MATHEMATICAL MODELING WITH APPLICATIONS IN HIGH-PERFORMANCE CODING

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the
Graduate School of The Ohio State University

By

Yong Su, B.E., M.E., M.S.

* * * * *

The Ohio State University

2005

Dissertation Committee:

Yuan F. Zheng, Co-Adviser
Hakan Ferhatosmanoglu, Co-Adviser
Charles A. Klein

Approved by

Co-Adviser

Co-Adviser
Graduate Program in
Electrical and Computer
Engineering
© Copyright by

Yong Su

2005
ABSTRACT

With the progress in scientific research and practical application, mathematical models are being improved continuously to explain the nature and advance the technology. This dissertation involves both theoretical and empirical studies on mathematical modeling. From a theoretical standpoint, it investigates model selection and analysis; from an empirical perspective, it explores channel and source coding.

The question of how to decide among competing explanations of data is at the heart of the scientific enterprise. Choosing competing models based solely on the goodness of fit can result in the selection of an unnecessarily complex model that overfits the data. The dilemma is how to compromise both goodness of fit and model complexity. Among various model selection criteria, the Minimum Description Length (MDL) principle is a relatively recent method for inductive inference, which embodies the principle of Occam’s razor. In applying MDL to the selection of parametric models, one of the main obstacles is to calculate Fisher information. This study presents a general formula to compute Fisher information with multinomial or normal distribution for any mathematical model.

Another focus of the current research is on componential analysis, which investigates how and how much each parameter affects mathematical model’s ability to fit arbitrary patterns of data. To assess the relative importance of each parameter for such an ability is critical to both model selection and model building. The goal of
the research along this venue is to establish a unified theory, under which complex modeling procedures can be analyzed in terms of the contribution of each parameter.

Essentially, coding is the direct implementation of mathematical modeling. Channel coding and source coding are the practical applications of two important concepts in the information theory: channel capacity and entropy. This study examines these concepts in two particular cases respectively: bandwidth efficient nonsystematic turbo codes and bitmap index compression through an integrated reorganization.

The investigation starts with an introduction on mathematical modeling and the overview of the study. It is then organized by four consecutive themes in the following chapters: MDL model selection, componential analysis, turbo codes, and bitmap compression.
For the future reading of Amy and Stone
ACKNOWLEDGMENTS

I would like to extend all my gratitude to many individuals whose guidance and support have accompanied me through the course of this work. I am obligated to acknowledge the financial support from the NSF, NIH, NASA, and DOE at different stages of my studies.

First of all, I would like to express a heartfelt thank to Professor Yuan F. Zheng. The encouragement I received from him, his constructive criticisms, his sense of balance, and his wisdom made the accomplishment of my dissertation study possible. He was not only an adviser, but a mentor and a friend. His confidence in my abilities and his support throughout the past several years have been sincerely appreciated.

A special thank you goes to my co-adviser Professor Hakan Ferhatosmanoglu who granted me the privilege of being his Research Assistant. His thoughtful feedback and productive critique helped to clarify my thoughts and to broaden my thinking. He gave me inestimable support and advice under all circumstances, and I gratefully acknowledge all the trusts he put on me. His noble-heartedness showed me the richness of human soul.

I am grateful to Professor Charles A. Klein for serving in my candidacy exam and dissertation committee. Thank you for the valuable suggestions, feedback and guidance. Your introspective contributions served to enhance and strengthen my dissertation on many levels.
I would like to thank Professor Jay I. Myung and Professor Mark A. Pitt, the promoters of this dissertation. I owe them, among many other things, my initiation into model selection research. They offered me the vision for this dissertation study. Their many technical advices have been a great source of inspiration on my work. I shall remain gratefully indebted to them.

My thanks also go to Professor David E. Orin, Professor Oscar Y. Takeshita, and Professor Stanley C. Ahalt, whose contributions have enriched my dissertation study. I wish to thank Dr. Ali Pinar from Lawrence Berkeley National Laboratory and Dr. Peter Grünwald from CWI Netherlands, whose insightful comments provided valuable support to this study.

Many thanks are due to my fellow graduate students and friends Zhigang Gao, Yi Liu, Ozgur Ozturk, Mike Gibas, Tan Apaydin, Guadalupe Canahuate, Yanfei Liu, Woojae Kim, and Daniel Navarro for their comments and suggestions during the course of this work.

Exclusive thanks go to my wife, Ye Wang, for sharing with me many precious moments over the years, for her understanding, her company, and her love. Last surely not least, I would like to thank our parents for their encouragement and unconditional support through this journey.
VITA

February 12, 1974 ......................... Born – Huangshi, China

1996 ........................................ B.E. Electrical Engineering,
Huazhong University of Sci. & Tech.,
Wuhan, China

1999 ........................................ M.E. Electrical Engineering,
Huazhong University of Sci. & Tech.,
Wuhan, China

2001 ........................................ M.S. Electrical Engineering,
The Ohio State University

1999–2000 ................................. Graduate Fellow,
The Ohio State University

2000–2002 ................................. Graduate Research Associate,
Department of Electrical and Com-
puter Engineering,
The Ohio State University

2002–2004 ................................. Graduate Research Associate,
Department of Psychology,
The Ohio State University

2004–present ............................... Graduate Research Associate,
Department of Computer Science and
Engineering,
The Ohio State University.

PUBLICATIONS

Research Publications


FIELDS OF STUDY

Major Field: Electrical and Computer Engineering

Studies in:

- Wavelet Compression, Prof. Yuan F. Zheng
- Databases, Prof. Hakan Ferhatosmanoglu
- Mathematical Model Selection, Prof. Jay I. Myung
- Parallel Computing, Prof. David E. Orin
- Error Control Coding, Prof. Oscar Y. Takeshita
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Vita</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xiii</td>
</tr>
<tr>
<td>Chapters:</td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Minimum Description Length for Model Selection</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Recent Formulations of MDL</td>
<td>12</td>
</tr>
<tr>
<td>2.2.1 FIA Criterion</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2 NML Criterion</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Example Complexity for Binomial Models</td>
<td>17</td>
</tr>
<tr>
<td>2.3.1 Algebraic Deduction of the Integrand</td>
<td>18</td>
</tr>
<tr>
<td>2.3.2 Complexity with Numerical Integration</td>
<td>20</td>
</tr>
<tr>
<td>2.4 Fisher Information</td>
<td>21</td>
</tr>
<tr>
<td>2.4.1 Models with Multinomial Distribution</td>
<td>22</td>
</tr>
<tr>
<td>2.4.2 Models with Normal Distribution</td>
<td>28</td>
</tr>
<tr>
<td>2.5 MDL Complexity Comparison</td>
<td>31</td>
</tr>
<tr>
<td>2.5.1 FIA Complexity</td>
<td>31</td>
</tr>
</tbody>
</table>
3. Componential Analysis .......................... 41
   3.1 Motivation and Significance .................. 41
   3.2 Simple Examples ............................ 43
   3.3 The Componential Analysis Problem .......... 45
      3.3.1 Formal Definition of the Problem ..... 46
      3.3.2 Variable Importance Problems ......... 48
   3.4 Complexity from Model Selection Criteria ... 55
   3.5 Global Approaches ........................ 60
      3.5.1 Best Point Method .................... 60
      3.5.2 Average Method ....................... 61
   3.6 Local Approaches .......................... 62
      3.6.1 Sub-model Method ..................... 62
      3.6.2 Eigenvalue Method .................... 63
   3.7 Simulation Results ......................... 64
   3.8 Summary .................................. 67

4. Bandwidth Efficient Nonsystematic Turbo Codes ....... 68
   4.1 Introduction ............................... 68
   4.2 Binary Nonsystematic Turbo Codes ........... 70
   4.3 Bandwidth Efficient Scheme ................ 72
      4.3.1 Puncturing Patterns .................. 73
      4.3.2 Gray Mappings ....................... 73
      4.3.3 Logarithm Likelihood Ratio .......... 75
   4.4 Threshold Analysis ........................ 77
   4.5 Performance .............................. 77
   4.6 Conclusion ................................ 78

5. Bitmap Index Compression with an Integrated Reorganization ........ 80
   5.1 Introduction ............................... 81
   5.2 Preliminaries and Related Work ............. 83
      5.2.1 The Bitmap Index ..................... 84
      5.2.2 Gray Code Ordering ................... 85
      5.2.3 Word-Aligned Hybrid Compression ...... 87
   5.3 Columnwise Permutation ..................... 88
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.1</td>
<td>The Column Reordering Problem</td>
<td>88</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Peculiarity of Gray Code Ordering</td>
<td>90</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Reordering by Bitmap Data</td>
<td>93</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Reordering by Query History</td>
<td>96</td>
</tr>
<tr>
<td>5.4</td>
<td>Results and Discussions</td>
<td>100</td>
</tr>
<tr>
<td>5.4.1</td>
<td>The Effects of Column Reordering</td>
<td>100</td>
</tr>
<tr>
<td>5.4.2</td>
<td>The Bias of Gray Code Ordering</td>
<td>101</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Performance of Reordering by Bitmap Data</td>
<td>104</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Performance of Reordering by Query History</td>
<td>108</td>
</tr>
<tr>
<td>5.5</td>
<td>Summary and Future Work</td>
<td>109</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>111</td>
</tr>
</tbody>
</table>

