CHROMOSOME 3D STRUCTURE MODELING AND NEW APPROACHES FOR GENERAL STATISTICAL INFERENCE

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Rongrong Zhang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Michael Yu Zhu, Chair

Department of Statistics

Bruce A. Craig

Department of Statistics

Chuanhai Liu

Department of Statistics

Xiao Wang

Department of Statistics

Approved by:

Jun Xie

Head of the Departmental Graduate Program

To my family.

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude towards my advisor, Professor Michael Yu Zhu, for his guidance and continuous encouragement during my PhD study at Purdue University. His guidance not only helped me in my PhD research, it will also have lasting impact on my future career.

I would also like to thank Professor Bruce A. Craig, Professor Chuanhai Liu, and Professor Xiao Wang, for their time and effort serving as my committee members. I thank them for their insightful comments, and valuable suggestions on my thesis. I sincerely thank Professor Ming Hu at Cleveland Clinic Foundation for his incisive suggestions and guidance in my research.

I am so grateful for being here at Purdue. I thank all the faculty members, the staff, and my fellow students at the Department of Statistics. They made my time at Purdue full of happiness, joy, and laughter. I am also so thankful to my fellow students in Professor Zhu's research group for their help on my research. I would never forget the time when we learned Spark and Tensorflow together, which made my study at Purdue more valuable.

Last but by no means the least, I must express my very profound gratitude to my parents, and my grandparents for providing me with unfailing support and unconditional love throughout my years of study in the US and through the process of writing this thesis. A special thank goes to my boyfriend, Bangda Zhou, for his companionship during the challenges of graduate, and for him always being there answering my coding related questions. I would never achieve this accomplishment without them. Thank you!

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

Zhang, Rongrong PhD, Purdue University, December 2018. Chromosome 3D Structure Modeling and New Approaches For General Statistical Inference . Major Professor: Michael Yu Zhu.

This thesis consists of two separate topics, which include the use of piecewise helical models for the inference of 3D spatial organizations of chromosomes and new approaches for general statistical inference.

The recently developed Hi-C technology enables a genome-wide view of chromosome spatial organizations, and has shed deep insights into genome structure and genome function. However, multiple sources of uncertainties make downstream data analysis and interpretation challenging. Specifically, statistical models for inferring three-dimensional (3D) chromosomal structure from Hi-C data are far from their maturity. Most existing methods are highly over-parameterized, lacking clear interpretations, and sensitive to outliers. We propose a parsimonious, easy to interpret, and robust piecewise helical curve model for the inference of 3D chromosomal structures from Hi-C data, for both individual topologically associated domains and whole chromosomes. When applied to a real Hi-C dataset, the piecewise helical model not only achieves much better model fitting than existing models, but also reveals that geometric properties of chromatin spatial organization are closely related to genome function.

For potential applications in big data analytics and machine learning, we propose to use deep neural networks to automate the Bayesian model selection and parameter estimation procedures. Two such frameworks are developed under different scenarios. First, we construct a deep neural network-based Bayes estimator for the parameters of a given model. The neural Bayes estimator mitigates the compu-

tational challenges faced by traditional approaches for computing Bayes estimators. When applied to the generalized linear mixed models, the neural Bayes estimator outperforms existing methods implemented in R packages and SAS procedures. Second, we construct a deep convolutional neural networks-based framework to perform simultaneous Bayesian model selection and parameter estimation. We refer to the neural networks for model selection and parameter estimation in the framework as the neural model selector and parameter estimator, respectively, which can be properly trained using labeled data systematically generated from candidate models. Simulation study shows that both the neural selector and estimator demonstrate excellent performances.

The theory of Conditional Inferential Models (CIMs) has been introduced to combine information for efficient inference in the Inferential Models framework for prior-free and yet valid probabilistic inference. While the general theory is subject to further development, the so-called regular CIMs are simple. We establish and prove a necessary and sufficient condition for the existence and identification of regular CIMs. More specifically, it is shown that for inference based on a sample from continuous distributions with unknown parameters, the corresponding CIM is regular if and only if the unknown parameters are generalized location and scale parameters, indexing the transformations of an affine group.

1. INFERRING SPATIAL ORGANIZATION OF CHROMSOMES VIA PIECEWISE HELICAL MODEL

In this chapter, we focus on the problem of inferring spatial organization of chromosomes via piecewise helical models, both locally and globally. We start with a review of comprehensive technique to capture the conformation of genomes - HIC techinique, and reviewed advantages and disadvantages of existing methods of reconstructing the struture of choromosomes in Section 1.1. Section 1.2 describes the proposed piecewise helical models. In Section 1.3 and 1.4, we explored the performance of proposed approach under different simulation studies and a real data example.

1.1 Introduction

Spatial organizations of chromosomes form the three-dimensional (3D) structural basis of gene expression regulation, DNA replication, and DNA repair [1]. Understanding how chromatin folds and its functional implications has attracted wide attention of the scientific community for many decades. Since early 1980s, scientists have been using microscopic-based methods, such as fluorescence in situ hybridization (FISH) [2] to study the relative positioning of genomic loci in each cell. Although very successful, these microscopic- based methods are limited by their low throughput capacities. To achieve higher throughput, Dekker et al. developed the revolutionary chromosome conformation capture (3C) technology, which simultaneously interrogates the entire cell population [3]. Lieberman-Aiden et al. further coupled 3C with next generation sequencing technologies, named the resulting method as Hi-C, and obtained the first genome-wide view of chromosome spatial organizations [4]. Recently, Hi-C and the related technologies, such as ChIA-PET [5], TCC [6], and single-cell

Hi-C [7] have been widely used and shed deep insights into genome structure and genome functions [8, 9].

In Hi-C experiments, two chromatin regions with close spatial proximity are first cross-linked and then cut into fragments by restriction enzyme. The ends of two nearby fragments are fused together to form a ligation product. Such ligation product is further sheared into short pieces, and sequenced from both ends by next generation sequencing technologies. Following sequencing, the resulting pairend reads are aligned to the reference genome. Hi-C experiments measure chromatin interactions in millions of cells simultaneously. A higher number of aligned pair-end reads indicates more frequent chromatin interactions and closer spatial proximity.

A fundamental question in Hi-C data analysis is to study spatial organizations of chromosomes. Biophysicists have proposed several polymer models to study biophysical principles governing chromatin folding [10–15]. The basic idea of polymer models is that chromatin interaction energy, derived from the underlying biophysical principles, determines the specific conformation of chromosome folding. However, these polymer models cannot fully accommodate and explain the inherent randomness in Hi-C data. To fill in this gap, data-driven statistical approaches have been developed to model uncertainties of Hi-C data and study chromatin dynamics in the cell population [16, 17]. Despite from different perspectives, both polymer models and statistical models characterize spatial organizations of chromosomes via similar beads-on-a-string representation: they first partition a genomic region of interest into non-overlapped, consecutive bins (beads) based on their genomic order (string), and then estimate the Euclidean coordinates of each bin (bead). Without any specific geometric assumptions, such a beads-on-a-string representation is highly over-parameterized and sensitive to outliers. It is also difficult to take full advantage of the knowledge of biophysical constraints on the resulting structure.

To overcome the limitations of the beads-on-a-string representation, we propose a parsimonious, easy to interpret, and robust piecewise helical model for inferring spatial organizations of individual topologically associated domains (TADs) from Hi-

C data. Similar helix models have been successfully applied to studies of protein folding [18, 19] and gene expression regulation [20]. To the best of our knowledge, such modeling strategies have not been employed in Hi-C data analysis yet.

The motivation of the proposed piecewise helical model comes from two perspectives. First of all, existing methods, such as BACH [17], ChromSDE [15] and pastis [21], have showed that although large scale chromatin folding is highly dynamic, local scale chromatin folding, especially within topologically associated domains (TADs) [22], can be very stable. Therefore, we assume that chromatin within a TAD exhibits a consensus spatial organization among the cell population. Noticeably, two recent Hi-C studies [23, 24] have shown that the paternal allele and maternal allele exhibit highly similar chromatin organization features for autosome chromosomes. Therefore, in this work, we assume that two homologous autosome alleles share the same 3D consensus structure, and do not account for allele-specific chromatin spatial organizations. Additionally, it is known from geometry that any 3D curve can be uniquely determined by its local curvature and torsion. As a special case, a constant curvature and constant torsion leads to a helical curve. Since any continuous function can be approximated by piecewise constant function, the curvature and torsion of an arbitrary 3D curve can be approximated in the same way. Therefore, any continuous 3D curve can be approximated by several well-connected helixes, which we refer to as a piecewise helical curve. From these two motivations, we propose a piecewise helical model to characterize the consensus 3D chromosomal structure within TADs.

We have performed exploratory analysis on a real Hi-C dataset [22] using the existing method BACH, and the results confirmed our motivation for using piecewise helical curve to locally characterize the 3D chromatin folding structure. Specially, Fig. 1.11 in Section 1.4 of this chapter demonstrates a BACH-predicted 3D structure of a 1Mb TAD (mouse chromosome 18,33,960,001 – 34,960,000) at 40kb resolution, which can be well approximated by a piecewise helical curve. More importantly, compared to the beads-on-astring model, our proposed piecewise helical model only

involves curvatures and torsions as primary unknown parameters, resulting in a highly efficient computational algorithm with straightforward geometric interpretations. In addition, genomic distance between any two loci could be easily incorporated in a piecewise helical model as the arc length between them to protect against potential outliers, leading to a robust 3D structural model for chromatin folding.

1.2 Methods

1.2.1 Piecewise helical curve representation

Let $\boldsymbol{r}$ represent a 3D curve, and $\boldsymbol{r}(s) = (r_1(s), r_2(s), r_3(s))$ represent the Euclidean coordinates of a point on the curve, where $s \in [0, S]$ is the arc length parameter such that $|\boldsymbol{r}'(s)| = 1$.

From curve geometry, the Frenet frame [25] as a local coordinate system determines the dynamics and geometry of a curve. The Frenet frame of $\boldsymbol{r}$ consists of the tangent vector $\boldsymbol{t}(s) = (t_1(s), t_2(s), t_3(s))$, the normal vector $\boldsymbol{n}(s) = (n_1(s), n_2(s), n_3(s))$, and the binormal vector $\boldsymbol{b}(s) = (b_1(s), b_2(s), b_3(s))$, and these three vectors are orthonormal to each other. At a given s, the collection of $\boldsymbol{t}(s)$, $\boldsymbol{n}(s)$, and $\boldsymbol{b}(s)$, denoted as $\{\boldsymbol{t}(s), \boldsymbol{n}(s), \boldsymbol{b}(s)\}$ is referred to as the orientation of the curve at s. Furthermore, together with $\boldsymbol{r}(s)$, the vectors satisfy a system of differential equations which are called the Frenet equations and given as follows:

$$\begin{cases} \boldsymbol{r}'(s) = \boldsymbol{t}(s), \\ \boldsymbol{t}'(s) = \kappa(s)\boldsymbol{n}(s), \\ \boldsymbol{n}'(s) = -\kappa(s)\boldsymbol{t}(s) + \tau(s)\boldsymbol{b}(s), \\ \boldsymbol{b}'(s) = -\tau(s)\boldsymbol{n}(s). \end{cases}$$

Here $()'$ denotes the differentiation with respect to the arc length parameter s. $\kappa(s)$ is the curvature of the curve at s, which represents how fast the curve deviates from a straight line at s, and $\tau(s)$ is the torsion of the curve at s, which represents how fast it twists from a flat plane at s. As noted previously, the geometry of the curve is controlled by the Frenet frame, the curvature and the torsion. Once the

curvature function $\kappa(s)$, the torsion function $\tau(s)$, the starting point $\boldsymbol{r}(0)$, and the orientation $\{\boldsymbol{t}(0), \boldsymbol{n}(0), \boldsymbol{b}(0)\}$ at the starting point are given, the entire curve can be constructed by solving the Frenet equations.

When both $\kappa(s)$ and $\tau(s)$ are constants, the resulting curve is a helix. Helixes are arguably the simplest 3D curves. They, however, can give rise to more complex curves. One way to construct more complex 3D curves from helixes is to connect the latter piece by piece, each of which may have different constant curvatures and torsions. We denote a curve constructed in such way as a piecewise helical curve.

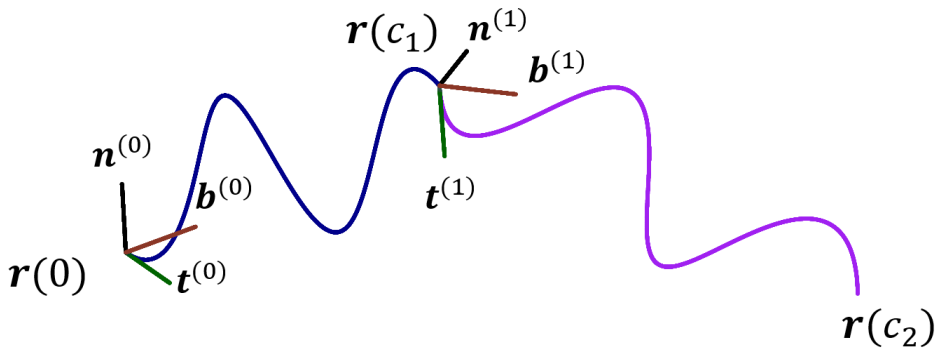


Fig. 1.1.: A piecewise helical curve consisting of two helices

Fig. 1.1 shows an example of a piecewise helical curve consisting of two helixes. The first helix starts at $\boldsymbol{r}(0)$ and its orientation at $s = 0$ is given by $\{\boldsymbol{t}^{(0)}, \boldsymbol{n}^{(0)}, \boldsymbol{b}^{(0)}\}$. The curvature and torsion of the first helix are 0.5 and 0.3, respectively. The ending point of the first helix is $\boldsymbol{r}(c_1)$ and the orientation at the ending point is $\{\boldsymbol{t}(c_1), \boldsymbol{n}(c_1), \boldsymbol{b}(c_1)\})$. The starting point of the second helix is $\boldsymbol{r}(c_1)$. Therefore, the two helixes are connected continuously. The initial orientation of the second helix at $\boldsymbol{r}(c_1)$ is $\{\boldsymbol{t}^{(1)}, \boldsymbol{n}^{(1)}, \boldsymbol{b}^{(1)}\}$. If it is identical to $\{\boldsymbol{t}(c_1), \boldsymbol{n}(c_1), \boldsymbol{b}(c_1)\})$, then the two helixes are set to be smoothly connected. Otherwise, they are set to be flexibly connected. Note that the two helixes in Fig. 1.1 are flexibly connected. The curvature of the second helix is 0.5 while the torsion is 0.2. The ending point is $\boldsymbol{r}(c_2)$. In summary, a piecewise helical curve of two helixes are uniquely determined by the starting point $\boldsymbol{r}(0)$,

the location c_1 where the two helixes are connected, the location c_2 where the second helix ends, a list of pairs giving the curvatures and torsions, $\{(\kappa^{(1)}, \tau^{(1)}), (\kappa^{(2)}, \tau^{(2)})\}$, and a list of triples giving the orientations, $\{(\boldsymbol{t}^{(0)}, \boldsymbol{n}^{(0)}, \boldsymbol{b}^{(0)}), (\boldsymbol{t}^{(1)}, \boldsymbol{n}^{(1)}, \boldsymbol{b}^{(1)})\}$. The curve can be constructed by solving the Frenet equations.

We use the notations from the previous work [18] and define a 12-dimensional vector:

$$\boldsymbol{Y} = (t_1, n_1, b_1, t_2, n_2, b_2, t_3, n_3, b_3, r_1, r_2, r_3),$$

whose entries are consisting of nine components of the three basis vectors in the Frenet frame and three coordinates of the point on the curve. Then the Frenet equations can be summarized into the following matrix form:

$$\boldsymbol{Y}' = M(s)(Y),$$

where

$$M = \begin{bmatrix} F & 0 & 0 & 0 \\ 0 & F & 0 & 0 \\ 0 & 0 & F & 0 \\ V_1 & V_2 & V_3 & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & \kappa(s) & 0 \\ -\kappa(s) & 0 & \tau(s) \\ 0 & -\tau(s) & 0 \end{bmatrix}$$

$$V_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, V_2 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, V_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

For the first helix with constant curvature $\kappa^{(1)}$ and torsion $\tau^{(1)}$ in Fig. 1.1, the matrix M is a constant matrix, and the Frenet equations become a system of homogeneous first order linear differential equations with constant coefficients. Given the initial value $\boldsymbol{Y}(0)$, the solution [18] to the Frenet equations is given by:

$$\boldsymbol{Y}^{(1)}(s) = A(\kappa^{(1)}, \tau^{(1)}, s)\boldsymbol{Y}(0), \quad 0 \leq s \leq c_1.$$

where

$$\boldsymbol{Y}(0) = (t_1^{(0)}, n_1^{(0)}, b_1^{(0)}, t_2^{(0)}, n_2^{(0)}, b_2^{(0)}, t_3^{(0)}, n_3^{(0)}, b_3^{(0)}, r_1^{(0)}, r_2^{(0)}, r_3^{(0)}),$$

and

$$A(\kappa, \tau, s) = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & a & 0 \\ b_1 & b_2 & b_3 & I_3 \end{bmatrix}$$

$$a = \begin{bmatrix} \frac{\tau^2 + \kappa^2 \cos(\alpha s)}{\alpha^2} & \frac{\kappa \sin(\alpha s)}{\alpha} & \frac{\kappa\tau(1 - \cos(\alpha s))}{\alpha^2} \\ -\frac{\kappa \sin(\alpha s)}{\alpha} & \cos(\alpha s) & \frac{\tau \sin(\alpha s)}{\alpha} \\ \frac{\kappa\tau(1 - \cos(\alpha s))}{\alpha^2} & -\frac{\tau \sin(\alpha s)}{\alpha} & \frac{\kappa^2 + \tau^2 \cos(\alpha s)}{\alpha^2} \end{bmatrix}$$

$$b_1 = \begin{bmatrix} \frac{\alpha s\tau^2 + \kappa^2 \sin(\alpha s)}{\alpha^3} & \frac{\kappa(1 - \cos(\alpha s))}{\alpha^2} & \frac{\kappa\tau(\alpha s - \sin(\alpha s))}{\alpha^3} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$b_2 = \begin{bmatrix} 0 & 0 & 0 \\ \frac{\alpha s\tau^2 + \kappa^2 \sin(\alpha s)}{\alpha^3} & \frac{\kappa(1 - \cos(\alpha s))}{\alpha^2} & \frac{\kappa\tau(\alpha s - \sin(\alpha s))}{\alpha^3} \\ 0 & 0 & 0 \end{bmatrix}$$

$$b_3 = \begin{bmatrix} 0 & 0 & 0 \\ \frac{\alpha s\tau^2 + \kappa^2 \sin(\alpha s)}{\alpha^3} & \frac{\kappa(1 - \cos(\alpha s))}{\alpha^2} & \frac{\kappa\tau(\alpha s - \sin(\alpha s))}{\alpha^3} \\ 0 & 0 & 0 \end{bmatrix}$$

$$\alpha = \sqrt{\kappa^2 + \tau^2},$$

and I_3 is the 3×3 identity matrix.

$\boldsymbol{Y}^{(1)}(c_1)$ explicitly gives the orientation at the ending point, and the coordinates of the ending point of the first helix are $\boldsymbol{r}(c_1) = (r_1^{(1)}, r_2^{(1)}, r_3^{(1)})$.

If the starting point $\boldsymbol{r}(0)$ is the origin, i.e. $\boldsymbol{r}(0) = (0,0,0)$ and the orientation at 0 is given by the standard basis of $\mathbb{R}^3$, i.e. $\{\boldsymbol{t} = \{1,0,0\}, \boldsymbol{n} = \{0,1,0\}, \boldsymbol{b} = \{0,0,1\}\}$, the parametric expression of the first helix could be simplified as follows:

$$\boldsymbol{r}(s) = \begin{pmatrix} r_1(s) \\ r_2(s) \\ r_3(3) \end{pmatrix} = \begin{pmatrix} \frac{\kappa}{\alpha^2}\cos(\alpha s) \\ \frac{\kappa}{\alpha^2}\sin(\alpha s) \\ \frac{\tau}{\alpha}s \end{pmatrix}, \tag{1.1}$$

$$\alpha = \sqrt{\kappa^2 + \tau^2}, \quad 0 \le s \le c_1$$

As discussed previously, to construct the second helix, $\boldsymbol{r}(c_1)$ is served as the starting point of the second helix and the orientation at starting point is $\{\boldsymbol{t}^{(1)}, \boldsymbol{n}^{(1)}, \boldsymbol{b}^{(1)}\}$. The solution of the Frenet equations of the second helix is given by:

$$\boldsymbol{Y}^{(2)}(s) = A(\kappa^{(2)}, \tau^{(2)}, s - c_1)\boldsymbol{Y}(c_1), \quad c_1 \le s \le c_2,$$

where

$$\boldsymbol{Y}(c_1) = (t_1^{(1)}, n_1^{(1)}, b_1^{(1)}, t_2^{(1)}, n_2^{(1)}, b_2^{(1)}, t_3^{(1)}, n_3^{(1)}, b_3^{(1)}, r_1^{(1)}, r_2^{(1)}, r_3^{(1)})$$

The last three components of the vector $\boldsymbol{Y}^{(2)}(c_2)$ give the coordinates of the ending point $\boldsymbol{r}(c_2)$.

It is straightforward to extend the piecewise helical curve with two helixes to that with H helixes. Suppose the latter is defined for $s \in [0, S]$, $\boldsymbol{r}(c_0) = \boldsymbol{r}(0)$ is the starting point, $c_1, c_2, \cdots, c_{H-1}$ are the $H-1$ locations where the H helixes connect to each other consecutively, and $\boldsymbol{r}(c_H) = r(S)$ is the ending point.

The j-th helix is defined from c_{j-1} to c_j, $\boldsymbol{r}^{(j-1)} = \boldsymbol{r}(c_{j-1})$ is the starting point, the curvature and torsion are $\kappa^{(j)}$ and $\tau^{(j)}$, the orientation at $\boldsymbol{r}^{(j-1)}$ is $\{\boldsymbol{t}^{(j-1)}, \boldsymbol{n}^{(j-1)}, \boldsymbol{b}^{(j-1)}\}$. By recursively applying the procedures used for two helixes, the solution to the Frenet equations of the j-th helix is given by:

$$\boldsymbol{Y}^{(j)}(s) = A(\kappa^{(j)}, \tau^{(j)}, s - c_{j-1})\boldsymbol{Y}(c_{j-1}), \quad c_{j-1} \le s \le c_j.$$

where

$$\boldsymbol{Y}(c_i) = (t_1^{(i)}, n_1^{(i)}, b_1^{(i)}, t_2^{(i)}, n_2^{(i)}, b_2^{(i)}, t_3^{(i)}, n_3^{(i)}, b_3^{(i)}, r_1^{(i)}, r_2^{(i)} r_3^{(i)}),$$

$$i = j - 1, j = 1, 2, \cdots, H.$$

If all the helixes are smoothly connected, the recursive solution to the Frenet equations could be simplified as

$$\boldsymbol{Y}^{(j)}(s) = A(\kappa^{(j)}, \tau^{(j)}, s - c_{j-1}) \left[\prod_{k=1}^{j-1} A(\kappa^{(k)}, \tau^{(k)}, c_k - c_{k-1}) \right] \boldsymbol{Y}(0), \tag{1.2}$$

$$c_{j-1} \leq s \leq c_j, j \geq 2.$$

Since, any arbitrary 1D function can be approximated by a piecewise constant function, the curvature and torsion functions of any 3D curve can be approximated in the same way. As discussed above, a curve with piecewise constant curvature and torsion functions is a piecewise helical curve. Hence the piecewise helical curve representation is flexible enough to model any 3D curve.

Compared with the over-parameterized beads-on-astring representation, the piecewise helical curve representation is much simpler. It is a parsimonious representation with a finite number of parameters, which can be much smaller than the total number of coordinates of loci in the domain of interest. Moreover, the piecewise helical curve representation is mathematical tractable. The spatial distance between any two genomic loci can be expressed recursively as a function of a series of curvatures, torsions and orientation vectors, and all these parameters have clear geometric interpretations. The proposed piecewise helical representation is a model for 3D curve, and doesn't need additional spatial constraints. On the other hand, the beads-on-a-string model, used by existing methods such as ChromSDE and pastis, do not impose any spatial constraints and entirely rely on the spatial pattern present in Hi-C data, making the model unstable and sensitive to outliers. Therefore, the piecewise helical model is more preferable to the beads-on-a-string model in terms of model robustness.

1.2.2 Piecewise Helical Model for Contact Frequencies within TAD

Piecewise Helical Model

Topological associate domain (TAD), the basic unit of genome structure and genome function, is much shorter domain compared with whole chromosome, and can be approximated by the piecewise helical curve. Using the piecewise helical curve representation, we propose a statistical model to link Hi-C data to pair-wise spatial distances between genomic loci in a TAD.

We use the piecewise helical curve with H helixes to approximate the 3D structure of the TAD of interest. Suppose the total number of loci in the TAD is N, and their corresponding positions on the curve are $\boldsymbol{r}(s_1), \cdots, \boldsymbol{r}(s_N)$. Then, the Euclidean distance between any two loci i and j , denoted as $d_{ij} = d(s_i, s_j)$, is given by:

$$d_{ij} = \sqrt{(r_1(s_i) - r_1(s_j))^2 + (r_2(s_i) - r_2(s_j))^2 + (r_3(s_i) - r_3(s_j))^2}.$$

Let Y_{ij} represent the number of pair-end reads spanning the two genomic loci i and j, $i, j \in \{1, \cdots, N\}$. To account for the over-dispersion in Hi-C data (see Section 4), we propose the following negative binomial model for Y_{ij}:

$$Y_{ij} \sim NB(\theta_{ij}, \phi).$$

In this model, Y_{ij} has mean θ_{ij}, variance $\theta_{ij} + \theta_{ij}^2/\phi$, and dispersion parameter ϕ. Furthermore, we model the relationship between $\log \theta_{ij}$ and $\log d_{ij}$ and local genomic features as follows:

$$\log \theta_{ij} = \log e_{ij} + \beta_0 + \beta_1(\log d_{ij}),$$

where

$$e_{ij} = \exp\left(\beta_{enz} \log(E_i E_j) + \beta_{gcc} \log(G_i G_j) + \beta_{map} \log(M_i M_j)\right).$$

In this model, $E_i, E_j, G_i, G_j, M_i, M_j$ are the restriction enzyme cutting frequency, GC content, and sequence mappability of loci i and loci j, respectively, and β_{enz}, β_{gcc}, β_{map}, β_0, and β_1 are unknown coefficients. β_0 represents the over-all sequencing

depth, and β_1 represents the link coefficient between spatial distance and chromatin interaction frequency. Note that β_1 is expected to be negative.

We assume that all Y_{ij}'s are independent. The joint likelihood is given as follows:

$$L\left(\{Y_{ij}\}_{1\le i,j\le N}|\{d_{ij}\}_{1\le i,j\le N},\phi\right)=\prod_{1\le i,j\le N}\left(\frac{\phi}{\phi+\theta_{ij}}\right)^{\phi}\frac{\Gamma(\phi+Y_{ij})}{\Gamma(\phi)Y_{ij}!}\left(\frac{\theta_{ij}}{\phi+\theta_{ij}}\right)^{Y_{ij}},$$

where $\log\theta_{ij}=\log e_{ij}+\beta_0+\beta_1\log d_{ij}$.

Model Implementation

Since Hi-C data do not contain information about the absolute spatial distance, piecewise helical models are not identifiable up to a scaling factor. For example, assume that the helix has curvature κ, torsion τ and total arc length S. For any positive constant C, two sets of parameters $(\beta_0,\beta_1,\kappa,\tau,S)$, and $(\beta_0+\beta_1\log C,\beta_1,C\kappa,C\tau,S/C)$ can achieve the same likelihood value in our posited negative binomial model. To resolve this non-identifiability issue without loss of generality, we normalize the absolute scale of the piecewise helical curve such that

$$\sum_{1\le i<j\le N}\log d_{ij}=0.$$

Recall the equation (1.1) for the helix starting from the origin. The first two coordinates of the helix are periodical, and we refer to each period as a loop of this helix. For a helix with curvature κ, torsion τ and total arc length S, the number of loops, which is denoted as L, is given as follows:

$$L=\frac{\sqrt{\kappa^2+\tau^2}*S}{2\pi}.$$

For any helix that contains N genomic loci, we add the following constraint on L:

$$L\le N/2.$$

The purpose of imposing this constraint is to exclude helixes that have an excessive number of loops. Equivalently, this constraint requires that on average each loop contains at least two genomic loci.

Additional assumptions are imposed on the piecewise constant curvatures and torsions in the piecewise helical model to avoid over-fitting. We require that the helixes within a piecewise helical curve to have equal length, which is equivalent to requiring that the helixes have an equal number of genomic loci. We use the Bayesian Information Criterion (BIC) [26] to determine the number of helixes for a TAD. Note that the curvatures and torsions for all helixes and orientation vectors $\boldsymbol{t}$, $\boldsymbol{n}$, $\boldsymbol{b}$ at the connection points are collectively the primary parameters of the piecewise helical model.

A fully Bayesian approach with Markov Chain Monte Carlo (MCMC) techniques is adopted for fitting the piecewise helical model and performing the subsequent statistical inference. Uniform prior $Unif[0.001, 1000]$ is chosen for curvature κ , torsion τ and over-dispersion parameter ϕ . Non-informative priors are used for the other parameters in the piecewise helical model. The random walk Metropolis within Gibbs algorithm [27] is implemented to sample the joint posterior distribution. Fig. 1.2 shows high correlation between curvature and torsion in the single helix model. In order to improve the sampling efficiency, the multiple-try Metropolis algorithm [27] is applied to sample curvature and torsion simultaneously. When fitting the piecewise helical model, we ran 50,000 MCMC iterations. The first 10,000 samples were discarded as the burn-in stage, and then every 50th sample in the last 40,000 samples were used for posterior inference.

1.2.3 Piecewise Helical Model for Contact Frequencies within whole Chromosome

Depending on sequencing depths in Hi-C technique, the commonly used sizes of bins can range from 1 kb to 1 Mb. The bin size of Hi-C interaction matrix is also referred to as 'resolution'. Then the size of contact matrix could range from hundreds to hundreds of thousands for different resolutions. It is clear that higher resolution (smaller bin size) will provide more insights about the structure of chromosome,

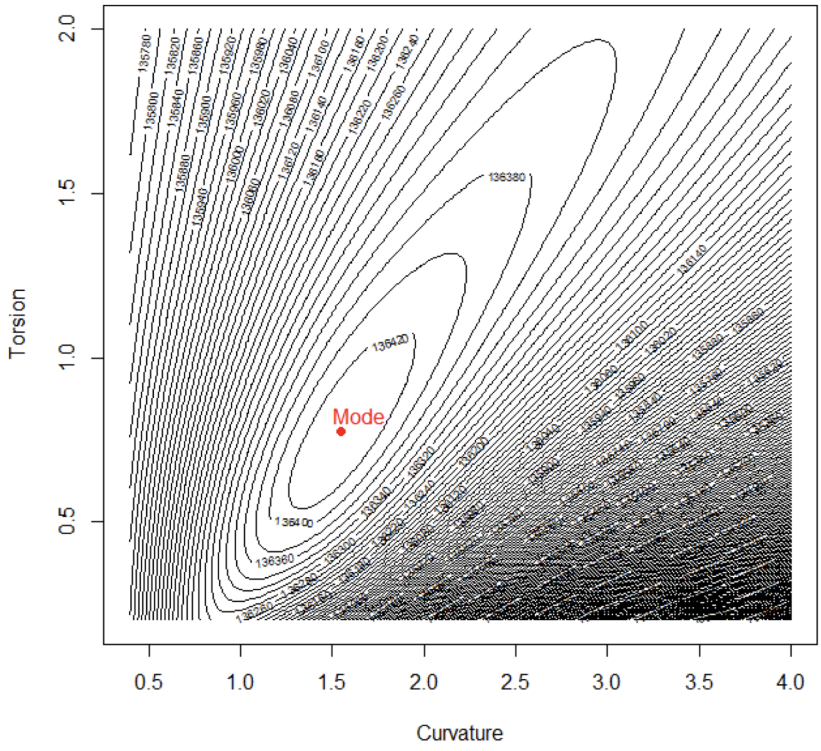


Fig. 1.2.: Contour plot of log likelihood shows high correlation between curvature and torsion in the single helix model. We simulate a Hi-C data from the posited helix model with curvature 1.55 and torsion 0.76. The contour plot shows the log likelihood of all combinations of 200 curvatures and 200 torsions. Red dot represents the location of the combination of true curvature and true torsion used in the simulation, which corresponds to the highest log likelihood.

but it also requires more computation. A critical hurdle in developing piecewise helical curve for the whole chromosome is that the computation cost. In the PHM model we proposed for TAD, we calculate the coordinates of the points on the helical curve iteratively, and we recalculate the coordinates of all points once we update one parameter in MCMC steps. The computation cost increases exponentially with respect to the resolution We proposed a two-step procedure to infer the structure for the whole chromosome. The main idea of this two-step procedure is divide and combine, in which we will first divide the whole chromosome into pieces, apply PHM to all pieces in parallel, and second find the way to connect all these pieces together to make the whole structure.

To be more specific, suppose that in the Hi-C experiment the whole chromosome is divided into N equally sized bins, and the size of the contact matrix is $N \times N$. We will use $A = (a_{ij})$ to denote this Hi-C contact matrix, where a_{ij} denotes the number of pair-end reads between loci i and j for $1 \leq i, j \leq N$. It's assumed that the structure of the whole chromosome is composed of contains H piecewise helical curves. We describe the two-step procedure of reconstructing the structure of the whole chromosome as follows.

Step 1: Parallel construction of the structures of domains. We first divide the whole chromosome into H equally sized sub-pieces. We denote the center points of these H pieces as $p_1, p_2, \cdots, p_H$, where p_0 and p_{H+1} are the starting and ending points of the whole chromosome. Then the distance from p_0 to p_1 is $n/2$, the distance between p_i and p_{i+1} is n for $1 \leq i \leq H-1$, and the distance between p_H to the end of the chromosome p_{H+1} is $n/2$, where $n = N/M$.

The first step is to reconstruct the structures of domains on the chromosome between consecutive points p_i and p_{i+1}, $i = 0, \cdots, H$. To reconstruct the structure of domain i, we simply apply the PHM method to the contact matrix for bins between

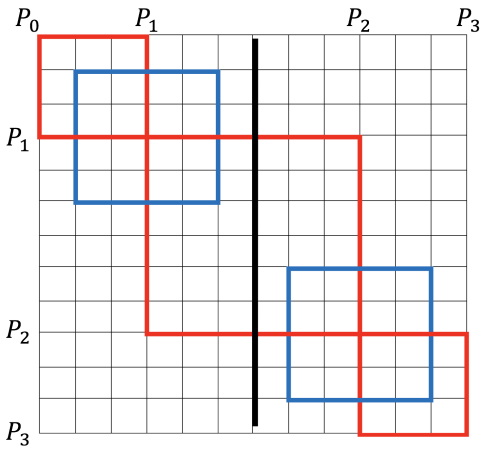


Fig. 1.3.: Demonstration of how to cut the original square matrix to pieces for two-step inference procedure.

p_i and p_{i+1}, which is the diagonal submatrix of the original contact matrix A from p_i to p_{i+1} and is denoted as A_i, refer to the red square matrix in Fig. 1.3.

$$A = \begin{pmatrix} A_0 & \cdots & \cdots & \cdots \\ \cdots & A_1 & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & A_H \end{pmatrix}. \tag{1.3}$$

Since those sub-pieces do not overlap, reconstructions can be done in parallel to save computation time. Note that the A_0 and A_H are just the half size of the others. Fig. 1.3 shows an example in which the curve is cut into three pieces. We will apply PHM method to red square matrices on the diagonal to reconstruct the structures of three pieces in parallel.

Step 2: Connect sub-pieces. After reconstruction of structures of $H+1$ domains, we will connect them continuously to make the structure for whole chromosome in step two. We need to know how two consecutive domains are connected to each other. To be specific, we need to know the relative orientation between two consecutive curves, which are defined by Frenet frames.

