$1.0 \cdot 10^{-10}$	$9.1 \cdot 10^{-9}$	$3.5 \cdot 10^{-10}$
	5	55	$2.9 \cdot 10^{-12}$	$4.4 \cdot 10^{-9}$	$1.2 \cdot 10^{-7}$	$3.8 \cdot 10^{-8}$
	6	84	$9.3 \cdot 10^{-11}$	$7.2 \cdot 10^{-8}$	$1.7 \cdot 10^{-6}$	$4.6 \cdot 10^{-7}$
	7	165	$1.4 \cdot 10^{-10}$	$1.4 \cdot 10^{-7}$	$4.1 \cdot 10^{-6}$	$1.7 \cdot 10^{-6}$
30	3	14	$3.2 \cdot 10^{-14}$	$7.1 \cdot 10^{-12}$	$2.2 \cdot 10^{-10}$	$4.3 \cdot 10^{-11}$
	4	21	$3.6 \cdot 10^{-13}$	$1.6 \cdot 10^{-10}$	$2.4 \cdot 10^{-8}$	$3.9 \cdot 10^{-10}$
	5	91	$3.3 \cdot 10^{-11}$	$1.5 \cdot 10^{-7}$	$5.3 \cdot 10^{-6}$	$6.2 \cdot 10^{-7}$
	6	136	$1.0 \cdot 10^{-10}$	$1.7 \cdot 10^{-7}$	$8.3 \cdot 10^{-6}$	$1.9 \cdot 10^{-6}$
	7	364	$2.4 \cdot 10^{-8}$	$2.4 \cdot 10^{-6}$	$2.7 \cdot 10^{-5}$	$4.1 \cdot 10^{-5}$
40	3	19	$9.4 \cdot 10^{-14}$	$5.9 \cdot 10^{-12}$	$7.6 \cdot 10^{-11}$	$2.4 \cdot 10^{-11}$
	4	29	$4.1 \cdot 10^{-13}$	$5.4 \cdot 10^{-11}$	$6.9 \cdot 10^{-10}$	$1.4 \cdot 10^{-10}$
	5	171	$1.6 \cdot 10^{-10}$	$1.3 \cdot 10^{-7}$	$1.4 \cdot 10^{-6}$	$1.1 \cdot 10^{-6}$
	6	286	$4.5 \cdot 10^{-9}$	$5.9 \cdot 10^{-6}$	$1.1 \cdot 10^{-4}$	$6.0 \cdot 10^{-5}$
	7	969	$7.8 \cdot 10^{-7}$	$1.2 \cdot 10^{-5}$	$4.3 \cdot 10^{-5}$	$3.7 \cdot 10^{-4}$

Then we present numerical experiments for exploring the incomplete symmetric tensor approximation quality of Algorithm 3.2.1. We first randomly generate the rank-r symmetric $\mathcal{F}$ as in (3.38). Then we generate a random tensor $\mathcal{E}$ with the same dimension and order as $\mathcal{F}_m$ and scale it to a given norm ϵ, i.e. $\|\mathcal{E}_m\| = \epsilon$. Let $\widehat{\mathcal{F}}_m = \mathcal{F}_m + \mathcal{E}_m$. Algorithm 3.2.1 is applied to the subtensor $(\widehat{\mathcal{F}}_m)_\Omega$ to compute the rank-r approximation $\mathcal{F}^*_m$. The approximation quality of $\mathcal{F}^*_m$ can be measured by the absolute error and the relative error

$$\text{abs-err}_m := \|(\mathcal{F}_m^* - \mathcal{F}_m)_{\Omega_m}\|, \quad \text{rel-err}_m := \frac{\|(\mathcal{F}_m^* - \widehat{\mathcal{F}}_m)_{\Omega_m}\|}{\|(\mathcal{E}_m)_{\Omega_m}\|}.$$

We choose the values of d, m, ϵ as

$$d = 15,\, 25, \quad m = 3,\, 4,\, 5,\, 6, \quad \epsilon = 0.1,\, 0.01,\, 0.001,$$

and r as largest computable rank in Theorem 3.1.7 given d and m. For each (d, m, r, ϵ), we generate 100 instances of $\widehat{\mathcal{F}}_m$(for the case $(25, 6, r, \epsilon)$, 20 instances are generated due to long computational time) and record the minimum, average, maximum of abs-err$_m$, rel-err$_m$ respectively. For the case when $d = 15$, the results are reported in Table 3.3. For the case when $d = 25$, the results are reported in Table 3.4. For all instances, the output tensor of Algorithm 3.2.1 provides a good rank-r approximation.