Appendices:

A. Characteristic Properties of Convex Function        116

B. Conversion Bitmaps                                  120

C. Zipf’s Law and Log Linear Distributions            122

Bibliography                                         125
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>20</td>
</tr>
<tr>
<td>2.2</td>
<td>39</td>
</tr>
<tr>
<td>3.1</td>
<td>65</td>
</tr>
<tr>
<td>3.2</td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>85</td>
</tr>
<tr>
<td>5.2</td>
<td>88</td>
</tr>
<tr>
<td>5.3</td>
<td>106</td>
</tr>
<tr>
<td>5.4</td>
<td>107</td>
</tr>
<tr>
<td>5.5</td>
<td>108</td>
</tr>
<tr>
<td>B.1</td>
<td>121</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Binomial model complexity by numerical integration.</td>
<td>21</td>
</tr>
<tr>
<td>2.2</td>
<td>Complexity as function of sample size.</td>
<td>35</td>
</tr>
<tr>
<td>2.3</td>
<td>Complexity as function of category.</td>
<td>36</td>
</tr>
<tr>
<td>2.4</td>
<td>Relative complexity difference with different categories.</td>
<td>36</td>
</tr>
<tr>
<td>3.1</td>
<td>Four toy models with data ensembles indexed by parameter.</td>
<td>44</td>
</tr>
<tr>
<td>3.2</td>
<td>The Venn diagram for $X_1$ and $X_2$ with $Y$.</td>
<td>49</td>
</tr>
<tr>
<td>3.3</td>
<td>Local parameter importancy index of parameter $a$ and $b$ for M1.</td>
<td>66</td>
</tr>
<tr>
<td>4.1</td>
<td>System diagram of the coding scheme.</td>
<td>70</td>
</tr>
<tr>
<td>4.2</td>
<td>A binary turbo encoder.</td>
<td>71</td>
</tr>
<tr>
<td>4.3</td>
<td>A decoder for a nonsystematic binary turbo codes.</td>
<td>71</td>
</tr>
<tr>
<td>4.4</td>
<td>The nonsystematic turbo encoder.</td>
<td>72</td>
</tr>
<tr>
<td>4.5</td>
<td>The puncturing and bits permutation patterns.</td>
<td>74</td>
</tr>
<tr>
<td>4.6</td>
<td>The ring.</td>
<td>74</td>
</tr>
<tr>
<td>4.7</td>
<td>The symmetry Gray mapping patterns.</td>
<td>75</td>
</tr>
<tr>
<td>4.8</td>
<td>The 16QAM Gray mapping.</td>
<td>76</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.9</td>
<td>Performance of the bandwidth efficient nonsystematic turbo codes.</td>
<td>78</td>
</tr>
<tr>
<td>5.1</td>
<td>The overview of the reorganization scheme.</td>
<td>83</td>
</tr>
<tr>
<td>5.2</td>
<td>Gray code ordering algorithm.</td>
<td>91</td>
</tr>
<tr>
<td>5.3</td>
<td>The query history models.</td>
<td>99</td>
</tr>
<tr>
<td>5.4</td>
<td>The compressed file sizes after column reordering.</td>
<td>101</td>
</tr>
<tr>
<td>5.5</td>
<td>The histogram of the file sizes.</td>
<td>102</td>
</tr>
<tr>
<td>5.6</td>
<td>The effects of column reordering.</td>
<td>103</td>
</tr>
<tr>
<td>5.7</td>
<td>The distribution of runs &amp; compressed column size.</td>
<td>104</td>
</tr>
<tr>
<td>5.8</td>
<td>The bias of Gray code ordering.</td>
<td>105</td>
</tr>
<tr>
<td>C.1</td>
<td>The log linear distribution.</td>
<td>123</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

A model is a schematic description of a complex system by ignoring nuisance details. Useful models can be applied for further study of the system in addition to accounting for known phenomena. However, some models are specified by verbal language, which leads to an unavoidable limitation: lack of precision. Mathematical modeling represents an alternative approach to overcoming this limitation by inferring structural and functional properties of a process from experimental data in explicit mathematical expressions.

In this dissertation research, the mathematical model or simply the model, is defined as a parametric family of probability distribution function \( f_{X|\theta}(x|\theta) \), as a Riemannian manifold [1] embedded in the space of distribution, with \( x \in \mathcal{X} \) and \( \theta \in \Omega \). \( \mathcal{X} \) and \( \Omega \) are the sets in sample space and parameter space respectively. The sample space or parameter space could be an Euclidean space with arbitrary dimensions. Thus, the dimension of the parameter here corresponds to what is commonly referred to the number of parameters in the model. Furthermore, \( \mathcal{X} \) and \( \Omega \) are inseparable components of the model. Hence, the model defined here is equivalent to the model class in some other literature whereas the model referred there corresponds to our model definition with \( \Omega \) as a singleton. Moreover, models with discrete and continuous
random quantity are not treated separately, since their probability density functions can be unified as Radon-Nikodym derivatives [2] with respect to counting measure or Lebesgue measure respectively.

After defining what a model is, let’s look at the model selection problem. In modeling measured data, we wish to identify the model, from a set of candidate models, which generated the observed data. This is an ill-posed problem because information in the finite data sample is rarely sufficient to point to a single model. Rather, multiple models may provide equally good descriptions of the data. In statistics, this ill-posedness of model selection is overcome by reformulating the inference problem as one of making a best guess as to which model provides the closest approximation, in some defined sense, to the true but unknown model that generated the data. The particular measure of such an approximation, which is widely recognized among modelers in statistics and computer science, is generalizability. Generalizability, or predictive accuracy, refers to a model’s ability to accurately predict future, as yet unseen, data samples from the same process that generated the currently observed sample.

A formal definition of generalizability can be given in terms of a discrepancy function that measures the degree of approximation or similarity between two probability distributions. A discrepancy function $D(f, g)$ between two distributions, $f$ and $g$, is some well-behaved function (e.g., Kullback-Leibler divergence [3]) that satisfies $D(f, g) > D(f, f) = 0$ for $f \neq g$. Generalizability could be defined as

\[
E^{f_T}[D(f_T, f_M)] \triangleq \int_{\mathcal{X}} D(f_T(x), f_M(\hat{\theta}(x))) f_T(x) dx, \tag{1.1}
\]

where $f_T$ and $f_M$ denote the probability distributions of the true and guessing models and $\hat{\theta}(x)$ is the Maximum Likelihood Estimator (MLE) of the parameter. According
to the above equation, generalizability is a mean discrepancy between the true model and the best-fitting member of the model of interest, averaged across all possible data that could be observed under the true model. The basic tenet of model selection is that among a set of competing model, one should select the one that optimizes generalizability (i.e., minimizes the quantity in equation (1.1)). However, generalizability is not directly observable and instead, one must estimate the measure from a data sample by considering the characteristics of the model under investigation.

Several generalizability estimates have been proposed. They include Akaike Information Criterion (AIC) [4], Bayesian Information Criterion (BIC) [5], Cross Validation (CV) [6], and Minimum Description Length (MDL) [7, 8, 9, 10, 11, 12]. It is well established in statistics that choosing any competing models based solely on goodness of fit can result in the selection of an unnecessarily complex model that overfits the data. The dilemma is how to compromise both goodness of fit and model complexity. Goodness of fit refers to how well a model fits the particular data set whereas model complexity or flexibility refers to a model’s ability to fit arbitrary patterns of data. For all the model selection criteria, generalizability is estimated by trading off a model’s goodness of fit with model complexity.

AIC and BIC are the two most commonly used estimations of generalizability. Developed by using expected log likelihood function as the risk of decision and finding its asymptotic approximation, AIC is defined as

\[
AIC \triangleq - \log f(x|\hat{\theta}) + k, \tag{1.2}
\]

where \(\log(\cdot)\) is the natural logarithm function of base \(e\) and \(k\) is the dimension of the parameter. The BIC criterion, obtained as an asymptotic approximation of a
quantity related to the Bayes factor [13], is defined similarly as

\[ \text{BIC} \triangleq - \log f_{X|\Theta}(x|\hat{\theta}) + \frac{k}{2} \log(n), \]

(1.3)

where \( n \) is the sample size. In both (1.2) and (1.3), the first term stands for a goodness of fit measure and the second term represents a complexity measure. From the AIC/BIC viewpoint, the number of parameter \( (k) \) and the sample size \( (n) \) are the only relevant facets of complexity. AIC/BIC, however, ignores another important facet of model complexity, namely, the functional form of the model equation [14]. Functional form refers to the way in which the model’s parameters are combined to define the model equation. For example, two models, \( x = at + b \) and \( x = at^b \), have the same number of parameters but differ in functional form.