Reconstruction procedure of piecewise helical curve returns the curvature vector, torsion vector, and the Frenet frames at all junction points. For example, after Step one, we obtain $\vec{\kappa}_0$, $\vec{\tau}_0$ and a set of $\boldsymbol{t}_0, \boldsymbol{n}_0, \boldsymbol{b}_0$'s for domain from p_0 to p_1 as shown in Fig. 1.3. For domain from p_1 to p_2, we obtain another set of $\vec{\kappa}_1$, $\vec{\tau}_1$ and $\boldsymbol{t}_1, \boldsymbol{n}_1, \boldsymbol{b}_1$'s. But we will need to rotate the domain from p_1 to p_2 before we continuous connect it to domain from p_0 to p_1.

The off diagonal cells in the contact matrix A provide insights to the relative structure of loci which are relatively far from each other. We will use these cells to infer the rotation transformation matrix. We will fit PHM models to the blue matrices as shown in Fig. 1.3 and set all p_i's as the change point. The size of blue matrix does not need to be large, size of 50 to 100 will be enough to infer the relative rotation. The relative transformation matrix from one set of $\{\boldsymbol{t}^1, \boldsymbol{n}^1, \boldsymbol{b}^2\}$ to another set $\{\boldsymbol{t}^2, \boldsymbol{n}^2, \boldsymbol{b}^2\}$ is

$$\begin{bmatrix} t_1^2 & t_2^2 & t_3^2 \\ n_1^2 & n_2^2 & n_3^2 \\ b_1^2 & b_2^2 & b_3^2 \end{bmatrix} \cdot \begin{bmatrix} t_1^1 & t_2^1 & t_3^1 \\ n_1^1 & n_2^1 & n_3^1 \\ b_1^1 & b_2^1 & b_3^1 \end{bmatrix}^{-1} . \tag{1.4}$$

To connect the domain from p_1 to p_2 to the previous domain, we multiply the relative transformation matrix inferred from the blue matrix to the $\{\boldsymbol{t}, \boldsymbol{n}, \boldsymbol{b}\}$ of the last helix piece in the piecewise helical curve for the first domain. We modify the all subsequent $\{\boldsymbol{t}, \boldsymbol{n}, \boldsymbol{b}\}$'s repeatly following this procedure. By doing so, we use the off-diagonal cells to connect all pieces together. Applying PHM model to all blue matrices can be added in Step one in parallel to save computation time.

This two-step procedure of reconstructing the structure of whole chromosome takes advantages PHM models. Further more, parallel computing the structures of the domains saves the computation time when the size (resolution) of contact matrix increases.

1.2.4 Heterogeneity and Mixture of Piecewise Helical Models

Although local chromatin folding, especially within TADs, can be very stable. This assumption may not be true for modelling the 3D structure of the whole chromosome in the cell population.The whole chromosome is highly dynamic, especially its euchromatin /open chromatin regions. It is unclear whether the cell population contains one dominant structure or multiple distinct 3D chromosomal structures with comparable mixing proportions. The whole chromosome is likely to exhibit multiple distinct 3D consensus structures among the entire cell population, and just using one single helical curve to characterize the structure of the whole chromosome is not adequate. Therefore, we propose to use a mixture of piecewise helical models to characterize the structure variability of the whole chromosome.

Suppose the cell population consists of M distinct sub-populations, each of them is a piecewise helical curve with curvature vector $\vec{\kappa}^m = (\kappa_1^m, \cdots, \kappa_{n_m}^m)$ and torsion vector $\vec{\tau}^m = (\tau_1^m, \cdots, \tau_{n_m}^m)$, $m \in \{1, 2, \cdots, M\}$, where n_m is the number of helices contained in m-th piecewise helical curve. The spatial distance between locus i and locus j in the m-th helical curve is $d_{ij}^m = f(i, j, \vec{\kappa}^m, \vec{\tau}^m)$. Here the link function f can be derived iteratively according to formula (1.2), which is the same as the PHM method. Let Y_{ij} represent the number of reads spanning locus i and locus j, which is assumed to follow a negative binomial distribution with rate θ_{ij} and a common over-dispersion parameter ϕ:

$$Y_{ij} \sim NB(\theta_{ij}, \phi). \tag{1.5}$$

We further impose an additive model on θ_{ij}:

$$\theta_{ij} = \sum_{m=1}^{M} \theta_{ij}^m, \tag{1.6}$$

where θ_{ij}^m is the expected number of reads from the m-th sub-population. we further assume a log linear model for θ_{ij}^m:

$$\log \theta_{ij}^m = \log e_{ij} + \beta_0 + \beta_1 \log d_{ij}^m = \log e_{ij} + \beta_0 + \beta_1 f(i, j, \vec{\kappa}^m, \vec{\tau}^m), \tag{1.7}$$

where e_{ij} is the expected reads, which can be calculated from the local genomic features, and β_0 and β_1 are two unknown parameters. β_0 is the over-all sequencing depth. $\beta_1 < 0$ is the link coefficient between spatial distance and chromatin interaction frequency. e_{ij}, β_0, and β_1 are shared by all sub-populations.

Combining the assumptions above, we have:

$$\theta_{ij} = e_{ij} e^{\beta_0} \sum_{m=1}^{M} f(i, j, \vec{\kappa}^m, \vec{\tau}^m)^{\beta_1}. \tag{1.8}$$

We assume that all Y_{ij}'s are independent. Then the joint likelihood is:

$$\begin{aligned} P(Y_{ij}, 1 \le i < j \le N | \theta_{ij}, \phi) &= \prod_{1 \le i < j \le N} P(Y_{ij} | \theta_{ij}, \phi) \\ &= \prod_{1 \le i < j \le N} P(Y_{ij} | e_{ij}, \beta_0, \beta_1, \phi, \vec{\kappa}^m, \vec{\tau}^m, 1 \le m \le M). \end{aligned} \tag{1.9}$$

As mentioned previously, the mixture of piecewise helical models also faces the problem of non-identifiability. To resolve the issue without loss of generality, we impose similar constraints as in PHM. The number of loops L in every piecewise helical curve satisfies $L \le N/2$, where N is the total number of loci. We also impose a constraint on β_0, such that $\exp(\beta_0)$ is equal to the average of all read counts:

$$\exp(\beta_0) = \frac{\sum_{1 \le i < j \le N} Y_{ij}}{N(N-1)/2}. \tag{1.10}$$

All the other constrains in original piecewise helical curve are preserved.

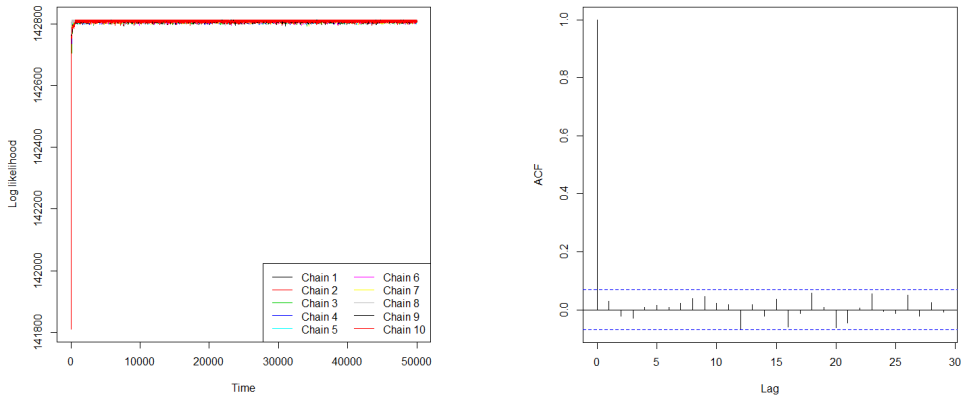
The unknown parameters of interests include: β_1, ϕ, M, $\vec{\kappa}^m$, and $\vec{\tau}^m, m = 1, \cdots, M$. We adopted a fully Bayesian approach and used random walk Metropolis within Gibbs sampler to iteratively sample each of them. To improve the sampling efficiency, the multiple-try Metropolis algorithm is applied to sample curvature and torsion simultaneously. We also start with different initial values of curvatures and torsions in the implementation. Then the Bayesian Information Criterion (BIC) is used to determine the number of sub-populations, the number of helices in each piecewise helical curve.

1.3 Simulation Studies

1.3.1 Simulation study when Hi-C data is simulated from a single helix

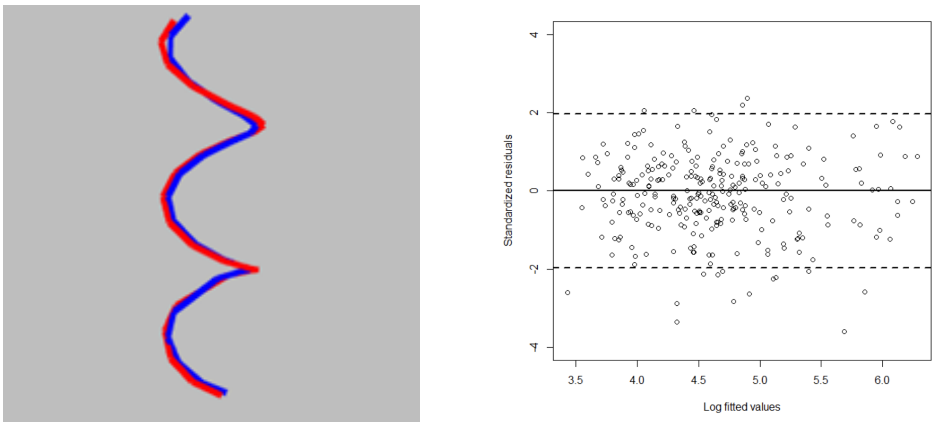
First, we conducted a simulation study to test the performance of the single helix model using a TAD in mouse chromosome 18, 33,960,001 - 34,960,000. Each 40kb bin is treated as a bead in the beads-on-a-string representation, and the chromatin is assumed to fold as a single helix with curvature $\kappa = 0.56$ and torsion $\tau = 0.28$. The arc length between the center of the i-th bin and the center of the j-th bin is proportional to the genomic distance between them ($|i - j|$). The parameters in the negative binomial model are set as: $\beta_0 = 6$, $\beta_1 = -1$, $\phi = 20$, $\beta_{enz} = 0.1$, $\beta_{gcc} = -0.1$, $\beta_{map} = 0.1$. The local genomic features of each bin, including restriction enzyme cutting frequency, GC content and sequence mappability, are calculated from the UCSC reference genome mm9. The Hi-C contact matrix $\{y_{ij}\}_{1 \le i < j \le 25}$ is then simulated from the posited negative binomial model. The constant shifted log likelihood of the simulated data at the true parameter values, referred to as true log likelihood, is 142811.16. We normalize the absolute scale of the piecewise helical curve as mentioned previously and transform the curvature, torsion and arc length parameters accordingly. After parameter transformation, curvature κ is 1.79, and torsion τ is 0.90.

The proposed algorithm is applied to the simulated Hi-C data, and statistical inference is conducted on the unknown parameters. Fig. 1.4 (a) shows the convergence of ten parallel chains. The Gelman-Rubin statistic [28] is 1.00, suggesting good mixing. Among the ten parallel chains, chain 8 achieves the highest posterior mode 142813.94, which is higher than the true log likelihood 142811.16. Therefore, chain 8 is used for posterior inference. Fig. 1.4 (b) shows the autocorrelation plot of every 50th posterior sample in the last 40,000 iterations in chain 8. After thinning, the posterior samples are independent. Table 1.1 lists the summary statistics of the posterior distribution of all unknown parameters. For each parameter, the 95% credible interval covers the true value. Fig. 1.5 (a) shows the structural alignment between



(a) (b)

Fig. 1.4.: Single helix simulation study. (a): Convergence plot of the log likelihood of ten parallel chains. (b): Autocorrelation plot of the log likelihood in the chain 8.



(a) (b)

Fig. 1.5.: (a) Structural alignment of the inferred helix (blue) and the true helix (red). Root mean square distance = 0.02. (b)Residual plot. The solid black line represents that the standardized residual equals to zero. The two dashed black lines represent that the absolute value of the standardized residual equals 1.96.

the inferred helix (blue line) and the true helix (red line). The root mean squared distance [2] between the two structures is 0.02, suggesting high similarity. The residual plot (Fig. 1.5 (b)) shows no obvious trend between the standardized residuals and the fitted values, suggesting good model fit.

Posterior predictive check [28] (Fig. 1.6) is also conducted to evaluate the goodness of fit. Nine summary statistics of the Hi-C contact matrix are used: minimal value, 10% percentile, 25% percentile, median, 75% percentile, 90% percentile, maximum value, mean and variance. For every 50th sample in the last 40,000 iterations in MCMC, a Hi-C contact matrix is simulated based on the posterior samples of parameters, and the summary statistics of the simulated Hi-C contact matrix are recorded. The summary statistics of the input Hi-C contact matrix are compared with the distribution of the summary statistics of the simulated Hi-C contact matrix. All p-values are between 0.05 and 0.95, suggesting good model fit.

Table 1.1.: Summary statistics of the posterior distribution of unknown parameters in simulation study when Hi-C data is simulated from a single helix

Parameter	Truth	Posterior Mode	Posterior distribution				
			Min	2.5% Percentile	Mean	97.5% Percentile	Max
κ	1.79	1.86	1.49	1.66	1.84	2.03	2.12
τ	0.90	1.06	0.80	0.87	1.11	1.41	1.68
β_1	-1.00	-0.95	-1.08	-1.03	-0.93	-0.85	-0.78
β_{enz}	0.10	0.09	0.04	0.06	0.09	0.12	0.14
β_{gcc}	-0.10	-0.08	-0.13	-0.12	-0.09	-0.06	-0.03
β_{map}	0.10	0.12	0.08	0.09	0.12	0.15	0.17
ϕ	20.00	20.34	15.05	16.53	20.31	24.79	26.97

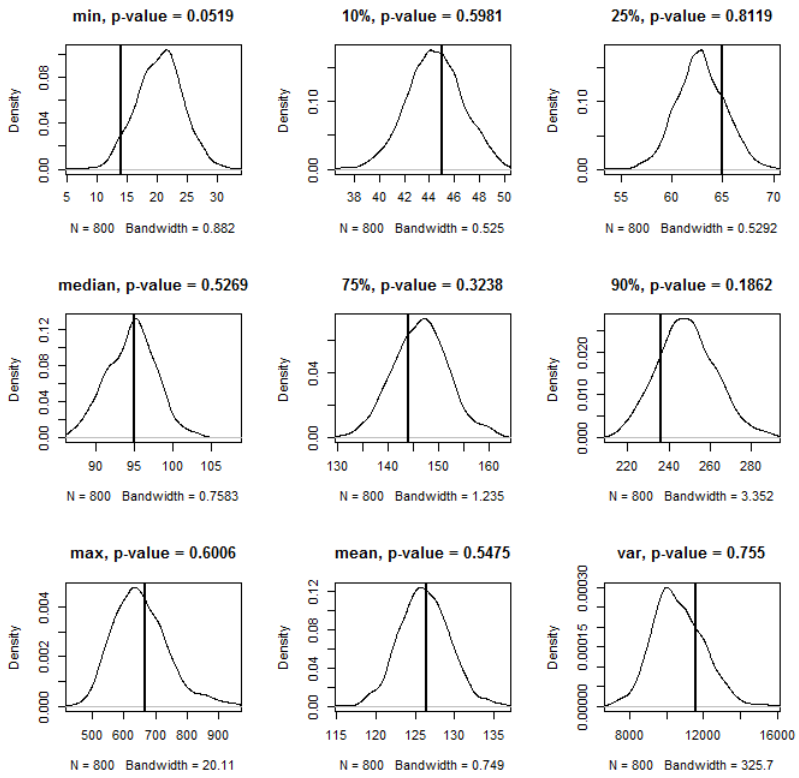


Fig. 1.6.: Posterior predictive check. The solid black line represents the summary statistics calculated from the Hi-C data.

1.3.2 Simulation study when Hi-C data is simulated from one piecewise helical curve

Next, we conducted a simulation study to test the performance of the piecewise helical model using a TAD in mouse chromosome 1, 92,800,001 - 94,800,000. Each 40kb bin is treated as a bead in the beads-on-a-string representation, and chromatin is assumed to fold as a piecewise helical curve, which consists of two equal-sized helixes with curvatures and torsions listed below:

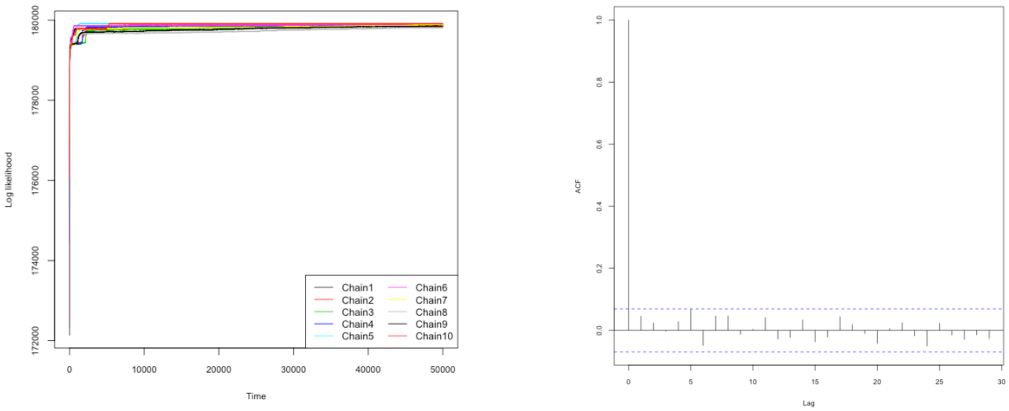
$$\text{Helix 1: } \kappa^{(1)} = 0.60, \quad \tau^{(1)} = 0.20;$$
$$\text{Helix 2: } \kappa^{(2)} = 0.50, \quad \tau^{(2)} = 0.10.$$

The arc length between the center of the i-th bin and the center of the j-th bin is proportional to the genomic distance between them ($|i-j|$). Linear link function is used to connect Hi-C data and spatial proximity, with $\beta_0 = 5$, $\beta_1 = -1$. We further set $\phi = 15$, $\beta_{enz} = 0.1$, $\beta_{gcc} = -0.1$, $\beta_{map} = 0.1$. The local genomic features of each bin, including restriction enzyme cutting frequency, GC content and sequence mappability, are calculated from the UCSC reference genome mm9. The Hi-C contact matrix $\{y_{ij}\}_{1\le i<j\le 50}$ is simulated from the posited negative binomial model. The true log likelihood is 179911.19. To account for the multi-collinearity in the regression, we normalize the absolute scale of the piecewise helical curve as mentioned previously and transform the curvature, torsion and arc length parameters accordingly. The parameters become: $\beta_0 = 3.73$, $\beta_1 = -1$,

$$\text{Helix 1: } \kappa^{(1)} = 2.13, \quad \tau^{(1)} = 0.71;$$
$$\text{Helix 2: } \kappa^{(2)} = 1.77, \quad \tau^{(2)} = 0.35.$$

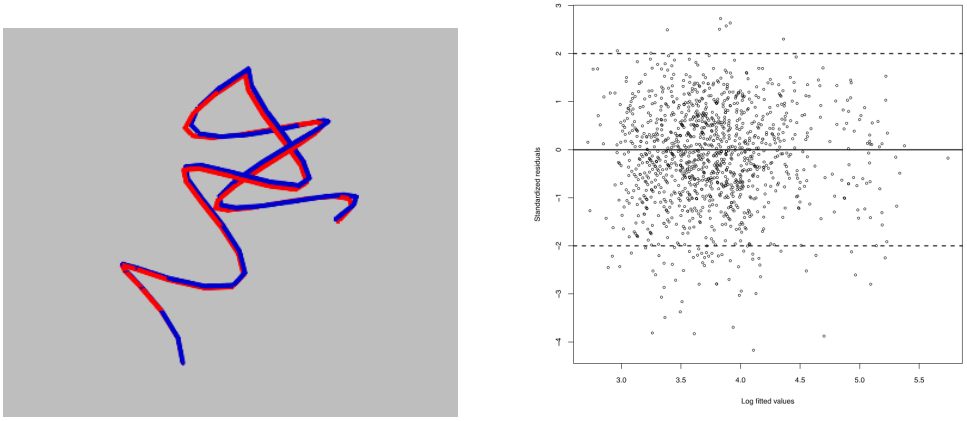
Our algorithm is applied to the simulated data, and statistical inference is conducted on the unknown parameters. Fig. 1.7 (a) shows the convergence of ten parallel chains. The Gelman-Rubin statistic is 1.00, suggesting good mixing. Among the ten parallel chains, chain 5 achieves the highest posterior mode 179915.52, which is higher than the true log likelihood 179911.19. Therefore, chain 5 is used for posterior inference. Fig. 1.7 (b) shows the autocorrelation plot of every 50th posterior sample in the last 40,000 iterations in chain 5. After thinning, the posterior samples are independent. Table 1.2 lists the summary statistics of the posterior distributions of some primary parameters. For each parameter, the 95% credible interval covers the true value.

Fig. 1.8 (a) shows the structural alignment between the inferred helical curve (blue line) and the true helical curve (red line). The root mean squared distance [16] between the two structures is 0.06, suggesting high similarity. Fig. 1.8 (b) shows the residual plot and suggests good model fit.



(a) (b)

Fig. 1.7.: Piecewise helical curve simulation study. (a): Convergence plot of the log likelihood of ten parallel chains. (b): Autocorrelation plot of the log likelihood in the chain 5.



(a) (b)

Fig. 1.8.: (a). Structural alignment of the inferred helical curve (blue) and the true helical curve (red). Root mean square distance = 0.06. (b). Residual plot. The solid black line represents that the standardized residual equals to zero. The two dashed black lines represent that the absolute value of the standardized residual equals 1.96.

Table 1.2.: Summary statistics of the posterior distribution of unknown parameters in simulation study when Hi-C data is simulated from a piecewise helical curve

Parameter	Truth	Posterior Mode	Posterior distribution				
			Min	2.5% Percentile	Mean	97.5% Percentile	Max
κ_1	2.13	2.12	2.03	2.07	2.11	2.16	2.22
τ_1	0.71	0.72	0.61	0.67	0.71	0.76	0.82
κ_2	1.77	1.78	1.70	1.74	1.78	1.82	1.86
τ_2	0.35	0.35	0.31	0.33	0.35	0.37	0.40
β_0	3.73	3.74	3.70	3.72	3.74	3.76	3.78
β_1	-1.00	-0.99	-1.07	-1.02	-0.99	-0.95	-0.91
β_{enz}	0.10	0.10	0.07	0.08	0.10	0.12	0.13
β_{gcc}	-0.10	-0.10	-0.14	-0.13	-0.11	-0.09	-0.06
β_{map}	0.10	0.08	0.03	0.06	0.08	0.10	0.12
ϕ	15.00	15.66	12.12	13.78	15.39	17.18	18.92

1.3.3 Simulation study when Hi-C data is simulated from multiple piecewise helical curves

We further conducted a simulation study to illustrate the performance of the piecewise helical model for inferring whole chromosome structure. For illustrative testing purpose, we only use the first 200 40kb bins in mouse chromosome 19, 3,080,001 - 11,320,000. We will show the result of whole chromosome analysis in the real data section. Each 40kb bin is treated as a bead in the beads-on-a-string representation, and chromatin is assumed to fold as two piecewise helical curves, each consists of two equal-sized helices with curvatures and torsions listed below:

	Helix 1	Helix 2
Curve 1	$\kappa_1^{(1)} = 0.6$	$\kappa_1^{(2)} = 0.4$
	$\tau_1^{(1)} = 0.2$	$\tau_1^{(2)} = 0.1$
Curve 2	$\kappa_2^{(1)} = 0.5$	$\kappa_2^{(2)} = 0.3$
	$\tau_2^{(1)} = 0.1$	$\tau_2^{(2)} = 0.2$

We randomly generate the orientation vectors $(\boldsymbol{t}, \boldsymbol{n}, \boldsymbol{b})$'s at the connection points, as well as the orientation between two helical curves. The arc length between the center of the i-th bin and the center of the j-th bin is proportional to the genomic distance between them ($|i-j|$). Linear link function is used to connect Hi-C data and spatial proximity, with $\beta_0 = 5$, $\beta_1 = -1$. We further set $\phi = 15$, $\beta_{enz} = 0.1$, $\beta_{gcc} = -0.1$, $\beta_{map} = 0.1$. The local genomic features of each bin, including restriction enzyme cutting frequency, GC content and sequence mappability, are calculated from the UCSC reference genome mm9. The Hi-C contact matrix $\{y_{ij}\}_{1 \le i < j \le 200}$ is simulated from the posited negative binomial model. The true log likelihood is 938630.54. We normalize the absolute scale of the piecewise helical curve and transform the curvature, torsion and arc length parameters accordingly. The parameters become: $\beta_0 = 2.68$, $\beta_1 = -1$,

	Helix 1	Helix 2
Curve 1	$\kappa_1^{(1)} = 6.15$	$\kappa_1^{(2)} = 4.10$
	$\tau_1^{(1)} = 2.05$	$\tau_1^{(2)} = 1.03$
Curve 2	$\kappa_2^{(1)} = 5.13$	$\kappa_2^{(2)} = 3.08$
	$\tau_2^{(1)} = 1.03$	$\tau_2^{(2)} = 2.05$

We applied our divide and combine two step algorithm to the simulated data, and statistical inference is conducted on the unknown parameters. We divide the whole piece into two equally sized helical curves and infer the structure in parallel just using the two squared 100×100 blocks (1-100, and 101-200) along the diagonal of the contact matrix. In order to infer the orientation between the two pieces, we use the middle 100×100 (from 51 to 150) squared matrix along the diagonal of contact matrix. Finally, we rotate the second piece according to the transformation of inferred $\boldsymbol{t}, \boldsymbol{n}, \boldsymbol{b}$ vectors, and connect it to the first piece to complete the inference procedure. Table 1.3 lists the summary statistics of the posterior distributions of some primary parameters. For each parameter, the 95% credible interval covers the true value.

Our proposed PHM method is not identifiable up to a scaling factor as mentioned earlier, and the structure of the whole chromosome is not identifiable up to rotation (with different starting orientations). For example, as shown in Fig. 1.9, the two helical curves have the same parameters except their starting orientations. Hence, they are essentially the same. In reconstruction of the structure of chromosome, our main focus is the relative distance between the bins; therefore, we would consider the two recovered helical curves in Fig. 1.9 both valid.

Thus, we calculate the pairwise Euclidean distance matrices of the structures from the simulation and prediction. The Spearman correlation between the two matrixes is calculated and used to assess the accuracy of proposed two-step model. To the simulated datset, our proposed two-step divide and combine PHM achieves 0.93 correlation. Though we didn't use all the cells of the original contact matrix, we can

Table 1.3.: Summary statistics of the posterior distribution of main parameters in simulation study when Hi-C data is simulated from multiple piecewise helical curves

Parameter	Truth	Posterior Mode	Posterior distribution				
			Min	2.5% Percentile	Mean	97.5% Percentile	Max
$\kappa_1^{(1)}$	6.15	6.21	6.03	6.08	6.21	6.33	6.40
$\tau_1^{(1)}$	2.05	2.07	1.89	1.95	2.07	2.16	2.25
$\kappa_2^{(1)}$	4.10	4.12	4.03	4.05	4.13	4.19	4.24
$\tau_2^{(1)}$	1.03	1.01	0.95	0.96	1.02	1.07	1.11
$\kappa_1^{(2)}$	5.13	4.95	4.75	4.81	4.97	5.12	5.17
$\tau_1^{(2)}$	1.03	0.96	0.87	0.90	0.97	1.03	1.07
$\kappa_2^{(2)}$	3.08	3.16	2.95	3.01	3.13	3.23	3.30
$\tau_2^{(2)}$	2.05	2.02	1.72	1.79	2.01	2.20	2.26

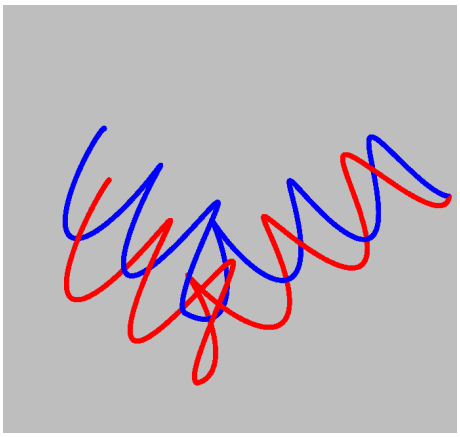


Fig. 1.9.: Two exactly same piecewise helical curve just with different starting orientations

still recover the main structure of the multiple piecewise helical curve and keep the relative relationship.

1.3.4 Simulation study when Hi-C data is simulated from mixture of two piecewise helical curves

Last, we conducted a simulation study to illustrate the performance of mixture piecewise helical model. For illustrative testing purpose, we used the first 50 40kb bins in mouse chromosome 19, 3,080,001 - 11,320,000. Each 40kb bin is treated as a bead in the beads-on-a-string representation, and the cell population contains two distinct structures, each is assumed to fold as piecewise helical curve.

The arc length between the center of the i-th bin and the center of the j-th bin is proportional to the genomic distance between them ($|i-j|$). Linear link function is used to connect Hi-C data and spatial proximity, with $\beta_0 = 5$, $\beta_1 = -1$. We further set $\phi = 15$, $\beta_{enz} = 0.1$, $\beta_{gcc} = -0.1$, $\beta_{map} = 0.1$. The local genomic features of each bin, including restriction enzyme cutting frequency, GC content and sequence mappability, are calculated from the UCSC reference genome mm9. The Hi-C contact matrix $\{y_{ij}\}_{1\le i<j\le 50}$ is simulated from the posited negative binomial model. The true log likelihood is 292478.0804. We normalize the absolute scale of the piecewise helical curve and transform the curvature, torsion and arc length parameters accordingly. After normalization, the curvatures and torsions are as follows:

	Helix 1	Helix 2
Portion 1	$\kappa_1^{(1)} = 3.29$	$\kappa_1^{(2)} = 3.29$
	$\tau_1^{(1)} = 0.55$	$\tau_1^{(2)} = 1.10$
Portion 2	$\kappa_2^{(1)} = 2.74$	$\kappa_2^{(2)} = 2.19$
	$\tau_2^{(1)} = 0.55$	$\tau_2^{(2)} = 1.10$

Table 1.4.: Summary statistics of the posterior distribution of main parameters in simulation study when Hi-C data is simulated from mixture piecewise helical model

Parameter	Truth	Posterior Mode	Posterior distribution				
			Min	2.5% Percentile	Mean	97.5% Percentile	Max
$\kappa_1^{(1)}$	3.29	3.19	2.61	2.70	3.10	3.57	3.78
$\tau_1^{(1)}$	0.55	0.50	0.22	0.25	0.37	0.51	0.56
$\kappa_2^{(1)}$	3.29	3.20	2.99	3.06	4.28	5.85	6.15
$\tau_2^{(1)}$	1.10	1.01	0.84	0.87	1.42	2.19	2.56
$\kappa_1^{(2)}$	2.74	2.62	2.40	2.44	4.68	7.47	7.78
$\tau_1^{(2)}$	0.55	0.51	0.37	0.41	1.30	2.59	2.80
$\kappa_2^{(2)}$	2.19	2.03	0.66	1.23	1.92	2.23	2.76
$\tau_2^{(2)}$	1.10	0.90	0.59	0.71	1.21	4.05	6.35

We applied mixture piecewise helical model to the simulated data, and statistical inference is conducted on the unknown parameters. Table 1.3 lists the summary statistics of the posterior distributions of some primary parameters. For each parameter, the 95% credible interval covers the true value.

We calculate the pairwise Euclidean distance matrices of structures for two sub-populations in the mixture model. The Spearman correlation between matrixes for simulation and prediction are calculated and used to assess the accuracy of proposed mixture model. The correlations for two sub-populations are 0.997 and 0.994, respectively, which indicates the high similarity between the inferred and simulated structures.

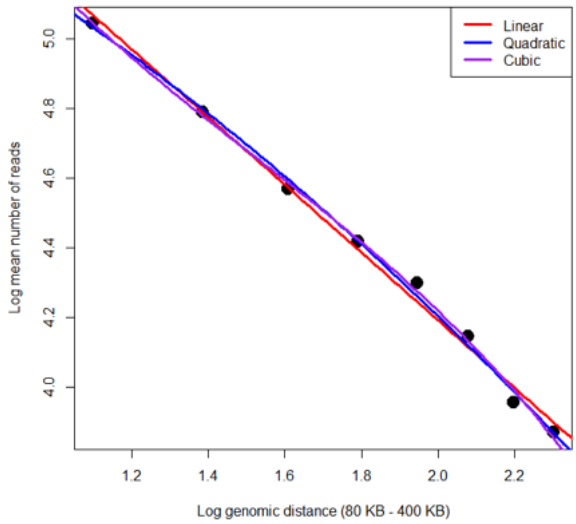
1.4 Real Data Application

1.4.1 Data description

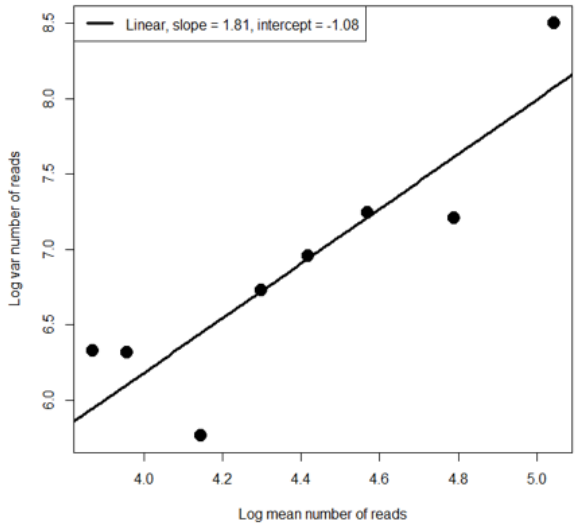
Simulation studies have shown that the piecewise helical model works well on the synthetic datasets. To evaluate the performance of the piecewise helical model on real Hi-C data, we choose a published real Hi-C dataset on mouse embryonic stem (ES) cells [22]. The Hi-C experiments were conducted using two restriction enzymes, HindIII and NcoI. The data sets with restriction enzyme HindIII and NcoI are referred as the HindIII sample and the NcoI sample, respectively.

We first investigate the log average number of Hi-C reads spanning two loci as a function of the log genomic distance between the loci. We observe that the function is approximately linear when the genomic distance is around the size of a TAD (Fig. 1.10 (a)). Therefore, a linear link function is used to analyze Hi-C data within each TAD.

Next, we evaluate the additional variation in Hi-C data after adjusting for genomic distance. We select all loci pairs with the same genomic distance, and plot the mean versus the variance of Hi-C read count spanning two loci. Interestingly, the variance is a quadratic function of the mean, suggesting a strong over-dispersion pattern (Fig. 1.10 (b)). Similar over-dispersion patterns have been observed in other types of next-generation sequencing count data, such as RNA-Seq data [29] and ChIP-Seq data [30]. Using the negative binomial model is a popular approach to analyzing such over-dispersed count data. Therefore, we apply the same approach to account for the over-dispersion phenomenon in our proposed model. In addition, we apply BACH to TADs in mouse chromosome 18. The 3D chromosomal structures predicted by BACH demonstrate a helix-like shape (Fig. 1.11). This observation motivates us to use the parsimonious piecewise helical curve representations for chromatin folding.



(a)



(b)

Fig. 1.10.: Exploratory analysis of 40 KB resolution Hi-C contact matrix of a topological domain in mouse chromosome 18, 33,960,001 – 34,960,000. (a): The decreasing trend of Hi-C data with the increase of genomic distance (80 KB – 400 KB). The linear line (red) fits data as well as the quadratic line (blue) and the cubic line (purple). (b): Over-dispersion of 40 KB resolution Hi-C data. Both x-axis and y-axis are in logarithm scale. The slope of the fitting line is 1.81, suggesting that the variance is a quadratic function of the mean.