Table 3.3. The performance of Algorithm 3.2.1 when $d = 15$

			rel-error			abs-error		
m	r	ϵ	min	average	max	min	average	max
3	6	0.1	0.8452	0.8953	0.9258	0.0378	0.0444	0.0534
		0.01	0.8549	0.8947	0.9280	0.0037	0.0045	0.0052
		0.001	0.8581	0.8969	0.9337	$3.5 \cdot 10^{-4}$	$4.4 \cdot 10^{-4}$	$5.1 \cdot 10^{-4}$
4	8	0.1	0.9382	0.9544	0.9666	0.0256	0.0298	0.0346
		0.01	0.9409	0.9569	0.9700	0.0024	0.0029	0.0034
		0.001	0.9333	0.9547	0.9692	$2.5 \cdot 10^{-4}$	$3.0 \cdot 10^{-4}$	$3.6 \cdot 10^{-4}$
5	15	0.1	0.9521	0.9612	0.9689	0.0248	0.0275	0.0306
		0.01	0.9529	0.9613	0.9690	0.0025	0.0028	0.0030
		0.001	0.9539	0.9615	0.9704	$2.4 \cdot 10^{-4}$	$2.7 \cdot 10^{-4}$	$3.0 \cdot 10^{-4}$
6	20	0.1	0.9625	0.9697	0.9767	0.0215	0.0244	0.0271
		0.01	0.9620	0.9694	0.9737	0.0023	0.0025	0.0027
		0.001	0.9619	0.9696	0.9769	$2.1 \cdot 10^{-4}$	$2.4 \cdot 10^{-4}$	$2.7 \cdot 10^{-4}$

Table 3.4. The performance of Algorithm 3.2.1 when $d = 25$

			rel-error			abs-error		
m	r	ϵ	min	average	max	min	average	max
3	11	0.1	0.9248	0.9377	0.9488	0.0316	0.0347	0.0380
		0.01	0.9257	0.9380	0.9484	0.0032	0.0035	0.0038
		0.001	0.9239	0.9383	0.9504	$3.1 \cdot 10^{-4}$	$3.4 \cdot 10^{-4}$	$3.8 \cdot 10^{-4}$
4	16	0.1	0.9809	0.9840	0.9861	0.0166	0.0178	0.0194
		0.01	0.9813	0.9839	0.9868	0.0016	0.0018	0.0019
		0.001	0.9808	0.9838	0.9860	$1.7 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	$1.9 \cdot 10^{-4}$
5	55	0.1	0.9854	0.9870	0.9878	0.0156	0.0161	0.0170
		0.01	0.9858	0.9871	0.9884	0.0015	0.0016	0.0017
		0.001	0.9856	0.9870	0.9882	$1.5 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$	$1.7 \cdot 10^{-4}$
6	84	0.1	0.9938	0.9940	0.9943	0.0106	0.0109	0.0111
		0.01	0.9939	0.9942	0.9946	0.0011	0.0011	0.0011
		0.001	1.0001	1.0046	1.0102	$1.6 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	$2.1 \cdot 10^{-4}$