CV is an easy-to-use, sampling-based method of estimating a model’s generalizability. In CV, the data are split into two samples, the calibration sample and the validation sample. The model of interest is fitted to the calibration sample and the best-fit parameter values are obtained. With these values fixed, the model is fitted again, this time to the validation sample. The resulting fit defines the model’s generalizability estimate. Note that this estimation is done without an explicit consideration of complexity. Unlike AIC or BIC, CV takes into account functional form as well as the number of parameters. But given the implicit nature of CV, it is not clear how this is achieved.

The principle of MDL was developed within the domain of algorithmic coding theory in computer science [15]. It represents a new conceptualization of the model selection problem. In MDL, both models and data are viewed as codes that can be compressed, and the goal of model selection is to choose the model that permits the
greatest compression of data in its description. The code here refers to the probabil-
ity distribution $p$ of a random quantity. The code length is justified as $\log(1/p)$ from
Shannon’s information theory [16]. The shortest code length obtainable with the help
of a given model is called the stochastic complexity of the model. In the present study,
we focus on two implementations of the stochastic complexity, a Fisher Information
Approximated normalized maximum likelihood (FIA) [11] and Normalized Maximum
Likelihood (NML) [11, 12]. Each, as an analytic realization of Occam’s razor, com-
bines measures of goodness of fit and model complexity in a way that remedies the
shortcomings of AIC, BIC, and CV; complexity is explicitly defined and functional
form is included in the definition.

In applying MDL to the selection of mathematical models, one of the main obsta-
cles is to calculate Fisher information. In Chapter 2, a general formula to calculate
Fisher information is derived in detail. We start with a special model with binomial
distribution. Then we generalize the calculating formula to models with multinomial
distribution or normal distribution. The usage of the formula for the models of cate-
gorization, information integration, retention, and psychophysics are also illustrated.
Further, we show that the two complexity measures of FIA and NML are close to
each other for a multinomial model. Moreover, the adequacy of MDL is demonstrated
in the selection of retention models.

This research also deals with decomposing model complexity to the contribution
of each parameter. Model complexity is the quantified measure of model’s ability to
fit arbitrary patterns of data. As discussed before, the notion of model complexity
originates from model selection problem; we found that choosing competing models
based solely on the goodness of fit can result in the selection of an unnecessarily
complex model that overfits the data. Accordingly, besides model selection, our next focus is on the problem of componential analysis on model complexity.

As an invaluable tool for scientific research, mathematical models are being improved continuously to explain the nature. To simplify the inquiry, a model is qualitatively characterized by finite factors. Each factor is then quantitatively represented by the parameter of the model. How to assess the relative importance of each parameter is crucial to both model selection and model building. The problem of componential analysis is to investigate how and how much each parameter affects mathematical model’s ability to fit arbitrary patterns of data. The goal of this investigation is to establish a unified theory under which complex modeling procedures can be analyzed in terms of the contribution of each parameter.

Methodologically, the importance index will be proposed globally or locally. A global approach is to evaluate importance index by choosing a certain model complexity measure. As the model complexity is a global property of the model, the derived importance index is a global attribute of the model. On the other hand, a local approach is to investigate how the complexity is evaluated at each point in the model. The importance index is then defined locally without explicitly borrowing certain model complexity measure. The global measure could also be induced by the procedure combining average by integration of the local measure.

In Chapter 3, the problem of componential analysis is introduced and prior approaches to the similar problems are examined. Along this line of research, the current study reviews several known measures of model complexity and explores various methods to define and estimate importance index of each parameter. Using a highly efficient calculating method for the Fisher information presented in Chapter 2,
various algorithms including those motivated by the problem of decision tree to find the importancy index are proposed. The preliminary results are also presented.

Essentially, coding is a direct implementation of mathematical modeling. In Shannon’s classical information theory [16], communication channel and information source are described by mathematical models. For example, in communication system, the Additive White Gaussian Noise (AWGN) channel model is the linear addition of wideband or white Gaussian noise with a constant spectral density, while the discrete memoryless source is presented by a discrete random process model. Channel coding and source coding are the practical applications of two important concepts in information theory: channel capacity and entropy. This study examines two particular cases of these concepts. Next, we introduce each case with more details.

Information theory proves that asymptotically error free communication is possible in a given noisy channel up to a certain information transmission rate, called channel capacity. One of the active research areas in the information and coding theory is to find simple coding schemes that work as close to the channel capacity as possible. The turbo code invented in 1993 [17] has performance almost achieving the channel capacity. The code is constructed by parallel concatenating two convolutional codes via an interleaver. However, the performance depends upon the particular structure of the code. Chapter 4 presents a coding scheme that uses a nonsystematic turbo code and a bandwidth efficient 16QAM modulation to get good performance at low SNR. Nonsystematic constituent code is used to expand the searching domain whereas puncturing and 16QAM improve the bandwidth efficiency. The performance of one such coding scheme is presented.
While channel coding increase the error free information transmission rate, the purpose of source coding is to presented information in a compact form. Designing efficient bitmap schemes for storage and retrieval of massive scientific data is a challenging problem. Through an integrated data reorganization method, this study improves bitmap index compression by a factor of 3 to 15. Our first step is to explore the speciality of Gray code ordering. After analyzing the algorithm, we find that Gray code ordering favors the first few columns of the bitmap. Furthermore, the number of the runs in the column could increase exponentially if the bits are uniformly distributed. Next, we arrange the columns of bitmap according to the bitmap itself and the query history. By the bitmap index itself, we design a novel integrated reorganization algorithm that maximally extends the longest run to facilitate the later run length encoding. Regarding the query history, we model the query access patterns by several statistical distributions and move the more frequently accessed columns ahead. Results on empirical data sets show the performance of the bitmap index has significantly improved in terms of the compressed bitmap size and the efficiency of the queries. In essence, our technique can be applied to both equality-encoded and range-encoded bitmap indices or even more general circumstances of bit matrices.

In a nutshell, our main discussion is organized by four consecutive themes in the coming chapters: MDL model selection, componential analysis, turbo codes, and bitmap compression.
CHAPTER 2

Minimum Description Length for Model Selection

The question of how one should choose among contesting explanations of limited observation is the model selection problem. Among various model selection criteria, the minimum description length principle is a relatively recent method for inductive inference. It embodies the principle of Occam’s razor, which states that one should not make more assumptions than the minimum needed. To apply MDL principle to the selection of statistical models, one of the major obstacles is to calculate Fisher information. In the present study, we provide a general formula to calculate Fisher information for models assuming multinomial or normal distributions. We also illustrate the usage of the formula for models of categorization, information integration, retention, and psychophysics. Further, we compute and compare the complexity penalty terms of two recent versions of MDL for a multinomial model. Finally, the adequacy of MDL is demonstrated in the selection of retention models.

2.1 Introduction

The study of modeling is concerned with identifying lawful patterns of phenomena, as well as developing theories that provide a concise explanation of the data. How one should find appropriate explanations of the data is at the heart of scientific
enterprises. Ideally, we want to identify the underline process that actually generates the data. In reality, however, it is impossible to find the true model from finite data sample. Therefore, we reformulate the inference problem as one of identifying the model that best approximates the truth among a set of candidates. The primary guideline for model selection is to estimate generalizability of the model by the principle of parsimony.

Generalizability measures the approximation of the candidate model to the true data generating model. It is a qualitative notation of how well the model will fit the further data which is not used in model specification. In other words, it reflects the predictive accuracy of the model to the sample from the same process that has generated the observed data sample. The idea is that a model should not be evaluated solely on how well it fits observed data but on how well it fits future data generated from the underlying regularity without the extrospective disturbance.

Model selection criterion is a quantitative estimation of the generalizability. Useful model selection criterion is a well-compromising of goodness of fit and model complexity. Goodness of fit measures a model’s fitness to particular data in hand, and as such, it depends upon observed data as well as the model of interest. Model complexity refers to model’s ability to fit arbitrary patterns of data; therefore, it is determined by the characteristics of the model itself. It is well established in statistics that choosing any competing models based solely on the goodness of fit can result in the selection of an unnecessarily complex model that overfits the data.

Several forms of model selection criterion have been proposed in the past. They include Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Cross Validation (CV), and Minimum Description Length (MDL). AIC and BIC are
both modifications of Maximum Likelihood Estimator (MLE) by making asymptotic assumptions. It works relatively well when the sample size is large. However, with the limitation of only taking into account the dimension of the parameter, but ignoring the difference in functional form (i.e., the way in which the model’s parameters and data are combined in the model equation), the application of AIC/BIC to the data with small sample size is restricted. CV is an easy to use, heuristic method of estimating a model’s capacity to accurately predict future data. The emphasis on prediction accuracy might make it reasonable to suppose that CV somehow takes into account the effects of functional form. But how, and how well it does this is not clear to us.

Remedying the shortcomings of AIC, BIC, and CV, minimum description length principle has been proposed to be an analytic realization of Occam’s razor [18]. MDL is a quantitative realization and transcendent compromise of both goodness of fit and model complexity. It is shown to be a plausible model selection method in cognitive science [19, 20]. Similar to a living thing, MDL develops constantly. Fisher Information Approximated normalized maximum likelihood (FIA) [11] and Normalized Maximum Likelihood (NML) [11, 12] are the two most recent MDL model selection criteria, characterizing a model through the parameter space and the sample space respectively.