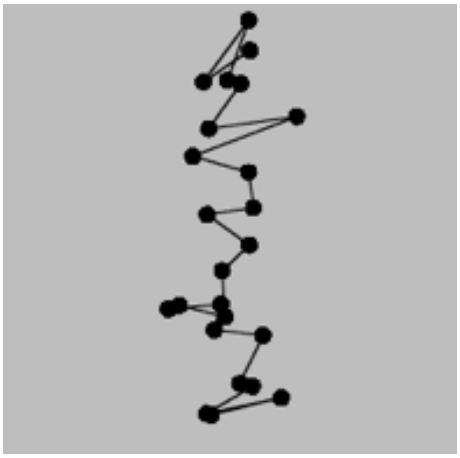
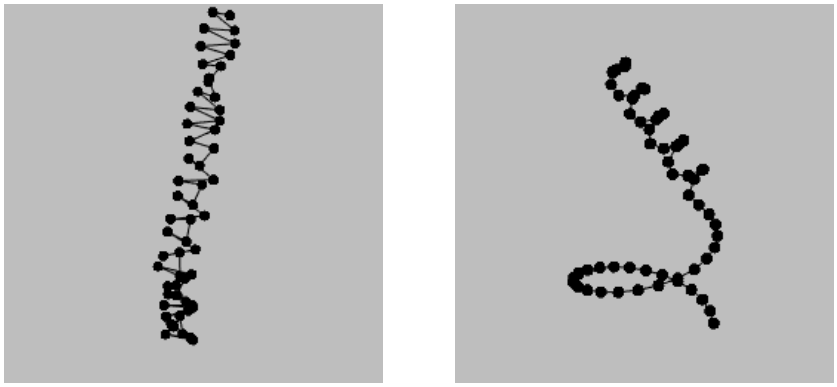


Fig. 1.11.: BACH-predicted 3D chromosomal structures under the beads-on-a-string representation. 3D model of a topological domain in mouse chromosome 18, 33,960,001 – 34,960,000. Each bead represents one 40 KB bin.



(a) (b)

Fig. 1.12.: 3D chromosomal structures predicted by the piecewise helical model.
(a) 3D model of a topological domain in mouse chromosome 2, 6,360,001 - 8,600,000.
(b) 3D model of a topological domain in mouse chromosome 8, 22,040,001 - 24,280,000.

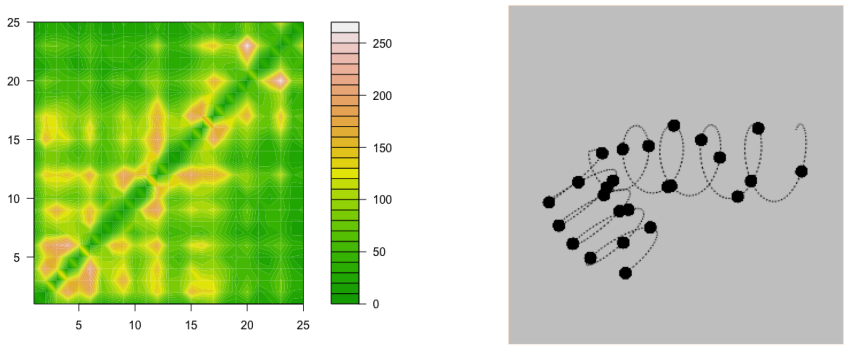
1.4.2 Results of PHM on TADs

Piecewise helical model is applied to TADs of mouse ES cells [22] to infer the geometric properties of chromatin folding. Among all 2,200 TADs in mouse ES cells, 1,639 TADs with size larger than 480kb (12 loci) are analyzed. For each TAD, the predicted piecewise helical curve is obtained and the number of loops is used to measure the geometric property (i.e. compactness) of the piecewise helical curve. Note that the number of loops only reflects a small amount of characteristics of the piecewise helical curve structure. A larger number of loops indicates a more compact curve, while a smaller number of loops indicates a looser curve. In Fig. 1.12, we compare the predicted piecewise helical models of the two TADs in mouse chromosome 2, 6,360,001 – 8,600,000 and chromosome 8, 22,040,001 – 24,280,000. Both of the two TADs have 56 40kb bins. We can see that the TAD in chromosome 2 contains 24 loops, and it is more compact than the TAD in chromosome 8 with 6 loops.

We next evaluate how the compactness of chromatin folding, as measured by the number of loops, is correlated with genetic and epigenetic features. We collect eleven markers for each TAD, including gene density (UCSC reference genome mm9), gene expression [31], promoter marker H3K4me3 [31], active chromatin marker RNA polymerase II [31], H3K36me3 [32], repressive chromatin marker H3K27me3 [33], H3K9me3 [34], H4K20me3 [33], DNA hypersensitivity marker DNaseI [35], DNA replication timing [36] and genome-nuclear lamina interaction [37]. The estimated Pearson correlations coefficients between the compactness of chromatin folding and the genetic and epigenetic features are presented in Table 1.5. Statistical hypothesis testing shows that they are all significant. It is clear that gene-rich, highly expressed and early replicated genomic regions are much looser, whereas gene-poor, lowly expressed and later replicated genomic regions are more compact.

To show the advantage of allowing flexible connection between two consecutive helixes within the piecewise helical curve, we used the TAD in mouse chromosome 1, 57,440,001 - 58,440,000 as an illustrative example. The piecewise helical curve with two helixes, which achieves the lowest BIC, is preferred for this TAD in PHM. Fig. 1.13 (a) is the raw Hi-C contact heatmap, which shows long-range chromatin interactions. In the reconstructed structure as shown in Fig. 1.13 (b), the two helixes bend at the connection point to account for the long-range interactions.

PHM is then applied to both of the HindIII and NcoI samples of the longest 500 TADs in mouse ES cells [22] to predict their 3D structures. We calculate the pairwise Euclidean distance matrices of the predicted structures from the two samples. The Spearman correlation between the two matrixes is calculated and used to assess the reproducibility of PHM. To compare with existing methods, we repeat this procedure and obtain the corresponding Spearman correlations for chromSDE, three variants of Pastis methods (Pastismds, Pastis-nmds, Pastis-pm1). The results are listed in Fig. 1.14 and Table 1.6. PHM achieves the highest correlation, indicating its higher reproducibility.



(a) (b)

Fig. 1.13.: 3D chromosomal structures predicted by the piecewise helical model. (a) raw Hi-C contact matrix heatmap of topological domain in mouse chromosome 1, 57,440,001 – 58,440,000. (b) reconstructed 3D model from PHM, and each bead represent one 40kb bin.

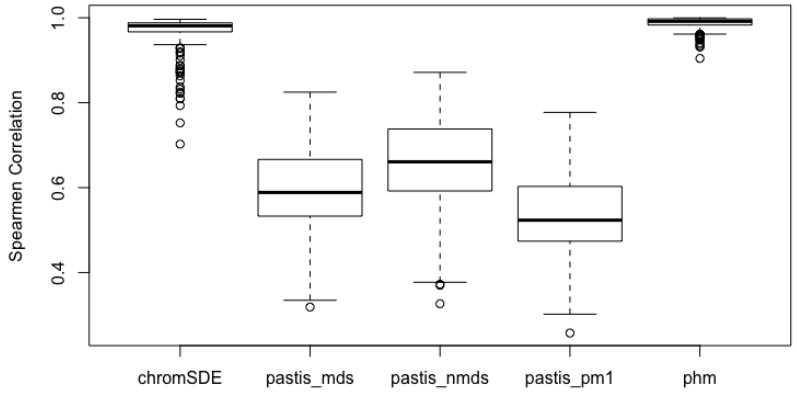


Fig. 1.14.: Reproducibility of inferred 3D chromosomal structure of 500-topological-domain genomic regions between the HindIII sample and the NcoI sample.

Table 1.5.: Pearson correlation coefficients between the number of loops and the genetic and epgenetic features.

Features	Number of Loops	
	HindIII	NcoI
Gene density	-0.26	-0.25
Gene expression	-0.09	-0.08
RNA polymerase II	-0.22	-0.21
H3K36me3	-0.33	-0.31
H3K27me3	-0.30	-0.29
H3K4me3	-0.31	-0.30
DNaseI	-0.38	-0.36
Replication timing	-0.39	-0.37
Lamina	0.39	0.37
H3K9me3	0.34	0.31
H4K20me3	0.30	0.29

Table 1.6.: Spearman correlation coefficients between the HindIII samples and the NcoI samples.

Chrom SDE	Pastis-mds	Pastis-nmds	Pastis-pm1	PHM
0.9691	0.5969	0.6621	0.5401	0.9878

Zhang et al. [15] showed that the mapping from contact counts to physical distance can vary from one resolution to another and proposed a procedure to assess the stability of a 3D structure reconstruction method with respect to change in res-

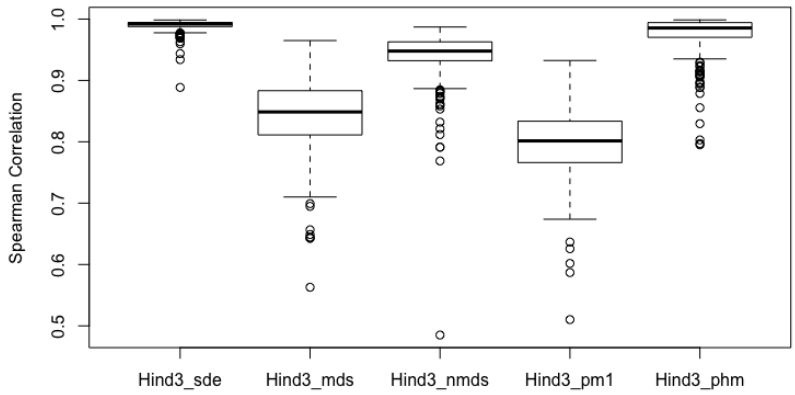


Fig. 1.15.: Stability across resolutions. Spearman correlation between pairs of structures at 40kb and 80kb resolutions of inferred 3D chromosomal structure of 500-topological-domain genomic regions of the HindIII sample.

olution. The procedure can be described as follows. Assume that $X \in \mathbb{R}^{3\times N}$ and $Y \in \mathbb{R}^{3\times M}$ denote predicted structures for different resolution levels with $N < M$, indicating that X is constructed at the lower resolution level. A downsampled structure $Y^* \in \mathbb{R}^{3\times N}$ from Y at the same resolution as X is obtained by averaging the coordinates of loci. Then the Spearman correlation between distance matrices from X and Y^* is calculated and used as a stability measure. We apply this procedure to PHM, ChromSDE, three variants of Pastis for the selected first 500 longest TADs separately. The resulting Spearman correlations are presented in Fig. 1.15, 1.16 and Table 1.7. Our PHM method achieves comparable Spearman correlation coefficients with ChromSDE, and significantly outperforms the three Pastis methods.

1.4.3 Model validation with gold standard FISH data

We further use a FISH dataset independently generated by [38] for the same mouse ES cells to validate our PHM method. In the FISH experiment, spatial distances of six

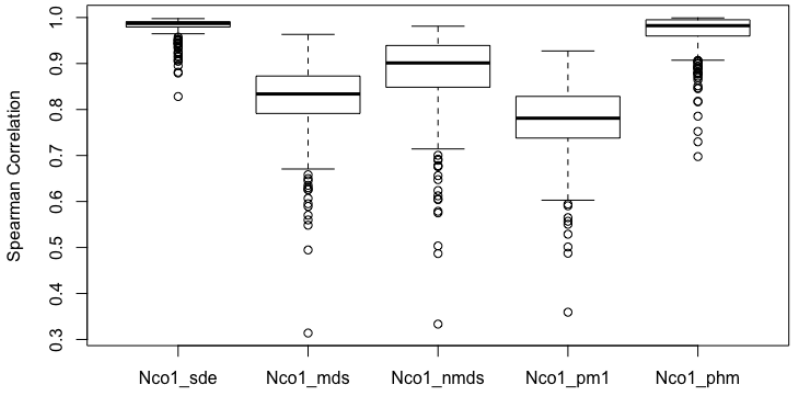


Fig. 1.16.: Stability across resolutions. Spearman correlation between pairs of structures at 40kb and 80kb resolutions of inferred 3D chromosomal structure of 500-topological-domain genomic regions of the NcoI sample.

Table 1.7.: Spearman correlation coefficients between inferred structures at different resolutions.

	Chrom SDE	Pastis-mds	Pastis-nmds	Pastis-pm1	PHM
HindIII	0.9901	0.8419	0.9419	0.7997	0.9773
NcoI	0.9822	0.8249	0.8836	0.7773	0.9697

pairs of genes, which belong to four TADs, were directly measured. These distances are referred to as the FISH distances and used as the gold standard. The detailed information about the FISH experiment and the four TADs (1-4) can be found in Table 1.8 and 1.9.

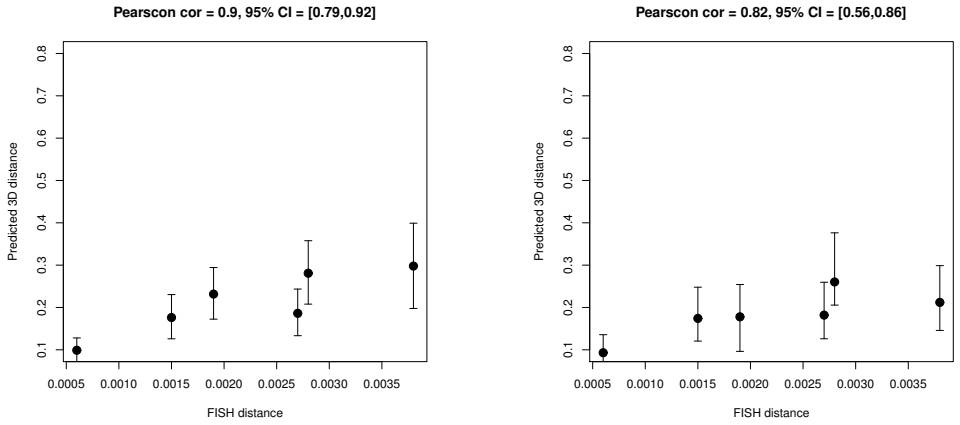
Table 1.8.: Six pairs of genes with FISH measurement.

Gene pairs	FISH distances
Lnp-GCR	0.0027
Hoxd3-Evx2	0.0015
Rcn1-1550J22	0.0038
Hbq-Il9r	0.0019
Hoxb1-Calcoco2	0.0028
Hoxb1-Hoxb9	0.0006

We apply PHM with two helixes to TADs 1 and 2, and PHM with one helix to TADs 3 and 4. The HindIII sample and NcoI sample are analyzed separately. Based on the results, the Pearson correlation coefficients between the predicted spatial

Table 1.9.: Topological domain annotation for genes with FISH measurement.

Domain ID	Chrom	Start	End	Size	Genes within domain
1	2	73762329	74606210	0.84	MB GCR, Lnp, Evx2, Hoxd3
2	2	105100659	106026498	0.93	MB Rcn1, 1550J22
3	11	32058671	33038791	0.98	MB Il9r, Hbq1
4	11	94495571	97482060	2.99	MB Calcoco2, Hoxb1, Hoxb9



(a) (b)

Fig. 1.17.: Predicted spatial distance using the piecewise helical model vs. the gold standard FISH data. (a) HindIII samples (b) NcoI samples.

distances and the gold standard FISH distances are calculated, which are 0.90 and 0.82 (Fig. 1.17) for HindIII sample and NcoI sample, respectively.

As a comparison, we also apply BACH, ChromSDE, and the three variants of Pastis to infer the structures of the four TADs. The resulting Pearson correlations with the gold standard FISH distances of all the methods are presented in Table 1.10. For the HindIII sample, the top three performers are Pastis-mds, Pastis-pm1, and

PHM. For the NcoI sample, the top three performers are PHM, BACH, and Pastis-pm1. Even though the two Pastis methods, Pastis-mds, Pastispm1 perform better for the HindIII sample than PHM, their performances for the NcoI sample are inferior. Considering both of the samples, PHM achieves better overall performance than the other competing methods.

Table 1.10.: Pearson correlation coefficients with FISH data.

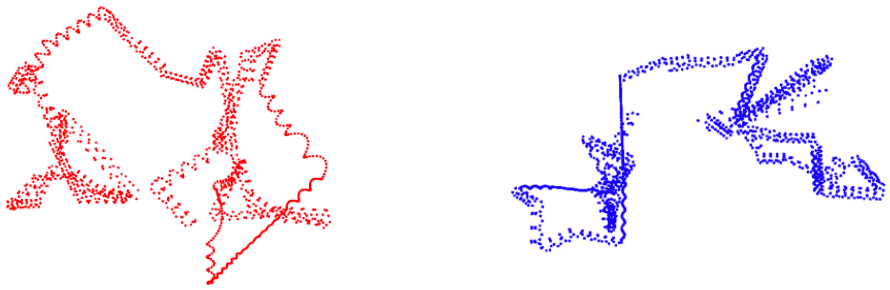
	HindIII	NcoI
PHM	0.90	0.82
BACH	0.88	0.78
ChromSDE	-0.14	-0.14
Pastis-mds	0.99	0.48
Pastis-nmds	0.83	0.03
Pastis-pm1	0.97	0.46

1.4.4 Results of PHM on whole Chromosomes

Validate parallel PHM using two enzyme replicates We applied the two-step divide and combine procedure to the 40 KB resolution Hi-C contact matrices to generate 3D chromosomal structure for each mouse chromosome.

We compute the Spearman correlation between the distance matrices of the estimated structure from one enzyme to the predicted structure from the other enzyme data. Spearman correlation instead of Pearson correlation is used because the Spearman correlation is independent of the scale of the piecewise helical curve.

Fig. 1.18 lists the alignment of two 3D chromosomal structures predicted by parallel PHM from mouse chromosome 19 in both HindIII sample and NcoI sample. The



(a) (b)

Fig. 1.18.: The predicted structure of chromosome 19 from mESC cells under 40kb resolution. (a) HindIII samples. (b) NcoI samples

high similarity between the two structures shows that the 3D structures of different enzymes predicted by parallel PHM method are highly reproducible.

2. DEEP NEURAL NETWORK BASED BAYESIAN ESTIMATORS AND MODEL SELECTORS

In this chapter, we focus on developing new Neural Network based methods to automate statistical analysis via supervised learning, including parameter estimation and model selection for large scale datasets. We proposed two inference frameworks based on whether we have knowledge about the model or not. We proposed neural network based Bayes estimator when the model is known but the involved parameters need to be estimated. When the model and its parameters are both unknown, we proposed a neural model selector and a neural parameter estimator to do perform model selection and parameter estimation simultaneously.

2.1 Introduction

Breiman [39] compared two general approaches to data analysis. The first approach is called the data modeling approach, and the second the algorithmic modeling approach. The data modeling approach starts with the assumption that the data set under consideration is sampled from a parametric model with known form and a set of unknown parameters. The main tasks of data analysis then focus on parameter estimation and hypotheses testing, and the results can be further used to support discovery, decision making, and prediction. The data modeling approach has been the main approach adopted by statisticians for the past one hundred years.

The second approach is essentially the machine learning approach, and as a visionary, Breiman correctly predicted that machine learning would become more and more powerful and popular, especially in the era of big data. According to the definitions of Gartner [40] and De Maro et al. [41], big data refer to information assets with characteristics such as high volume, high velocity, and/or high variety, and the

transformation from big data to value requires specific analytical methods. Currently, machine learning methods are used as the main tools for big data analytics, which emphasize algorithms instead of statistical analysis (SA).

The quick rise of machine learning does not mean that the data modeling approach becomes obsolete or useless. First of all, data generated from well-designed experiments following sound statistical principles admit proper statistical models and thus can be better analyzed by the data modeling approach. Second, for applications in which interpretability is important, the data modeling approach may still be more preferable. The advent of big data era raises new challenges to the data modeling approach. For example, with the help of new technologies, modern large scale health-focused cohort studies can generate big and diverse data, including genomic data, behavioral data, medical record data, environmental data, etc. The integrative analysis of all of those different types of data requires parametric models that are much larger and more complex than those considered in the past. Conventional statistical methods for parameter estimation and hypothesis testing can become ineffective under such models.

Due to the complexity of the data analysis process, we will narrow it down to two most important procedures, which are model building and parameter estimation. We first reformulate model building and parameter estimation as a machine learning problem. Suppose $\mathcal{M} = \{M_k : 1 \leq k \leq K\}$ is a collection of K prespecified models/distributions. Let $f(y|\theta_k, M_k)$ be the density function of a general statistical model M_k, where $\theta_k \in \Theta_k$ represents the parameter and can be a vector. Assume that we have a random sample from one of the models, which is $\{y_j\}_{1\leq j\leq N}$, but we do not know the data-generating procedure and corresponding parameters. The goal is to identify the model that fits the data best and further estimate the parameters of the model.

Neural Bayes Estimator

Model selection and parameter estimation are two tasks which interplay with each other. First we review the three major paradigms for handling parameter estimation given that we know the model from which the sample is drawn from. The three paradigms are frequentist, Bayesian, and decision-theory paradigms, respectively. The frequentist's estimator of θ is a mapping from a sample to the parameter space, which is expected to have nice frequentist properties such as unbiasedness and minimum variance. The maximum likelihood principle is popularly used to produce the frequentist estimator called the Maximum Likelihood Estimator (MLE).

In the Bayesian paradigm, the posterior distribution of θ is obtained to represent the current knowledge or state about θ, combining the information from the sample and the prior distribution. And if necessary, the posterior distribution can be further used to produce a point estimator of θ. Typically, Monte Carlo Markov Chain (MCMC) [42] algorithms are used to calculate the posterior distribution. MCMC works even if the likelihood function can only be computed conditionally or up to a normalizing constant. However, when a large and complex model is postulated, the calculations of both the MLE and the posterior distribution of θ can become computationally challenging and even infeasible [43]. When likelihoods of models are unknown or computationally intractable, Approximate Bayesian Computation (ABC) [44–50] methods can be used to approximate posterior distributions. ABC algorithms sample from the posterior distribution of the parameters by finding values that yield simulated data sufficiently resembling the observed data. Approximation accuracy and computational efficiency of ABC depend on the choice of summary statistics. In the literature, there are two classes of ABC methods. The first class employs best subset selection, which chooses the best summary statistics based on information-based criteria among a set of candidate summary statistics. The second class uses linear regression based approach, which constructs summary statistics by linear regression of response on candidate summary statistics [51]. In [52], the authors proposed to

automate the process of constructing summary statistics by training deep neural networks and the resulting summary statistics are approximately posterior means of the parameters. The authors of [52] mainly focused on the regime when the dimension of X is moderately high (e.g. sample size is 100) and the dimension of parameter θ is low (e.g. 1, 2, 3).

The third paradigm of parameter estimation is based on decision theory [53]. An estimator of θ is considered a decision rule (denoted by δ) of using a function of the sample to approximate θ. Let $l(\delta, \theta)$ denote the loss function. Then the best estimator is chosen to be the rule δ^* such that it has the best performance according to a certain type of overall performance measure. One such overall performance measure is the so-called average risk, which is defined to be the weighted average of the expected loss (or risk) over sampling. The exact definition is deferred to the next section. If the weight function used to calculate the average risk is treated as a prior imposed on θ, then by applying the Bayes formula and the exchange of integration order with respect to x and θ, the estimator with minimum average risk is the minimizer of the expectation of $l(\delta, \theta)$ over the posterior distribution of θ given the sample, and therefore the estimator is called the Bayes estimator.

The Bayes estimator can be considered a specific way to construct a point estimator from the posterior distribution produced in the Bayesian paradigm. Good estimators produced from the frequentist paradigm often are limits of Bayes estimators. Bayes estimators enjoy various nice properties such as admissibility [54]. The explicit expression of Bayes estimators are available only under simple models and standard loss functions. When under large and complex models or nonstandard loss functions, computing algorithms must be used to calculate Bayes estimators. The conventional approach is again to first apply MCMC algorithms to calculate the posterior distribution of θ and second to solve the optimization problem with respect to the posterior distribution. As discussed previously, using MCMC algorithms to calculate the posterior distribution can quickly become infeasible when the model becomes larger and more complex. The fundamental reason is that the calculation

of the posterior distribution is by itself a difficult task, and probably is more difficult than what the definition of Bayes estimator requires.

Recall that Bayes estimators are originally defined as solutions of optimization problems involving integration over the statistical model $f(x|\theta)$ and the prior function instead of the posterior distribution. One may wonder if the Bayes estimator can be calculated through solving the original optimization problem. After some exploration, it turns out that directly solving the original problem can run into other difficulties and can be just as challenging.

Deep neural networks (DNNs) ([55], [56]) have achieved remarkable performances in a variety of applications including competitive game play, object recognition, machine translation, and speech recognition in recent years, and promise to deliver artificial intelligence (AI) in almost every aspect of human life and society. We believe that DNNs can also be used to automate the data analysis process, bring AI to large scale statistical analysis, and make data analysis popular for big data analytics.

We propose to use Deep Neural Nets (DNN) to construct the Bayes estimator through supervised machine learning, and we call the proposed estimator the Neural Bayes Estimator (NBE). The description of the exact construction and learning will be presented in the next section.

The Neural Bayes Estimator possesses a number of other advantages over the conventional approach. First, the conventional approach needs to engineer new algorithms to compute the posterior distribution whenever a new model is considered, and therefore, it is ad hoc; whereas for the neural estimator, only the input data need to be generated from the new model, but the algorithm for training the neural Bayes estimator remains the same. Therefore, the neural Bayes estimator is a unified approach for all different models. Second, thanks to the fast progress in artificial intelligence and deep learning, there are a number of powerful platforms such as TensorFlow [57] and PyTorch [58], which can be directly used to train the neural Bayes estimator with a minimum amount of additional coding. As a matter of fact, we will adopt Tensorflow as the platform to train the neural Bayes estimator in this chapter.

Thirdly, even under the same model, when the parameters may change and new sample are available, the Bayes estimator needs to recalculated. This is not necessary for the neural Bayes estimator. Once the neural Bayes estimator has been trained, as long as the model form does not change, the trained model can generate the Bayes estimator for different samples.

The proposed neural Bayes estimator is intended for handling any statistical models with any number of parameters, not limited to a specific model. For example, it can be used for the FRAME (Filters, Random field, And Maximum Entropy) model [59] for texture modeling; topic models for uncovering the underlying semantic structures of document collections based on hierarchical Bayesian analysis [60], among other statistical models for complex machine learning tasks. For ease in discussion and illustration, we choose the class of powerful statistical models called the generalized linear mixed effects models (GLMM) [61] as the main working model for both illustration and implementation purposes. They play a key role in epidemiology, clinical research, and social sciences. GLMMs are a powerful class of statistical models that incorporates random effects into the linear predictor of a generalized linear model (GLM) to account for the correlation of responses. The parameters of interests in GLMMs are the fixed-effect parameters (effects of covariates, differences among treatments and interactions) and random-effect parameters (the standard deviations of the random effects). Though these exist accurate techniques for parameter estimation in simple GLMMs, it's still challenging for complex GLMMs.

Neural Model Selector and Parameter Estimator

What if we do not know the model from which the sample is drawn from, and can be any in the candidate set? For this case, we need to perform the model building procedure and further estimate the corresponding parameters. Conventionally, model building is either done by the analyst based on the results of exploratory data analysis and his/her own knowledge, or it is done via model selection using statistical principles

and criteria. Both of these two approaches are however difficult to automate. There exists an extensive statistical literature on model selection [62–64]. Numerous model selection methods have been proposed. Some of these methods are not applicable to the setting we consider here, while others, though applicable, may run into various difficulties; we will talk about difficulties later. Numerous statistical methods for parameter estimation [65–67], including the major three major paradigms mentioned earlier, rely on full or partial knowledge of the model and are based on statistical principles.

Here, we will briefly discuss several representative approaches for model selection, which include the Kolmogorov-Smirnov (KS) distance [68], Bayesian Information Criterion (BIC) [69], and Bayes factor [70]. The KS distance method calculates the distance between the population Cumulative Distribution Function (CDF) and the empirical CDF based on the sample $\{y_j\}$ for each model. The model that achieves the minimum distance will be selected as the true model. The BIC criterion calculates the BIC score for each model as follows:

$$\mathrm{BIC}(M_k) = -2\log L(\hat{\theta}_k) + \log(n)p$$

where $L(\cdot)$ is the likelihood function, $\hat{\theta}_k$ is the maximum likelihood estimate, and p is the number of parameters in the model M_k. Note that for the scenario considered here, $p = 1$. The model that achieves the minimum BIC score will be selected as the true model.

The Bayes factor method will impose a prior distribution to the models, $\pi(M_k)$, and further impose a prior distribution to the parameter $\pi(\theta_k)$. Then, given the model, the likelihood can be calculated, which is denoted as $\pi(\{y_j\}|M_k)$. The Bayes factor between any two models, M_{k_1} and M_{k_2}, can be calculated as $\mathrm{BF}(M_{k_1}, M_{k_2}) = \pi(\{y_j\}|M_{k_1})/\pi(\{y_j\}|M_{k_2})$, which can be used to discriminate between the two models. The model the BF scores support the most will be selected as the true model.

Our criticism for the conventional statistical approaches discussed above is two-fold. First, for the goal of automating model selection, the model set $\mathcal{M}$ usually consists of a large number of candidate models, and the models are of huge variety.

The conventional statistical methods such as the KS distance and BIC only work for selection between nested or other well-structured models. Second, for a given sample, all the conventional methods will have to calculate a score for each of the candidate models, and then compare them to pick the winner. This can become computationally intensive or even intractable, especially for the Bayes factor approach.

Instead, we propose to use CNNs and machine learning to automate model selection and parameter estimation. Our main idea is that the procedures for model selection and parameter estimation can be considered mappings from the sample to a model and a value of the model parameter, that is,

$$G : \{y_j\} \to \begin{pmatrix} G_1(\{y_j\}) \\ G_2(\{y_j\}) \end{pmatrix} \in \mathcal{M} \times \Theta,$$

where $G = (G_1, G_2)$ consists of the model selection mapping G_1 and the parameter estimation mapping G_2, and Θ is the parameter space. Instead of using statistical principles to derive G_1 and G_2, we propose to use CNNs to approximate them. From here on in the rest of the paper, we refer to G_1 as the *neural model selector*, and G_2 the *neural parameter estimator*. We propose to construct CNNs to approximate G_1 and G_2. Data systematically simulated from candidate models will be used to train the neural model selector. The idea of using neural networks to estimate parameters in a stochastic model is not entirely new, but it is neither well-known nor a common practice in the statistical community.

We further explore the possible interplay between the neural model selector and the neural parameter estimator. As will be discussed in Section 2.3, the two CNNs can be entirely separate, almost identical, or partially joint, leading to different performances in training as well as in application. We carry out extensive simulation studies, and show that the proposed neural model selector and parameter estimator can be properly trained, and the trained CNNs demonstrate excellent performance.

To our best knowledge, there is no prior work about redefining the model selection problem as a machine learning classification problem and training CNN to learn and perform model selection with labeled simulated data. After conducting intensive

literature search, we only found one paper [71], in which the authors proposed to use artificial neural networks and simulated data to construct estimates for parameters of a stochastic differential equation. However, the idea of using CNNs and simulated data to automate parameter estimation and model selection and bring AI to the general data analysis process appears to be novel to our best knowledge.

2.2 Neural Bayes Estimator

2.2.1 Proposed Method

Let $\mathcal{F} = \{f(x|\theta) : \theta \in \Theta\}$, where Θ is the parameter space (i.e., the set of all possible values of θ). We call $\mathcal{F}$ the model space. Let X be the space of samples drawn from a model in $\mathcal{F}$. As discussed in the Introduction, an estimator δ is a mapping or decision rule from sample space X to parameter space Θ. Recall that the loss function is $L(\delta, \theta)$. The risk of δ, denoted as R, is defined as follows.

$$R(\delta, \theta) = \int_X L(\delta(x), \theta) f(x|\theta) dx.$$

It is clear that R depends on θ, and the decision rule that achieves minimum R for all θ does not exit. Instead, we consider the average risk, which is denoted as r and defined as follows.

$$r(\delta) = \int_\Theta R(\delta, \theta)\lambda(\theta) d\theta = \int_\Theta \int_X L(\delta(x), \theta) f(x|\theta)\lambda(\theta) dx d\theta,$$

where $\lambda(\theta)$ is the weight function. The decision rule or estimator with minimum average risk is defined as the estimator δ^* such that it minimizes $r(\delta)$ among all possible estimators, that is,

$$\delta^* = \arg\min_\delta r(\delta). \tag{2.1}$$

Let $f(x)$ be the marginal density for X, and $\lambda(\theta|x) = f(x|\theta)\lambda(\theta)/f(x)$ the posterior distribution of θ given x. Then under general conditions,

$$r(\delta) = \int_X \left[\int_\Theta L(\delta(x), \theta)\lambda(\theta|x) d\theta \right] f(x) dx,$$

and

$$\delta^* = \arg\min_{\delta} \int_{\Theta} L(\delta(x), \theta)\lambda(\theta|x)d\theta. \tag{2.2}$$

As discussed in the Introduction, δ^* is the Bayes estimator, and the conventional wisdom is to find it via first deriving the posterior distribution $\lambda(\theta|x)$ and then solving the minimization problem above. In the Introduction section, we argued that for a large and complex model, finding the posterior $\lambda(\theta|x)$ can be a much more challenging problem than solving the original minimization problem in defining δ, and it can be more advantageous to directly solving the original problem (2.1).

We propose to use deep convolutional neural networks and supervised learning to solve the original problem as follows.

1. Generate a data set $\{(\theta^{(i)}, X^{(i)})\}_{1\leq i\leq N}$ by repeated sampling $\theta^{(i)}$ from the prior distribution π and sampling $X^{(i)}$ from $f(x|\theta^{(i)})$, and split it into 80% training data and 20% validation data.

2. Construct a deep convolutional neural network.

3. Train the neural network with $\{X^{(i)}\}_{1\leq i\leq N}$ as input and $\{\theta^{(i)}\}_{1\leq i\leq N}$ as output, with loss function $L(\delta_\beta(X), \theta)$, where δ_β is a CNN with parameter β.

If we employ the mean squared error loss function $L(\delta_\beta(X), \theta) = \frac{1}{N}\sum_{i=1}^{N} \|\delta_\beta(X^{(i)}) - \theta^{(i)}\|^2$, then the resulting estimator is posterior mean while L_1 loss results in the posterior median. If we choose the following loss:

$$L(\delta, \theta) = \begin{cases} K_1(\theta - \delta) & \theta \geq \delta, \\ K_2(\delta - \theta) & \theta < \delta, \end{cases}$$

then the resulting estimator is the posterior $\dfrac{K_1}{K_1 + K_2}$-quantile.

The choice of deep convolutional neural networks over the fully connected neural networks is necessary especially for the complex models with large number of parameters and systematic structures. Convolutional layers are intended to learn features that can be efficiently and effectively used for identifying underlying structures. The

size of the training data, sample size, CNN structures, and hyperparameters in the training process etc. are all depending on the complexity of the model. We will illustrate the detailed choices of all these parameters and comparisons with different models in Sections 2.2.2 and 2.2.3.

2.2.2 Simulation Studies

We conduct simulation studies to demonstrate the properties and performance of the proposed neural Bayes estimator here, and further compare its performance with several conventional statistical methods. We start from simple Normal distributions with Normal prior and then to generalized linear mixed models (GLMM) with different response distributions and different experiment designs in Section 2.2.3.

The structure of deep neural networks we used in the simulation studies is a simple CNN which consists of three convolutional layers with 64, 128 and 128 filters, respectively, and followed by two fully-connected layers with 512 and 256 neurons, respectively. Different from image data, our training samples which serve as the input of CNN could be any numbers and have high variability, batch normalization is employed to stabilize the training. Dropout is also used to reduce overfitting.

Gaussian distribution with conjugate prior

In order to test the performance of the proposed Bayes estimator, we performed the simulation studies based on the conjugate Bayesian analysis of the Gaussian distribution.

Assume $x_i|\mu \overset{\text{i.i.d.}}{\sim} \mathsf{N}(\mu, \sigma^2)$ for $i = 1, \cdots, n$, μ is unknown and the prior distribution of μ is $\mathsf{N}(\mu_0, \sigma_0^2)$, where σ^2, μ_0, and σ_0^2 are all known. Then the posterior distribution of μ is given by

$$\mu|X = (x_1, \cdots, x_n) \sim \mathsf{N}\left(\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2}\bar{x} + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2}\mu_0, \frac{\sigma_0^2\sigma^2}{n\sigma_0^2 + \sigma^2}\right). \tag{2.3}$$

The Bayes estimator under L_2 loss is the posterior mean,

$$\hat{\mu} = \frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2}\bar{x} + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2}\mu_0.$$

Since the data-generating mechanism is very clear in this case, and we limited the space of μ_0 to be within $[-5, 5]$ for demonstration purpose, it's easy to generate the training data pairs $(\mu^{(i)}, X^{(i)})_{1 \le i \le N}$, where N is the total number of training samples. We trained a deep neural network with the structure described above and saved the trained estimator.

We tested the performance of the trained Bayes estimator over 100 different μ_0's uniformly sampled from $[-5, 5]$. For each fixed μ_0, we first generate 1000 pairs $(\mu^{(i)}, X^{(i)} = (x_1, \cdots, x_n))_{1 \le i \le 1000}$ by repeatedly drawing $\mu^{(i)}$ from the prior distribution $\mathsf{N}(\mu_0, \sigma_0^2)$ and drawing $X^{(i)}$ from $\mathsf{N}(\mu, \sigma^2)$. We calculated the mean squared error loss between the obtained estimate and the theoretical posterior mean and the variance of predictions for all μ's. We averaged the results over the 100 cases and summarized the results in Table 2.1.