Next, we explore the performance of Algorithm 3.3.1 for learning diagonal Gaussian mixture model. We compare it with the classical EM algorithm using the `MATLAB` function `fitgmdist` (`MaxIter` is set to be 100 and `RegularizationValue` is set to be 0.001). The dimension $d = 15$ and the orders of tensors $m = 3, 4, 5, 6$ are tested. The largest possible values of r as in Theorem 3.1.7 are tested for each (d, m). We generate 20 random instances of $\{(\omega_i, \mu_i, \Sigma_i) : i = 1, \ldots, r\}$ for each (d, m). For the weights $\omega_1, \ldots, \omega_r$, we randomly generate a positive vector $s \in \mathbb{R}^r$ and let $\omega_i = \frac{s_i}{\sum_{i=1}^r \omega_i}$. For each diagonal covariance matrix $\Sigma_i \in \mathbb{R}^{d \times d}$, we use the square of a random vector generated by `MATLAB` function `randn` to be diagonal entries. Each example is generated from one of r component Gaussians and the probability that the sample comes from the ith Gaussian is the weight ω_i. Algorithm 3.3.1 and EM algorithm are applied to learn the Gaussian mixture model from samples. After obtaining estimated parameters $(\omega_i, \mu_i, \Sigma_i)$ of the model, the likelihood of the sample for each component Gaussian distribution is calculated and we assign the sample to the group that corresponds to the maximum likelihood. We use classification accuracy, i.e. the ratio of correct assignments, to measure the performance of two algorithms. The accuracy comparison between two algorithms is shown in Table 3.5. As one can see, the performance of Algorithm 3.3.1 is better than EM algorithm in all tested cases.

Table 3.5. Comparison between Algorithm 3.3.1 and EM for learning Gaussian mixtures

			accuracy	
d	m	r	Algorithm 3.3.1	EM
15	3	6	0.9839	0.9567
	4	8	0.9760	0.9451
	5	15	0.9639	0.9382
	6	20	0.9423	0.9285

Acknowledgement. The Chapter 3, in full, has been submitted for publication [65]. The dissertation author coauthored this paper with Nie, Jiawang and Yang, Zi.

Bibliography

[1] D. Achlioptas and F. McSherry, On spectral learning of mixtures of distributions, *International Conference on Computational Learning Theory*, pp. 458–469, 2005.

[2] A. Anandkumar, R. Ge, D. Hsu, S. Kakade, and M. Telgarsky, Tensor decompositions for learning latent variable models, *Journal of Machine Learning Research*, 15, pp. 2773–2832, 2014.

[3] M. Belkin and K. Sinha, Toward learning Gaussian mixtures with arbitrary separation, *COLT*, 2010.

[4] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches, *IEEE journal of selected topics in applied earth observations and remote sensing*, 5(2):354–379, 2012.

[5] P. Breiding and N. Vannieuwenhoven. The condition number of join decompositions. *SIAM Journal on Matrix Analysis and Applications*, 39(1):287–309, 2018.

[6] P. Breiding and N. Vannieuwenhoven. The condition number of Riemannian approximation problems. *SIAM Journal on Optimization*, 31(1):1049–1077, 2021.

[7] C. Brubaker and S. Vempala, Isotropic PCA and affine-invariant clustering, *Building Bridges*, pages 241–281. Springer, 2008.

[8] F. Chatelin, Eigenvalues of matrices: revised edition, *SIAM*, 2012.

[9] K. Chaudhuri, S. Kakade, K. Livescu, and K. Sridharan, Multi-view clustering via canonical correlation analysis, *Proceedings of the 26th annual international conference on machine learning*, pages 129–136, 2009.

[10] K. Chaudhuri and S. Rao, Learning Mixtures of Product Distributions Using Correlations and Independence, *COLT*, pages 9–20, 2008.

[11] P. Comon, L.-H. Lim, Y. Qi, and K. Ye, Topology of tensor ranks, *Advances in Mathematics*, 367:107128, 2020.

[12] S. Dasgupta, Learning mixtures of Gaussians, *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 634–644. IEEE, 1999.

[13] S. Dasgupta and L. Schulman, A two-round variant of EM for Gaussian mixtures, *arXiv preprint arXiv:1301.3850*, 2013.

[14] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[16] V. De Silva and L.-H. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, *SIAM. J. Matrix Anal. Appl.*, 30(3), 1084–1127, 2008.

[17] M. Dixit, N. Rasiwasia, and N. Vasconcelos, Adapted Gaussian models for image classification, *CVPR 2011*, pages 937–943, 2011.

[18] I. Domanov and L. De Lathauwer, On the uniqueness of the canonical polyadic decomposition of third-order tensors—Part II: Uniqueness of the overall decomposition, *SIAM Journal on Matrix Analysis and Applications*, 34(3), pages 876–903, 2013.