The purpose of the current chapter is threefold. The first is to address an issue in applying MDL to the selection of parametric models. Calculation of the Fisher information can sometimes be a deterrent to using the measure. We walk the reader through the derivation and application of an efficient formula for computing Fisher information for two broad classes of models, those that have multinomial or normal distributions. Next, we compare the relative performance of FIA and NML for a
saturated multinomial model. Finally, we present an example application of the MDL criteria in the selection of retention models in cognitive science. We begin by briefly reviewing the recent formulations of MDL.

2.2 Recent Formulations of MDL

The MDL approach to model selection is developed within the domain of algorithmic coding theory [15], where the goal of model selection is to choose the model that permits the greatest compression of data in its description. The properties of a model that any selection method should be sensitive to include goodness of fit and model complexity. In terms of trade-off between goodness of fit and model complexity, both FIA and NML criteria can be decomposed as the sum of goodness of fit measure and model complexity measure. The goodness of fit measure for both criteria is the negative log likelihood function evaluated at the MLE of the parameter. The crucial difference of these two criteria is in the model complexity measure, which is the focus of our current investigation and will be revisited in later sections. In the following section, we present and discuss FIA and NML criteria.

2.2.1 FIA Criterion

Utilizing the notion of Fisher information, Rissanen [11] has the following model selection criterion in 1996.

\[
\text{FIA} \triangleq - \log f_{X|\Theta}(x|\hat{\theta}) + C_{\text{FIA}},
\]

where \( \log(\cdot) \) is the natural logarithm function to the base \( e \), \( \hat{\theta} \) is the MLE of the parameter, and

\[
C_{\text{FIA}} \triangleq \frac{k}{2} \log \frac{n}{2\pi} + \log \int_{\Omega} \sqrt{|I(\theta)|}d\theta, \quad (2.1)
\]
where \( k \) is the dimension of the parameter, \( n \) is the sample size, \( \Omega \) is the range of the parameter, and \( I(\theta) \) is the Fisher information in a single observation

\[
I_{i,j}(\theta) \triangleq \text{Cov}_\theta \left[ \frac{\partial}{\partial \theta_i} \log f_{X|\Theta}(X|\theta), \frac{\partial}{\partial \theta_j} \log f_{X|\Theta}(X|\theta) \right]. \tag{2.2}
\]

The inspection of \( C_{\text{FIA}} \) reveals four discernible facets of model complexity: the number of parameters \( k \), sample size \( n \), parameter range \( \Omega \), and functional form. As defined in (2.1), \( C_{\text{FIA}} \) consists of two additive terms. The first term seizes the number of parameter facet of complexity. The second term captures the functional form facet of complexity through the Fisher information matrix \( I(\theta) \). The sample size only appears in the first term. An implication is that when the sample size becomes large, the relative contribution of the second term to complexity in relation to that of the first term becomes negligible, essentially reducing \( C_{\text{FIA}} \) to the complexity penalty of BIC. Being a logarithmic function of sample size, the first term is also a linear function of the number of parameters. So the impact of sample size on model complexity is less dramatic than that of the number of parameters. The calculation of the second term depends on the parameter range, over which the integration of a nonnegative quantity in the parameter space is required. As such, the greater the ranges of the parameters are, the larger the value of the integral is, and therefore, the more complex the model is.

There are at least two nontrivial challenges to overcome in computing \( C_{\text{FIA}} \). First, we need the evaluation of integral in multidimensional parameter space. In general, a closed-form solution is not possible, especially for high dimensions, and numerical integration methods must be used. Markov Chain Monte Carlo (MCMC) might be one of the practical solutions to this problem [21]. In Section 2.3, we provide an illustrative example using simple numerical integration methods. Second, with partial
derivatives and expectation in the definition of Fisher information, it is obvious that
directly calculating the Fisher information matrix element by element is a daunting
task. First, the number of elements in the Fisher information matrix is the square
of the dimension of the parameter. For example, with 100-dimension parameter, we
need to find 10,000 elements of the Fisher information matrix, which will be quite a
chore. Furthermore, the functional form of the model and design matrix of the data
to be modeled might make the computation even harder, if not impossible. The first
main goal of the present study is to address this issue. In Section 2.4, we develop
simple algebraic formula solving all the above difficulties for models with multinomial
or normal distribution.

2.2.2 NML Criterion

From a coding perspective, Rissanen [12] shows the strong optimality of normal-
ized maximum likelihood distribution as a universal code for the data with the aid of
a model. The code here refers to the probability distribution \( p \) of a random quantity.
The code length is justified as \( \log(1/p) \) from Shannon’s information theory. The NML
criterion is the length of ideal code, which comes from following minimax problem
that Rissanen formulates in his derivation of NML:

\[
\arg\inf_{p} \sup_{q \in Q} \mathbb{E}_{q} \left[ \log \frac{f_{x|\Theta}(X|\hat{\Theta})}{p(X)} \right],
\]

where \( p \) is the ideal code, \( q \) is the data generating model, \( \mathbb{E}_{q}[\cdot] \) is the expectation with
respect to distribution \( q \), and \( Q \triangleq \{ q(x) : \mathbb{E}_{q}[\log \frac{q(X)}{f_{x|\Theta}(X|\Theta)}] < \infty \} \) with \( \hat{\Theta} \) as the MLE
of the parameter.

Given a parametric family of probability distribution \( f_{x|\Theta}(x|\Theta) \) and a true model
\( Q \), the minimax problem is to identify one probability distribution \( p \) that minimizes
the worst prediction error. It is not required that the true model be a member of the models of interest (i.e., the model need not to be correctly specified), nor that the desired probability distribution, as a solution to the minimax problem, be a member of the model. For these reasons, the NML distribution is also called the universal model.

Rissanen’s derivation [12] of a solution to the minimax problem is roughly the following:

\[
\inf_{p} \sup_{q \in Q} \mathbb{E}^q [ \log \frac{f_{X|\theta}(X|\hat{\theta})}{p(X)} ] \\
\geq \sup_{q \in Q} \inf_{p} \mathbb{E}^q [ \log \frac{f_{X|\theta}(X|\hat{\theta})}{p(X)} ] \\
= \sup_{q \in Q} \inf_{p} D[q \parallel p] - \mathbb{E}^q [ \log \frac{q(X)}{f_{X|\theta}(X|\hat{\theta})} ] \\
= \sup_{q \in Q} -D[q \parallel g] + C_{NML} \\
= C_{NML}.
\]

Equality occurs when \( p = q = g \) which is the NML distribution; that is, the maximized likelihood given the particular set of observed data divided by the sum of maximized likelihoods over all possible data samples. Here \( D[\cdot \parallel \cdot] \) is the relative entropy or Kullback-Leibler divergence between the two distributions and

\[
C_{NML} \triangleq \log \int_{\theta(x) \in \Omega} f_{X|\Theta}(x|\hat{\theta}(x)) dx. \tag{2.3}
\]

So

\[
NML \triangleq -\log f_{X|\Theta}(x|\hat{\theta}) + C_{NML}.
\]

As the solution to the above minimax problem, NML generalizes MLE, which is employed to identify the parameter values that optimize likelihood (i.e., goodness of fit) within a given model. Similarly, NML is employed to identify the model, among a set of models, which optimizes generalizability or equivalently, prediction error.
Being the value of the *minimized* worst prediction error and the complexity penalty of NML, \( C_{NML} \) also amounts to a normalizing constant of the NML distribution or the *sum of all best fits*, summed over all possible data patterns.

As presented above, the difference between FIA and NML is their complexity measure: \( C_{FIA} \) and \( C_{NML} \). Because \( C_{FIA} \) is obtained as a Taylor series expansion of \( C_{NML} \) under the assumption of large sample sizes, \( C_{NML} \) might capture the full scope of model complexity; thereby, representing a better solution to the long-standing problem of quantifying model complexity. In contrast, \( C_{FIA} \) is a partial solution to this problem because the higher order contributions of complexity are ignored in its complexity measure.