Table 2.1.: Simulation results of conjugate Bayesian estimation of Normal distribution

n	MSE	Var	Theoretical Var
49	0.0111	0.0249	0.0200
100	0.0099	0.0128	0.0099
400	0.0039	0.0051	0.0025

We can see that the Bayes estimator we proposed approximates the posterior mean well and the prediction accuracy increases as the sample size increases. The variances of the neural Bayes estimators are also close to their corresponding theoretical variances.

2.2.3 Application in GLMM

Generalized linear mixed models (GLMMs) are expansion of linear models (LMs) with variance components and accommodation of non-normal response. According to the definition in McCulloch and Searle (2001) [61], the GLMMs have two special features compared to the LMs. One feature is the normality assumption is no longer needed, and the other is the mean does not need to be a linear combination of the involved covariates. To be more specific, consider a sample of n independent random multivariate response $\{\mathbf{y}_i = (y_{i1}, \cdots, y_{im})\}_{1 \le i \le n}$, where y_{ij} is the j-th response of the i-th observation. We shall assume that y_{ij} depends on a $p \times 1$ vector of fixed covariates x_{ij} associated with a vector of fixed effects $\beta = (\beta_1, \cdots, \beta_p)'$ and on a $q \times 1$ vector of random covariates z_{ij} associated with the multivariate $q \times 1$ random effect u_i. GLMM satisfies the following conditions.

- Given u_i, the variables $y_{i1}, \cdots, y_{in}$ are mutually independent with a density function given by

$$f(y_{ij}|u_i, \beta) = \exp\left\{\frac{y_{ij}\theta_{ij} - b(\theta_{ij})}{d_{ij}(\phi)} + c(y_{ij}, \phi)\right\}, \tag{2.4}$$

  where θ_{ij} is the canonical parameter and ϕ is the scale parameter. The function d_{ij} and c are specific to each distribution.

- The conditional mean and variance of y_{ij} are given by

$$E(y_{ij}|u_i) = \mu_{ij} = h^{-1}(x'_{ij}\beta + z'_{ij}u_i), \tag{2.5}$$

$$Var(y_{ij}|u_i) = v(\mu_{ij})d_{ij}(\phi), \tag{2.6}$$

  where h and v are the link and variance function, respectively.

- The random effects $u_1, \cdots, u_n$, are mutually independent with a common underlying distribution G which depends on the unknown parameters σ.

We are interested in estimation of both fixed effect and random effect $\theta = (\beta, \sigma)$. As mentioned earlier, the difficulty of estimation arises due to the increase of the

number of variance components, i.e. the dimension of σ. We usually do not have strong prior knowledge of the parameters in GLMM, we will just employ the non-informative prior. Uniformly random sampling or Latin Hypercube sampling will be used to first generate the parameter samples. Then based on the statistical model (2.4), we can generate the labeled samples $\{(\mathbf{y}^k = (\mathbf{y}_1^k, \cdots, \mathbf{y}_n^k), \theta^k)\}_{k=1,\cdots,N}$ which will be further fed into the deep neural networks.

One often assumes G to be of a specific parametric form, generally a multivariate normal with mean vector 0 and covariance matrix $\Sigma = \Sigma(\sigma) = (\sigma_{jl})_{j,l=1,\cdots,q}$, and uses a parametric or semiparametric approach. In this case, the marginal likelihood of $\mathbf{y} = (\mathbf{y_1}, \cdots, \mathbf{y_N})$ is given by

$$L(\beta, \sigma|y) = (2\pi)^{-Nq/2}|\Sigma|^{-N/2} \prod_{i=1}^{N} \int_{\mathbb{R}^q} \prod_{j=1}^{n} f(y_{ij}|u_i, \beta) \exp(-\frac{1}{2} u_i' \Sigma^{-1} u_i) du_i. \qquad (2.7)$$

The maximization of the above function is computationally difficult and requires evaluation of integrals where the integral's dimension is equal to the number of random effects.

The popularly used statistical methods for estimating the parameters in GLMM can be divided into two categories. The first category is based on analytical Laplace approximation, such as penalized quasi-likelihood (PQL) [72, 73]. The second category is based on numeric techniques, and includes Bayesian approach with sampling [74], MCEM algorithm [75], Gauss-Hermite quadrature (GHQ) [76], and Randomized QMC method [77, 78].

Comparing different approaches for parameter estimation in GLMMs and selecting the best approach based on real data is very difficult because we do not know what the true values are. It is also hard to solve this analytically. That is why we are forced to implement a simulation study. We conduct simulation studies to demonstrate the properties and performance of the proposed Neural Bayes Estimator in this section. In addition, we will compare the relative performance and accuracy between our proposed approach and three R packages (lme4 [79], MCMCglmm [80], and glmmADMB [81]).

lme4 is a widely used package that uses the glmer function to fit a GLMM model. lme4 implements AGHQ to approximate the log-likelihood function using numerical integration only for single and scalar random effects. MCMCglmm uses MCMC instead of ML to fit the model. It is using Gibbs sampler with inverse-Wishart prior for the variance components and multivariate normal distribution for the fixed effects. glmmADMB is built on the open-source AD Model Builder platform. It's flexible, but much slower than other R packages.

Our proposed estimator is based on learning of the simulated data, and its performance depends on the number of responses in one experiment. Based on the setup of the GLMM, we mainly focus on the balanced design in this study. We will vary the number of replicates ($K = 5, 10, 20$) in the experiment to test the performance of the proposed estimator. We will also test on several GLMMs with different response distributions. We first set the parameter space of all the parameters in the GLMM, including fixed effect and random effect parameters to be an interval, like $[0, 2]$ or $[0, 4]$. We generate the labeled data based on GLMM of interest, 80% of which is used for training, and the other 20% is used for validation. Note that we use the definitions given in Ripley (2007) [82]: a training set is used for learning, a validation set is used to tune the network parameters, and a test set is used only to assess the performance of the network.

Gaussian mixed models

We will first use a Gaussian mixed model, which is a special case of GLMM. The model is formulated as follows.

$$
\begin{aligned}
Y_{ijk} &= \mu + \alpha_i + \beta_j + \epsilon_{ijk} \\
\beta_j &\sim Normal(0, \sigma_\beta^2), \\
(\epsilon)_{ijk} &\sim Nomral(0, \sigma), \\
i = 0, \cdots, 4; j = 0, \cdots, 3; k = 0, \cdots, 4 &
\end{aligned} \tag{2.8}
$$

where α_i's are the fixed effects, β_j's are the random main effects, and ϵ_{ijk}' are the error terms. To avoid the non-identifiability problem, we add a constraint on the base level of the fixed effects (i.e. $\alpha_0 = 0$). The response Y is assumed to follow the Gaussian distribution. The parameters of interest are $\theta = (\mu, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \sigma_\beta, \sigma) \in \mathbb{R}^7$.

In order to perform fair comparisons with R packages, especially with MCMCglmm, we employ the same prior distributions for the parameters as in MCMCglmm in this example, which are

$$
\begin{aligned}
&\mu, \alpha_i \sim Normal(0, 2), \\
&\sigma_\beta, \sigma \sim invgamma(scale = 1, shape = 1),
\end{aligned}
$$

The inverse-Gamma is a special case of the inverse-Wishart used in MCMCglmm for the variance components. We truncate the inverse-Gamma distribution on the right at 10 to avoid extreme large value responses.

We generate the training data by repeatedly drawing parameters samples from the priors distributions, and generating responses according to model 2.8. In total, we generate 10,000 samples for the training data. We employed a five-layer deep neural network that consists of three convolutional layers and two fully connected layers. For the testing, we randomly generate 1000 sets of parameters from the prior distributions. For each parameter set, we generate one sample with 100 observations. So the size of testing set is 1000. We compared the trained neural Bayes estimator with existing packages lmer and MCMCglmm in R on the testing dataset, and report the mean squared error in Table 2.2.

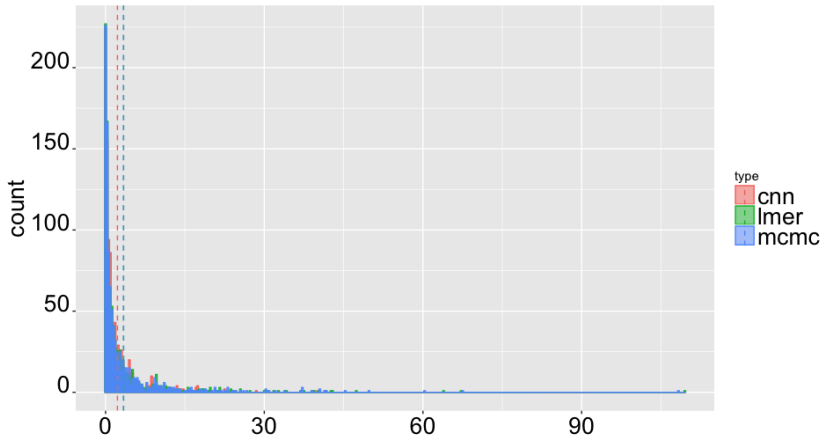
From Table 2.2, we can see that our proposed neural Bayes estimator outperforms the two R packages by a significant margin for the fixed effects and random effect

Table 2.2.: MSE for Gaussian mixed model with two variance components: comparison with R results, mean squared error over 1000 test cases. Sample size = 100

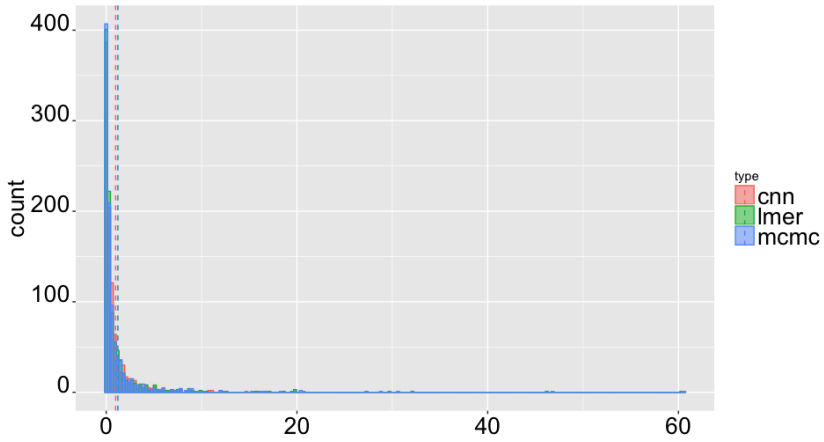
	Fixed effects					Random effects	
	(Intercept)	α_1	α_2	α_3	α_4	σ_β	σ
CNN 3+2	1.6815	0.8003	0.7630	0.8241	0.7135	1.3245	0.1542
R-lmer	3.4101	1.2137	1.1307	1.2485	1.0947	1.9893	0.0520
R-MCMCglmm	3.4051	1.2153	1.1354	1.2501	1.0960	2.8036	0.0544

σ_β. For the standard deviation of the error term, the two R packages did a much better job than us. One possible reason for the better performance of lmer and MCMCglmm over our proposed estimator is that the former two use the maximum likelihood principle, which leads to more degrees of freedom for σ. While in our method, σ and other parameters are treated equally.

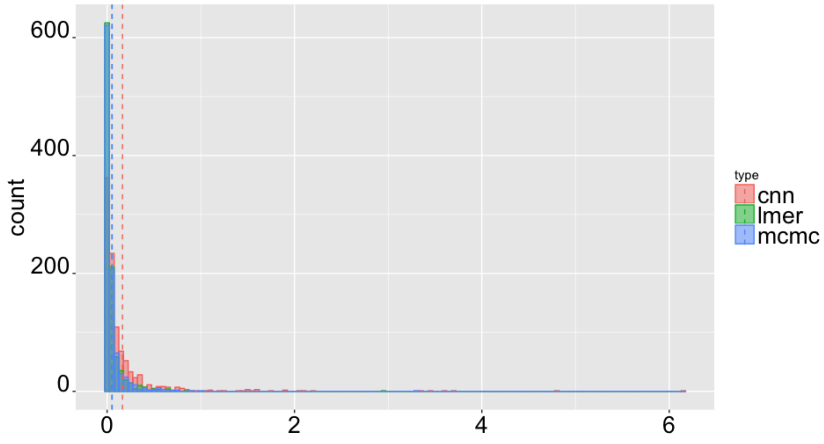
In Fig. 2.1, we plot the histograms of squared errors between predicted parameter values and the truth of all 1,000 test cases. Three colors represent three different methods, which are CNN based neural Bayes estimator, lmer, and MCMCglmm. Three panels are for the intercept, α_1 and σ, respectively. The dashed lines represent the mean squared errors. We can see that in panel (a), our proposed neural Bayes estimator performs better in terms of mean squared error over these test cases. The histograms of MCMCglmm has a long tail on the right. MCMCglmm did not perform well in 53 cases, in which the squared errors of these cases are over 15. This resulted in a higher mean squared error. The performances did not improve even if we increase the number of iterations in MCMCglmm. Similar patterns can be found in panel (b). However, our proposed neural Bayes estimator performs consistently well over all test cases. Overall, our proposed estimator will not encounter computation difficulties in predictions once the training is done, and it performs better on average.



(a) Intercept



(b) α_1



(c) σ

Fig. 2.1.: Histograms of squared errors of 1,000 test cases. Different colors represent three different methods. Dashed lines represent the mean squared errors over all test cases.

Poisson regression

For non-normal response mixed model examples, we will start with Poisson mixed model with two variance components. The model is formulated as follows.

$$\begin{aligned} Y_{ijk} &\sim Poisson(\mu_{ij}), \\ \log \mu_{ij} &= \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij}, \\ \alpha_0 &= 0, \\ \beta_j &\sim Normal(0, \sigma_\beta^2), \\ (\alpha\beta)_{ij} &\sim Nomral(0, \sigma_{\alpha\beta}^2), \\ i = 0, \cdots, 4; j = 0, \cdots, 3; k &= 0, \cdots, 19. \end{aligned} \tag{2.9}$$

Here α_i's are the fixed effects, β_j's are the random main effects, and $(\alpha\beta)_{ij}$' are the random interaction effects. To avoid the non-identifiability problem, we add a constrains on the base level of the fixed effect (i.e. $\alpha_0 = 0$). The response Y is assumed to follow the Poisson distribution. The parameters of interest are $\theta = (\mu, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \sigma_\beta, \sigma_{\alpha\beta}) \in \mathbb{R}^7$.

We put the non-informative prior $Uniform(0, 2)$ on the parameters since in the real examples, the responses are usually within thousands. We first generate the training data from 2.9. Then we train a neural Bayes estimator for this Poisson regression model.

For the testing, we randomly generate 1000 sets of parameters from the parameter space. For each parameter set, we generate one sample with 400 observations. So there are in total 1000 samples in the testing set. We will vary the size of the training data from 1,000 to 100,000, which are denoted in the parenthesis in Table 2.3. We compared the Neural Bayes estimator with existing packages in R on the testing dataset, and report the mean squared error in Table 2.3 . Note that several packages run into problems with some test cases. For example, R-MCMCglmm gave convergence errors for one case out of 1000 test cases. R-glmmADMB have problems with 147 cases. From Table 2.3, we can see that the performance of Neural Bayes

estimators improve when we user larger size of training data. It is clear that the neural Bayes estimator outperforms the three listed R packages.

Table 2.3.: MSE for Poisson mixed model with two variance components: comparison with R results, mean squared error over 1000 test cases. Sample size for the CNN training is denoted in the parenthesis.

	Fixed effects					Random effects	
	(Intercept)	α_1	α_2	α_3	α_4	σ_β	$\sigma_{\alpha\beta}$
CNN (10,000)	0.1114	0.1482	0.1726	0.1478	0.1559	0.0751	0.0565
CNN (100,000)	0.0841	0.1010	0.1015	0.1017	0.112	0.0552	0.0297
CNN (1,000,000)	0.0840	0.0948	0.0936	0.0952	0.1042	0.0518	0.0277
R-glmer	0.1702	0.1811	0.1888	0.1838	0.1907	0.0753	0.0211
R-MCMCglmm	0.1644	0.1664	0.1660	0.1769	0.1806	0.2476	0.0236
R-glmmADMB	0.1733	0.1700	0.1796	0.1912	0.1983	0.1502	0.1584

As mentioned earlier, the difficulty of estimation in GLMM increases as the number of variance components in the model increases. Additionally, we increase the number of variance components in the Poisson mixed regression model. Aside from the treatment, block, an interaction between treatment and block, we also consider a random subject effect. The model is specified as follows.

$$
\begin{aligned}
Y_{ijk} &\sim Poisson(\mu_{ijk}), \\
\log \mu_{ijk} &= \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}, \\
\alpha_0 &= 0, \\
\beta_j &\sim Normal(0, \sigma_\beta^2), \\
(\alpha\beta)_{ij} &\sim Nomral(0, \sigma_{\alpha\beta}^2), \\
\epsilon_{ijk} &\sim Nomral(0, \sigma^2), \\
i = 0, \cdots, 4; j = 0, \cdots, 3; k &= 0, \cdots, 19.
\end{aligned} \tag{2.10}
$$

The parameters of interest involved in this model are: $\theta = (\mu, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \sigma_\beta, \sigma_{\alpha\beta}, \sigma)$. We trained the CNN using different sizes of training data, which are denoted in the parenthesis in Table 2.4, and tested the performance of trained CNNs on a testing dataset with 1000 samples. We compared CNN based Bayes estimators with existing packages in R, and report the mean squared error in Table 2.4. From the table, it is clear that the neural Bayes estimator outperforms the listed statistical packages over 1000 test cases. MCMCglmm and glmer outperform the Neural Bayes estimator in the estimation of the subject effect. When we increase the number of variance components in GLMM, the accuracy decreases for the three R packages, while our proposed neural Bayes estimator performs consistently.

Logistic regression

We consider the similar setup as Poisson mixed model with three variance components in the previous section. Here, we consider the mixed effects logistic regression model. All the other settings are the same as above. The model of interest is given as follows.

Table 2.4.: MSE for Poisson mixed model with three variance components : comparison with R results, mean squared error over 1000 test cases. Sample size for the CNN training is denoted in the parenthesis.

	Fixed effects					Random effects		
	Intercept	α_1	α_2	α_3	α_4	σ_β	$\sigma_{\alpha\beta}$	σ
CNN (10,000)	0.2512	0.3422	0.2774	0.2515	0.3654	0.0959	0.0870	0.0881
CNN (100,000)	0.1048	0.1118	0.1072	0.0949	0.1177	0.0660	0.0465	0.0145
CNN (1,000,000)	0.1058	0.1247	0.1183	0.0981	0.1229	0.0598	0.0439	0.0265
R-glmer	0.1813	0.1816	0.1722	0.1568	0.1583	0.0823	0.0225	0.0013
R-MCMCglmm	0.1758	0.1807	0.1684	0.1513	0.1547	0.3290	0.0295	0.0013
R-glmmADMB	0.1935	0.2079	0.1856	0.1737	0.1735	0.4499	0.4561	0.0353

$$
\begin{aligned}
Y_{ijk} &\sim Bernoulli(p_{ij}),\\
\mu_{ij} &= \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk},\\
p_{ij} &= \frac{\exp(\mu_{ij})}{1+\exp(\mu_{ij})},\\
\alpha_0 &= 0,\\
\beta_j &\sim Normal(0, \sigma_\beta^2),\\
(\alpha\beta)_{ij} &\sim Nomral(0, \sigma_{\alpha\beta}^2),\\
\epsilon_{ijk} &\sim Nomral(0, \sigma^2)\\
i = 0, \cdots, 4; j = 0, \cdots, 7; k &= 1, \cdots, 10.
\end{aligned}
$$

Table 2.5 presents the performance of the neural Bayes estimator on the testing dataset together with the performance of other existing methods. This time, R-glmer gives convergence errors for 384 cases out of 1000 test cases. R-glmmADMB has

problems with 303 cases. From the table, it is clear that the neural Bayes estimator outperforms the four listed statistical packages by a significant margin. The larger MSEs in the table are resulted from some problematic cases, e.g. singular matrix, non-convergence issue.

Table 2.5.: MSE for Mixed effect Logistic regression with three variance components: comparison with R results, mean squared error over 1000 test cases. Sample size for the training is denoted in the parenthesis.

	Fixed effects				Random effects			
	(Intercept)	α_1	α_2	α_3	α_4	σ_β	$\sigma_{\alpha\beta}$	σ
CNN (10,000)	0.1185	0.1643	0.1659	0.1725	0.1944	0.0696	0.0806	0.0969
CNN (100,000)	0.1105	0.1512	0.1558	0.1626	0.1766	0.0657	0.0738	0.0951
R-glmer	2.0191	1.0975	0.3722	0.3787	1.2261	0.1052	0.1308	37.9541
R-glmmADMB	0.4093	1.0856	0.3942	0.6118	11.7821	0.2690	0.1905	83.3700

2.2.4 Selection of hyper parameters in training

Random sampling V.S. Latin Hypercube sampling

In order to reduce the size of the training data and at the same time not sacrifice the accuracy, we propose to use Latin Hypercube sampling method in the data generating step. We compared the performances of different sampling methods under Model 2.10 and summarized the L_2 loss on the test data in Table 2.6. We can see that Latin Hypercube sampling achieves the lowest loss and significantly improves the estimation accuracy.

Table 2.6.: Comparison of different sampling methods. The MSE of CNN predictions over 1000 test cases for a Poisson regression with three variance components, and 100 sample size (5 replicates in the experiment).

Sampling method	Fixed effects					Random effects		
	Intercept	α_1	α_2	α_3	α_4	σ_β	$\sigma_{\alpha\beta}$	σ
Equal grid	0.6067	0.5588	0.6481	0.6373	0.5902	0.1440	0.1341	0.1705
Random	0.2275	0.4255	0.4430	0.4980	0.4194	0.1386	0.1502	0.1388
LHS	0.2332	0.3618	0.3456	0.3298	0.3443	0.1355	0.1260	0.1010

Number of training iterations

One drawback of our proposed method is that it needs to be trained every time we have a new model. We want to propose a unified approach without sacrificing the computing time. The proposed Bayes estimator is based on deep neural networks, and we believe that the training time is what one will be most interested in. We employed a moderately sized CNN in the simulation studies across this chapter. For the Poisson model 2.10, we vary the training data size and the number of training steps, and we summarize the performance of the trained neural Bayes estimators under different scenarios in Fig. 2.2. We can see that for this moderately sized model, training size of 10000 is enough and training with 20,000 steps is also enough to achieve acceptable results. Large training data requires a long training time to achieve stable results.

2.3 Neural Model Selector and Parameter Estimator

As discussed in the Introduction, we focused on two different scenarios in which statistical analysis is needed. One scenario is that we know the data generating

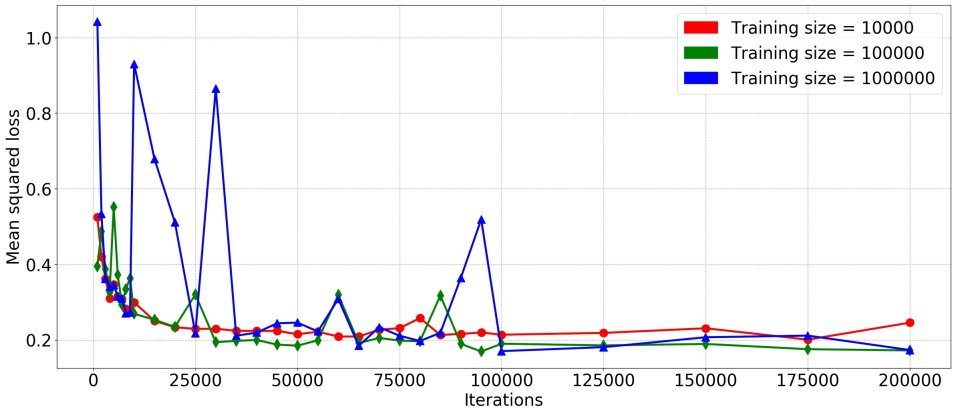


Fig. 2.2.: Mean squared error loss on the testing data over different number of iterations under the Poisson model 2.10

procedure (model) and are only interested in the involved parameters. We proposed the neural Bayes estimator. The other scenario we are going to discuss here is that we just have the samples collected from an experiment, but we do not know the model and only have a set of candidate models. In this scenario, we will need to not only identify the best model that fits the data, but also estimate the parameters involved in the model. In this section, we will limit the candidate model space $\mathcal{M}$ to be the models with scalar parameter. As mentioned in Section 2.1, model selection and parameter estimation are two mappings from the sample space to the model class and parameter space, which we denote as G_1 and G_2, respectively. We propose to use CNNs to approximate the model selection mapping G_1 and the parameter estimation mapping G_2, and we refer the trained CNNs as the neural model selector and the neural parameter estimator.

2.3.1 Labeled data and loss functions

Recall that $\mathcal{M}$ is the collection of prespecified candidate models, and the density function of model M_k is $f(y|\theta_k, M_k)$, where $\theta_k \in \Theta_k \subset \mathbb{R}$ is the parameter. We have

a random sample from one of the models, which we denote as $\{y_j\}_{1\leq j\leq N}$, where N is the sample size. The situation here is that we do not know the exact model from which $\{y_j\}$ is drawn from, and we want to find the true model M_k and estimate θ_k.

As an analogy, the sample $\{y_j\}$ can be considered the image of an object, G_1 the classifier for object recognition, and G_2 the regression procedure for object localization in image processing [83]. In order to train G_1 and G_2, just like in image processing, labeled data must be available. We propose to generate the labeled data as follows.

Let N be a prespecified sample size. For each model M_k, we first place an equally space grid over the parameter space Θ_k, which is $\{\theta_{k,1}, \theta_{k,2}, \cdots, \theta_{k,n_k}\}$. For each value of the grid $\theta_{k,l}$, we generate D samples of size N from $f(y|\theta_{k,l}, M_k)$. We denote the samples as $\{y_{(k,l,d)j}\}_{1\leq j\leq N}$ for $1 \leq k \leq K$, $1 \leq l \leq n_k$, and $1 \leq d \leq D$. In total, we have the following collection of labeled data $Y = \left\{\left(\{y_{(k,l,d)j}\}_{1\leq j\leq N}, M_k, \theta_{k,l}\right)_{k,l,d}\right\}$, and Y will be used to train and validate both G_1 and G_2.

In order to train G_1 and G_2, we need to choose proper loss functions. As mentioned previously, the neural model selector is essentially a classifier and similar to the classifier for object recognition. Therefore, we choose the commonly used softmax loss function for training G_1. For details of the softmax loss function, the reader is referred to [84]. The neural parameter estimator G_2 is essentially a regression function and similar to the regression CNN for localizing an object in an image. For object localization, the L_2 loss is typically used, which is $L(G_2, \theta) = \|G_2 - \theta\|_2^2$. The L_2 loss function works well for image processing. For the neural parameter estimator, the L_2 loss is sensitive to extreme observations generated from models with long tails in $\mathcal{M}$, which makes the training process unstable. Resolve this issue, the Huber loss [66] is employed in training of neural estimator to improve the robustness against outliers. The Huber loss is defined as follows:

$$L_\delta(\theta, \hat{\theta}) = \begin{cases} \frac{1}{2}(\theta - \hat{\theta})^2 & \text{for } |\theta - \hat{\theta}| \leq \delta, \\ \delta(|\theta - \hat{\theta}| - \frac{1}{2}\delta) & \text{otherwise.} \end{cases}$$

2.3.2 Two types of architectures

The last important issue is about the architectures of the neural model selector and parameter estimator. There are two different types of architectures involved. The first type is regarding the architectures of the CNNs, which are about the numbers and sizes of the covolutional and fully connected layers. We refer to this type of architecture as the CNN architecture. The second type of architectures is regarding the interplay between the model selector G_1 and the parameter estimator G_2. Because this type of architecture directly affects how the overall analysis performed, we refer to it as the SA architecture.

There are three possible SA architectures. The first SA architecture uses two separate CNNs for the model selector and the parameter estimator, respectively, which we refer to as the Non-Shared Architecture (NSA). The second SA architecture uses one single CNN for both G_1 and G_2, and they part their ways only at the output layer. We refer to this architecture as the Fully Shared Architecture (FSA). The third architecture uses two partially joint CNNs for G_1 and G_2, respectively. The two CNNs can share from one to all common convolutional and fully connected layers. We refer to this architecture as the Partially Shared Architecture (PSA). The PSA sharing l layers is denoted as PSA-l. The three SA architectures NSA, FSA, and PSA are illustrated in Fig. 2.3.

NSA, FSA, and PSA have their own advantages and disadvantages. Using again the analogy of object recognition and localization, the convolutional layers are used to learn features that can be efficiently and effectively used for identifying the true model and its parameter. When NSA is used, the two separate CNNs are learning the features for model selection and parameter estimation, separately, and this simple SA architecture allows easy implementation of the training algorithms. The disadvantage of NSA is that it uses only the marginal distribution of model label and true parameter value separately, instead of the entire information in their joint distribution.

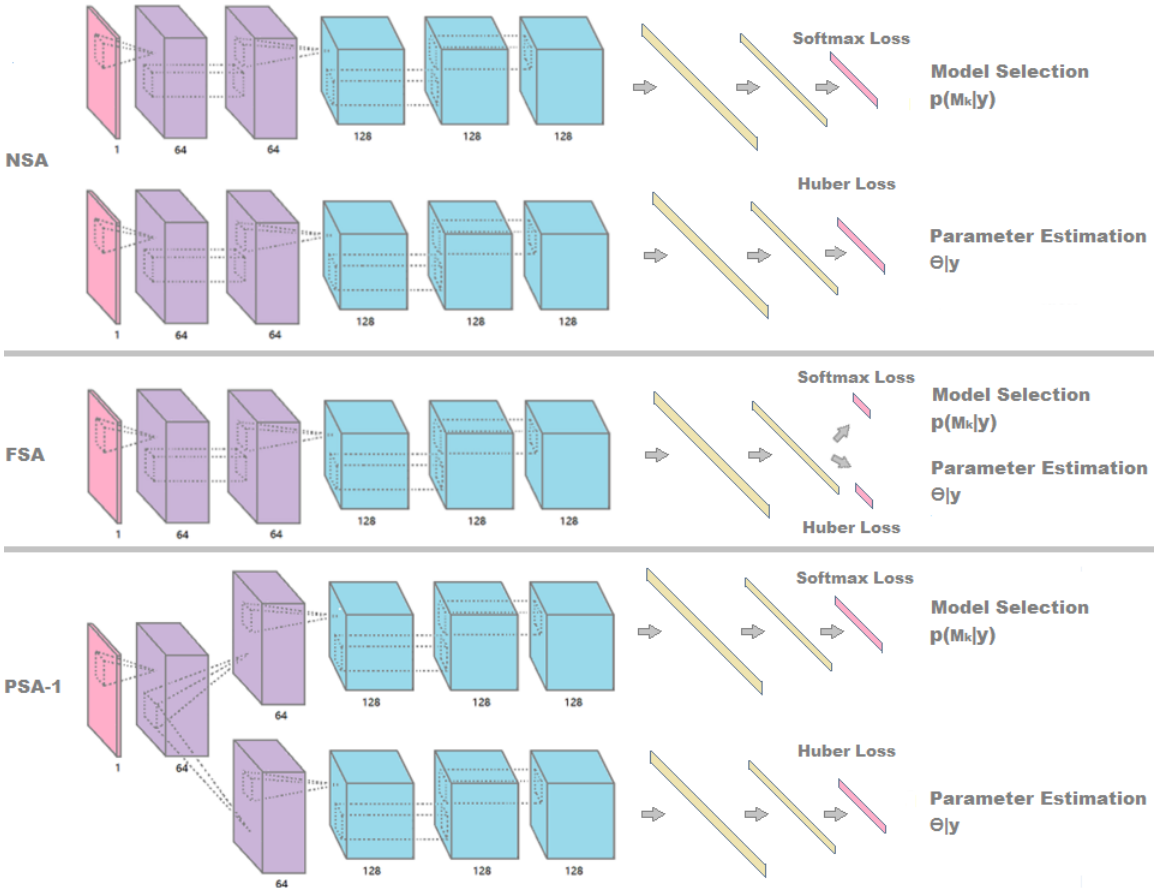


Fig. 2.3.: SA Architecture, from top to bottom: NSA, FSA, PSA-1

When FSA is used, the single CNN is trying to learn the same set of features, and hope that they can be used to not only select the model correctly but also estimate the parameter accurately. This is based on the assumption that such a set of common features exists. This assumption holds for distributions that belong to the Exponential family, and minimal sufficient statistics can serve as the set of common features. This assumption however may not hold in general. Therefore, FSA is expected to work well under one set of candidate models, but may fail under another set of candidate models.

The most promising architecture is PSA. The intuition underlying PSA is that the early convolutional layers will produce low-level features that are common for both model selection and parameter estimation, and information in the true model

label and parameter values can be shared. Because model selection and parameter estimation are two different tasks, we should not expect they would be relaying on the same set of high-level features. Our simulation studies reported in later sections support this intuition. In terms of training, PSA is more demanding than the other two architectures. Furthermore, PSA leads to another important issue, that is, how many convolutional layers should be shared by G_1 and G_2. We will investigate this issue in the next section.

2.3.3 Relationship between neural model selector and Bayes factor

As discussed earlier, there are many traditional approaches for model selection, the Bayes factor method is one of them. It imposes a prior distribution to the models, $\pi(M_k)$, and a prior distribution to the parameter $\pi(\theta_k)$. The Bayes factor between any two models, M_{k_1} and M_{k_2}, is calculated as the ratio between the likelihood values of the models with their parameters integrated out with respect to the corresponding priors,

$$BF(M_{k_1}, M_{k_2}) = \frac{\pi(\{y_j\}|M_{k_1})}{\pi(\{y_j\}|M_{k_2})} = \frac{\int f(y|\theta_{k_1}, M_{k_1})\pi(\theta_{k_1})d\theta_{k_1}}{\int f(y|\theta_{k_2}, M_{k_2})\pi(\theta_{k_2})d\theta_{k_2}}. \tag{2.11}$$

The model the BF scores support the most will be selected as the best model.

In NSA, we proposed to use two separate CNNs to approximate the model selection and the parameter estimation, respectively. We train the first CNN for model selection with cross-entropy loss, which is defined as

$$\begin{aligned} L &= \int_Y \sum_{i=1}^{K} L(O_i(y), l_i) f(y|M_i)\pi(M_i)dy \\ &= -E\left\{\sum_{i=1}^{K} E[l_i]\log O_i(Y) + (1 - E[l_i])\log(1 - O_i(Y))\right\}, \end{aligned} \tag{2.12}$$

where $\{O_i(Y), i = 1, 2, \cdots, K\}$ is the output layer of the neural network, which is a function of the input data Y, and $\{l_i, i = 1, 2, \cdots, K\}$ is the desired output, and

$$f(y|M_i) = \int_\Theta p(y|\theta, M_i)\pi(\theta|M_i)d\theta. \tag{2.13}$$

Note that in the model selection, $\{l_i, i = 1, 2, \cdots, K\}$ is the one-hot encoding, and it follows

$$E[l_i] = P(M_i|Y).$$

Take the first derivative in (2.12) with respect to $O_i(Y)$ and set it to be zero, we have

$$O_i(Y) = E[l_i] = P(M_i|Y).$$

The cross-entropy cost function (2.12) is minimized when $O_i(Y) = P(M_i|Y)$ for $i = 1, 2, \cdots, K$.

In the training process of model selector, we use the data $\{y_j\}$ as the input and model label as the output. Therefore, if we train the CNN with large enough training data, the output of trained model selector approximates the posterior distributions $P(M_i|y)$, and the trained model selector essentially is a Bayes classifier. We generate the same amount of training data for each model, i.e. we impose the uniform prior on the candidate models $\pi(M_i) = 1/K$. Based on the relationship between likelihood values of models and posterior distributions,

$$P(y|M_k) = \frac{P(M_k|y)P(y)}{P(M_k)},$$

we have

$$BF(M_{k_1}, M_{k_2}) = \frac{P(\{y_j\}|M_{k_1})}{P(\{y_j\}|M_{k_2})} = \frac{P(M_{k_1}|\{y_j\})P(M_{k_2})}{P(M_{k_2}|\{y_j\})P(M_{k_1})}. \tag{2.14}$$

Thus, the Bayes factor is equal to the ratio of posterior probabilities of model M_{k_1} and model M_{k_2} since the prior probabilities on all candidate models are the same. In prediction, we choose the model that has the highest posterior probability to be the best model. This is exactly equivalent to using the Bayes factor method according to (2.14), but without the explicit calculation of posterior distribution $\pi(\{y_j\}|M_k)$.