[19] M. Dressler, J. Nie, and Z. Yang, Separability of Hermitian tensors and PSD decompositions, *Preprint*, 2020. arXiv:2011.08132

[20] J. Nie, *Sum of squares methods for minimizing polynomial forms over spheres and hypersurfaces*, Frontiers of Mathematics in China 7, 321–346, 2012.

[21] J. Nie, *Moment and Polynomial Optimization*, SIAM, 2023.

[22] J. Nie and X. Zhang, Real eigenvalues of nonsymmetric tensors, *Computational Optimization and Applications* 70(1), 1–32, 2018.

[23] L. Fialkow and J. Nie, The truncated moment problem via homogenization and flat extensions, *Journal of Functional Analysis*, 263(6), 1682–1700, 2012.

[24] S. Friedland, Remarks on the symmetric rank of symmetric tensors, *SIAM J. Matrix Anal. Appl.*, 37(1), 320–337, 2016.

[25] S. Friedland and L.-H. Lim, Nuclear norm of higher-order tensors, *Mathematics of Computation*, 87(311), 1255–1281, 2018.

[26] R. Ge, Q. Huang, and S. M. Kakade, Learning mixtures of Gaussians in high dimensions, *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 761–770, 2015.

[27] F. Ge, Y. Ju, Z. Qi, and Y. Lin. Parameter estimation of a gaussian mixture model for wind power forecast error by riemann l-bfgs optimization. *IEEE Access*, 6:38892–38899, 2018.

[28] M. Haas, S. Mittnik, and M. S. Paolella, Modelling and predicting market risk with Laplace–Gaussian mixture distributions, *Applied Financial Economics*, 16(15), 1145–1162, 2006.

[29] C. J. Hillar and L.-H. Lim, Most tensor problems are NP-hard, *J. ACM*, 60(6), Art. 45, 39, 2013

[30] D. Hsu and S. M. Kakade, Learning mixtures of spherical Gaussians: moment methods and spectral decompositions, *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 11–20, 2013.

[31] A. T. Kalai, A. Moitra, and G. Valiant, Efficiently learning mixtures of two Gaussians, *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 553–562, 2010.

[32] R. Kannan, H. Salmasian, and S. Vempala, The spectral method for general mixture models, *International Conference on Computational Learning Theory*, pages 444–457. Springer, 2005.

[33] S. Karpagavalli and E. Chandra. A review on automatic speech recognition architecture and approaches. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 9(4):393–404, 2016.

[34] J. M. Landsberg, *Tensors: geometry and applications*, Graduate Studies in Mathematics, vol. 128, American Mathematical Society, Providence, RI, 2012.

[35] D. -S. Lee, Effective Gaussian mixture learning for video background subtraction, *IEEE transactions on pattern analysis and machine intelligence*, 27(5):827–832, 2005.

[36] L.-H. Lim, *Tensors and hypermatrices*, in: L. Hogben (Ed.), *Handbook of linear algebra*, 2nd Ed., CRC Press, Boca Raton, FL, 2013.

[37] Y. Ma, Q. Jin, X. Mei, X. Dai, F. Fan, H. Li, and J. Huang, Hyperspectral unmixing with Gaussian mixture model and low-rank representation, *Remote Sensing*, 11(8), 911, 2019.

[38] M. Magdon-Ismail and J. T. Purnell, Approximating the covariance matrix of gmms with low-rank perturbations, *International Conference on Intelligent Data Engineering and Automated Learning*, pages 300–307, 2010.

[39] C. Mu, B. Huang, J. Wright, and D. Goldfarb, *Square deal: lower bounds and improved relaxations for tensor recovery*, Proceeding of the International Conference on Machine Learning (PMLR), 32(2),73-81, 2014.

[40] J. Nie and B. Sturmfels, Matrix cubes parameterized by eigenvalues, *SIAM journal on matrix analysis and applications* 31 (2), 755–766, 2009.

[41] J. Nie, The hierarchy of local minimums in polynomial optimization, *Mathematical programming* 151 (2), 555–583.

[42] J. Nie, Linear optimization with cones of moments and nonnegative polynomials, *Mathematical Programming*, 153(1), 247–274, 2013.