Like \( C_{FIA} \), \( C_{NML} \) is also nontrivial to compute. They both require evaluation of an integral, though different kinds: integration over parameter space in \( C_{FIA} \) compared to integration over sample space in \( C_{NML} \). In Section 2.4, we present a general formula to calculate Fisher information which is used to compute \( C_{FIA} \). With this formula, calculation of \( C_{NML} \) is more challenging than that of \( C_{FIA} \). That is, the former requires two-step expensive computation: maximization of likelihood function over parameter space given any data set in the sample space, followed by integration of the maximized likelihood over the sample space. In practice, the parameter estimation is mostly done numerically, which is complicated by the local maxima problem. Making the computation of \( C_{NML} \) even harder, the sample space normally has a much higher dimension than the parameter space. The second goal of the present study is to compare these two complexity measures for some known models (see Section 2.5) to gain further insight into the concept of their relationship.
2.3 Example Complexity for Binomial Models

To provide an example of how the model complexity is computed, we calculate the complexity penalty $C_{\text{FIA}}$ for three binomial models. Since the first term on the right-hand side of (2.1) is obtained by plugging in several constants, we basically want

$$\log \int_{\Omega} \sqrt{|I(\theta)|} d\theta.$$  

(2.4)

The three models have the same set of parameters with

$$\Omega \triangleq \{\theta = [\theta_1, \theta_2]^T : \theta_1 \in [0.4, 0.9999], \theta_2 \in [0.15, 0.7], \theta_1 - \theta_2 > 0.05\}.$$  

The sample vector is 15-dimensional; that is, $X|\Theta = [X_1|\Theta, X_2|\Theta, \ldots, X_{15}|\Theta]^T$. \{$X_k|\Theta$\} are independent and $X_k|\Theta \sim \text{Ber}(p_k(\theta))$, where $p_k(\theta)$ is the re-parameterization of $f(a, b, t_k)$ with $t_k = 0.1 \times 2^{k-1}$, $k = 1, 2, \ldots, 15$,

$$\begin{align*}
\theta_1 &= f(a, b, 0.1) \\
\theta_2 &= f(a, b, 12.8)
\end{align*}$$

(2.5)

and the three example models are given by

$$f(a, b, t) = \begin{cases} 
ae^{-bt} & (f_1) \\
1/(a + bt) & (f_2) \\
a(1 + t)^{-b} & (f_3)
\end{cases}.$$  

(2.6)

The complexity (2.4) for the three models in (2.6) is computed in two steps. First, we calculate the determinant of Fisher information matrix for these three binomial models. We then make the numerical integration by both linear integration and simple Monte Carlo integration.
2.3.1 Algebraic Deduction of the Integrand

After checking that Fisher information regularity conditions [2] are hold for the three models, the Fisher information matrix is defined as

\[
I_{i,j}(\theta) \triangleq \text{Cov}_{\theta} \left[ \frac{\partial}{\partial \theta_i} \log f_{X|\Theta}(X|\theta), \frac{\partial}{\partial \theta_j} \log f_{X|\Theta}(X|\theta) \right] \\
= -E_{\theta} \left[ \frac{\partial^2}{\partial \theta_i \partial \theta_j} \log f_{X|\Theta}(X|\theta) \right].
\]

Since \( X_k|\Theta \sim \text{Ber}(p_k(\theta)) \),

\[
f_{X_k|\Theta}(x_k|\theta) = p_k(\theta)^x_k(1 - p_k(\theta))^{1-x_k}, \quad x_k = 0, 1.
\]

Since \( \{X_k|\Theta\} \) independent,

\[
f_{X|\Theta}(x|\theta) = \prod_{k=1}^{15} f_{X_k|\Theta}(x_k|\theta) = \prod_{k=1}^{15} p_k(\theta)^x_k(1 - p_k(\theta))^{1-x_k}.
\]

Therefore,

\[
\log f_{X|\Theta}(x|\theta) = \sum_{k=1}^{15} x_k \log p_k(\theta) + (1 - x_k) \log(1 - p_k(\theta)),
\]

\[
\frac{\partial \log f_{X|\Theta}(x|\theta)}{\partial \theta_i} = \sum_{k=1}^{15} \left( \frac{x_k}{p_k(\theta)} + \frac{x_k - 1}{1 - p_k(\theta)} \right) \frac{\partial p_k(\theta)}{\partial \theta_i},
\]

\[
\frac{\partial^2 \log f_{X|\Theta}(x|\theta)}{\partial \theta_i \partial \theta_j} = \sum_{k=1}^{15} \left( \frac{-x_k}{p_k^2(\theta)} + \frac{x_k - 1}{(1 - p_k(\theta))^2} \right) \frac{\partial p_k(\theta)}{\partial \theta_i} \frac{\partial p_k(\theta)}{\partial \theta_j}
\]

\[
+ \left( \frac{x_k}{p_k(\theta)} + \frac{x_k - 1}{1 - p_k(\theta)} \right) \frac{\partial^2 p_k(\theta)}{\partial \theta_i \partial \theta_j},
\]

\[
I_{i,j}(\theta) = -E_{\theta} \left[ \frac{\partial^2 \log f_{X|\Theta}(X|\theta)}{\partial \theta_i \partial \theta_j} \right]
\]

\[
= \sum_{k=1}^{15} \left( \frac{1}{p_k(\theta)} + \frac{1}{1 - p_k(\theta)} \right) \frac{\partial p_k(\theta)}{\partial \theta_i} \frac{\partial p_k(\theta)}{\partial \theta_j}
\]

\[
= \sum_{k=1}^{15} \frac{1}{p_k(\theta)(1 - p_k(\theta))} \frac{\partial p_k(\theta)}{\partial \theta_i} \frac{\partial p_k(\theta)}{\partial \theta_j},
\]

\[
|I(\theta)| = I_{11}(\theta)I_{22}(\theta) - I_{12}(\theta)I_{21}(\theta)
\]
\[
\begin{align*}
\sum_{k,l=1}^{15} & \frac{\partial p_k(\theta)}{\partial \theta_1} \frac{\partial p_l(\theta)}{\partial \theta_2} \\
& \cdot \left( \frac{\partial p_k(\theta)}{\partial \theta_1} \frac{\partial p_l(\theta)}{\partial \theta_2} - \frac{\partial p_k(\theta)}{\partial \theta_2} \frac{\partial p_l(\theta)}{\partial \theta_1} \right) \\
= & \sum_{k,l=1}^{15} \frac{\partial p_k(\theta)}{\partial \theta_1} \frac{\partial p_l(\theta)}{\partial \theta_2} \frac{p_k(\theta)(1-p_k(\theta))p_l(\theta)(1-p_l(\theta))}{p_k(\theta)(1-p_k(\theta))p_l(\theta)(1-p_l(\theta))}^2.
\end{align*}
\]

From (2.7), to calculate \(|I(\theta)|\), we need the functions: \(p_k(\theta), \frac{\partial p_k(\theta)}{\partial \theta_1}\), and \(\frac{\partial p_k(\theta)}{\partial \theta_2}\). We derive them from the model function \(f(a, b, t_k)\) with

\[
\begin{align*}
\theta_1 &= f(a, b, 0.1) \\
\theta_2 &= f(a, b, 12.8).
\end{align*}
\]

If \(f(a, b, t) = ae^{-bt}\),

\[
\begin{align*}
\Rightarrow \begin{cases} \\
\theta_1 &= ae^{-0.1b} \\
\theta_2 &= ae^{-12.8b} \\
\end{cases} \Rightarrow \begin{cases} \\
\theta_1 &= \theta_1 \left( \frac{\theta_1}{\theta_2} \right)^{0.1} \\
\theta_2 &= \frac{1}{12.7} \log \frac{\theta_1}{\theta_2} \\
\end{cases} \Rightarrow p_k(\theta) = \theta_1 \left( \frac{\theta_1}{\theta_2} \right)^{0.1-t_k}. \\
\end{align*}
\]

So,

\[
\frac{\partial p_k(\theta)}{\partial \theta_1} = \frac{12.8-t_k}{12.7} \left( \frac{\theta_1}{\theta_2} \right)^{0.1-t_k}, \quad \frac{\partial p_k(\theta)}{\partial \theta_2} = \frac{t_k-0.1}{12.7} \left( \frac{\theta_1}{\theta_2} \right)^{12.8-t_k}.
\]

If \(f(a, b, t) = \frac{1}{a+bt}\),

\[
\begin{align*}
\Rightarrow \begin{cases} \\
\theta_1 &= \frac{1}{a+0.1b} \\
\theta_2 &= \frac{1}{a+12.8b} \\
\end{cases} \Rightarrow p_k(\theta) = \frac{127}{(128-10t_k)^{\theta_1^{-1} + (10t_k-1)^{\theta_2^{-1}}}}. \\
\end{align*}
\]

So,

\[
\begin{align*}
\frac{\partial p_k(\theta)}{\partial \theta_1} &= \frac{127}{(128-10t_k)^{\theta_1^{-1} + (10t_k-1)^{\theta_2^{-1}}}} \frac{128-10t_k}{\theta_1}, \\
\frac{\partial p_k(\theta)}{\partial \theta_2} &= \frac{127}{(128-10t_k)^{\theta_1^{-1} + (10t_k-1)^{\theta_2^{-1}}}} \frac{10t_k-1}{\theta_2}.
\end{align*}
\]
If \( f(a, b, t) = a(1 + t)^{-b} \),

\[
\begin{align*}
\theta_1 &= a \cdot 1.1^{-b} \\
\theta_2 &= a \cdot 13.8^{-b}
\end{align*}
\]

\[ p_k(\theta) = \theta_1 \left( \frac{\theta_1}{\theta_2} \right)^{\log \frac{1.1}{13.8}} \cdot \theta_2 \left( \frac{\theta_1}{\theta_2} \right)^{\log \frac{1.1}{13.8}}.
\]

So,

\[ \frac{\partial p_k(\theta)}{\partial \theta_1} = \log \frac{1.1}{13.8} \left( \frac{\theta_1}{\theta_2} \right)^{\log \frac{1.1}{13.8}}, \quad \frac{\partial p_k(\theta)}{\partial \theta_2} = -\log \frac{1.1}{13.8} \left( \frac{\theta_1}{\theta_2} \right)^{\log \frac{1.1}{13.8}}. \]

### 2.3.2 Complexity with Numerical Integration

The results of numerical integration by both linear sampling method and Simple Monte Carlo sampling method [22] are listed in Table 2.1 and Figure 2.1. In the table and the figure, the left half contains the results for linear integration whereas the right half contains the results for simple Monte Carlo integration. Table 2.1 lists the numerical data with different sampling points whereas Figure 2.1 is the graphic representation of the corresponding data. In the tables, the first column represents the number of sampling points of the integration. The remaining \( i \)th column represents the complexity of the model corresponding to \( f_i \).