2.3.4 Simulation results

We conduct simulation studies to demonstrate the properties and performance of the proposed model selector and parameter estimator in this section, and further compare them with several conventional statistical methods.

Setup and datasets

The difficulty of model selection and parameter estimation depends on the number of candidate models (K) in $\mathcal{M}$. We consider three levels of K, which are $5, 20$ and 50. We take 50 probability distributions from the textbook [65] and some R packages [85–89]. The 50 distributions are listed in Table 2.7 and 2.8. When $K = 5$, the set of candidate models $\mathcal{M}$ includes the top five distributions in the list; when $K = 20$, $\mathcal{M}$ includes the top 20 distributions in the list; and when $K = 50$, $\mathcal{M}$ includes all the 50 distributions.

The performance of the proposed model selector and parameter estimator depends on the sample size. We consider three levels of the sample size N, which are $100, 400$, and 900. Each sample will be resized to a square matrix to feed into CNNs. According to the data-generating scheme described in Section 2.3.1, the total amount of training samples further depends on the number of parameter values on the grid (n_k) and the number of replicated samples (D). We specify $D = 1000$, and grid size n_k is set to be between 10 and 12. For each distribution, if the parameter space is bounded, like probability p in Bernoulli distribution, we will place the grid on the original space. If the parameter space is not bounded, like μ in Normal distribution, we will set the parameter space to be a bounded interval, $[0, 12]$. Following the scheme of Section 2.3.1, we generate all the labeled data, 80% of which is used for training, and the other 20% is used for validation. Note that we use the definitions given in [82]: a training set is used for learning, a validation set is used to tune the network parameters, and a test set is used only to assess the performance of the network.

We further generate the test data as follows. For each model, we first randomly sample 100 parameter values from the parameter space, those values are just in the same range as parameters in the training set, but not the same; and second, for each sampled parameter value, we generate 10 random samples of size N. We test the trained model selector and parameter estimator using the test datasets. Counting both the labeled data and test data, in total, we have generated roughly 400 thousand

Table 2.7.: List of 50 models used in the simulation study: part I

	Distribution	Parameter of interest	Other parameters
1	Bernoulli(p)	p	
2	Discrete Uniform(N)	N	
3	Geometric(p)	p	
4	Negtive Binomial(r, p)	r	$p = 0.3$
5	Exponential(λ)	λ	
6	Normal(μ, σ)	μ	$\sigma = 1$
7	Poisson(λ)	λ	
8	Beta(α, β)	α	$\beta = 3$
9	Weibull(λ, k)	k	$\lambda = 3$
10	Double Exponential(μ, λ)	μ	$\lambda = 3$
11	Chi Square(k)	k	
12	F(d_1, d_2)	d_2	$d_1 = 3$
13	Gamma(α, β)	β	$\alpha = 0.5$
14	Logistic(μ, s)	μ	$s = 0.5$
15	Lognormal(μ, σ)	μ	$\sigma = 0.5$
16	Pareto(x_m, α)	x_m	$\alpha = 2$
17	Student's t(ν, ncp)	ν	$ncp = 2$
18	Uniform(a, $a + 2$)	a	
19	Hypergeometric(m, n, k)	n	$m = 3, k = 2$
20	Binomial(n, p)	n	$p = 0.5$
21	One-Inflated Logarithmic($shape$, $pstr_1$)	$shape$	$pstr_1 = 0$
22	Triangle(θ, $lower$, $upper$)	θ	$lower = 0$, $upper = 33$
23	Wilcoxon Signed Rank Statistic(n)	n	
24	Benini(y_0, $shape$)	$shape$	$y_0 = 1$
25	Beta-Geometric($shape_1$, $shape_2$)	$shape_2$	$shape_1 = 5$
26	Beta-Normal($shape_1$, $shape_2$, $mean$, sd)	$shape_1$	$shape_2 = 10$, $mean = 5$, $sd = 11$
27	Birnbaum-Saunders($scale$, $shape$)	$shape$	$scale = 1$
28	Dagum($scale$, $shape_{1.a}$, $shape_{2.p}$)	$shape_{1.a}$	$scale = 1$, $shape_{2.p} = 2$
29	Frechet($location$, $scale$, $shape$)	$shape$	$location = 0$, $scale = 1$
30	Dirichlet($shape_1$, $shape_2$, $shape_3$)	$shape_1$	$shape_2 = 2$, $shape_3 = 4$

Table 2.8.: List of 50 models used in the simulation study: part II

	Distribution	Parameter of interest	Other parameters
31	Huber's Least Favourable(k, μ, σ)	k	$\mu = 0$, $\sigma = 1$
32	Gumbel($location$, $scale$)	$scale$	$location = 1$
33	Gompertz($scale$, $shape$)	$shape$	$scale = 1$
34	Kumaraswamy($shape_1$, $shape_2$)	$shape_2$	$shape_1 = 10$
35	Laplace($location$, $scale$)	$scale$	$location = 5$
36	Log-Gamma($location$, $scale$, $shape$)	$scale$	$location = 0$, $shape = 2$
37	Lindley(θ)	θ	
38	Lomax($scale$, $shape_{3.q}$)	$shape_{3.q}$	$scale$
39	Makeham($scale$, $shape$, ϵ)	$shape$	$scale = 0$, $\epsilon = 0$
40	Maxwell($rate$)	$rate$	
41	Nakagami($scale$, $shape$, $Smallno$)	$shape$	$scale = 1$, $Smallno = 1.0e-6$
42	Perks($scale$, $shape$)	$shape$	$scale = 1$
43	Rayleigh($scale$)	$scale$	
44	Rice(σ, vee)	vee	$\sigma = 1$
45	Simplex(μ, $dispersion$)	$dispersion$	$\mu = 0.5$
46	Singh-Maddala($scale$, $shape_{1.a}$, $shape_{3.q}$)	$shape_3.q$	$scale = 1$, $shape_{1.a} = 5$
47	Skellam(μ_1, μ_2)	μ_2	$\mu_1 = 5$
48	Tobit($mean$, sd, $lower$, $upper$)	$mean$	$sd = 1$, $lower = 0$, $upper = Inf$
49	Paralogistic($scale$, $shape_{1.a}$)	$scale_{1.a}$	$scale = 1$
50	Zipf(N, $shape$)	$shape$	$N = 10$

training samples, 100 thousand validation samples, and 50 thousand test samples for the 50 models.

Architecture setup and training

In Section 2.3.2, we discussed the three possible SA architectures (NSA, FSA, and PSA), but have not discussed the CNN architectures. In our simulation studies, we employ three different sizes of CNNs, which are referred to as small, medium, and

large, respectively. The small CNN architecture consists of three convolutional layers with 64, 128 and 128 filters, respectively, which are followed by two fully-connected layers with 512 and 256 neurons, respectively. The medium CNN architecture consists of five convolutional layers with 64 filters each, which are followed by two fully connected layers each with 64 neurons. The large CNN architecture consists of five convolutional layers with 64, 64, 128, 128 and 128 filters, respectively, which are followed by two fully connected layers with 1024 and 512 neurons, respectively. In all three CNN architectures, convolutional filters are connected to a 5×5 region of their input, 2×2 max pooling and 2×2 average pooling are performed between some consecutive convolutional layers. The same CNN architecture is used for both the neural model selector and parameter estimator except for the output layers.

Under each combination of SA architectures (NSA, FSA, PSA), CNN architectures (small, medium, large), number of candidate models ($K = 5, 20, 50$), sample sizes ($N = 100, 400, 900$), we use the generated labeled data to train, validate, and test the proposed neural selector and parameter estimator. For PSA, we further vary the number of shared layers (l) in training. Each training run is replicated six times to assess the stability of the training procedure and results. All training is performed using the Caffe implementation [90] on one GTX-1080 GPU. The running time each training run takes ranges from five minutes to one hour depending on the values of K and N.

Performance of neural selector and estimator

Overall, the trained model selector and parameter estimator demonstrate excellent performances.

Accuracy of the model selector Fig. 2.10 shows the confusion matrix based on large CNN and PSA-5 neural model selector on test dataset with $K = 20$ distributions. Table 2.9 presents the performance of the model selector on the test dataset under all the combinations of SA architecture, CNN architecture, number of candidate models

K, and sample size N. The selection accuracy together with standard deviation in parentheses based on repeated six runs are reported, the better result between NSA and PSA SA architectures is denoted as bold. For PSA, we report the best results based on layer analysis, they are PSA-3, PSA-2, and PSA-5 for small, medium, and large CNN architectures, respectively.

The table shows that the selection accuracy decreases as K increases under fixed N, and the accuracy increases as we have larger sample size. When we have a moderate sample size, 400, the partially shared CNN architecture can achieve more than 90% selection accuracy. In order to maintain high accuracy for large number of candidate models, large sample sizes should be used.

Accuracy of the parameter estimator The performance of the parameter estimator on the test dataset under different scenarios is reported in Table 2.11 in Appendix. Fig. 2.4 shows the scatter plots of true parameter values and predicted values estimated by parameter estimator under different combinations of architectures and sample sizes N. Fig. 2.12, 2.12, 2.13, 2.14, 2.15 show the scatter plot of true parameter values and predicted values for 50 distributions under the use of architecture PSA-5 on the test dataset with $N = 900$. Overall, the large CNN PSA-5 parameter estimator performs the best on almost all the distributions in the model set.

Impact of SA architectures on learning rate Fig. 2.5 is used to compare the impacts of the NSA and PSA-l SA architectures on the learning rates of the model selector and the parameter estimator, respectively. The medium and large CNN architectures are used, and all the three sample sizes are considered. The upper panel is for medium CNN architecture while the lower panel is for large CNN architecture. The upper left panel of Fig. 2.5 plots the accuracy of the model selector evaluated on the validation dataset against the number of iterations during the training process, whereas the upper right panel plots the log Huber loss of the parameter estimator. Solid curves are for the PSA-2 SA architecture, and dotted curves are for the NSA architectures. It is clear from the plots that the learning rate under PSA-2 is faster than that

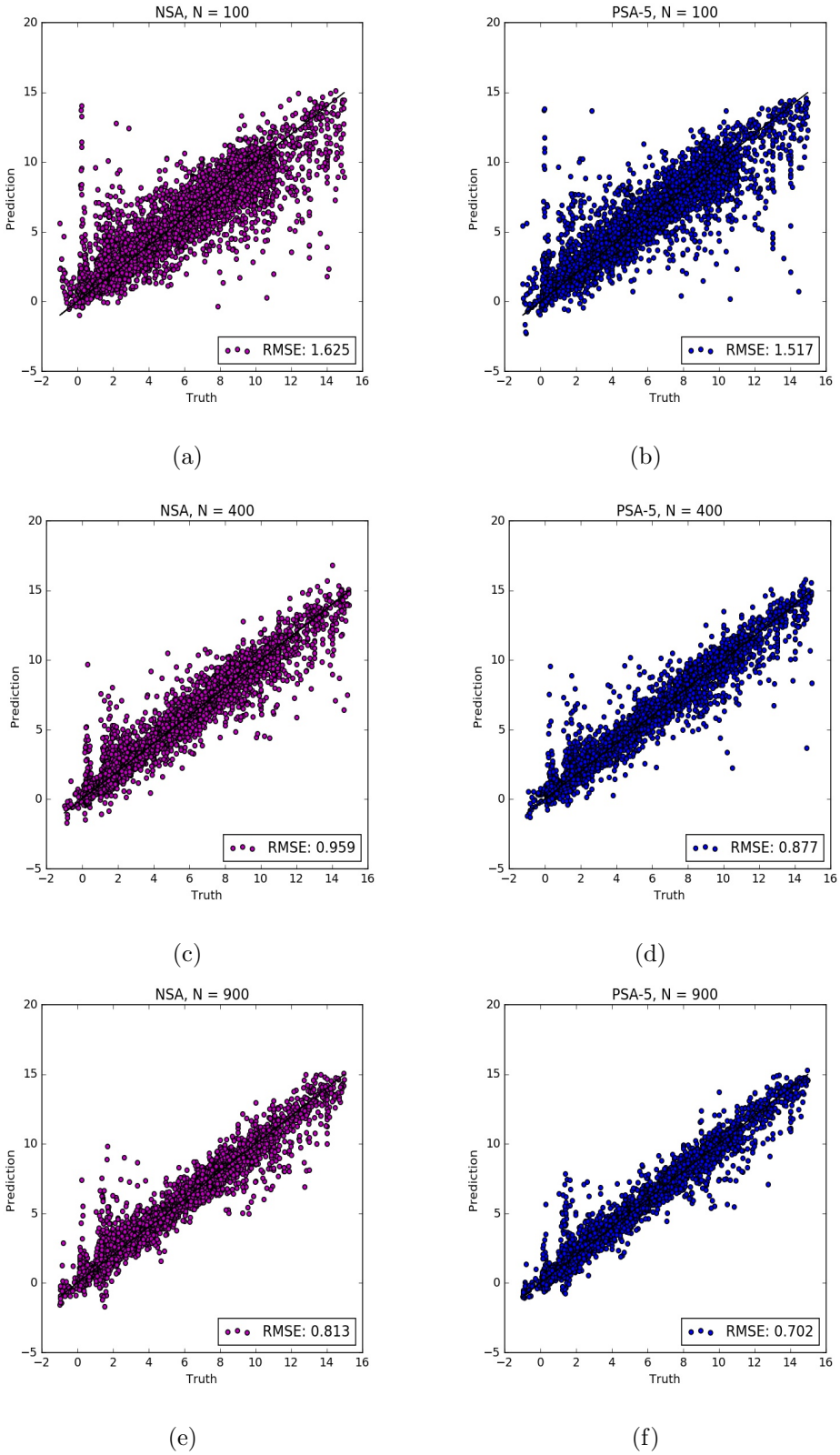


Fig. 2.4.: Parameter estimation results on the test dataset with $K = 50$. The left panel is the results estimated by large CNN and NSA parameter estimator, while the right panel is estimated by large CNN and PSA-5 parameter estimator. The x-axis in each plot is the ground truth for the parameters and the y-axis is the estimation.

Table 2.9.: Model selection results under all the combinations of SA architecture, CNN architecture, number of candidate models K, and sample size N.

	CNN Architecture	$N = 100$		$N = 400$		$N = 900$	
		NSA	PSA	NSA	PSA	NSA	PSA
$K = 5$	small	96.88% (0.12%)	**96.92%** (0.11%)	97.68% (0.19%)	**97.78%** (0.13%)	97.98% (0.13%)	**98.01%** (0.07%)
	medium	96.06% (0.29%)	**96.62%** (0.21%)	97.68% (0.25%)	**97.93%** (0.16%)	97.85% (0.08%)	**97.90%** (0.16%)
	large	**97.30%** (0.19%)	97.13% (0.13%)	**97.88%** (0.08%)	97.77% (0.08%)	98.01% (0.13%)	98.01% (0.17%)
$K = 20$	small	90.76% (0.17%)	**91.44%** (0.20%)	96.34% (0.33%)	**96.59%** (0.27%)	97.79% (0.20%)	**97.81%** (0.24%)
	medium	67.73% (3.09%)	**88.98%** (0.86%)	95.11% (0.47%)	**96.07%** (0.46%)	97.78% (0.14%)	**98.03%** (0.08%)
	large	92.18% (0.23%)	**92.53%** (0.35%)	97.09% (0.23%)	**97.19%** (0.32%)	98.37% (0.29%)	**98.47%** (0.14%)
$K = 50$	small	73.33% (0.54%)	**74.00%** (0.88%)	86.34% (0.29%)	**86.52%** (0.59%)	**90.10%** (0.59%)	90.00% (0.35%)
	medium	48.93% (1.92%)	**71.61%** (0.66%)	83.86% (0.37%)	**87.21%** (0.41%)	88.72% (0.25%)	**90.38%** (0.34%)
	large	75.77% (0.64%)	**78.19%** (0.48%)	88.58% (0.50%)	**88.98%** (0.31%)	91.08% (0.28%)	**91.11%** (0.42%)

under NSA, indicating that sharing convolutional layers between the model selector and parameter estimator can expedite their training rates. Similar patterns could be found in other scenarios as showed in Fig. 2.6, 2.7, 2.8.

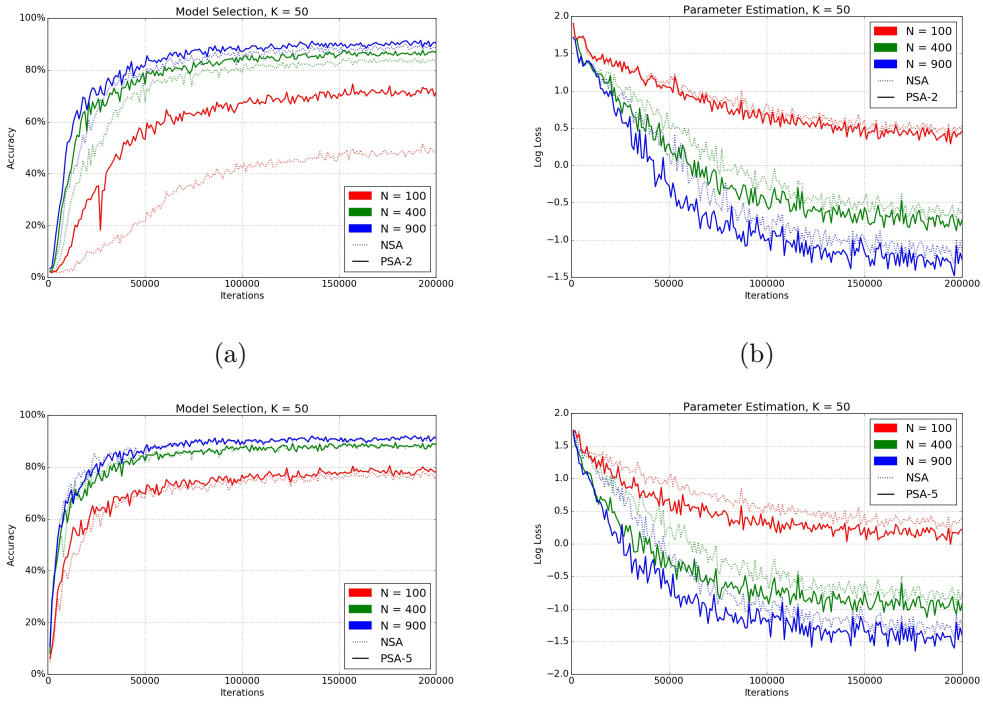


Fig. 2.5.: Comparison between NSA and PSA-l neural model selector and parameter estimator, different colours denote for different sample sizes, upper panel is for medium CNN architecture and lower panel is for large CNN architecture.

How many layers should be shared? Fig. 2.9 shows the impact of the number of shared layers between the model selector and parameter estimator on their performances. We consider the scenario with $K = 50$, $N = 100$, and the medium and large CNN architectures, and vary the SA architectures from NSA to FSA. The left panel of Fig. 2.9 presents the boxplots of accuracy of the model selector under various SA architectures, whereas the right panel presents the boxplots of the Huber loss of the parameter estimator. In terms of model selection accuracy, for medium CNN archi-

tecture, PSA-1 shows significant improvement over NSA, and PSA-2 further improves upon PSA-1, though the amount of improvement from PSA-1 to PSA-2 is small. PSA-3 performs almost the same as PSA-2, and further increasing the number of shared layers leads to slight decrease in selection accuracy. In terms of estimation accuracy (i.e. Huber loss), we can observe similar patterns as the selection accuracy for NSA, PSA-1 and PSA-2. As the number of shared layers further increases, the estimation accuracy declines fairly fast. The results suggest that the PSA-2 SA architecture is optimal for both of the model selector and parameter estimator for the medium CNN architecture. For the large CNN architecture, the optimal SA architecture turns out to be PSA-5 instead.

Comparison with conventional methods

We apply the three conventional model selection methods, the KS distance, BIC, and Bayes factor to the test datasets under the scenario with $K = 20$, and compare their performances with that of the trained neural model selector. Table 2.10 reports the accuracy of the three statistical methods as well as the trained neural model selector under various sample sizes. From the table, it is clear that the neural model selector outperforms the three statistical methods by a significant margin.

In terms of accuracy in parameter estimation, conventional statistical estimators and the proposed neural estimator are not directly comparable. The former is based on the knowledge of the model, whereas the latter does not assume the underlying model is known. If the model is known, then statistical methods such as the maximum likelihood estimation (MLE) method are shown to enjoy certain optimality. For example, MLEs are asymptotically most efficient under some regularity conditions. If the model is unknown, then most conventional statistical methods are not applicable, but the neural parameter estimator can still work well.

Table 2.10.: Comparison of model selection methods on model set with $K = 20$.

	$N = 100$		$N = 400$		$N = 900$	
	Top-1	Top-2	Top-1	Top-2	Top-1	Top-2
KS distance	72.5%	83.2%	83.3%	85.0%	84.7%	85.0%
BIC	69.9%	74.6%	74.7%	75.0%	75.0%	75.0%
Bayes factor	75.5%	84.8%	77.8%	83.3%	70.0%	75.0%
Neural selector	92.1%	99.2%	96.4%	99.7%	97.9%	99.7%

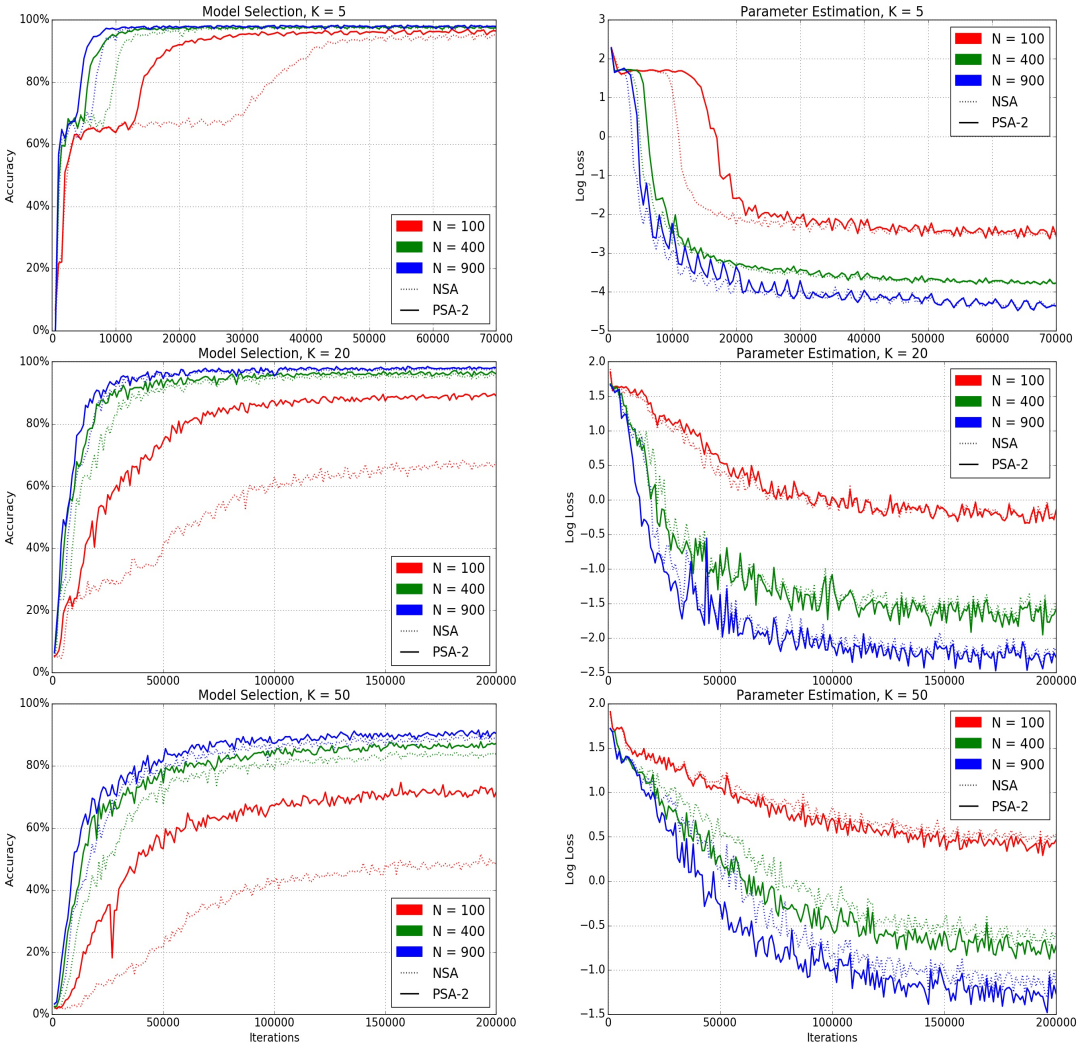
2.3.5 Neural selector for models with covariates

In the Introduction, we propose to build neural network based framework to automate the process of model selection and parameter estimation. In the previous sections, we developed the neural model selector and parameter estimator targeting on only univariate models with single parameter. Our idea and proposed framework can be extended to handle more sophisticated models. In this section, we extend the neural selector to a group of commonly used simple regression models.

Let the model set $\mathcal{M}$ include the following seven regression models: simple linear regression model, Poisson regressoin model, Logistic regression model, Negative Binomial regression model, Lognormal regression model, Loglinear regression model, and multinormial regression model. Let $\{(y_j, x_j)\}_{1\leq j\leq N}$ be a sample generated from one of the seven model. As before, the neural model selector is a CNN-based classifier that maps the sample to its generating model, and we will use labeled data systematically generated from the seven models to train this neural model selector.

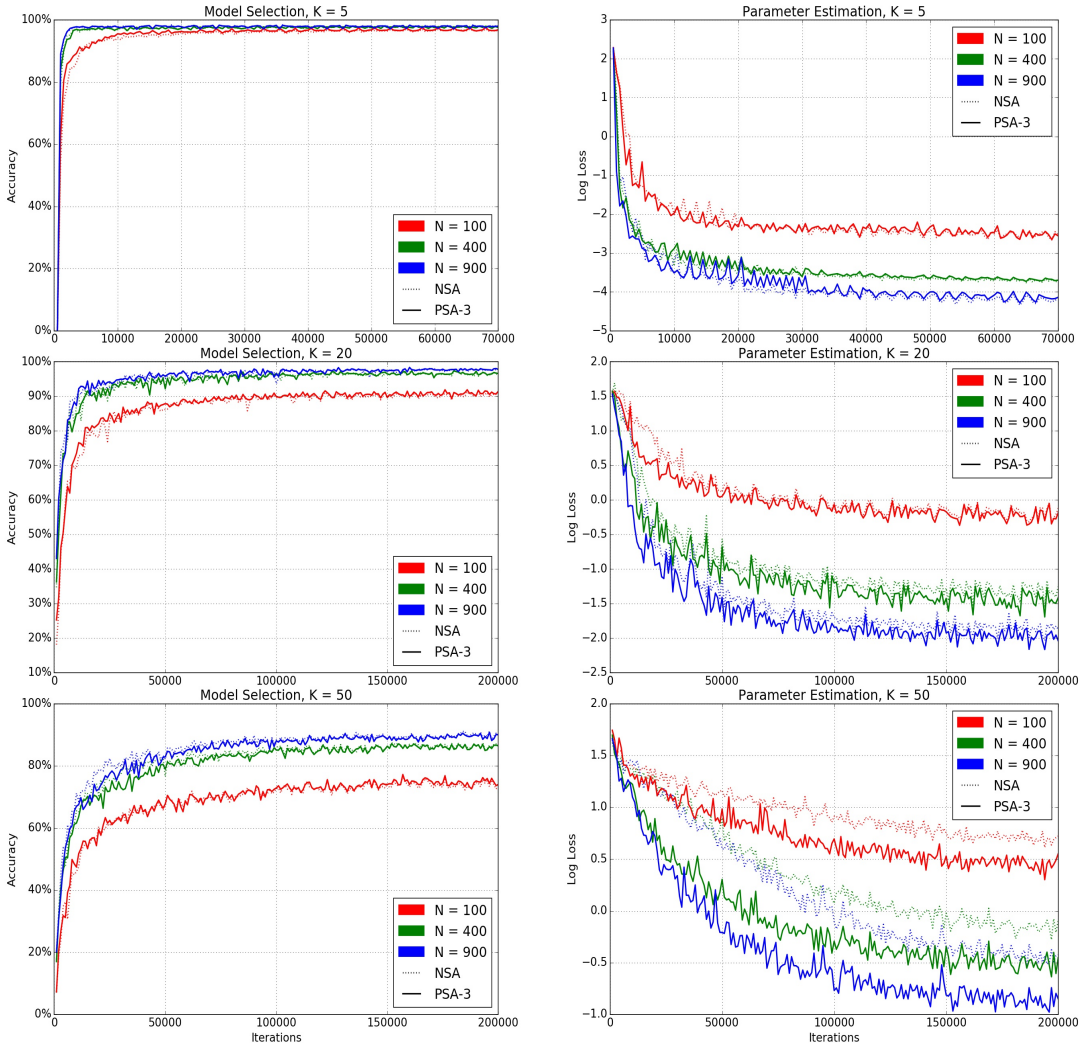
The labeled data are generated as follows. For each regression model, we place an evenly spaced grid over its parameter space. For each vector of the parameter values on the grid, 1000 samples with sample size N are randomly drawn from the model.

The generated data are further partitioned into 70% for training, 20% for validation, and 10% for test. We use the medium CNN architecture, employ the Caffe to train the model selector, and further test the performance of the trained selector on the test dataset. The results show that the trained model selector can achieve 87.86% in accuracy when the sample size is 100, and can achieve 97.86% in accuracy when the sample size is 400.



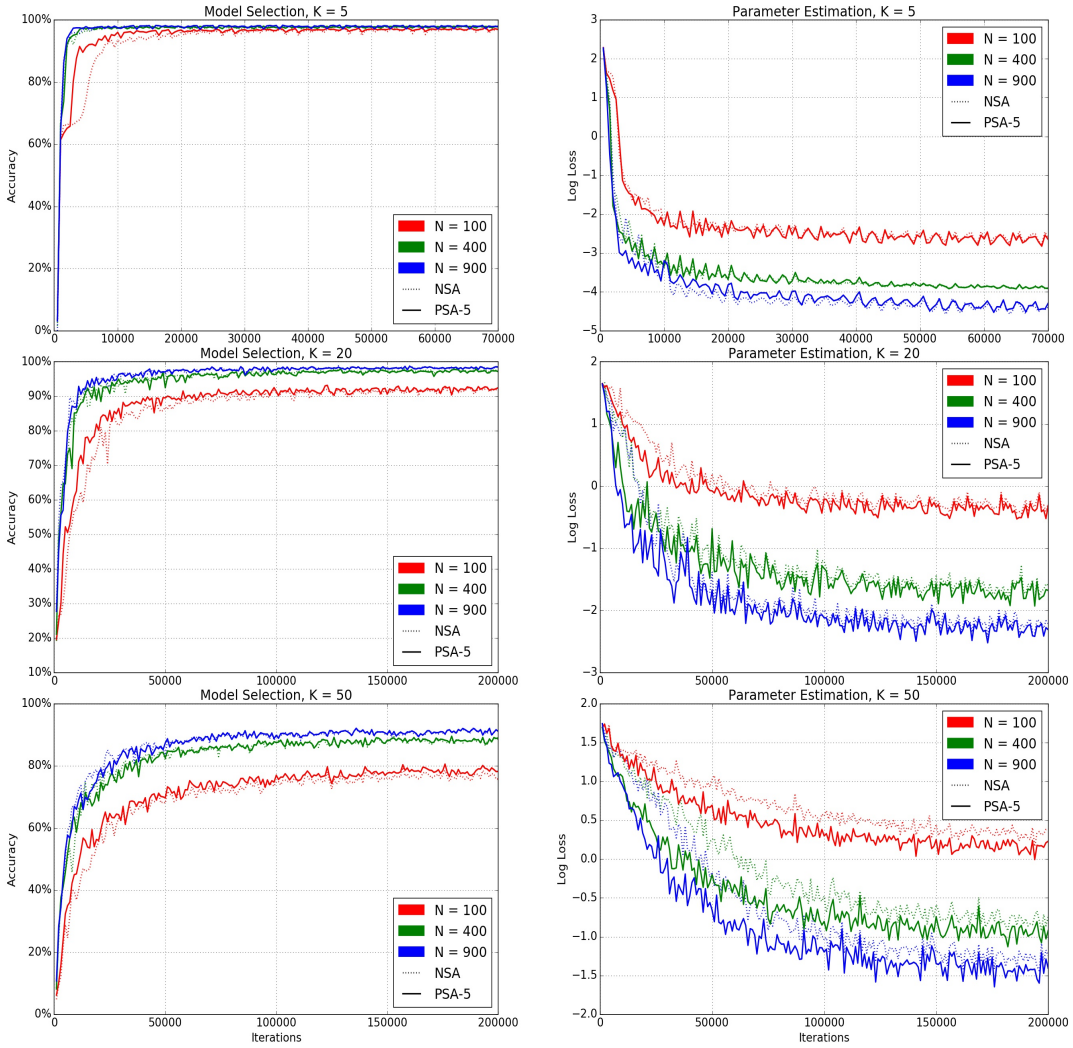
(a) Model Selection (b) Parameter Estimation

Fig. 2.6.: Comparison between NSA and PSA-2: learning curves of selection accuracy and estimation Huber loss for different samples sizes and different number of candidate models. This is based on medium CNN architecture. The left panel plots the selection accuracy of the model selector evaluated on the validation dataset against the number of iterations during the training process, whereas the right panel plots the log Huber loss of the parameter estimator on the validation dataset against the number of iterations during the training process. Solid curves are for the PSA-2, and dotted curves are for the NSA. Different colors denote for different sample sizes.



(a) Model Selection (b) Parameter Estimation

Fig. 2.7.: Comparison between NSA and PSA-3: learning curves of selection accuracy and estimation Huber loss for different samples sizes and different number of candidate models. This is based on small CNN architecture. The left panel plots the selection accuracy of the model selector evaluated on the validation dataset against the number of iterations during the training process, whereas the right panel plots the log Huber loss of the parameter estimator on the validation dataset against the number of iterations during the training process. Solid curves are for the PSA-3, and dotted curves are for the NSA. Different colors denote for different sample sizes.



(a) Model Selection (b) Parameter Estimation

Fig. 2.8.: Comparison between NSA and PSA-5: learning curves of selection accuracy and estimation Huber loss for different samples sizes and different number of candidate models. This is based on large CNN architecture. The left panel plots the selection accuracy of the model selector evaluated on the validation dataset against the number of iterations during the training process, whereas the right panel plots the log Huber loss of the parameter estimator on the validation dataset against the number of iterations during the training process. Solid curves are for the PSA-5, and dotted curves are for the NSA. Different colors denote for different sample sizes.

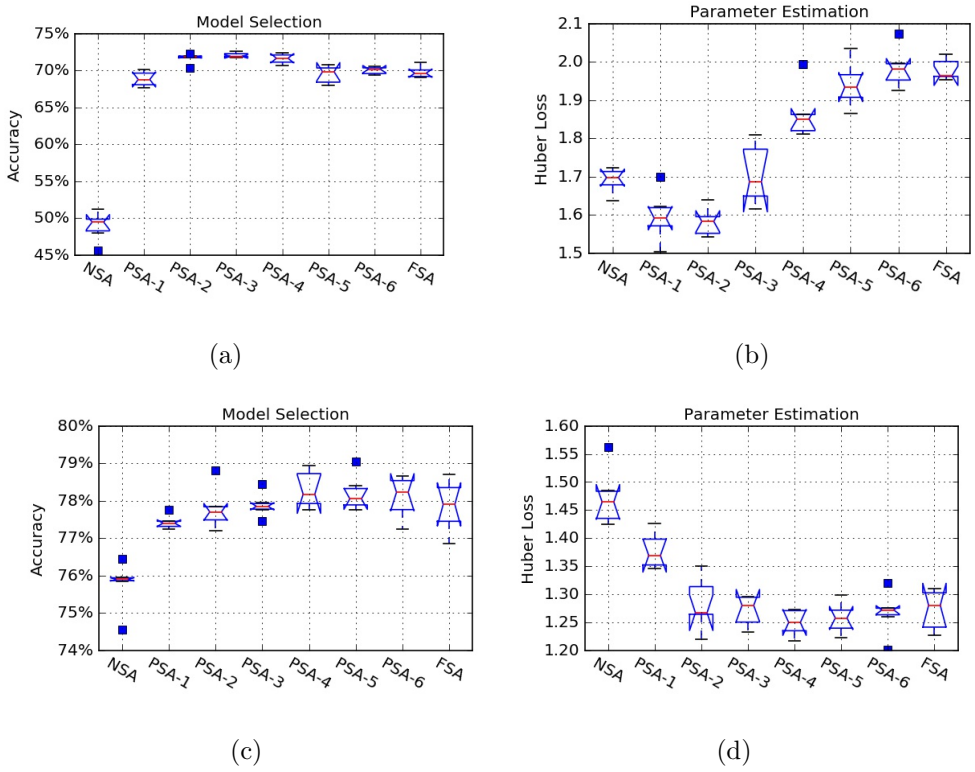


Fig. 2.9.: Information sharing comparison for medium and large CNN architectures, $K = 50$ and $N = 100$. The upper panel is for medium CNN architecture and the lower panel is for large CNN architecture.