[43] J. Nie, Generating polynomials and symmetric tensor decompositions, *Foundations of Computational Mathematics*, 17(2), 423–465, 2017.

[44] J. Nie, Symmetric tensor nuclear norms, *SIAM J. Appl. Algebra Geometry*, 1(1), 599–625, 2017.

[45] J. Nie, Low rank symmetric tensor approximations, *SIAM Journal on Matrix Analysis and Applications*, 38(4), 1517–1540, 2017.

[46] J. Nie, Tight relaxations for polynomial optimization and Lagrange multiplier expressions, *Mathematical Programming* 178 (1-2), 1–37, 2019.

[47] J. Nie and K. Ye, Hankel tensor decompositions and ranks, *SIAM Journal on Matrix Analysis and Applications*, vol. 40, no. 2, pp. 486–516, 2019.

[48] J. Nie and Z. Yang, Hermitian tensor decompositions, *SIAM Journal on Matrix Analysis and Applications*, 41 (3), 1115-1144, 2020

[49] K. Pearson, Contributions to the mathematical theory of evolution, *Philosophical Transactions of the Royal Society of London. A*, 185, 71–110, 1894.

[50] H. Permuter, J. Francos, and I. Jermyn, A study of Gaussian mixture models of color and texture features for image classification and segmentation, *Pattern Recognition*, 39(4), 695–706, 2006.

[51] D. Povey, L. Burget, M. Agarwal, P. Akyazi, F. Kai, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, A. Rastrow, et al, The subspace Gaussian mixture model—a structured model for speech recognition, *Computer Speech & Language*, 25(2), 404–439, 2011.

[52] R. A. Redner and H. F. Walker, Mixture densities, maximum likelihood and the EM algorithm, *SIAM review*, 26(2), 195–239, 1984.

[53] D. A. Reynolds, Speaker identification and verification using Gaussian mixture speaker models, *Speech communication*, 17(1-2), 91–108, 1995.

[54] B. Romera-Paredes and M. Pontil, A New Convex Relaxation for Tensor Completion, *Advances in Neural Information Processing Systems 26*, 2967–2975, 2013.

[55] A. Sanjeev and R. Kannan, Learning mixtures of arbitrary Gaussians, *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 247–257, 2001.

[56] Y. Shekofteh, S. Jafari, J. C. Sprott, S. M. R. H. Golpayegani, and F. Almasganj, A Gaussian mixture model based cost function for parameter estimation of chaotic biological systems, *Communications in Nonlinear Science and Numerical Simulation*, 20(2), 469–481, 2015.

[57] G. Tang and P. Shah, *Guaranteed tensor decomposition: a moment approach*, Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pp. 1491-1500, 2015. *Journal of Machine Learning Research:* W&CP volume 37.

[58] S. Vempala and G. Wang, A spectral algorithm for learning mixture models, *Journal of Computer and System Sciences*, 68(4),841–860, 2004.

[59] T. Veracini, S. Matteoli, M. Diani, and G. Corsini, Fully unsupervised learning of Gaussian mixtures for anomaly detection in hyperspectral imagery, *2009 Ninth International Conference on Intelligent Systems Design and Applications*, 596–601, 2009.

[60] Y. Wu, P. Yang, Optimal estimation of Gaussian mixtures via denoised method of moments, *Annals of Statistics*, 48(4), pp. 1981–2007, 2020.

[61] M. Yuan and C.-H. Zhang, On tensor completion via nuclear norm minimization, *Found. Comput. Math.*, 16(4), 1031–1068, 2016.

[62] H. Zhang, C. L. Giles, H. C. Foley, and J. Yen, Probabilistic community discovery using hierarchical latent Gaussian mixture model, *AAAI*, 7, 663–668, 2007.

[63] Z. Zivkovic, Improved adaptive Gaussian mixture model for background subtraction, *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2, 28–31, 2004.

[64] B. Guo, J. Nie and Z. Yang, Learning diagonal Gaussian mixture models and incomplete tensor decompositions, *Vietnam Journal of Mathematics*, pages 1–26, 2021.

[65] B. Guo, J. Nie and Z. Yang, Diagonal Gaussian Mixture Models and Higher Order Tensor Decompositions, *arXiv preprint arXiv:2401.01337*, 2024.