<table>
<thead>
<tr>
<th># of points</th>
<th>Linear Integration</th>
<th>Simple Monte Carlo Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( f_1 = ae^{-\theta_1} )</td>
<td>( f_1 = ae^{-\theta_1} )</td>
</tr>
<tr>
<td></td>
<td>( f_2 = \frac{ae^{-\theta_1}}{\theta_2} )</td>
<td>( f_2 = \frac{ae^{-\theta_1}}{\theta_2} )</td>
</tr>
<tr>
<td></td>
<td>( f_3 = a(1 + t)^{-\theta_2} )</td>
<td>( f_3 = a(1 + t)^{-\theta_2} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># of points</th>
<th>( f_1 = ae^{-\theta_1} )</th>
<th>( f_2 = \frac{ae^{-\theta_1}}{\theta_2} )</th>
<th>( f_3 = a(1 + t)^{-\theta_2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>2.4144664</td>
<td>2.4441487</td>
<td>2.509244</td>
</tr>
<tr>
<td>20000</td>
<td>2.3666880</td>
<td>2.3079888</td>
<td>2.538433</td>
</tr>
<tr>
<td>30000</td>
<td>2.3699368</td>
<td>2.2865738</td>
<td>2.511282</td>
</tr>
<tr>
<td>40000</td>
<td>2.3621502</td>
<td>2.2774117</td>
<td>2.502698</td>
</tr>
<tr>
<td>50000</td>
<td>2.3572473</td>
<td>2.2702339</td>
<td>2.498024</td>
</tr>
<tr>
<td>60000</td>
<td>2.3525286</td>
<td>2.2641767</td>
<td>2.488561</td>
</tr>
<tr>
<td>70000</td>
<td>2.3546926</td>
<td>2.2646140</td>
<td>2.477401</td>
</tr>
<tr>
<td>80000</td>
<td>2.3569996</td>
<td>2.2660966</td>
<td>2.484432</td>
</tr>
<tr>
<td>90000</td>
<td>2.3520303</td>
<td>2.2605165</td>
<td>2.483039</td>
</tr>
<tr>
<td>100000</td>
<td>2.3452102</td>
<td>2.2537408</td>
<td>2.477195</td>
</tr>
</tbody>
</table>

Table 2.1: The values of binomial model complexity.
2.4 Fisher Information

Fisher information is designed to assess the amount of information that a data set provides about a parameter in a parametric family. It has also been used in finding the performance bound of an estimator, investigating the asymptotic behavior of maximum likelihood estimates, as well as formulating Jeffreys prior in Bayesian inference. Geometrically, Fisher information induces a Riemannian distance measure [23, 24], where the parametric family of probability distribution function $f_{X|\Theta}(x|\theta)$ could be regarded as a Riemannian manifold embedded in the space of distribution. The Fisher
information matrix $I(\theta) = [I_{i,j}(\theta)]$ is defined \cite{2} by

$$I_{i,j}(\theta) \triangleq \text{Cov}_\theta \left[ \frac{\partial}{\partial \theta_i} \log f_{X|\Theta}(X|\theta), \frac{\partial}{\partial \theta_j} \log f_{X|\Theta}(X|\theta) \right]. \quad (2.8)$$

As discussed in Section 2.2.1, one of the two challenges of using FIA is to compute the Fisher information matrix $I(\theta)$, especially when the dimension of the parameter in the model is large. The mathematical models for the data in science and engineering are mostly with multinomial or independent normal distribution. In this section, we derive an unified, easy-to-use formula to compute Fisher information for those models having arbitrary dimensional parameter. Eliminating the need for numerical expectation or the second derivative of the likelihood function, the formula greatly simplifies the computation of the Fisher information. We also demonstrate the application of this formula in four areas of cognitive modeling: categorization, information integration, retention and psychophysics. We begin by the calculation of the Fisher information for general multinomial models.

### 2.4.1 Models with Multinomial Distribution

For the model $f_{X|\Theta}(x|\theta)$ with multinomial distribution, the parameter and sample vectors are

$$\theta = [\theta_1, \theta_2, \ldots, \theta_K]^T, \quad X|\Theta = [X_1|\Theta, X_2|\Theta, \ldots, X_N|\Theta]^T,$$

where $\{X_n|\Theta\}$ is independent whereas each follows a multinomial distribution of $C$ categories and sample size $n'$; that is, $X_n|\Theta \sim \text{Mult}_C(n', p_{n,1}(\theta), p_{n,2}(\theta), \ldots, p_{n,C}(\theta))$. $K$ is the parameter dimension number, $N$ is the random vector dimension number, and $C$ is the number of categories. Different choice of $\{p_{n,c}(\theta)\}$ corresponds to different
model. Therefore,
\[ f_{X_n|\theta}(x_n|\theta) = \left( \begin{array}{c} n' \\ x_{n,1}, \ldots, x_{n,C} \end{array} \right) \prod_{c=1}^{C} p_{n,c}(\theta)^{x_{n,c}} \]
with respect to counting measure on \( \{(x_{n,1}, \ldots, x_{n,C}) : \sum_{c=1}^{C} x_{n,c} = n', \ x_{n,c} \in \{0, \ldots, n'\} \} \).

Since \( \{X_n|\Theta\} \) is independent,
\[
\begin{align*}
    f_{X|\Theta}(x|\theta) &= \prod_{n=1}^{N} \left( \begin{array}{c} n' \\ x_{n,1}, \ldots, x_{n,C} \end{array} \right) \prod_{c=1}^{C} p_{n,c}(\theta)^{x_{n,c}}, \\
    \log f_{X|\Theta}(x|\theta) &= \sum_{n=1}^{N} \log \left( \begin{array}{c} n' \\ x_{n,1}, \ldots, x_{n,C} \end{array} \right) + \sum_{c=1}^{C} x_{n,c} \log p_{n,c}(\theta), \\
    \frac{\partial \log f_{X|\Theta}(x|\theta)}{\partial \theta_i} &= \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{x_{n,c}}{p_{n,c}(\theta)} \frac{\partial p_{n,c}(\theta)}{\partial \theta_i}, \\
    \frac{\partial^2 \log f_{X|\Theta}(x|\theta)}{\partial \theta_i \partial \theta_j} &= \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{-x_{n,c}}{p_{n,c}(\theta)} \frac{\partial p_{n,c}(\theta)}{\partial \theta_i} \frac{\partial p_{n,c}(\theta)}{\partial \theta_j} + \frac{x_{n,c}}{p_{n,c}(\theta)} \frac{\partial^2 p_{n,c}(\theta)}{\partial \theta_i \partial \theta_j}.
\end{align*}
\]

After checking that Fisher information regularity conditions [2] are hold for the \( f_{X|\Theta}(x|\theta) \) in question, we have the Fisher information matrix with a sample of size one.
\[
I_{i,j}(\theta) \triangleq \text{Cov}_\theta \left[ \frac{\partial}{\partial \theta_i} \log f_{X|\Theta}(X|\theta), \frac{\partial}{\partial \theta_j} \log f_{X|\Theta}(X|\theta) \right] \\
= -\text{E}_\theta \left[ \frac{\partial^2}{\partial \theta_i \partial \theta_j} \log f_{X|\Theta}(X|\theta) \right] \\
= \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{1}{p_{n,c}(\theta)} \frac{\partial p_{n,c}(\theta)}{\partial \theta_i} \frac{\partial p_{n,c}(\theta)}{\partial \theta_j} + \frac{\partial^2 p_{n,c}(\theta)}{\partial \theta_i \partial \theta_j} \\
= \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{1}{p_{n,c}(\theta)} \frac{\partial p_{n,c}(\theta)}{\partial \theta_i} \frac{\partial p_{n,c}(\theta)}{\partial \theta_j} \\
= \sum_{l=1}^{l} \frac{\partial p_l(\theta)}{\partial \theta_i} \frac{\partial p_l(\theta)}{\partial \theta_j}
\]
with \( p_l(\theta) \triangleq p_{n,c}(\theta), \ l = (n-1)C + c. \)

Finally,
\[ I(\theta) = P^T \Lambda^{-1} P \quad (2.9) \]
with
\[ P \triangleq \frac{\partial (p_1(\theta), \ldots, p_{NC}(\theta))}{\partial (\theta_1, \ldots, \theta_K)}, \]
\[ \Lambda \triangleq \text{diag}(p_1(\theta), \ldots, p_{NC}(\theta)), \]
where \( \frac{\partial (p_1(\theta), \ldots, p_{NC}(\theta))}{\partial (\theta_1, \ldots, \theta_K)} = [P_{i,j}] \) is the \( NC \times K \) Jacobian matrix with \( P_{i,j} = \frac{\partial p_i(\theta)}{\partial \theta_j} \), and \( \Lambda \) is the diagonal matrix with \( p_1(\theta), \ldots, p_{NC}(\theta) \) as the diagonal elements.