Table 2.11.: Parameter estimation results under all the combinations of SA architecture, CNN architecture, number of candidate models K, and sample size N. The Huber Loss with standard deviation in parentheses based on six repeated runs are reported, the better result between NSA and PSA SA architectures is denoted as bold. For PSA, we report the best results based on layer analysis, they are PSA-3, PSA-2, and PSA-5 for small, medium, and large CNN architectures respectively. We can see that PSA performs better than NSA in most cases.

	CNN Architecture	$N = 100$		$N = 400$		$N = 900$	
		NSA	PSA	NSA	PSA	NSA	PSA
$K = 5$	small	0.074 (0.0024)	**0.072** (0.0041)	0.022 (0.0008)	0.022 (0.0004)	**0.014** (0.0007)	0.015 (0.0003)
	medium	**0.071** (0.0011)	0.072 (0.0032)	0.020 (0.0003)	0.020 (0.0003)	0.012 (0.0003)	0.012 (0.0002)
	large	0.068 (0.0022)	**0.067** (0.0020)	0.019 (0.0002)	**0.018** (0.0003)	**0.011** (0.0002)	0.012 (0.0001)
$K = 20$	small	0.887 (0.0310)	**0.830** (0.0261)	0.285 (0.0097)	**0.243** (0.0090)	0.154 (0.0106)	**0.130** (0.0077)
	medium	**0.851** (0.0274)	0.864 (0.0216)	0.223 (0.0105)	**0.207** (0.0103)	0.111 (0.0043)	**0.102** (0.0032)
	large	0.752 (0.0472)	**0.732** (0.0431)	0.203 (0.0076)	**0.187** (0.0043)	0.110 (0.0045)	**0.099** (0.0039)
$K = 50$	small	2.056 (0.0724)	**1.734** (0.0344)	0.91 (0.1323)	**0.636** (0.0137)	0.654 (0.0483)	**0.428** (0.0176)
	medium	1.691 (0.0314)	**1.596** (0.0521)	0.561 (0.0169)	**0.484** (0.0153)	0.313 (0.0049)	**0.282** (0.0132)
	large	1.472 (0.0505)	**1.258** (0.0273)	0.480 (0.0114)	**0.399** (0.0062)	0.288 (0.0313)	**0.246** (0.0051)

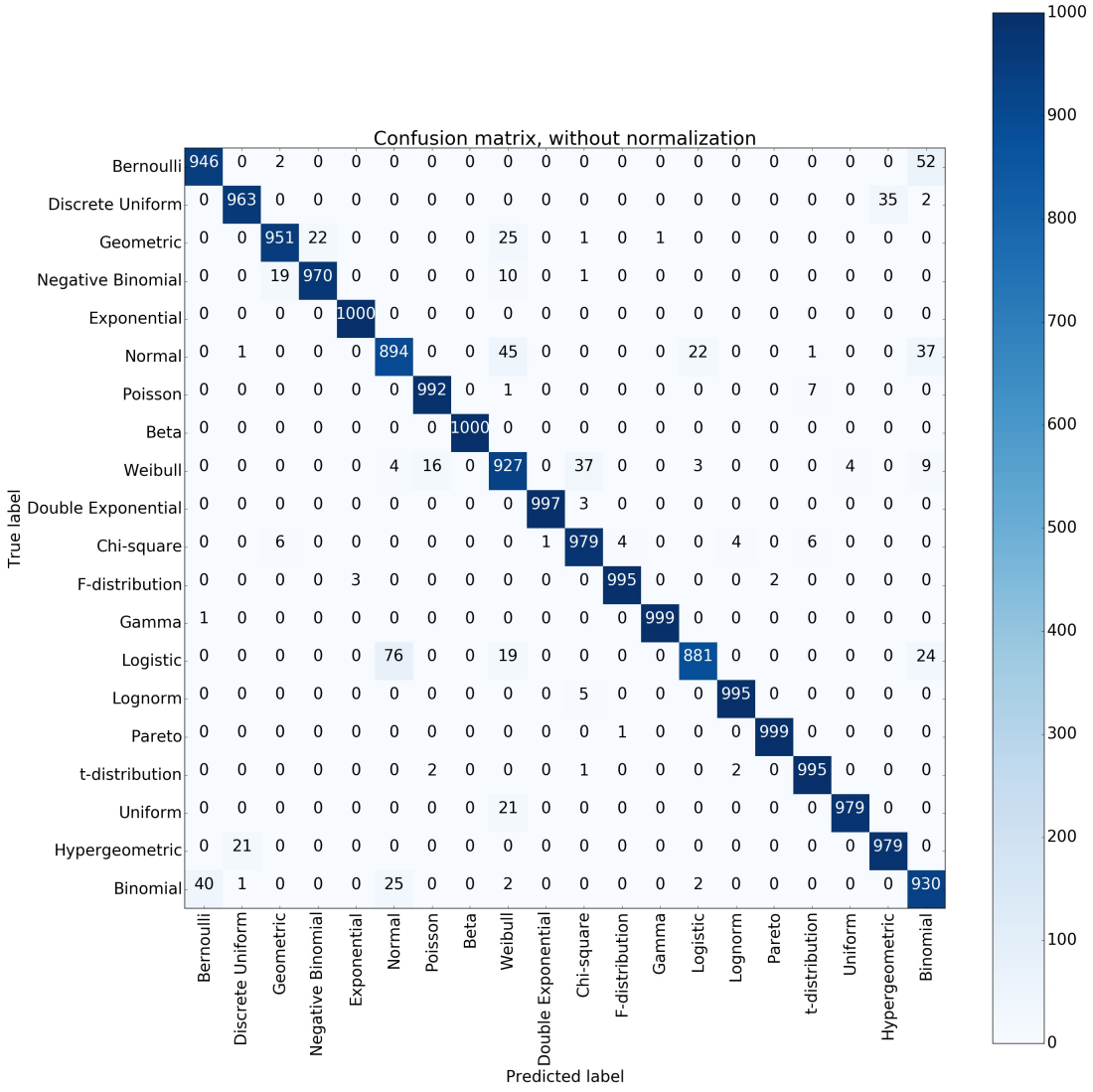


Fig. 2.10.: Confusion matrix based on large CNN and PSA-5 neural model selector on test dataset with $K = 20$

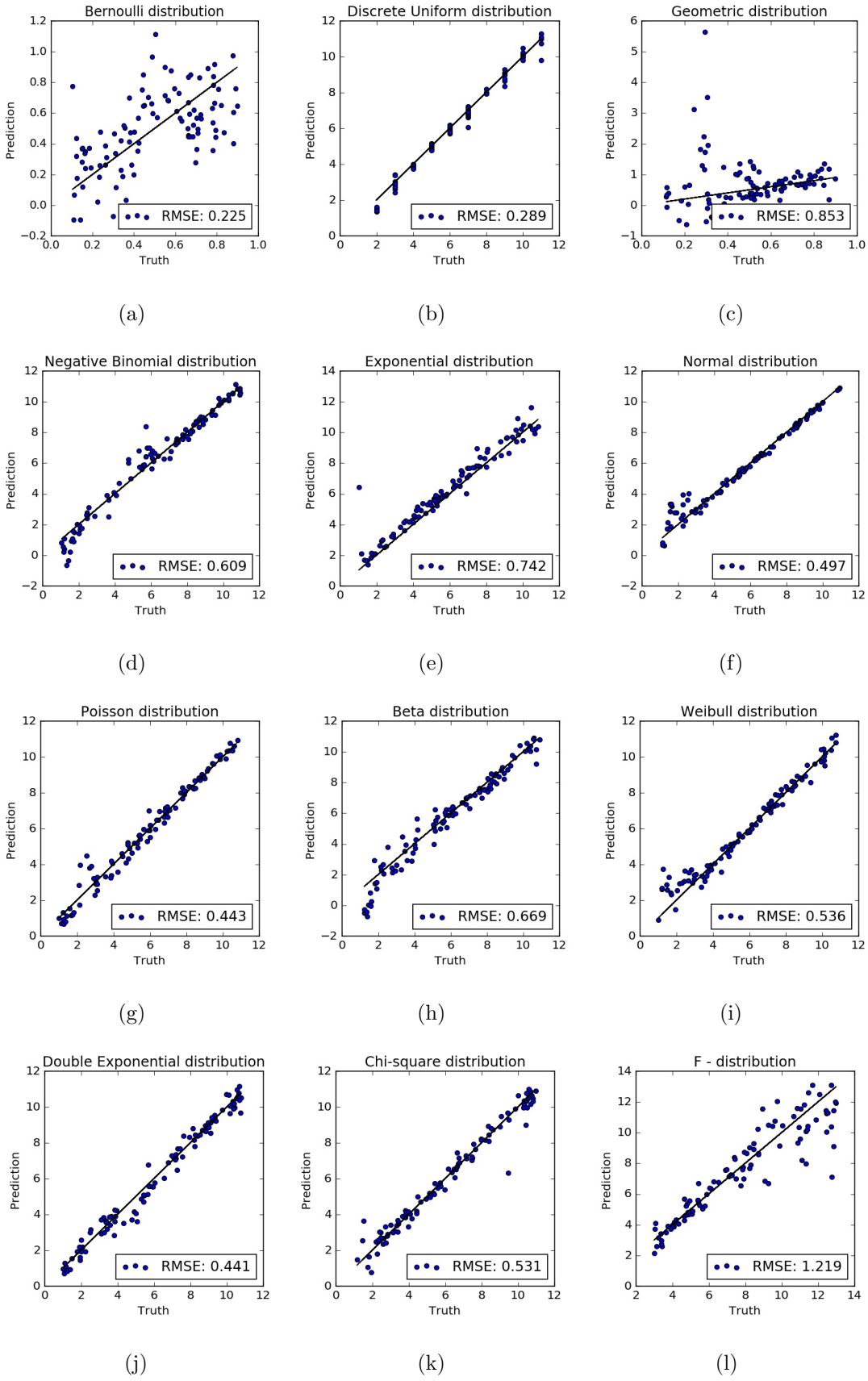


Fig. 2.11.: Distribution wise performance of PSA-5 neural parameter estimator under large CNN on the test dataset with $N = 900$. Part 1.

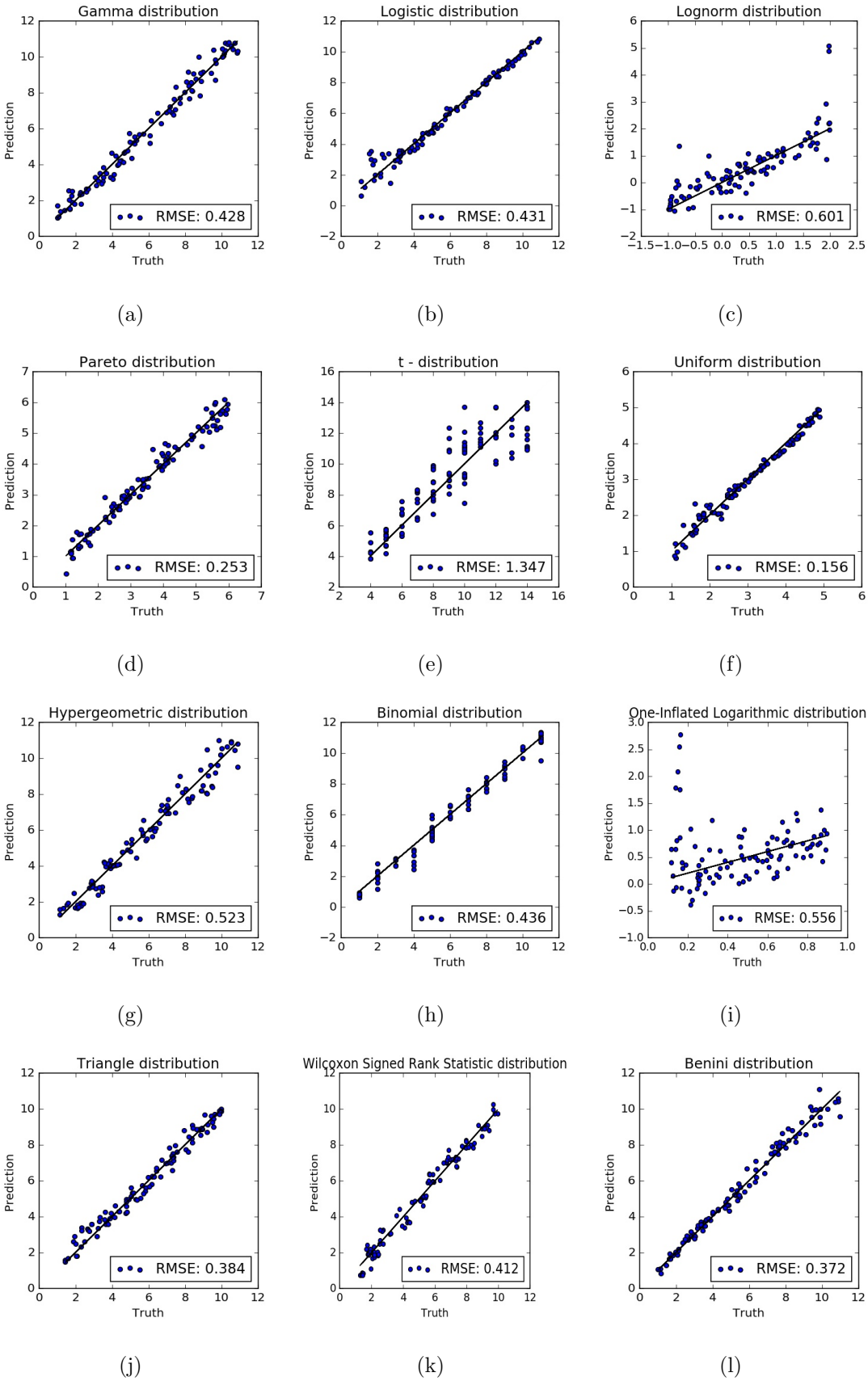


Fig. 2.12.: Distribution wise performance of PSA-5 neural parameter estimator under large CNN on the test dataset with $N = 900$. Part 2.

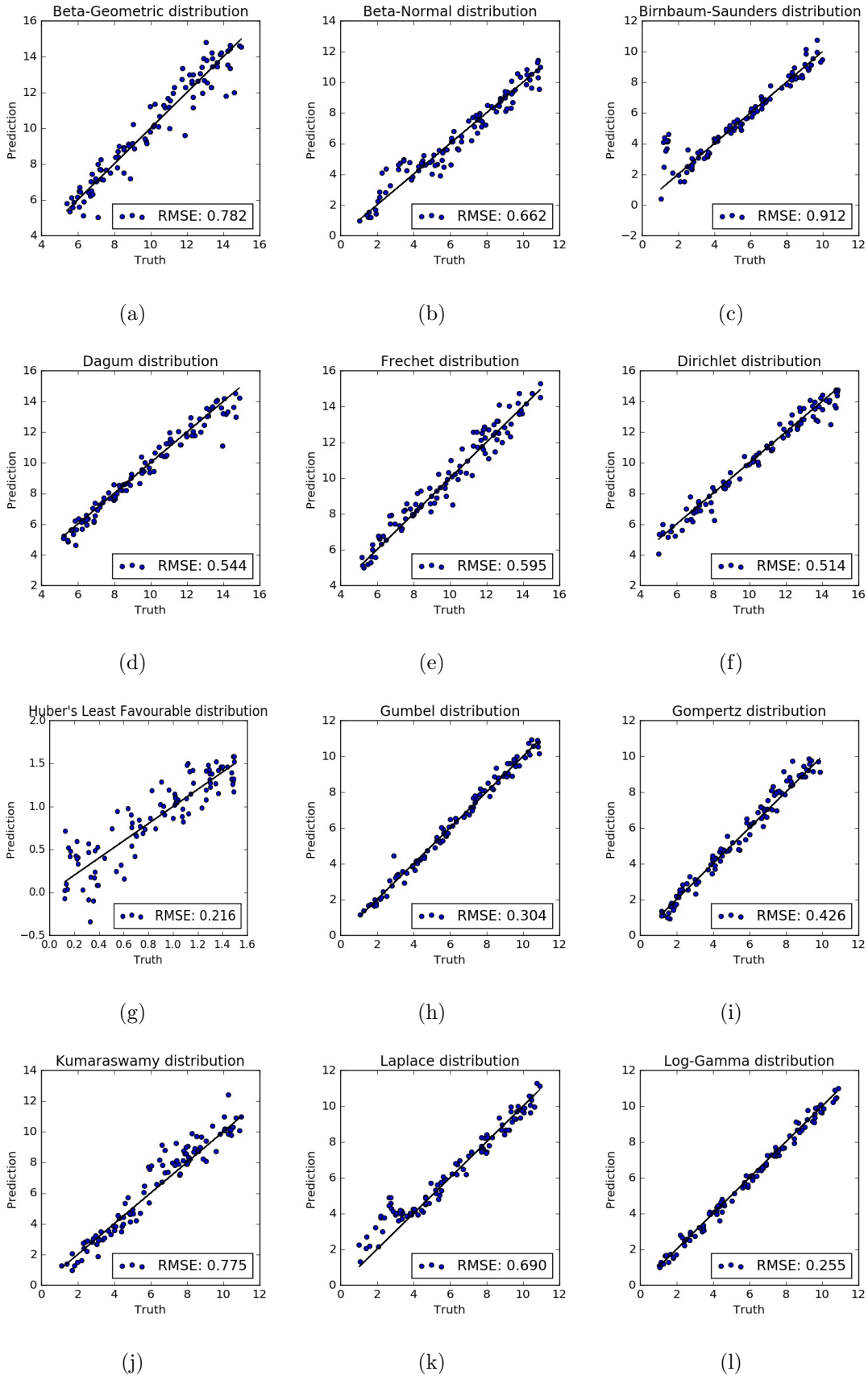


Fig. 2.13.: Distribution wise performance of PSA-5 neural parameter estimator under large CNN on the test dataset with $N = 900$. Part 3.

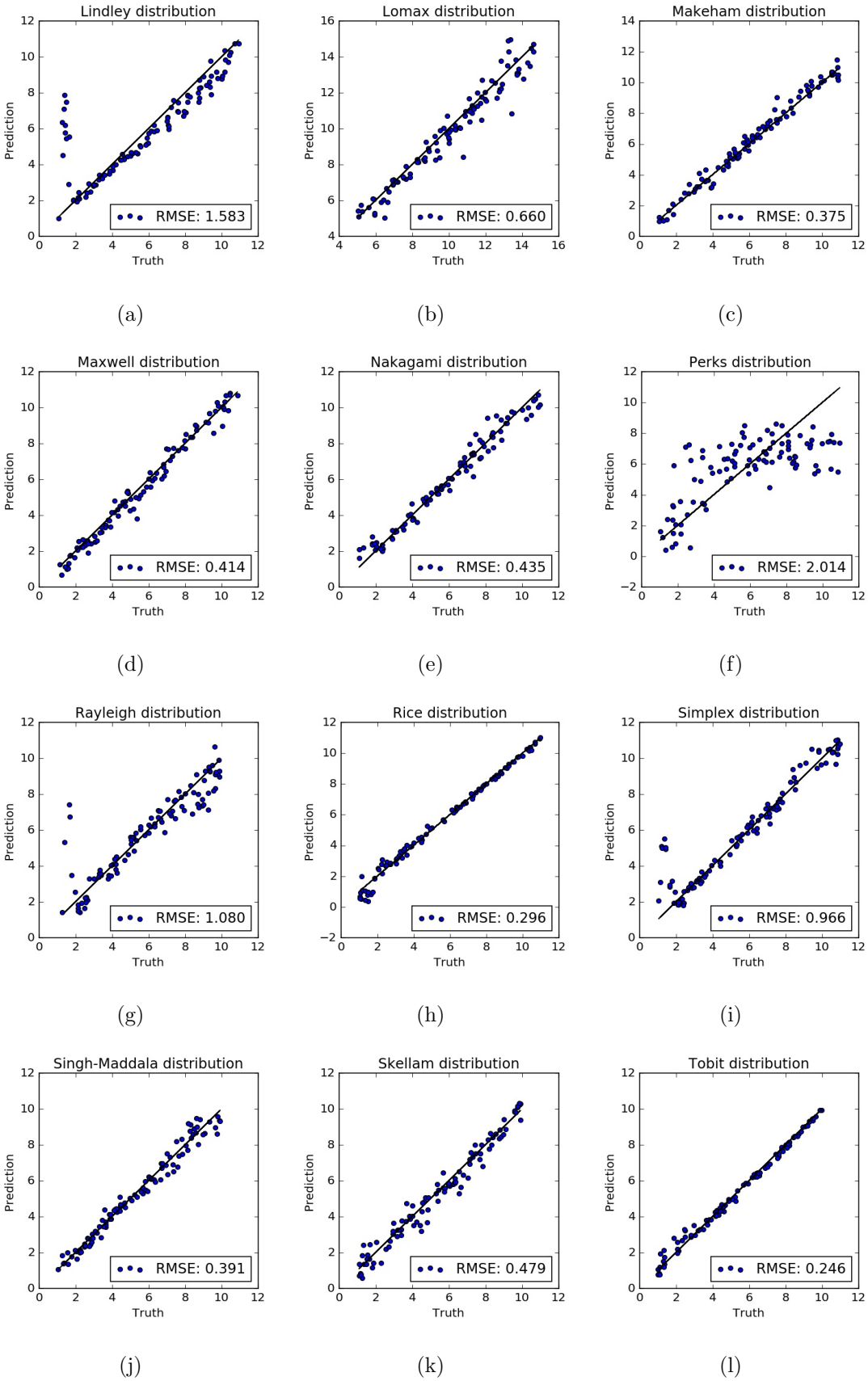
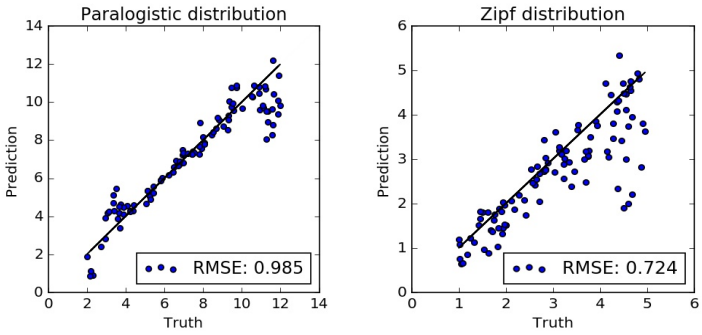


Fig. 2.14.: Distribution wise performance of PSA-5 neural parameter estimator under large CNN on the test dataset with $N = 900$. Part 4.



(a) (b)

Fig. 2.15.: Distribution wise performance of PSA-5 neural parameter estimator under large CNN on the test dataset with $N = 900$. Part 5.

3. NECESSARY AND SUFFICIENT CONDITIONS FOR REGULAR CONDITIONAL INFERENTIAL MODELS

3.1 Introduction

Prior-free probabilistic inference is desirable but challenging. This is evidenced by the century-long efforts starting with fiducial inference introduced by Fisher [91] and including structural equations of Fraser [92], the Dempster-Shafer theory of belief functions [93–95], generalized fiducial inference [96], and confidence distribution [97]. A new framework called Inferential Models (IMs) has been proposed recently by Martin and Liu (2015) [98] for prior-free and yet valid probabilistic inference. It is valid in the sense that the numerical probabilities in its output are consistent with frequency interpretations of probability.

It is well-known that in Bayesian inference, where priors are required to be specified for everything, the centuries-old Bayes theorem is the tool for combining information and thereby playes a fundamental role in the Bayesian framework. In lack of sufficient prior information, the method of Inferential models can be used. In this case, the counterpart or extension of the Bayes theorem is the theory of Conditional Inferential Models (CIMs), which has been introduced to combine information for efficient inference.

The basic idea of behind the theory of CIMs is to sharpen predictive inference via conditioning. The theory of CIMs is thus quite general, and corresponding methods are subject further development. There exists however a class of CIMs, called regular CIMs, that are easy to use. For example, Martin and Liu (2015) [98] showed that when the dimension of minimal sufficient statistics is the same as that of unknown parameters, CIMs are regular and can be obtained based on the minimal sufficient statistics.

In this work we provide a necessary and sufficient condition for the identification of regular CIMs. More specifically, it is shown that for inference based on a sample from continuous distributions with unknown parameters, CIMs are regular *iff* the unknown parameters are generalized location and scale parameters, indexing the transformations of an affine group. This result helps make a new connection between Inferential Models with previous work on fiducial inference by Lindley (1958) [99], Dempster (1963), [100], Dawid and Stone (1982) [101], Taraldsen and Lindqvist (2013) [102].

The remainder of this chapter is organized as follows. Section 2 briefly review Inferential Models and Conditional Inferential Models. Section 3 presents our main results on the necessary and sufficient condition for CIMs to be regular. Section 4 concludes with a remark on related theoretical results found in fiducial inference. The proofs of the theorems are given in Section 5.

3.2 Inferential Models

This section provides a brief review of Inferential Models to introduce the context of discussion and necessary notations.

3.2.1 Basic Inferential Models

Here, we will use the same notations as in [98]. let $X \in \mathbb{X}$ be the observable sample data, where $\mathbb{X}$ is the sample space, and let $\theta \in \Theta$ be the parameter of interest, where Θ is the parameter space. Here, both X and θ could be multi-dimensional.

The starting point of the IM framework is an auxiliary variable, denoted by $U \in \mathbb{U}$ and equipped with probability measure P_U, it is associated with X and θ. The sampling distribution for X is characterized by this association for given θ. We can write this association as

$$X = a(\theta, U), \quad U \sim \mathsf{P}_U. \tag{3.1}$$

Note that the subscripts on P indicate which quantity is random.

The IM approach treats the unobserved value of U as the fundamental quantity, and the goal is to predict this unobserved value with a random set before conditioning on $X = x$ and inverting (3.1). Let $(U, \mathscr{U}, P_U)$ be a probability space, where $\mathscr{U}$ is rich enough to contain all closed subsets of $\mathbb{U}$. Take a nested collection $\mathbb{S}$ of closed (hence P_U-measurable) subsets of $\mathbb{U}$, assumed to contain $\varnothing$ and $\mathbb{U}$. An admissible predictive random set $\mathcal{S} \sim \mathsf{P}_{\mathbb{S}}$ is defined based on the support of the collection $\mathbb{S}$ and with distribution $\mathsf{P}_{\mathcal{S}}$ satisfying

$$\mathsf{P}_{\mathcal{S}}\{\mathcal{S} \subseteq K\} = \sup_{\mathcal{S} \in \mathbb{S}: \mathcal{S} \subseteq K} \mathsf{P}_U(\mathcal{S}), \quad K \subseteq \mathbb{U}.$$

IMs take the sampling model and observed data as the input, and produce prior-free, probabilistic measures of certainty about any assertion/hypothesis of interest. The fundamental idea of IM is that uncertainty about the parameter θ, given observed data $X = x$, is fully characterized by an unobservable auxiliary variable U. Thus, the problem of inference about θ can be translated into one of predicting this unobserved U with a random set.

To be more specific, the IM is constructed with three steps, association, predict, and combine.

A-step Associate X, θ, and $U \sim \mathsf{P}_U$, consistent with the sampling distribution $X \sim \mathsf{P}_{X|\theta}$, such that, for all (x, u), there is a unique subset $\Theta_x(u) = \{\theta : x = a(\theta, u)\} \subseteq \Theta$, possibly empty, containing all possible candidate values of θ given (x, u).

P-step Predict the unobserved value $u^\star$ of U associated with the observed data by an admissible predictive random set $\mathcal{S}$.

C-step Combine $\mathcal{S}$ and the association $\Theta_x(u)$ specified in the A-step to obtain

$$\Theta_x(\mathcal{S}) = \bigcup_{u \in \mathcal{S}} \Theta_x(u).$$

Then, for any assertion or hypothesis $A \subseteq \Theta$ about parameter of interest θ, we can compute the *belief probability* that the random set $\Theta_x(\mathcal{S})$ is a subset of A as a measure of the available evidence in x supporting A.

$$\mathsf{bel}_x(A; \mathcal{S}) = \mathsf{P}_{\mathcal{S}}\{\Theta_x(\mathcal{S}) \subseteq A | \Theta_x(\mathcal{S}) \neq \varnothing\}.$$

The *plausibility function* is defined as

$$\mathsf{pl}_x(A; \mathcal{S}) = 1 - \mathsf{bel}_x(A^c; \mathcal{S}).$$

Then the pair $(\mathsf{bel}_x, \mathsf{pl}_x)(A; \mathcal{S})$ characterizes the IM output and provide a probabilistic summary of the evidence in data $X = x$ supporting the truthfulness of assertion A.

3.2.2 Conditional Inferential Models

Construction of efficient predictive random sets is relatively easy in the case of scalar auxiliary variable. But usual case is that the model rarely admits a scalar auxiliary variable representation but multi-dimensional auxiliary variable U, and efficient prediction of U would be challenging. If we can reduce the dimension of auxiliary variable U, ideally to dimension one, then choosing the efficient predictive random set is as easy as dealing with the scalar case.

Let's start with a simple normal mean example with $n = 2$. The baseline association is

$$X_1 = \theta + U_1, \qquad X_2 = \theta + U_2, \tag{3.2}$$

where $U = (U_1, U_2) \sim \mathsf{N}_2(0, I)$. The dimension of auxiliary variable is two. First, let's use this example to show how three steps in IM framework as described in the previous section. In the A-step, we consider the change of variable: $Y_1 = X_1 + X_2$, $Y_2 = X_1 - X_2$, and the new association:

$$Y_1 = 2\theta + V_1, \qquad Y_2 = V_2, \tag{3.3}$$

where $V = (V_1, V_2) \sim \mathsf{N}_2(0, 2I)$. The predictive random set $\mathcal{S}$ for prediction of (V_1, V_2) is a random square.

$$\mathcal{S} = \{(v_1, v_2) : \max(|v_1|, |v_2|) \leq \max(|V_1|, |V_2|)\}, \quad (V_1, V_2) \sim \mathsf{N}_2(0, 2I).$$

The plausibility function is the C-step for a singleton assertion $\{\theta\}$ is

$$\mathsf{pl}_y(\theta) = \frac{1 - G(2^{-1/2}\max\{|y_1 - 2\theta|, |y_2|\})^2}{1 - G(2^{-1/2}|y_2|)^2},$$

where $G(z) = 1 - 2(1 - \Phi(z))$ is the $|\mathsf{N}(0,1)|$ distribution function. This completes the three steps in IM procedure.

Note that in the association 3.3, the value of V_2 is actually known once we observe Y_2. Instead of predicting this component, we can rely on the observation of Y_2 to sharpen the uncertainty about prediction of V_1. Then for A-step, we have $Y_1 = 2\theta + V_1$, the prediction random set in P-step is as simple as $\mathcal{S} = \{v_1 : |v_1| \leq V_1\}$, where $V_1 \sim \mathsf{N}(0, 2)$. We can get a more efficient plausibility function in C-step

$$\mathsf{pl}_y(\theta) = 1 - |2\Phi(2^{-1/2}(y_1 - 2\theta)) - 1|.$$

The key here is that the function of original auxiliary variables, $V_2 = U_1 - U_2$ is fully observed, and we can condition on what is fully observed to sharpen prediction of what is not observed, this directly leads to a dimension reduction.

The general strategy of conditional IM is as follows:

1. Identify an observed characteristic, $\eta(U) = H(x)$, of the auxiliary variable U whose distribution is free (or at least mostly free) of θ;

2. define a *conditional association* that relates an unobserved characteristic, $\tau(U)$, of the auxiliary variable U to θ and some function $T(X)$ of the data X.

Then the original baseline association $x = a(u; \theta)$ can be decomposed as

$$H(x) = \eta(u), \tag{3.4a}$$

$$T(x) = b(\tau(u), \theta). \tag{3.4b}$$

This decomposition indicates an alternative conditional association. Since $H(x)$ does not provides any information about θ, we can take a new association

$$T(X) = b(\tau(U), \theta), \quad V \sim \mathsf{P}_{V|H(x)}, \tag{3.5}$$

where $\mathsf{P}_{V|H(x)}$ is the conditional distribution of V, given $H(x)$. Usually we can choose τ to make V have lower dimension than U so that the construction of efficient predictive sets for V is easier and condition on the observed naturally provide more information to make the predictive ability more efficient. Once we find the decomposition, it's easy to construct the corresponding IM framework as described in subsection 3.2.1.

3.3 The problem and Main results

Recall that the association is $X = a(\theta; U)$, the dimension of X and U is n and the dimension of parameter θ is m such that $m \leq n$. In CIM, we want to find the observed characteristics $\eta(U)$ to reduce the dimension.

Definition 3.3.1 *The association* (3.1) *is regular if it can be written in the form* (3.4).

In this section, we will give the necessary and sufficient conditions for the identification of regular CIM. Since the proof techniques vary for different n and m, we will state the results in different theorems in this section.

3.3.1 Differential equations-based technique for finding conditional associations

In [98], authors proposed a novel technique for finding conditional associations based on differential equations. The method can be used for going directly from the baseline association to something lower-dimensional.

We will take the single parameter case as an example, similar technique can be used to deal with multi-parameter case. Suppose $\Theta \subseteq \mathbb{R}$. The intuition is that τ should map $\mathbb{U} \subseteq \mathbb{R}^n$ to Θ, while η maps $\mathbb{U}$ into a $(n-1)$-dimensional manifold in $\mathbb{R}^n$ such that $V = \tau(U)$ is one-dimensional and η is insensitive to changes in θ. Suppose that $u_{x;\theta}$ is the unique solution for u in the baseline association $x = a(\theta; u)$. We

require $\eta(u_{x;\theta})$ be constant with respect to θ for fixed x. Mathematically, we require that $\partial u_{x;\theta}/\partial\theta$ exists and

$$0 = \frac{\partial \eta(u_{x;\theta})}{\partial \theta} = \frac{\partial \eta(u)}{\partial u} \cdot \frac{\partial u_{x;\theta}}{\partial \theta}. \tag{3.6}$$

It is clear that, if there exists a solution η to partial differential equation 3.6, then the value of $\eta(U)$ is fully observed. If we choose τ carefully, we can find such a corresponding function H that $H(X) = \eta(U)$ which does not depend on θ. The solution η of 3.6 determines the decomposition 3.4. The differential equation-based technique will serve as the base of the construction of sufficient and necessary conditions of the identification of regular CIM.

3.3.2 Single parameter case

Let's start with the simplest case that when the parameter space is in $\mathbb{R}$. Suppose that we have a simple association with $n = 2$.

$$X_1 = g_1(U_1, \theta), \tag{3.7}$$

$$X_2 = g_2(U_2, \theta), \tag{3.8}$$

where g_1 and g_2 are differentiable functions with respect to both θ and U. The dimension of auxiliary variable $U = (U_1, U_2)$ is two. In regular CIM, we want to first construct one observed characteristic $H(X) = \eta(U)$ which is free of θ.

In the simple normal mean example 3.2 mentioned previously, there indeed exists the decomposition as showed in 3.3 and the observed characteristic is $H(X) = U_1 - U_2$, $\tau(U) = U_1 + U_2$. This is an easy example that we can just eyeball the observed characteristic without using partial differential-based technique. In this case, we can see that θ is a location parameter. Actually this is not a coincidence. For $n = 2$, the necessary and sufficient condition for the identification of observed characteristic H is that θ is a generalized location parameter. Here generalized location parameter means that there exists transformation of U and θ, such that transformed θ is a

location parameter. By this definition, we can see that actually scale parameter is a generalized location parameter if we do log transformation.