Although the formula (2.9) has a similar form as the formula for reparameterization of Fisher information [2], they are essentially different. Reparameterization formula reveals the relationship between Fisher information under different parameterization and all the matrices are square matrices. Formula (2.9) actually tells us how to calculate Fisher information explicitly and \( P \) could be rectangular instead.

With formula (2.9) to calculate \( I(\theta) \), we only need functions \( \{p_i(\theta)\} \) and \( \{\frac{\partial p_i(\theta)}{\partial \theta_k}\} \). \( \{p_i(\theta)\} \) is given from a particular model and its first derivative can easily be determined in analytical form. By placing restrictions on the parameter dimension number \( K \), the random vector dimension number \( N \) and the number of categories \( C \), we could apply (2.9) to many scenarios of mathematical modeling. In the following, we demonstrate its application to the models of categorization, information integration and retention.

Categorization

Categorization is one of the most fundamental and pervasive cognitive activities. It is the process by which distinct entities are treated as equivalent. Each class of equivalent entities is regarded as one category. For categorization models, the entity in each category usually refers to the stimulus which is represented as a vector in Euclidean space. In a categorization experiment, the stimulus is presented to the subject and one of several categories is chosen by the subject.
As the most general case of our application examples, the model of categorization assumes multinomial probability distribution and requires no special restrictions on $K$, $N$, and $C$. We now illustrate the application of (2.9) to Generalized Context Model (GCM) [25]. For this model, the probability of choosing category $c$ in response of stimulus $n$ is given by

$$p_{n,c} = \frac{\sum_{m \in C_c} s_{nm}}{\sum_{q} \sum_{p \in C_q} s_{np}},$$

where $C_c$ is the set of all indexes of the prototype stimuli in category $c$ and

$$s_{ij} = \exp \left( -s \cdot \left( \sum_{t=1}^{K-1} w_t \left| x_{it} - x_{jt} \right|^r \right)^{1/r} \right),$$

where $x_{it}$ is the $t$th component of multidimensional stimulus $i$. $\sum_{t=1}^{K-1} w_t = 1$. With $\theta = [\theta_1, \theta_2, \ldots, \theta_K]^T \triangleq [w_1, w_2, \ldots, w_{K-2}, s, r]^T$, we have

$$\frac{\partial p_{n,c}}{\partial \theta_k} = \left( \sum_{m \in C_c} \frac{\partial s_{nm}}{\partial \theta_k} \right) \left( \sum_{q} \sum_{p \in C_q} s_{np} \right) - \left( \sum_{m \in C_c} s_{nm} \right) \left( \sum_{q} \sum_{p \in C_q} \frac{\partial s_{np}}{\partial \theta_k} \right) \left( \sum_{q} \sum_{p \in C_q} s_{np} \right)^2,$$

with

$$\frac{\partial s_{ij}}{\partial \theta_k} = \left\{ \begin{array}{ll}
    \frac{s_{ij}}{r} \cdot \frac{T_{ij}^{\frac{1-r}{r}}}{T_{ij}^{\frac{1}{r}}} \cdot (|x_{ik} - x_{jk}|^r - |x_{iK-1} - x_{jK-1}|^r) & k = 1, \ldots, K-2 \\
    s_{ij} \cdot T_{ij}^{\frac{1}{r}} & k = K-1 \\
    s_{ij} \cdot \log s_{ij} \cdot \left( \frac{1}{r^2} \cdot \log T_{ij} + \frac{\sum_{t=1}^{K-2} w_t |x_{it} - x_{jt}|^r \log |x_{it} - x_{jt}|}{T_{ij}} \right) & k = K
\end{array} \right.$$
modeled as a stimulus which is generated from a factorial manipulation. In a typical information integration experiment, stimulus is presented to participants for categorization as one of two or more possible response alternatives. The data are scored as the proportion of responses in each category across various stimuli combinations.

The model of information integration represents the slightly more restricted case comparing to categorization models in the previous section. Similar to the categorization model, the stimulus is also represented by a vector in Euclidean space. For an information integration models, $K$ is the summation of all stimuli dimensions and $N$ is the production of all stimuli dimensions. A special model (maybe FLMP or LIM) with the response probability of one category in a two-factor experiment is considered as following.

$$p_{ij} = f(\theta_i, \lambda_j),$$

where $i \in \{1, \ldots, I\}$, $j \in \{1, \ldots, J\}$. So $K = I + J$, $N = I \cdot J$, and $C = 2$. With the restriction on $C$, this model will degrade to binomial distribution. Formula (2.9) can still be used because the binomial distribution is a special case of multinomial distribution. Moreover, there exists a simpler formula considering $C=2$, $\sum_{c=1}^{C} p_{n,c} = 1$ and the result of former derivation.

$$I_{i,j}(\theta) = \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{1}{p_{n,c}(\theta)} \frac{\partial p_{n,c}(\theta)}{\partial \theta_i} \frac{\partial p_{n,c}(\theta)}{\partial \theta_j}$$

$$= \sum_{n=1}^{N} \frac{1}{p_{n,1}(\theta)} \frac{\partial p_{n,1}(\theta)}{\partial \theta_i} \frac{\partial p_{n,1}(\theta)}{\partial \theta_j} + \frac{1}{p_{n,2}(\theta)} \frac{\partial p_{n,2}(\theta)}{\partial \theta_i} \frac{\partial p_{n,2}(\theta)}{\partial \theta_j}$$

$$= \sum_{n=1}^{N} \frac{1}{p_{n,1}(\theta)(1 - p_{n,1}(\theta))} \frac{\partial p_{n,1}(\theta)}{\partial \theta_i} \frac{\partial p_{n,1}(\theta)}{\partial \theta_j}$$

$$= \sum_{n=1}^{N} \frac{1}{p_{n}(\theta)(1 - p_{n}(\theta))} \frac{\partial p_{n}(\theta)}{\partial \theta_i} \frac{\partial p_{n}(\theta)}{\partial \theta_j}$$

$^1$Fuzzy Logical Model of Perception [26], Linear Integration Model [27]
with \( p_n(\theta) \triangleq p_{n,1}(\theta) = p_{ij}(\theta), \ n = (i - 1)J + j, \) and \( \theta = [\theta_1, \theta_2, \ldots, \theta_{I+J}]^T \triangleq [\theta_1, \theta_2, \ldots, \theta_I, \lambda_1, \lambda_2, \ldots, \lambda_J]^T. \) Consequently, we have the expression in matrix notation.

\[
I(\theta) = B^T \Delta^{-1} B \tag{2.10}
\]

with

\[
B \triangleq \frac{\partial (p_1(\theta), \ldots, p_{I+J}(\theta))}{\partial (\theta_1, \ldots, \theta_{I+J})},
\]

\[
\Delta \triangleq \text{diag}(p_1(\theta)(1 - p_1(\theta)), \ldots, p_{I+J}(\theta)(1 - p_{I+J}(\theta))).
\]

Now \( \Delta \in \mathbb{R}^{N \times N} \) which reduces the number of diagonal elements by one half as that of \( \Lambda \) in (2.9).

The formula (2.10) can be applied to any model with binomial distribution. For information integration model, since \( n = (i - 1)J + j \) and \( i \in \{1, \ldots, I\}, \ j \in \{1, \ldots, J\}, \)

we have

\[
i = \lceil(n - 1)/J \rceil + 1, \]

\[
j = n - J\lfloor(n - 1)/J \rfloor,
\]

where \( \lceil x \rceil \triangleq \max \{n \in \mathbb{Z} : n \leq x \}. \) Hence,

\[
p_n(\theta) = f(\theta_{\lceil(n-1)/J\rceil+1}, \theta_{n-J\lfloor(n-1)/J\rfloor}), \]

\[
\frac{\partial p_n(\theta)}{\partial \theta_k} = \frac{\partial f(\theta_{\lceil(n-1)/J\rceil+1}, \theta_{n-J\lfloor(n-1)/J\rfloor})}{\partial \theta_k}.
\]

**Retention**

Retention normally refers to the rate at which information is kept over time by human memory [28, 29]. Exponential model and power model are two common retention models [30]. The retention model is the most special case of our application examples. The parameter dimension \( K = 2 \). The number of categories \( C = 2 \). With
such restrictions, it is straightforward to show that
\[
|I(\theta)| = \sum_{n_{l=1}, \; n<l}^N \left( \frac{\partial p_n(\theta)}{\partial \theta_1} \frac{\partial p_l(\theta)}{\partial \theta_2} - \frac{\partial p_n(\theta)}{\partial \theta_2} \frac{\partial p_l(\theta)}{\partial \theta_1} \right)^2 p_n(\theta)(1-p_n(\theta))p_l(\theta)(1-p_l(\theta)).
\] (2.11)

Note that (2.7) is a special case of (2.11). Also derived previously in [20] for the case of \(K = 1\),
\[
|I(\theta)| = \sum_{n=1}^N \left( \frac{dp_n(\theta)}{d\theta} \right)^2 p_n(\theta)(1-p_n(\theta)).
\] (2.12)

By comparing (2.11) and (2.12), we might have the conjecture for the case with arbitrary \(K\) dimensional parameter.
\[
|I(\theta)| = \sum_{n_1<n_2<\cdots<n_K=1}^N \left( \frac{\partial(p_{n_1}(\theta), p_{n_2}(\theta), \ldots, p_{n_K}(\theta))}{\partial(\theta_1, \theta_2, \ldots, \theta_K)} \right)^2 \prod_{k=1}^K p_{n_k}(\theta)(1-p_{n_k}(\theta)).
\] (2.13)

This might be proved by mathematical induction on \(K\) as the initial cases with \(K = 1\) and \(K = 2\) are already true. However, by simulation and algorithm analysis, we find that (2.10) is much more efficient than (2.13) besides its representational simplicity.