Definition 3.3.2 *In a baseline association $X = a(U, \theta)$, we call θ is a generalized location and scale parameter if there exist transformations $V = \tau(U)$ and $\delta = \phi(\theta)$ such that*

$$X = a(\tau^{-1}(V), \phi^{-1}(\delta)) = b(V + \delta).$$

Now we can summarize the necessary and sufficient condition for two observations and one parameter case in the following theorem.

Theorem 3.3.1 *Suppose we have the baseline association*

$$X_1 = g_1(U_1, \theta), \tag{3.9}$$

$$X_2 = g_2(U_2, \theta), \tag{3.10}$$

where g_1 and g_2 are differentiable with respect to both θ and U.

The sufficient and necessary condition for the existence of fully observed characteristic $\eta(U_1, U_2)$ which could be used as condition is that θ is a generalized location parameter.

Proof of this theorem merely depends on the characteristic method 3.5.2 for solving partial differential equations. See Subsection 3.5.3 for details. This theorem says that as long as θ is a generalized location parameter when $n = 2$, we can always find the observed characteristic that we could use for dimension reduction. This is easily extended to n observations with single parameter.

Theorem 3.3.2 *Let's consider n observations, one parameter included,*

$$\begin{cases} X_1 &= g_1(U_1, \theta), \\ X_2 &= g_2(U_2, \theta), \\ \cdots & \\ X_n &= g_n(U_n, \theta). \end{cases} \tag{3.11}$$

The sufficient and necessary condition for the existence of fully observed variable $\boldsymbol{\eta}(U_1, \cdots, U_n) = (\eta_1, \cdots, \eta_{n-1})$ *is that there exist transformations such that* θ *is a generalized location parameter.*

See Subsection 3.5.4 for the proof. We can see that if we want to reduce the dimension to that same as the parameter space, we need $n-1$ independent fully observed characteristics. In general, if we have n observations with m parameters, for regular CIM, we will need $n-m$ independent fully observed characteristics.

3.3.3 Multi-parameter case

In this section, we want to build the sufficient and necessary condition for the identification of regular CIM. We will start with the two parameters case since the proof technique is quite different though the condition is the same as the single parameter case.

Two parameters case

Now let's expand the dimension of parameter space to two. For example, let's revisit the simple normal mean example with three observations, but both mean and standard deviation are unknown. The baseline association is

$$X_1 = \sigma U_1 + \mu, \quad X_2 = \sigma U_2 + \mu, \quad \text{and} \quad X_3 = \sigma U_3 + \mu,$$

where U_1, U_2, U_3 are independent $\mathsf{N}(0,1)$, $\theta = (\mu, \sigma)$. An observed characteristic could be

$$\eta(\boldsymbol{U}) = \frac{U_1 - U_2}{U_1 - U_3} = \frac{X_1 - X_2}{X_1 - X_3} = H(X),$$

and the original baseline association can be decomposed as

$$\begin{aligned} &\frac{X_1 - X_2}{X_1 - X_3} = \frac{U_1 - U_2}{U_1 - U_3}, \\ &X_1 - X_3 = \sigma(U_1 - U_3), \\ &X_1 = \sigma U_1 + \mu. \end{aligned} \tag{3.12}$$

This decomposition suggests the following alternative conditional association,

$$X_1 = \sigma U_1 + \mu \quad X_1 - X_3 = \sigma(U_1 - U_3),$$

where $V_1 = U_1 \sim \mathsf{N}(0,1)$ and $V_2 = U_1 - U_3 \sim \mathsf{N}(0,2)$. Note that this decomposition and conditional association are not unique. We can see that in this simple example, both σ and μ are scale and location parameters.

We considered the special case that the three observations share the same form of association here, and summarized the necessary and sufficient conditions for the regular CIM as following theorem:

Theorem 3.3.3 *For the baseline association with 3 observations and 2 parameters included,*

$$\begin{cases} X_1 &= a(\theta_1, \theta_2, U_1), \\ X_2 &= a(\theta_1, \theta_2, U_2), \\ X_3 &= a(\theta_1, \theta_2, U_3). \end{cases} \tag{3.13}$$

The sufficient and necessary condition for the existence of fully observed variable $\eta(U_1, U_2, U_3)$ is that $\theta = (\theta_1, \theta_2)$ is generalized location and scale parameters.

See Subsection 3.5.5 for the proof. It's easy to generalize the result to n observations case, the sufficient and necessary condition is the same, please see Subsection 3.5.6 for the proof.

Three parameters case

When it comes to the associations with 3 parameters, the case changed to be different. We fund that it's degenerated. When we say it's degenerated, we mean that there is a map

$$\theta = (\theta_1, \theta_2, \theta_3) \mapsto (T_1(\theta), T_2(\theta))$$

such that $(T_1(\theta), T_2(\theta))$ is generalized scale and location parameter. We summarize this result in the following theorem.

Theorem 3.3.4 *For the baseline association with n observations and 3 parameters included,*

$$\begin{cases} X_1 &= a(\theta_1, \theta_2, \theta_3, U_1), \\ X_2 &= a(\theta_1, \theta_2, \theta_3, U_2), \\ \cdots & \\ X_n &= a(\theta_1, \theta_2, \theta_3, U_n). \end{cases} \tag{3.14}$$

The sufficient and necessary condition for the regular CIM is that $\theta = (\theta_1, \theta_2, \theta_3)$ is degenerated generalized location and scale parameter, i.e. there exist maps such that $(T_1(\theta), T_2(\theta))$ is generalized location and scale parameters.

See Subsection 3.5.8 for the proof with $n = 4$, and the same proof technique could be easily applied with any n.

3.4 Discussion

We only focused on associations that share the same form for all observations and limited the number of parameters within three. For more complicated associations or associations with any number of parameters, the sufficient condition for the identification of regular CIM is still that if θ is generalized location and scale parameters, but it needs more work and technique to prove the necessary part.

3.5 Proofs

3.5.1 Existence of First Order Ordinary Differential Equations

Theorem 3.5.1 *The initial value problem we consider is*

$$\frac{du}{dx} = F(x, u(x)), \quad u(a) = b, \tag{3.15}$$

where F is a function and a, b are given real numbers. If F and $\frac{\partial F}{\partial u}$ are continuous at (a, b) then there is an $\epsilon > 0$ such that there is a unique solution to (3.15) on the interval $a - \epsilon < x < a + \epsilon$

3.5.2 Method of Characteristics

Consider the following quasilinear equation:

$$a(t,x,u)\partial_t u(x,t) + b(t,x,u)\partial_x u(t,x) = f(t,x,u). \tag{3.16}$$

Suppose $u = u(x,t)$ is a smooth solution of 3.16 and let

$$S = \{(t,x,u) \in \mathbb{R}^3 : u = u(x,t)\}.$$

Then S is said to be a solution surface for 3.16. The smoothness of the solution u means that S has a tangent plane at each point $(t,x,u) \in S$. The normal vector $\overrightarrow{n}$ to the tangent plane has the direction numbers $(\partial_t u, \partial_x u, -1)$; i.e. $u(x,t) - u = 0$ is the equation of S and $\partial_t u dt + \partial_x u dx - du = 0$ is the equation of the tangent plane. Now consider a curve $C = \{t = t(s), x = x(s), u = u(s), s \in I\}$ in a 3-space defined as a solution curve for the system

$$\frac{dt}{ds} = a(t,x,u), \quad \frac{dx}{ds} = b(t,x,u), \quad \frac{du}{ds} = f(t,x,u). \tag{3.17}$$

If $\overrightarrow{T}$ denotes a vector tangent to C at (t,x,u) then the direction numbers of $\overrightarrow{T}$ must be (a,b,f). But then 3.16implies that $\overrightarrow{T} \perp \overrightarrow{n}$, which is to say, $\overrightarrow{T}$ lies in the tangent plane to the surface S. But if $\overrightarrow{T}$ lies in the tangent plane, then C must lie in S. Evidently, solution curves of 3.16 lie in the solution surface S associated with 3.16. Such curves are called characteristic curves for 3.16. Note that if C is a solution curve for 3.17 then

$$\frac{du}{ds} = \partial_t u(x,t)\frac{dt}{ds} + \partial_x u(x,t)\frac{dx}{ds} = a(t,x,u)\partial_t u(x,t) + b(t,x,u)\partial_x u(t,x) = f(t,x,u).$$

3.5.3 Proof of Theorem 1

Definition 3.5.1 *Let's consider a bivariate function $f(x,y)$, we say that f is not seperable if $f(x,y)$ can not be written as $g(x) \cdot h(y)$.*

Proof • Sufficiency:

If θ is generalized location and scale parameter, then there exist transformations $V_1 = \tau_1(U_1)$, $V_2 = \tau_2(U_2)$ and $\delta = \phi(\theta)$ such that

$$X_1 = g_1(\tau^{-1}(V_1), \phi^{-1}(\delta)) = b_1(V_1 + \delta),$$
$$X_2 = g_2(\tau^{-1}(V_2), \phi^{-1}(\delta)) = b_2(V_2 + \delta).$$

Then we can construct

$$\eta(U) = b_1^{-1}(X_1) - b_2^{-1}(X_2) = V_1 - V_2 = \tau_1(U_1) - \tau_2(U_2),$$

and η is a function only of U_1 and U_2.

- Necessity:

Based on the partial differential equations-based technique for finding conditional associations, we want to find *eta* such that

$$\frac{\partial \eta(u)}{\partial u} \cdot \frac{\partial u_{x;\theta}}{\partial \theta} = 0.$$

Now we are dealing with the two-dimensional unobserved variable $U = (U_1, U_2)$, we want to find $\eta(U_1, U_2)$ such that

$$\frac{\partial \eta}{\partial U_1}\frac{\partial U_1}{\partial \theta} + \frac{\partial \eta}{\partial U_2}\frac{\partial U_2}{\partial \theta} = 0. \tag{3.18}$$

At the same time, according to (3.9) and (3.10), we have

$$\frac{\partial g_1}{\partial \theta} + \frac{\partial g_1}{\partial U_1}\frac{\partial U_1}{\partial \theta} = 0,$$
$$\frac{\partial g_2}{\partial \theta} + \frac{\partial g_2}{\partial U_2}\frac{\partial U_2}{\partial \theta} = 0.$$

Combine the above three equations, we have

$$\frac{\partial U_1/\partial \theta}{\partial U_2/\partial \theta} = \frac{-\frac{\partial g_1}{\partial \theta}\Big/\frac{\partial g_1}{\partial U_1}}{-\frac{\partial g_2}{\partial \theta}\Big/\frac{\partial g_2}{\partial U_2}} = -\frac{\partial \eta/\partial U_2}{\partial \eta/\partial U_1} = h(U_1, U_2). \tag{3.19}$$

Since g_1 is the function of U_1 and θ, and g_2 is the function of U_2 and θ, $h(U_1, U_2)$ should be separable, i.e. be the product of a function of U_1 and a function of U_2, $h(U_1, U_2)$ could be written as

$$h(U_1, U_2) = C_1(U_1)/C_2(U_2).$$

Thus (3.19) could be written as

$$\frac{\frac{\partial g_1}{\partial \theta}/\frac{\partial g_1}{\partial U_1}}{\frac{\partial g_2}{\partial \theta}/\frac{\partial g_2}{\partial U_2}} = \frac{C_1(U_1)}{C_2(U_2)} = \frac{C_1(U_1)/C(\theta)}{C_2(U_2)/C(\theta)}.$$

This gives us the following two partial differential equations:

$$\frac{\partial g_1}{\partial \theta}/\frac{\partial g_1}{\partial U_1} = \frac{C_1(U_1)}{C(\theta)},$$
$$\frac{\partial g_2}{\partial \theta}/\frac{\partial g_2}{\partial U_2} = \frac{C_2(U_2)}{C(\theta)}.$$

Rewrite as follows.

$$C(\theta)\frac{\partial g_1}{\partial \theta} - C_1(U_1)\frac{\partial g_1}{\partial U_1} = 0, \tag{3.20}$$
$$C(\theta)\frac{\partial g_2}{\partial \theta} - C_2(U_2)\frac{\partial g_2}{\partial U_2} = 0. \tag{3.21}$$
$$\tag{3.22}$$

We want to know under what condition g_1 and g_2 should meet.

Without loss of generality, consider the following equation first:

$$C(x)\frac{\partial f}{\partial x} - D(y)\frac{\partial f}{\partial y} = 0. \tag{3.23}$$

Our characteristic equations are given by

$$\frac{dx}{ds}(r, s) = C(x),$$
$$\frac{dy}{ds}(r, s) = -D(y),$$
$$\frac{dz}{ds}(r, s) = 0.$$

The solution of (3.23) is

$$f = F\left(\int^x \frac{1}{C(t)}dt + \int^y \frac{1}{D(t)}dt\right),$$

where F is any arbitrary function.

Based on this, the solutions to (3.20), (3.21) are as follows.

$$g_1 = G_1(\int^\theta \frac{1}{C(t)}dt + \int^{U_1} \frac{1}{C_1(t)}dt),$$
$$g_2 = G_2(\int^\theta \frac{1}{C(t)}dt + \int^{U_2} \frac{1}{C_2(t)}dt),$$

where G_1 and G_2 are arbitrary functions.

Consider new random variables and a new parameter given by

$$V_1 = \int^{U_1} \frac{1}{C_1(t)}dt \quad V_2 = \int^{U_2} \frac{1}{C_2(t)}dt \quad \delta = \int^\theta \frac{1}{C(t)}dt,$$

then

$$g_1 = G_1(V_1 + \delta) \quad g_2 = G_2(V_2 + \delta).$$

Thus θ is generalized location and scale parameter. ■

3.5.4 Proof of n observations with single parameter

Proof Based on the given association 3.11, we have

$$\frac{\partial X_j}{\partial \theta} = \frac{\partial g_j}{\partial \theta} + \frac{\partial g_j}{\partial U_j}\frac{\partial U_j}{\partial \theta} = 0.$$

So

$$\frac{\partial U_j}{\partial \theta} = -\frac{\partial g_j/\partial \theta}{\partial g_j/\partial U_j}.$$

In order to find the fully observed characteristics $\boldsymbol{\eta}(U_1, U_2, \cdots, U_n)$, it requires

$$\frac{\partial \eta_i}{\partial \theta} = \sum_{j=1}^n \frac{\partial \eta_i}{\partial U_j}\frac{\partial U_j}{\partial \theta} = -\sum_{j=1}^n \frac{\partial \eta_i}{\partial U_j}\frac{\partial g_j/\partial \theta}{\partial g_j/\partial U_j} = 0 \quad i = 1, \cdots, n-1. \tag{3.24}$$

We consider the matrix representation of the above system of equations, let

$$A = \begin{pmatrix} \frac{\partial \eta_1}{\partial U_1} & \frac{\partial \eta_1}{\partial U_2} & \cdots & \frac{\partial \eta_1}{\partial U_n} \\ \frac{\partial \eta_2}{\partial U_1} & \frac{\partial \eta_2}{\partial U_2} & \cdots & \frac{\partial \eta_2}{\partial U_n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \frac{\partial \eta_{n-1}}{\partial U_1} & \frac{\partial \eta_{n-1}}{\partial U_2} & \cdots & \frac{\partial \eta_{n-1}}{\partial U_n} \end{pmatrix}$$

and $\boldsymbol{Z} = (f_1(\theta, U_1), f_2(\theta, U_1), \cdots, f_n(\theta, U_1))^T$, where $f_j(\theta, U_j) = \dfrac{\partial g_j/\partial \theta}{\partial g_j/\partial U_j}$. Then 3.24 becomes

$$A\boldsymbol{Z} = 0.$$

Let A_i denote the matrix which is from deleting the i-th column of matrix A. Then one solution of $A\boldsymbol{Z} = 0$ is:

$$\boldsymbol{Z} = (det(A_1), -det(A_2), \cdots, (-1)^{n-1} det(A_n))^T.$$

And since all the η_i's should be linearly independent. The independency here means any η_i can not be a function of any other η_j's ($j \neq i$), and all the solutions actually form a one dimensional space. Thus

$$f_j(\theta, U_j) = (-1)^{j-1} c(\theta, U) det(A_j) \quad j = 1, \cdots, n,$$

where $c(\theta, U) = c(\theta, U_1, U_2, \cdots, U_n)$ and there exist $f(\theta)$ and $c(U)$ such that $c(\theta, U) = f(\theta)c(U)$. Otherwise, if θ and U is not separable in $c(\theta, U)$, suppose it contains a factor like $f(\theta, U_k)$, then since $det(A_j)$ is just function of U_i's, $f_j(\theta, U_j)$ will contains $f(\theta, U_k)$ for all $j \neq k$.

So we have

$$f_1(\theta, U_1) = f(\theta) f_1(U_1) \quad f_2(\theta, U_2) = f(\theta) f_2(U_2) \quad \cdots \quad f_n(\theta, U_n) = f(\theta) f_n(U_n),$$

where $f_j(U_j) = (-1)^{j-1} c(U) det(A_j)$.

Thus there are transformations such that θ is a generalized location parameter.

■

3.5.5 Proof for 3 observations and 2 parameters case

Proof We want to find one $\eta(U_1, U_2, U_3)$, which is fully observed and is constant with respect to θ, i.e.

$$\frac{\partial \eta}{\partial \theta_i} = 0 \quad i = 1, 2.$$

$$\begin{cases} \dfrac{\partial \eta}{\partial \theta_1} = -\dfrac{\partial \eta}{\partial U_1}\dfrac{a_{\theta_1}(\theta, U_1)}{a_U(\theta, U_1)} - \dfrac{\partial \eta}{\partial U_2}\dfrac{a_{\theta_1}(\theta, U_2)}{a_U(\theta, U_2)} - \dfrac{\partial \eta}{\partial U_3}\dfrac{a_{\theta_1}(\theta, U_3)}{a_U(\theta, U_3)} = 0 \\ \dfrac{\partial \eta}{\partial \theta_2} = -\dfrac{\partial \eta}{\partial U_1}\dfrac{a_{\theta_2}(\theta, U_1)}{a_U(\theta, U_1)} - \dfrac{\partial \eta}{\partial U_2}\dfrac{a_{\theta_2}(\theta, U_2)}{a_U(\theta, U_2)} - \dfrac{\partial \eta}{\partial U_3}\dfrac{a_{\theta_2}(\theta, U_3)}{a_U(\theta, U_3)} = 0 \end{cases} \tag{3.25}$$

Solve for $\dfrac{\partial \eta}{\partial U_3}$, we have

$$\begin{cases} \dfrac{\partial \eta}{\partial U_1}\dfrac{a_{\theta_1}(\theta, U_1)}{a_U(\theta, U_1)}\dfrac{a_U(\theta, U_3)}{a_{\theta_1}(\theta, U_3)} + \dfrac{\partial \eta}{\partial U_2}\dfrac{a_{\theta_1}(\theta, U_2)}{a_U(\theta, U_2)}\dfrac{a_U(\theta, U_3)}{a_{\theta_1}(\theta, U_3)} = -\dfrac{\partial \eta}{\partial U_3} \\ \dfrac{\partial \eta}{\partial U_1}\dfrac{a_{\theta_2}(\theta, U_1)}{a_U(\theta, U_1)}\dfrac{a_U(\theta, U_3)}{a_{\theta_2}(\theta, U_3)} + \dfrac{\partial \eta}{\partial U_2}\dfrac{a_{\theta_2}(\theta, U_2)}{a_U(\theta, U_2)}\dfrac{a_U(\theta, U_3)}{a_{\theta_2}(\theta, U_3)} = -\dfrac{\partial \eta}{\partial U_3} \end{cases} \tag{3.26}$$

Equate the left side of the two equations above, we have

$$\begin{aligned} \frac{\partial \eta / \partial U_1}{\partial \eta / \partial U_2} &= \frac{a_{\theta_1}(\theta, U_3)a_{\theta_2}(\theta, U_2) - a_{\theta_1}(\theta, U_2)a_{\theta_2}(\theta, U_3)}{a_{\theta_1}(\theta, U_1)a_{\theta_2}(\theta, U_3) - a_{\theta_1}(\theta, U_3)a_{\theta_2}(\theta, U_1)} \cdot \frac{a_U(\theta, U_1)}{a_U(\theta, U_2)} \\ &= \frac{F(\theta, U_2, U_3)}{F(\theta, U_3, U_1)}\frac{G(\theta, U_1)}{G(\theta, U_2)}. \end{aligned} \tag{3.27}$$

Since the left-hand side of (3.27) is free of θ, the non-separable factors of θ and U_i's in F must also be in G, so we have

$$F(\theta, U_2, U_3) = A(\theta, U_3)B(\theta, U_2)C_1(\theta)D_1(U_2, U_3), \tag{3.28}$$

$$G(\theta, U_2) = B(\theta, U_2)C_2(\theta)D_2(U_2). \tag{3.29}$$

$$\tag{3.30}$$

Similarly, we have

$$\begin{aligned} F(\theta, U_3, U_1) &= A(\theta, U_1)B(\theta, U_3)C_1(\theta)D_1(U_3, U_1), \\ G(\theta, U_1) &= A(\theta, U_1)C_3(\theta)D_3(U_1). \end{aligned}$$

where $A(\theta, U)$, and $B(\theta, U)$ are functions in which θ and U can not be separated. By comparing the form of G, we have

$$B(\theta, U) = A(\theta, U), \quad C_2(\theta) = C_3(\theta), \quad D_2(U) = D_3(U).$$

Thus

$$\begin{cases} a_{\theta_1}(\theta, U_3)a_{\theta_2}(\theta, U_2) - a_{\theta_1}(\theta, U_2)a_{\theta_2}(\theta, U_3) = A(\theta, U_2)A(\theta, U_3)C_1(\theta)D_1(U_2, U_3) \\ a_U(\theta, U_1) = A(\theta, U_1)C_2(\theta)D_2(U_1) \end{cases} \tag{3.31}$$

$$\begin{aligned} a_{\theta_1}(\theta, U_3)a_{\theta_2}(\theta, U_2) - a_{\theta_1}(\theta, U_2)a_{\theta_2}(\theta, U_3) &= \frac{a_U(\theta, U_2)}{C_2(\theta)D_2(U_2)}A(\theta, U_3)C_1(\theta)D_1(U_2, U_3) \\ &= a_U(\theta, U_2)A(\theta, U_3)C(\theta)D(U_2, U_3), \end{aligned} \tag{3.32}$$

where $D(U_2, U_3) = D_1(U_2, U_3)/D_2(U_2)$ and $C(\theta) = C_1(\theta)/C_2(\theta)$. In equation(3.32), without loss of generality, let $z = U_2$, $U_3 = 0$, $x = \theta_1$, $y = \theta_2$, then we have

$$a_x(x, y, 0)a_y(x, y, z) - a_y(x, y, 0)a_x(x, y, z) - A(x, y, 0)C(x, y)D(z, 0)a_z(x, y, z) = 0. \tag{3.33}$$

If we consider the equation of the form

$$A(x, y)\frac{\partial a}{\partial x} + B(x, y)\frac{\partial a}{\partial y} + C(x, y)D(z)\frac{\partial a}{\partial z} = 0, \tag{3.34}$$

then the solution must have the following form by characteristic method,

$$a(x, y, z) = F(f(x, y), \int \frac{C(x, y)}{A(x, y)}dx - \int \frac{1}{D(z)}dz). \tag{3.35}$$

This indicates that the original association a must have the form as

$$a(\theta, U) = F(f(\theta), \int \frac{C(\theta)}{A(\theta)}d\theta_1 - \int \frac{1}{D(U)}dU), \tag{3.36}$$

$\theta = (\theta_1, \theta_2)$ is generalized location and scale parameter. ■

3.5.6 Proof for n observations and 2 parameters case

Proof Let's consider n observations, and 2 parameters.

$$\begin{cases} X_1 & = a(\theta, U_1) \\ X_2 & = a(\theta, U_2) \\ \cdots & \\ X_n & = a(\theta, U_n) \end{cases} \tag{3.37}$$

where $\theta = (\theta_1, \theta_2)'$.

We want to find $\eta(U_1, U_2, \cdots, U_n) = (\eta_1, \eta_2, \cdots, \eta_{n-2})$, which is fully observed, i.e.

$$\begin{cases} \frac{\partial \eta_1}{\partial \theta_1} = -\frac{\partial \eta_1}{\partial U_1}\frac{a_{\theta_1}(\theta, U_1)}{a_U(\theta, U_1)} - \frac{\partial \eta_1}{\partial U_2}\frac{a_{\theta_1}(\theta, U_2)}{a_U(\theta, U_2)} - \cdots - \frac{\partial \eta_1}{\partial U_n}\frac{a_{\theta_1}(\theta, U_n)}{a_U(\theta, U_n)} = 0 \\ \frac{\partial \eta_1}{\partial \theta_2} = -\frac{\partial \eta_1}{\partial U_1}\frac{a_{\theta_2}(\theta, U_1)}{a_U(\theta, U_1)} - \frac{\partial \eta_1}{\partial U_2}\frac{a_{\theta_2}(\theta, U_2)}{a_U(\theta, U_2)} - \cdots - \frac{\partial \eta_1}{\partial U_n}\frac{a_{\theta_2}(\theta, U_n)}{a_U(\theta, U_n)} = 0 \\ \cdots \\ \frac{\partial \eta_{n-2}}{\partial \theta_1} = -\frac{\partial \eta_{n-2}}{\partial U_1}\frac{a_{\theta_1}(\theta, U_1)}{a_U(\theta, U_1)} - \frac{\partial \eta_{n-2}}{\partial U_2}\frac{a_{\theta_1}(\theta, U_2)}{a_U(\theta, U_2)} - \cdots - \frac{\partial \eta_{n-2}}{\partial U_n}\frac{a_{\theta_1}(\theta, U_n)}{a_U(\theta, U_n)} = 0 \\ \frac{\partial \eta_{n-2}}{\partial \theta_2} = -\frac{\partial \eta_{n-2}}{\partial U_1}\frac{a_{\theta_2}(\theta, U_1)}{a_U(\theta, U_1)} - \frac{\partial \eta_{n-2}}{\partial U_2}\frac{a_{\theta_2}(\theta, U_2)}{a_U(\theta, U_2)} - \cdots - \frac{\partial \eta_{n-2}}{\partial U_n}\frac{a_{\theta_2}(\theta, U_n)}{a_U(\theta, U_n)} = 0 \end{cases} \tag{3.38}$$

We can rewrite this in matrix form as follows,

$$\nabla \eta \cdot f = 0 \tag{3.39}$$

$$\nabla \eta \cdot g = 0 \tag{3.40}$$

where $\eta = (\eta_1, \eta_2, \cdots, \eta_{n-2})$, $U = (U_1, U_2, \cdots, U_n)$.

$$\nabla \eta = \frac{\partial \eta}{\partial U} = \begin{pmatrix} \frac{\partial \eta_1}{\partial U_1} & \frac{\partial \eta_1}{\partial U_2} & \cdots & \frac{\partial \eta_1}{\partial U_n} \\ \frac{\partial \eta_2}{\partial U_1} & \frac{\partial \eta_2}{\partial U_2} & \cdots & \frac{\partial \eta_2}{\partial U_n} \\ \cdot & \cdot & & \cdot \\ & & & \\ \cdot & \cdot & & \cdot \\ \frac{\partial \eta_{n-2}}{\partial U_1} & \frac{\partial \eta_{n-2}}{\partial U_2} & \cdots & \frac{\partial \eta_{n-2}}{\partial U_n} \end{pmatrix}$$

$f = (f_1, f_2, \cdots, f_n)$, $f_i = \frac{a_{\theta_1}(\theta, U_1)}{a_U(\theta, U_i)}$, $g = (g_1, g_2, \cdots, g_n)$, and $g_i = \frac{a_{\theta_2}(\theta, U_1)}{a_U(\theta, U_i)}$. Let $U^* = (U_1, U_2, \cdots, U_{n-2})$, $U^{**} = (U_{n-1}, U_n)$, then

$$\nabla \eta \cdot f = \frac{\partial \eta}{\partial U} \cdot f = \begin{pmatrix} \frac{\partial \eta}{\partial U^*} & \frac{\partial \eta}{\partial U^{**}}. \end{pmatrix} \cdot f = 0$$

Since all the η_i's are independent, by Theorem 3.5.2, we know that $\frac{\partial \eta}{\partial U^*}$ is non-singular.

$$(\frac{\partial \eta}{\partial U^*})^{-1} \begin{pmatrix} \frac{\partial \eta}{\partial U^*} & \frac{\partial \eta}{\partial U^{**}} \end{pmatrix} \cdot f = 0$$

$$\begin{pmatrix} I_{n-1} & (\frac{\partial \eta}{\partial U^*})^{-1}\frac{\partial \eta}{\partial U^{**}} \end{pmatrix} \cdot f = 0$$

$$\begin{pmatrix} f_1 \\ f_2 \\ . \\ . \\ f_{n-2} \end{pmatrix} = -[(\frac{\partial W}{\partial U^*})^{-1}\frac{\partial W}{\partial U^{**}}] \begin{pmatrix} f_{n-1} \\ f_n \end{pmatrix} = A \cdot \begin{pmatrix} f_{n-1} \\ f_n \end{pmatrix}$$

We can rewrite the above system equations by considering a_{ij} as variables:

$$\begin{pmatrix} f_{n-1} & f_n & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ g_{n-1} & g_n & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & f_{n-1} & f_n & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & g_{n-1} & g_n & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & f_{n-1} & f_n & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & g_{n-1} & g_n & \cdots & 0 & 0 \\ . & . & . & . & . & . & & . & . \\ . & . & . & . & . & . & & . & . \\ . & . & . & . & . & . & & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & f_{n-1} & f_n \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & g_{n-1} & g_n \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ a_{31} \\ a_{32} \\ . \\ . \\ . \\ a_{n-2,1} \\ a_{n-2,2} \end{pmatrix} = \begin{pmatrix} f_1 \\ g_1 \\ f_2 \\ g_2 \\ f_3 \\ g_3 \\ . \\ . \\ . \\ f_{n-2} \\ g_{n-2} \end{pmatrix}$$

$$\begin{pmatrix} f_{n-1} & f_n \\ g_{n-1} & g_n \end{pmatrix} \begin{pmatrix} a_{i1} \\ a_{i2} \end{pmatrix} = \begin{pmatrix} f_i \\ g_i \end{pmatrix} \quad i = 1, 2, \cdots, n-2$$

By solving this system of equations, we have

$$\begin{aligned} a_{i1} &= \frac{f_i g_n - f_n g_i}{f_{n-1} g_n - f_n g_{n-1}} \\ &= \frac{a_{\theta_1}(\theta, U_i) a_{\theta_2}(\theta, U_n) - a_{\theta_1}(\theta, U_n) a_{\theta_2}(\theta, U_i)}{a_{\theta_1}(\theta, U_{n-1}) a_{\theta_2}(\theta, U_n) - a_{\theta_1}(\theta, U_n) a_{\theta_2}(\theta, U_{n-1})} \cdot \frac{a_U(\theta, U_{n-1})}{a_U(\theta, U_i)} \\ &= \frac{F(\theta, U_i, U_n)}{F(\theta, U_{n-1}, U_n)} \frac{G(\theta, U_{n-1})}{G(\theta, U_i)} \end{aligned}$$

Since a_{i1} is just a function of U_k's and free of θ, the non-separable factors of θ and U_i's in F must also be in G in order to cancel, so we have

$$F(\theta, U_i, U_n) = A(\theta, U_n)B(\theta, U_i)C_1(\theta)D_1(U_i, U_n) \tag{3.41}$$

$$G(\theta, U_i) = B(\theta, U_i)C_2(\theta)D_2(U_i) \tag{3.42}$$

$$\tag{3.43}$$

Similarly, we have

$$F(\theta, U_{n-1}, U_n) = A(\theta, U_n)B(\theta, U_{n-1})C_1(\theta)D_1(U_{n-1}, U_n)$$
$$G(\theta, U_{n-1}) = B(\theta, U_{n-1})C_3(\theta)D_3(U_{n-1})$$

where $A(\theta, U)$, and $B(\theta, U)$ are functions in which θ and U can not be separated. By comparing the form of G, we have

$$B(\theta, U) = A(\theta, U), \quad C_2(\theta) = C_3(\theta), \quad D_2(U) = D_3(U).$$

Thus

$$\begin{cases} a_{\theta_1}(\theta, U_i)a_{\theta_2}(\theta, U_n) - a_{\theta_1}(\theta, U_n)a_{\theta_2}(\theta, U_i) = A(\theta, U_n)B(\theta, U_i)C_1(\theta)D_1(U_i, U_n) \\ a_U(\theta, U_n) = B(\theta, U_n)C_2(\theta)D_2(U_n) \end{cases} \tag{3.44}$$

$$\begin{aligned} a_{\theta_1}(\theta, U_i)a_{\theta_2}(\theta, U_n) - a_{\theta_1}(\theta, U_n)a_{\theta_2}(\theta, U_i) &= \frac{a_U(\theta, U_i)}{C_2(\theta)D_2(U_i)}A(\theta, U_n)C_1(\theta)D_1(U_i, U_n) \\ &= a_U(\theta, U_i)A(\theta, U_n)C(\theta)D(U_i, U_n) \end{aligned} \tag{3.45}$$

where $D(U_i, U_n) = D_1(U_i, U_n)/D_2(U_i)$ and $C(\theta) = C_1(\theta)/C_2(\theta)$. Without loss of generality, let $z = U_i$, $U_n = 0$, $x = \theta_1$, $y = \theta_2$, then we have

$$a_x(x, y, z)a_y(x, y, 0) - a_y(x, y, z)a_x(x, y, 0) - A(x, y, 0)C(x, y)D(z, 0)a_z(x, y, z) = 0. \tag{3.46}$$

If we consider the equation of the form

$$A(x, y)\frac{\partial a}{\partial x} + B(x, y)\frac{\partial a}{\partial y} + C(x, y)D(z)\frac{\partial a}{\partial z} = 0, \tag{3.47}$$

then the solution must have the following form by characteristic method,

$$a(x,y,z) = F(f(x,y), \int \frac{C(x,y)}{A(x,y)}dx - \int \frac{1}{D(z)}dz). \tag{3.48}$$

This means that the original association a must have the form as

$$a(\theta, U) = F(f(\theta), \int \frac{C(\theta)}{A(\theta)}d\theta_1 - \int \frac{1}{D(U)}dU), \tag{3.49}$$

$\theta = (\theta_1, \theta_2)$ is generalized location and scale parameter.

■

3.5.7 Independence theorem proof

Theorem 3.5.2 *If two functions' gradients are parallel, then they are not independent, one is the function of the other.*

Proof

$$f_x(x,y) = a(x,y)g_x(x,y)$$
$$f_y(x,y) = a(x,y)g_y(x,y)$$

This is equivalent to

$$\frac{f_x}{f_y} = \frac{g_x}{g_y}.$$

In order to have $f_{xy} = f_{yx}$, this must be true:

$$f_{xy}(x,y) = a_y(x,y)g_x(x,y) + a(x,y)g_{xy}(x,y),$$
$$f_{yx}(x,y) = a_x(x,y)g_y(x,y) + a(x,y)g_{yx}(x,y).$$

Thus

$$a_y(x,y)g_x(x,y) = a_x(x,y)g_y(x,y).$$

This is saying that

$$\frac{f_x}{f_y} = \frac{g_x}{g_y} = \frac{a_x}{a_y}.$$

Let's assume that $(x(t), y(t))$ is contour line of $f(x, y)$, i.e. a curve along which the function has a constant value. Then we have:

$$f(x(t), y(t)) = C \quad t \in A,$$
$$(f_x, f_y) \cdot (x_t, y_t)' = 0,$$

$G(t) = g(x(t), y_t)$, then we have

$$a(x, y)G'(t) = a(x, y)(g_x, g_y) \cdot (x_t, y_t)' = (f_x, f_y) \cdot (x_t, y_t)' = 0.$$

When $a(x, y) \neq 0$, we have $G'(t) = 0$, which means that $f(x, y)$ and $g(x, y)$ have same contours. Thus we can consider f is a function of g. ■

3.5.8 Proof for 4 observations and 3 parameters case

Proof We want to find $\eta = \eta(U_1, U_2, U_3, U_4)$, which is fully observed, i.e.

$$\begin{cases} \dfrac{\partial \eta}{\partial \theta_1} = -\dfrac{\partial \eta}{\partial U_1}\dfrac{a_{\theta_1}(\theta, U_1)}{a_U(\theta, U_1)} - \dfrac{\partial \eta}{\partial U_2}\dfrac{a_{\theta_1}(\theta, U_2)}{a_U(\theta, U_2)} - \dfrac{\partial \eta}{\partial U_3}\dfrac{a_{\theta_1}(\theta, U_3)}{a_U(\theta, U_3)} - \dfrac{\partial \eta}{\partial U_4}\dfrac{a_{\theta_1}(\theta, U_4)}{a_U(\theta, U_4)} = 0 \\ \dfrac{\partial \eta}{\partial \theta_2} = -\dfrac{\partial \eta}{\partial U_1}\dfrac{a_{\theta_2}(\theta, U_1)}{a_U(\theta, U_1)} - \dfrac{\partial \eta}{\partial U_2}\dfrac{a_{\theta_2}(\theta, U_2)}{a_U(\theta, U_2)} - \dfrac{\partial \eta}{\partial U_3}\dfrac{a_{\theta_2}(\theta, U_3)}{a_U(\theta, U_3)} - \dfrac{\partial \eta}{\partial U_4}\dfrac{a_{\theta_2}(\theta, U_4)}{a_U(\theta, U_4)} = 0 \\ \dfrac{\partial \eta}{\partial \theta_3} = -\dfrac{\partial \eta}{\partial U_1}\dfrac{a_{\theta_3}(\theta, U_1)}{a_U(\theta, U_1)} - \dfrac{\partial \eta}{\partial U_2}\dfrac{a_{\theta_3}(\theta, U_2)}{a_U(\theta, U_2)} - \dfrac{\partial \eta}{\partial U_3}\dfrac{a_{\theta_3}(\theta, U_3)}{a_U(\theta, U_3)} - \dfrac{\partial \eta}{\partial U_4}\dfrac{a_{\theta_3}(\theta, U_4)}{a_U(\theta, U_4)} = 0 \end{cases} \tag{3.50}$$

Let

$$A = \begin{pmatrix} \dfrac{a_{\theta_1}(\theta, U_1)}{a_U(\theta, U_1)} & \dfrac{a_{\theta_1}(\theta, U_2)}{a_U(\theta, U_2)} & \dfrac{a_{\theta_1}(\theta, U_3)}{a_U(\theta, U_3)} & \dfrac{a_{\theta_1}(\theta, U_4)}{a_U(\theta, U_4)} \\ \dfrac{a_{\theta_2}(\theta, U_1)}{a_U(\theta, U_1)} & \dfrac{a_{\theta_2}(\theta, U_2)}{a_U(\theta, U_2)} & \dfrac{a_{\theta_2}(\theta, U_3)}{a_U(\theta, U_3)} & \dfrac{a_{\theta_2}(\theta, U_4)}{a_U(\theta, U_4)} \\ \dfrac{a_{\theta_3}(\theta, U_1)}{a_U(\theta, U_1)} & \dfrac{a_{\theta_3}(\theta, U_2)}{a_U(\theta, U_2)} & \dfrac{a_{\theta_3}(\theta, U_3)}{a_U(\theta, U_3)} & \dfrac{a_{\theta_3}(\theta, U_4)}{a_U(\theta, U_4)} \end{pmatrix}$$
$$= \frac{1}{a_U(\theta, U_1)a_U(\theta, U_2)a_U(\theta, U_3)a_U(\theta, U_4)} \begin{pmatrix} a_{\theta_1}(\theta, U_1) & a_{\theta_1}(\theta, U_2) & a_{\theta_1}(\theta, U_3) & a_{\theta_1}(\theta, U_4) \\ a_{\theta_2}(\theta, U_1) & a_{\theta_2}(\theta, U_2) & a_{\theta_2}(\theta, U_3) & a_{\theta_1}(\theta, U_4) \\ a_{\theta_3}(\theta, U_1) & a_{\theta_3}(\theta, U_2) & a_{\theta_3}(\theta, U_3) & a_{\theta_1}(\theta, U_4) \end{pmatrix}$$

First if Rank(A)=2, then $\theta_1, \theta_2, \theta_3$ are degenerated.

Without loss of generality, we can assume that

$$\begin{vmatrix} a_{\theta_1}(\theta, U_1) & a_{\theta_1}(\theta, U_2) & a_{\theta_1}(\theta, U_3) \\ a_{\theta_2}(\theta, U_1) & a_{\theta_2}(\theta, U_2) & a_{\theta_2}(\theta, U_3) \\ a_{\theta_3}(\theta, U_1) & a_{\theta_3}(\theta, U_2) & a_{\theta_3}(\theta, U_3) \end{vmatrix} = 0$$

There exist constant c_1, c_2 such that

$$\begin{cases} c_1 a_{\theta_1}(\theta, U_1) + c_2 a_{\theta_2}(\theta, U_1) = a_{\theta_3}(\theta, U_1) \\ c_1 a_{\theta_1}(\theta, U_2) + c_2 a_{\theta_2}(\theta, U_2) = a_{\theta_3}(\theta, U_2) \\ c_1 a_{\theta_1}(\theta, U_3) + c_2 a_{\theta_2}(\theta, U_3) = a_{\theta_3}(\theta, U_3) \end{cases}$$

Assume

$$\begin{vmatrix} a_{\theta_1}(\theta, U_1) & a_{\theta_1}(\theta, U_2) \\ a_{\theta_2}(\theta, U_1) & a_{\theta_2}(\theta, U_2) \end{vmatrix} \neq 0,$$

then c_1, c_2 are uniquely determined. Let u_3 goes through from $-\infty$ to $+\infty$, and we have

$$c_1 a_{\theta_1}(\theta, U) + c_2 a_{\theta_2}(\theta, U) = a_{\theta_3}(\theta, U) \quad U \in \mathbb{R}.$$

The solution of above differential equation is

$$a = F(\frac{\theta_1}{c_1} - \frac{\theta_2}{c_2}, \frac{\theta_2}{c_2} + \theta_3),$$

where F is arbitrary differentiable function.

Now let's assume that Rank(A)=3 at point $\omega^* = (U_1, U_2, U_3, U_4, \theta_1, \theta_2, \theta_3) \in \mathbb{R}^7$. By continuity, we can assume that there's a neighborhood of ω^*, let's denote that as Ω. Without loss of generality, let's assume that

$$\begin{vmatrix} a_{\theta_1}(\theta, U_1) & a_{\theta_1}(\theta, U_2) & a_{\theta_1}(\theta, U_3) \\ a_{\theta_2}(\theta, U_1) & a_{\theta_2}(\theta, U_2) & a_{\theta_2}(\theta, U_3) \\ a_{\theta_3}(\theta, U_1) & a_{\theta_3}(\theta, U_2) & a_{\theta_3}(\theta, U_3) \end{vmatrix} \neq 0.$$

Then if η exists, it satisfies

$$\nabla\eta \parallel \begin{vmatrix} e_1 & e_2 & e_3 & e_4 \\ \frac{\partial U_1}{\partial \theta_1} & \frac{\partial U_2}{\partial \theta_1} & \frac{\partial U_3}{\partial \theta_1} & \frac{\partial U_4}{\partial \theta_1} \\ \frac{\partial U_1}{\partial \theta_2} & \frac{\partial U_2}{\partial \theta_2} & \frac{\partial U_3}{\partial \theta_2} & \frac{\partial U_4}{\partial \theta_2} \\ \frac{\partial U_1}{\partial \theta_3} & \frac{\partial U_2}{\partial \theta_3} & \frac{\partial U_3}{\partial \theta_3} & \frac{\partial U_4}{\partial \theta_3} \end{vmatrix}$$

$$\begin{aligned} \frac{\partial\eta/\partial U_1}{\partial\eta/\partial U_4} &= -\frac{\begin{vmatrix} a_{\theta_1}(\theta,U_2) & a_{\theta_1}(\theta,U_3) & a_{\theta_1}(\theta,U_4) \\ a_{\theta_2}(\theta,U_2) & a_{\theta_2}(\theta,U_3) & a_{\theta_2}(\theta,U_4) \\ a_{\theta_3}(\theta,U_2) & a_{\theta_3}(\theta,U_3) & a_{\theta_3}(\theta,U_4) \end{vmatrix}}{\begin{vmatrix} a_{\theta_1}(\theta,U_1) & a_{\theta_1}(\theta,U_2) & a_{\theta_1}(\theta,U_3) \\ a_{\theta_2}(\theta,U_1) & a_{\theta_2}(\theta,U_2) & a_{\theta_2}(\theta,U_3) \\ a_{\theta_3}(\theta,U_1) & a_{\theta_3}(\theta,U_2) & a_{\theta_3}(\theta,U_3) \end{vmatrix}} \cdot \frac{a_U(\theta,U_1)}{a_U(\theta,U_4)} \\ &= -\frac{f_1(\theta)f_2(\theta,U_2)f_2(\theta,U_3)f_2(\theta,U_4)}{f_1(\theta)f_2(\theta,U_1)f_2(\theta,U_2)f_2(\theta,U_3)} \cdot \frac{g_1(\theta)g_2(U_1)g_3(\theta,U_1)}{g_1(\theta)g_2(U_4)g_3(\theta,U_4)} \end{aligned}$$

where f_2, g_3 are all non-seperatable functions.

Since the left hand side of the above equation is a function of U_1, U_2, U_3, U_4, then we have

$$f_2(\theta,U_1) = g_3(\theta,U_1) = \frac{a_U(\theta,U_1)}{g_1(\theta)g_2(U_1)}$$

$$\begin{vmatrix} a_{\theta_1}(\theta,U_1) & a_{\theta_1}(\theta,U_2) & a_{\theta_1}(\theta,U_3) \\ a_{\theta_2}(\theta,U_1) & a_{\theta_2}(\theta,U_2) & a_{\theta_2}(\theta,U_3) \\ a_{\theta_3}(\theta,U_1) & a_{\theta_3}(\theta,U_2) & a_{\theta_3}(\theta,U_3) \end{vmatrix} = \frac{f_1(\theta)}{g_1(\theta)^3} \cdot a_U(\theta,U_1)a_U(\theta,U_2)a_U(\theta,U_3) \cdot \frac{1}{g_2(U_1)g_2(U_2)g_2(U_3)}$$

$$\begin{vmatrix} g^*(U_1)f^*(\theta)\frac{a_{\theta_1}(\theta,U_1)}{a_U(\theta,U_1)} & g^*(U_2)f^*(\theta)\frac{a_{\theta_1}(\theta,U_2)}{a_U(\theta,U_2)} & g^*(U_3)f^*(\theta)\frac{a_{\theta_1}(\theta,U_3)}{a_U(\theta,U_3)} \\ g^*(U_1)f^*(\theta)\frac{a_{\theta_2}(\theta,U_1)}{a_U(\theta,U_1)} & g^*(U_2)f^*(\theta)\frac{a_{\theta_2}(\theta,U_2)}{a_U(\theta,U_2)} & g^*(U_3)f^*(\theta)\frac{a_{\theta_2}(\theta,U_3)}{a_U(\theta,U_3)} \\ g^*(U_1)f^*(\theta)\frac{a_{\theta_3}(\theta,U_1)}{a_U(\theta,U_1)} & g^*(U_2)f^*(\theta)\frac{a_{\theta_3}(\theta,U_2)}{a_U(\theta,U_2)} & g^*(U_3)f^*(\theta)\frac{a_{\theta_3}(\theta,U_3)}{a_U(\theta,U_3)} \end{vmatrix} = 1$$

where $g^*(U) = g_2(U)$ and $f^*(\theta) = \dfrac{g_1(\theta)}{f_1^{1/3}(\theta)}$. According to Lemma 1, the above equation could not be true. So if η exists, rank of matrix A can not be 3.

Exactly same proof could be extended to n observations, where $n \geq 4$.

■

Lemma 1 *Consider functions $f(x), g(x), h(x)$, which are not all equal, the following is impossible.*

$$\begin{vmatrix} f(x_1) & f(x_2) & f(x_3) \\ g(x_1) & g(x_2) & g(x_3) \\ h(x_1) & h(x_2) & h(x_3) \end{vmatrix} \equiv 1$$

of a neighborhood Ω of (x_1, x_2, x_3).

Proof Let's take derivative with respect to x_1,

$$\begin{vmatrix} f'(x_1) & f(x_2) & f(x_3) \\ g'(x_1) & g(x_2) & g(x_3) \\ h'(x_1) & h(x_2) & h(x_3) \end{vmatrix} = 0.$$

Continue taking derivatives with respect to x_2 and x_3, we have

$$\begin{vmatrix} f'(x_1) & f'(x_2) & f'(x_3) \\ g'(x_1) & g'(x_2) & g'(x_3) \\ h'(x_1) & h'(x_2) & h'(x_3) \end{vmatrix} = 0.$$

So we have $f'(x), g'(x), h'(x)$ are linearly dependent. Without loss of generality, there exists non-zero constants c_1, c_2 such that

$$f'(x) + c_1 g'(x) + c_2 h'(x) = 0.$$

Integrating the above equation with respect to x, we have

$$f(x) + c_1 g(x) + c_2 h(x) + c_3 = 0.$$

$$0 = \begin{vmatrix} f(x_1) + c_3 & f(x_2) + c_3 & f(x_3) + c_3 \\ g(x_1) & g(x_2) & g(x_3) \\ h(x_1) & h(x_2) & h(x_3) \end{vmatrix} = \begin{vmatrix} f(x_1) & f(x_2) & f(x_3) \\ g(x_1) & g(x_2) & g(x_3) \\ h(x_1) & h(x_2) & h(x_3) \end{vmatrix} + \begin{vmatrix} c_3 & c_3 & c_3 \\ g(x_1) & g(x_2) & g(x_3) \\ h(x_1) & h(x_2) & h(x_3) \end{vmatrix}$$

$$\begin{vmatrix} c_3 & c_3 & c_3 \\ g(x_1) & g(x_2) & g(x_3) \\ h(x_1) & h(x_2) & h(x_3) \end{vmatrix} = -1.$$

Repeating the above steps, we can easily get

$$\begin{vmatrix} c_3 & c_3 & c_3 \\ c_4 & c_4 & c_4 \\ h(x_1) & h(x_2) & h(x_3) \end{vmatrix} = 1.$$

which can not be true and result a contradiction. So we conclude that $f(x), g(x), h(x)$ do not exist.

■

4. FUTURE WORK

4.1 Future Research Topics for Modeling Chromosome Structures Using Hi-C data

Hi-C technology substantially advances our understanding of spatial organizations of chromosomes and their implications on genomic functions. However, appropriate statistical models for analyzing Hi-C data and inferring chromatin folding are still lacking. We proposed a piecewise helical model with mixture extension to reconstruct 3D chromosomal structure not only within each topologically associated domain, but also for the whole chromosome. Compared with existing approaches with over-parameterized beads on-a-string representation, the piecewise helical model has the following four advantages: (i) it is parsimonious, yet captures key features of the 3D chromosomal structure; (ii) unknown parameters—curvature and torsion—have clear geometric interpretation; (iii) it is straightforward to incorporate the genomic distance into the arc length of the piecewise helical model, so that the inferred 3D chromosomal structures are robust to outliers, and show high reproducibility between different samples; and (iv) the computational cost for the parsimonious model is significantly lower than other complex models. Additionally, the simplicity of the piecewise helical model makes it affordable to employ a computationally more demanding, yet much more suitable, negative binomial regression model to link the spatial distance with Hi-C data.

We noticed a few limitations of the piecewise helical model. First, we use BIC to select the number of helixes, but assume that all helixes within the same helical curve are of equal size. In theory, we can treat the helix boundary points as unknown parameters. Statistical inference of the number of helixes and helix boundary points can be formulated into the change point estimation problem. However, such model is

over-parameterized, and results are usually unstable. Second, in the piecewise helical model, we use piecewise constant functions to model curvature and torsion changes. But for an arbitrary 3D curve, they can be any continuous functions. It is possible to model curvature and torsion functions by smoothing spline functions, which may result in a much higher computational cost. Modeling 3D chromosomal structure via an arbitrary continuous curve clearly deserves further effort in the future research.

Although our work is designed for the modeling the 3D structure of chromosomes, the idea of modeling a 3D curve as piecewise helicel curve can be extended to approximate any curves. Piecewise helical curve approximation of 3D curves enjoys not only the advantages of simplicity and easy fitting, but also the advantages of interpretability. It is worthwhile to investigate more applications of this curve fitting method.

4.2 Future Research Topics for Deep Neural Network based Automated Statistical Analysis

In this subsection, we will briefly discuss several future work directions for the proposed automated neural network based framework for statistical analysis.

Simulation study have showed that proposed Bayes neural estimator and neural selector and estimator can be properly trained with simulated labeled data, and further demonstrate excellent performance. We consider this work a demonstration of the validity of our grand proposal that is to use DNNs to automate the entire statistical analysis process. There remains a lot of work we need to do before the grand proposal can be finally materialized.

First, we will extend the neural model selector and parameter estimator to models with multiple parameters as well as regression models involving a large number of explanatory variables. Second, we will investigate how CNNs or other DNNs can be used to automate other tasks such as hypotheses testing and diagnostics of the SA process in the near future. Our ultimate goal to develop AI systems or software

that can conduct principled SA for big data analytics without the need of human interventions.

REFERENCES

REFERENCES

[1] J. Dekker, "Gene regulation in the third dimension," *Science*, vol. 319, no. 5871, pp. 1793–1794, 2008.

[2] C. Lanctôt, T. Cheutin, M. Cremer, G. Cavalli, and T. Cremer, "Dynamic genome architecture in the nuclear space: regulation of gene expression in three dimensions," *Nature Reviews Genetics*, vol. 8, no. 2, p. 104, 2007.

[3] J. Dekker, K. Rippe, M. Dekker, and N. Kleckner, "Capturing chromosome conformation," *science*, vol. 295, no. 5558, pp. 1306–1311, 2002.

[4] E. Lieberman-Aiden, N. L. Van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B. R. Lajoie, P. J. Sabo, M. O. Dorschner *et al.*, "Comprehensive mapping of long-range interactions reveals folding principles of the human genome," *science*, vol. 326, no. 5950, pp. 289–293, 2009.

[5] M. J. Fullwood, M. H. Liu, Y. F. Pan, J. Liu, H. Xu, Y. B. Mohamed, Y. L. Orlov, S. Velkov, A. Ho, P. H. Mei *et al.*, "An oestrogen-receptor-α-bound human chromatin interactome," *Nature*, vol. 462, no. 7269, p. 58, 2009.

[6] R. Kalhor, H. Tjong, N. Jayathilaka, F. Alber, and L. Chen, "Genome architectures revealed by tethered chromosome conformation capture and population-based modeling," *Nature biotechnology*, vol. 30, no. 1, p. 90, 2012.

[7] T. Nagano, Y. Lubling, T. J. Stevens, S. Schoenfelder, E. Yaffe, W. Dean, E. D. Laue, A. Tanay, and P. Fraser, "Single-cell hi-c reveals cell-to-cell variability in chromosome structure," *Nature*, vol. 502, no. 7469, p. 59, 2013.

[8] J. Dekker, M. A. Marti-Renom, and L. A. Mirny, "Exploring the three-dimensional organization of genomes: interpreting chromatin interaction data," *Nature Reviews Genetics*, vol. 14, no. 6, p. 390, 2013.

[9] A. D. Schmitt, M. Hu, and B. Ren, "Genome-wide mapping and analysis of chromosome architecture," *Nature reviews Molecular cell biology*, vol. 17, no. 12, p. 743, 2016.

[10] J. Mateos-Langerak, M. Bohn, W. de Leeuw, O. Giromus, E. M. Manders, P. J. Verschure, M. H. Indemans, H. J. Gierman, D. W. Heermann, R. Van Driel *et al.*, "Spatially confined folding of chromatin in the interphase nucleus," *Proceedings of the National Academy of Sciences*, vol. 106, no. 10, pp. 3812–3817, 2009.

[11] M. Bohn and D. W. Heermann, "Diffusion-driven looping provides a consistent framework for chromatin organization," *PloS one*, vol. 5, no. 8, p. e12218, 2010.

[12] M. Barbieri, M. Chotalia, J. Fraser, L.-M. Lavitas, J. Dostie, A. Pombo, and M. Nicodemi, "Complexity of chromatin folding is captured by the strings and binders switch model," *Proceedings of the National Academy of Sciences*, vol. 109, no. 40, pp. 16 173–16 178, 2012.

[13] M. Barbieri, J. Fraser, L.-M. Lavitas, M. Chotalia, J. Dostie, A. Pombo, and M. Nicodemi, "A polymer model explains the complexity of large-scale chromatin folding," *Nucleus*, vol. 4, no. 4, pp. 267–273, 2013.

[14] A. Lesne, J. Riposo, P. Roger, A. Cournac, and J. Mozziconacci, "3d genome reconstruction from chromosomal contacts," *Nature methods*, vol. 11, no. 11, p. 1141, 2014.

[15] Z. Zhang, G. Li, K.-C. Toh, and W.-K. Sung, "3d chromosome modeling with semi-definite programming and hi-c data," *Journal of computational biology*, vol. 20, no. 11, pp. 831–846, 2013.

[16] M. Rousseau, J. Fraser, M. A. Ferraiuolo, J. Dostie, and M. Blanchette, "Three-dimensional modeling of chromatin structure from interaction frequency data using markov chain monte carlo sampling," *BMC bioinformatics*, vol. 12, no. 1, p. 414, 2011.

[17] M. Hu, K. Deng, Z. Qin, J. Dixon, S. Selvaraj, J. Fang, B. Ren, and J. S. Liu, "Bayesian inference of spatial organizations of chromosomes," *PLoS computational biology*, vol. 9, no. 1, p. e1002893, 2013.

[18] A. C. Hausrath and A. Goriely, "Repeat protein architectures predicted by a continuum representation of fold space," *Protein Science*, vol. 15, no. 4, pp. 753–760, 2006.

[19] A. Hausrath and A. Goriely, "Continuous representations of proteins: Construction of coordinate models from curvature profiles," *Journal of structural biology*, vol. 158, no. 3, pp. 267–281, 2007.

[20] G. Xiao, X. Wang, and A. B. Khodursky, "Modeling three-dimensional chromosome structures using gene expression data," *Journal of the American Statistical Association*, vol. 106, no. 493, pp. 61–72, 2011.

[21] N. Varoquaux, F. Ay, W. S. Noble, and J.-P. Vert, "A statistical approach for inferring the 3d structure of the genome," *Bioinformatics*, vol. 30, no. 12, pp. i26–i33, 2014.

[22] J. R. Dixon, S. Selvaraj, F. Yue, A. Kim, Y. Li, Y. Shen, M. Hu, J. S. Liu, and B. Ren, "Topological domains in mammalian genomes identified by analysis of chromatin interactions," *Nature*, vol. 485, no. 7398, p. 376, 2012.

[23] S. S. Rao, M. H. Huntley, N. C. Durand, E. K. Stamenova, I. D. Bochkov, J. T. Robinson, A. L. Sanborn, I. Machol, A. D. Omer, E. S. Lander *et al.*, "A 3d map of the human genome at kilobase resolution reveals principles of chromatin looping," *Cell*, vol. 159, no. 7, pp. 1665–1680, 2014.

[24] J. R. Dixon, I. Jung, S. Selvaraj, Y. Shen, J. E. Antosiewicz-Bourget, A. Y. Lee, Z. Ye, A. Kim, N. Rajagopal, W. Xie *et al.*, "Chromatin architecture reorganization during stem cell differentiation," *Nature*, vol. 518, no. 7539, p. 331, 2015.

[25] H. W. Guggenheimer, *Differential Geometry*. University Microfilms, 1963.

[26] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[27] J. S. Liu, *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[28] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. CRC press, 2013.

[29] J. H. Bullard, E. Purdom, K. D. Hansen, and S. Dudoit, "Evaluation of statistical methods for normalization and differential expression in mrna-seq experiments," *BMC bioinformatics*, vol. 11, no. 1, p. 94, 2010.

[30] Z. S. Qin, J. Yu, J. Shen, C. A. Maher, M. Hu, S. Kalyana-Sundaram, J. Yu, and A. M. Chinnaiyan, "Hpeak: an hmm-based algorithm for defining read-enriched regions in chip-seq data," *BMC bioinformatics*, vol. 11, no. 1, p. 369, 2010.

[31] Y. Shen, F. Yue, D. F. McCleary, Z. Ye, L. Edsall, S. Kuan, U. Wagner, J. Dixon, L. Lee, V. V. Lobanenkov *et al.*, "A map of the cis-regulatory sequences in the mouse genome," *Nature*, vol. 488, no. 7409, p. 116, 2012.

[32] A. Marson, S. S. Levine, M. F. Cole, G. M. Frampton, T. Brambrink, S. Johnstone, M. G. Guenther, W. K. Johnston, M. Wernig, J. Newman *et al.*, "Connecting microrna genes to the core transcriptional regulatory circuitry of embryonic stem cells," *Cell*, vol. 134, no. 3, pp. 521–533, 2008.

[33] T. S. Mikkelsen, M. Ku, D. B. Jaffe, B. Issac, E. Lieberman, G. Giannoukos, P. Alvarez, W. Brockman, T.-K. Kim, R. P. Koche *et al.*, "Genome-wide maps of chromatin state in pluripotent and lineage-committed cells," *Nature*, vol. 448, no. 7153, p. 553, 2007.

[34] S. Bilodeau, M. H. Kagey, G. M. Frampton, P. B. Rahl, and R. A. Young, "Setdb1 contributes to repression of genes encoding developmental regulators and maintenance of es cell state," *Genes & development*, vol. 23, no. 21, pp. 2484–2489, 2009.

[35] M. P. Schnetz, L. Handoko, B. Akhtar-Zaidi, C. F. Bartels, C. F. Pereira, A. G. Fisher, D. J. Adams, P. Flicek, G. E. Crawford, T. LaFramboise *et al.*, "Chd7 targets active gene enhancer elements to modulate es cell-specific gene expression," *PLoS genetics*, vol. 6, no. 7, p. e1001023, 2010.

[36] I. Hiratani, T. Ryba, M. Itoh, J. Rathjen, M. Kulik, B. Papp, E. Fussner, D. P. Bazett-Jones, K. Plath, S. Dalton *et al.*, "Genome-wide dynamics of replication timing revealed by in vitro models of mouse embryogenesis," *Genome research*, vol. 20, no. 2, pp. 155–169, 2010.

[37] D. Peric-Hupkes, W. Meuleman, L. Pagie, S. W. Bruggeman, I. Solovei, W. Brugman, S. Gräf, P. Flicek, R. M. Kerkhoven, M. van Lohuizen *et al.*, "Molecular maps of the reorganization of genome-nuclear lamina interactions during differentiation," *Molecular cell*, vol. 38, no. 4, pp. 603–613, 2010.

[38] R. Eskeland, M. Leeb, G. R. Grimes, C. Kress, S. Boyle, D. Sproul, N. Gilbert, Y. Fan, A. I. Skoultchi, A. Wutz *et al.*, "Ring1b compacts chromatin structure and represses gene expression independent of histone ubiquitination," *Molecular cell*, vol. 38, no. 3, pp. 452–464, 2010.

[39] L. Breiman *et al.*, "Statistical modeling: The two cultures (with comments and a rejoinder by the author)," *Statistical science*, vol. 16, no. 3, pp. 199–231, 2001.

[40] G. P. Release, "Gartner says the internet of things will transform the data center," *Retrieved from http://www.gartner.com/newsroom/id/2684616*, 2014.

[41] A. D. Mauro, M. Greco, and M. Grimaldi, "A formal definition of big data based on its essential features," *Library Review*, vol. 65, no. 3, pp. 122–135, 2016. [Online]. Available: http://dx.doi.org/10.1108/LR-06-2015-0061

[42] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. CRC press, 1995.

[43] M. J. Beal *et al.*, *Variational algorithms for approximate Bayesian inference*. university of London London, 2003.

[44] M. A. Beaumont, W. Zhang, and D. J. Balding, "Approximate bayesian computation in population genetics," *Genetics*, vol. 162, no. 4, pp. 2025–2035, 2002.

[45] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf, "Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems," *Journal of the Royal Society Interface*, vol. 6, no. 31, pp. 187–202, 2009.

[46] J. Lopes and M. Beaumont, "Abc: a useful bayesian tool for the analysis of population data," *Infection, Genetics and Evolution*, vol. 10, no. 6, pp. 825–832, 2010.

[47] M. A. Beaumont, "Approximate bayesian computation in evolution and ecology," *Annual review of ecology, evolution, and systematics*, vol. 41, pp. 379–406, 2010.

[48] K. Csilléry, M. G. Blum, O. E. Gaggiotti, and O. François, "Approximate bayesian computation (abc) in practice," *Trends in ecology & evolution*, vol. 25, no. 7, pp. 410–418, 2010.

[49] J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder, "Approximate bayesian computational methods," *Statistics and Computing*, vol. 22, no. 6, pp. 1167–1180, 2012.

[50] M. Sunnåker, A. G. Busetto, E. Numminen, J. Corander, M. Foll, and C. Dessimoz, "Approximate bayesian computation," *PLoS computational biology*, vol. 9, no. 1, p. e1002803, 2013.

[51] M. G. Blum, M. A. Nunes, D. Prangle, S. A. Sisson *et al.*, "A comparative review of dimension reduction methods in approximate bayesian computation," *Statistical Science*, vol. 28, no. 2, pp. 189–208, 2013.

[52] B. Jiang, T.-y. Wu, C. Zheng, and W. H. Wong, "Learning summary statistic for approximate bayesian computation via deep neural network," *arXiv preprint arXiv:1510.02175*, 2015.

[53] A. Wald, "Basic ideas of a general theory of statistical decision rules," in *Proceedings of the International congress of Mathematicians*, vol. 1, 1950, pp. 308–325.

[54] E. L. Lehmann and G. Casella, *Theory of point estimation*. Springer Science & Business Media, 2006.

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[56] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[57] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[58] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[59] S. C. Zhu, Y. N. Wu, and D. Mumford, "Minimax entropy principle and its application to texture modeling," *Neural computation*, vol. 9, no. 8, pp. 1627–1660, 1997.

[60] D. M. Blei and J. D. Lafferty, "Topic models," *Text mining: classification, clustering, and applications*, vol. 10, no. 71, p. 34, 2009.

[61] C. E. McCulloch and J. M. Neuhaus, *Generalized linear mixed models*. Wiley Online Library, 2001.

[62] H. Bozdogan, "Model selection and akaike's information criterion (aic): The general theory and its analytical extensions," *Psychometrika*, vol. 52, no. 3, pp. 345–370, 1987.

[63] K. P. Burnham and D. R. Anderson, *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media, 2003.

[64] ——, "Multimodel inference understanding aic and bic in model selection," *Sociological methods & research*, vol. 33, no. 2, pp. 261–304, 2004.

[65] G. Casella and R. L. Berger, *Statistical inference*. Duxbury Pacific Grove, CA, 2002, vol. 2.

[66] P. J. Huber *et al.*, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[67] J. Norton, E. Walter, and L. Pronzato, *Identification of Parametric Models: from Experimental Data*, ser. Communications and Control Engineering. Springer London, 2010. [Online]. Available: https://books.google.com.hk/books?id=BFmkcQAACAAJ

[68] I. Chakravarti, R. Laha, and J. Roy, *Handbook of methods of applied statistics*, ser. Wiley series in probability and mathematical statistics. Wiley, 1967, no. v. 1. [Online]. Available: https://books.google.com.hk/books?id=vtI-AAAAIAAJ

[69] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist*, vol. 6, no. 2, pp. 461–464, 03 1978. [Online]. Available: http://dx.doi.org/10.1214/aos/1176344136

[70] R. E. Kass and A. E. Raftery, "Bayes factors," *Journal of the american statistical association*, vol. 90, no. 430, pp. 773–795, 1995.

[71] Z. Xie, D. Kulasiri, S. Samarasinghe, and C. Rajanayaka, "The estimation of parameters for stochastic differential equations using neural networks," *Inverse Problems in Science and Engineering*, vol. 15, no. 6, pp. 629–641, 2007.

[72] N. E. Breslow and D. G. Clayton, "Approximate inference in generalized linear mixed models," *Journal of the American statistical Association*, vol. 88, no. 421, pp. 9–25, 1993.

[73] X. Lin and N. E. Breslow, "Bias correction in generalized linear mixed models with multiple components of dispersion," *Journal of the American Statistical Association*, vol. 91, no. 435, pp. 1007–1016, 1996.

[74] M. R. Karim and S. L. Zeger, "Generalized linear models with random effects; salamander mating revisited," *Biometrics*, pp. 631–644, 1992.

[75] J. G. Booth and J. P. Hobert, "Maximizing generalized linear mixed model likelihoods with an automated monte carlo em algorithm," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 1, pp. 265–285, 1999.

[76] J. Pan and R. Thompson, "Gauss-hermite quadrature approximation for estimation in generalised linear mixed models," *Computational Statistics*, vol. 18, no. 1, pp. 57–78, 2003.

[77] ——, "Quasi-monte carlo estimation in generalized linear mixed models," *Computational Statistics & Data Analysis*, vol. 51, no. 12, pp. 5765–5775, 2007.

[78] E. M. Al-Eid and S. P. (Professor), *Parameter estimation in generalized linear mixed models using quasi-Monte Carlo methods*. University of Manchester, 2007.

[79] D. Bates, M. Maechler, B. Bolker, S. Walker *et al.*, "lme4: Linear mixed-effects models using eigen and s4," *R package version*, vol. 1, no. 7, pp. 1–23, 2014.

[80] J. D. Hadfield *et al.*, "Mcmc methods for multi-response generalized linear mixed models: the mcmcglmm r package," *Journal of Statistical Software*, vol. 33, no. 2, pp. 1–22, 2010.

[81] B. Bolker, H. Skaug, A. Magnusson, and A. Nielsen, "Getting started with the glmmadmb package," *Available at glmmadmb. r-forge. r-project. org/glmmADMB. pdf*, 2012.

[82] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[83] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[84] C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, pp. 1–58, 2006.

[85] C. Hennig, *smoothmest: Smoothed M-estimators for 1-dimensional location*, 2012, r package version 0.1-2. [Online]. Available: https://CRAN.R-project.org/package=smoothmest

[86] I. M. Johnstone, Z. Ma, P. O. Perry, and M. Shahram, *RMTstat: Distributions, Statistics and Tests derived from Random Matrix Theory*, 2014, r package version 0.3.

[87] Statisticat and LLC., *LaplacesDemon: Complete Environment for Bayesian Inference*, 2016, r package version 16.0.1. [Online]. Available: https://web.archive.org/web/20150206004624/http://www.bayesian-inference.com/software

[88] B. Swihart and J. Lindsey, *rmutil: Utilities for Nonlinear Regression and Repeated Measurements Models*, 2016, r package version 1.1.0. [Online]. Available: https://CRAN.R-project.org/package=rmutil

[89] T. W. Yee, *VGAM: Vector Generalized Linear and Additive Models*, 2017, r package version 1.0-3. [Online]. Available: https://CRAN.R-project.org/package=VGAM

[90] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[91] R. A. Fisher, "Inverse probability," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 26. Cambridge University Press, 1930, pp. 528–535.

[92] D. A. Fraser, "The fiducial method and invariance," *Biometrika*, vol. 48, no. 3/4, pp. 261–280, 1961.

[93] A. P. Dempster, "New methods for reasoning towards posterior distributions based on sample data," *The Annals of Mathematical Statistics*, pp. 355–374, 1966.

[94] ——, "Upper and lower probabilities induced by a multivalued mapping," *The annals of mathematical statistics*, pp. 325–339, 1967.

[95] G. Shafer, *A mathematical theory of evidence*. Princeton university press, 1976, vol. 42.

[96] J. Hannig, "On generalized fiducial inference," *Statistica Sinica*, pp. 491–544, 2009.

[97] M. Xie and K. Singh, "Confidence distribution, the frequentist distribution estimator of a parameter: A review," *International Statistical Review*, vol. 81, no. 1, pp. 3–39, 2013.

[98] R. Martin and C. Liu, *Inferential models: reasoning with uncertainty*. Chapman and Hall/CRC, 2015.

[99] D. V. Lindley, "Fiducial distributions and bayes' theorem," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 102–107, 1958.

[100] A. P. Dempster, "On direct probabilities," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 100–110, 1963.

[101] A. P. Dawid and M. Stone, "The functional-model basis of fiducial inference," *The Annals of Statistics*, pp. 1054–1067, 1982.

[102] G. Taraldsen, B. H. Lindqvist *et al.*, "Fiducial theory and optimal inference," *The Annals of Statistics*, vol. 41, no. 1, pp. 323–341, 2013.

VITA

VITA

Rongrong Zhang was born and raised in Lanzhou, China. She received her bachelor's degree in Mathematics from Lanzhou University in 2010 and master's degree in Mathematics from Peking University in 2013. She then joined the Department of Statistics at Purdue University in 2013.