### 2.4.2 Models with Normal Distribution

Interestingly, for models with independent normal distribution, the form of Fisher information formula turns out to be similar to that of multinomial distribution. The details are presented in the following.

For \(f_{X|\Theta}(x|\theta)\) with independent normal distribution, The parameter \(\theta = [\theta_1, \theta_2, \ldots, \theta_K]^T\), and \(X|\Theta = [X_1|\Theta, X_2|\Theta, \ldots, X_N|\Theta]^T\) with \(\{X_n|\Theta\}\) independent, \(X|\Theta \sim \mathcal{N}_N(\mu(\theta), \sigma(\theta))\), \(\mu \in \mathbb{R}^N\), and \(\sigma \in \mathbb{R}^{N \times N}\). Different choice of \(\mu(\theta)\) and \(\sigma(\theta)\) corresponds to different model. Since \(X|\Theta \sim \mathcal{N}_N(\mu(\theta), \sigma(\theta))\),
\[
f_{X|\Theta}(x|\theta) = (2\pi)^{-\frac{N}{2}}|\sigma(\theta)|^{-\frac{1}{2}} \exp \left( -\frac{1}{2} (x - \mu(\theta))^T \sigma(\theta)^{-1} (x - \mu(\theta)) \right)
\]
with respect to Lebesgue measure on $\mathbb{R}^N$. So

$$
\log f_{X|\Theta}(x|\theta) = -\frac{N}{2} \log(2\pi) - \frac{1}{2} \log |\sigma(\theta)| - \frac{1}{2} (x - \mu(\theta))^T \sigma(\theta)^{-1} (x - \mu(\theta)),
$$

$$
\frac{\partial \log f_{X|\Theta}(x|\theta)}{\partial \theta_i} = -\frac{1}{2} \left( \frac{\partial \log |\sigma(\theta)|}{\partial \theta_i} - \frac{\partial \mu(\theta)^T}{\partial \theta_i} \sigma(\theta)^{-1} (x - \mu(\theta)) + (x - \mu(\theta))^T \frac{\partial \sigma(\theta)^{-1}}{\partial \theta_i} (x - \mu(\theta)) - (x - \mu(\theta))^T \sigma(\theta)^{-1} \frac{\partial \mu(\theta)}{\partial \theta_i} \right),
$$

$$
I_{i,j}(\theta) = \frac{1}{2} \left( \frac{\partial^2 \log |\sigma(\theta)|}{\partial \theta_i \partial \theta_j} + \frac{\partial \mu(\theta)^T}{\partial \theta_i} \sigma(\theta)^{-1} \frac{\partial \mu(\theta)}{\partial \theta_j} + \sum_{m,n=1}^N \sigma_{m,n}(\theta) \frac{\partial^2 (\sigma(\theta)^{-1})_{m,n}}{\partial \theta_i \partial \theta_j} + \frac{\partial \mu(\theta)^T}{\partial \theta_i} \sigma(\theta)^{-1} \frac{\partial \mu(\theta)}{\partial \theta_j} \right)
$$

$$
= \frac{1}{2} \left( \frac{\partial^2 \log |\sigma(\theta)|}{\partial \theta_i \partial \theta_j} + \sum_{m,n=1}^N \sigma_{m,n}(\theta) \frac{\partial^2 (\sigma(\theta)^{-1})_{m,n}}{\partial \theta_i \partial \theta_j} + \frac{\partial \mu(\theta)^T}{\partial \theta_i} \sigma(\theta)^{-1} \frac{\partial \mu(\theta)}{\partial \theta_j} \right)
$$

$$
= \frac{\partial^2 \log |\sigma(\theta)|}{\partial \theta_i \partial \theta_j} + \sum_{m,n=1}^N \sigma_{m,n}(\theta) \frac{\partial^2 (\sigma(\theta)^{-1})_{m,n}}{\partial \theta_i \partial \theta_j} + \frac{\partial \mu(\theta)^T}{\partial \theta_i} \sigma(\theta)^{-1} \frac{\partial \mu(\theta)}{\partial \theta_j}.
$$

$\{X_n|\Theta\}$ is independent; that is, $\sigma$ is diagonal. Therefore,

$$
\frac{\partial^2 \log |\sigma(\theta)|}{\partial \theta_i \partial \theta_j} = \frac{\partial^2 \log \prod_{n=1}^N \sigma_{n,n}(\theta)}{\partial \theta_i \partial \theta_j}
$$

$$
= \sum_{n=1}^N \frac{\partial^2 \log \sigma_{n,n}(\theta)}{\partial \theta_i \partial \theta_j}
$$

$$
= \sum_{n=1}^N \frac{1}{\sigma_{n,n}(\theta)^2} \frac{\partial \sigma_{n,n}(\theta)}{\partial \theta_i} \frac{\partial \sigma_{n,n}(\theta)}{\partial \theta_j}
$$

$$
+ \frac{1}{\sigma_{n,n}(\theta)} \frac{\partial^2 \sigma_{n,n}(\theta)}{\partial \theta_i \partial \theta_j},
$$

$$
\sum_{m,n=1}^N \sigma_{m,n}(\theta) \frac{\partial^2 (\sigma(\theta)^{-1})_{m,n}}{\partial \theta_i \partial \theta_j} = \sum_{n=1}^N \frac{2}{\sigma_{n,n}(\theta)^2} \frac{\partial \sigma_{n,n}(\theta)}{\partial \theta_i} \frac{\partial \sigma_{n,n}(\theta)}{\partial \theta_j}
$$

$$
- \frac{1}{\sigma_{n,n}(\theta)} \frac{\partial^2 \sigma_{n,n}(\theta)}{\partial \theta_i \partial \theta_j}.
$$

Consequently,

$$
I_{i,j}(\theta) = \frac{\partial \mu(\theta)^T}{\partial \theta_i} \sigma(\theta)^{-1} \frac{\partial \mu(\theta)}{\partial \theta_j} + \sum_{n=1}^N \frac{1}{2\sigma_{n,n}(\theta)^2} \frac{\partial \sigma_{n,n}(\theta)}{\partial \theta_i} \frac{\partial \sigma_{n,n}(\theta)}{\partial \theta_j}.
$$
We also have

\[ I(\theta) = P^T \Lambda^{-1} P \]  

(2.14)

with

\[ P \triangleq \frac{\partial (\mu_1(\theta), \ldots, \mu_N(\theta), \sigma_{1,1}(\theta), \ldots, \sigma_{N,N}(\theta))}{\partial (\theta_1, \ldots, \theta_K)}, \]

\[ \Lambda \triangleq \text{diag}(\sigma_{1,1}(\theta), \ldots, \sigma_{N,N}(\theta), 2\sigma_{1,1}(\theta)^2, \ldots, 2\sigma_{N,N}(\theta)^2). \]

Similarly as in (2.9), \( P \) in (2.14) is the \( 2N \times K \) Jacobian matrix and \( \Lambda \) is diagonal matrix. Hence, it is straightforward to plug the mean vector \( \mu \) and covariance matrix \( \sigma \) into (2.14) to calculate the Fisher information matrix for models with independent normal distribution.

As a simple application of (2.14), Fechner’s logarithmic model of psychophysics [31] is considered as an example.

\[ X = \theta_1 \log(Y + \theta_2) + E, \]

where the data sample \( X \in \mathbb{R}^N \), independent variable \( Y = [Y_1, \ldots, Y_N]^T \in \mathbb{R}^N \), the parameter \( \theta = [\theta_1, \theta_2]^T \in \mathbb{R}^2 \), and the error \( E \sim \mathcal{N}(0, c) \) with constant variance \( c \in \mathbb{R} \). Then \( X|\Theta \sim \mathcal{N}_N(\mu(\theta), \sigma(\theta)) \) with \( \mu(\theta) = \theta_1 \log(Y + \theta_2) \) and \( \sigma(\theta) = cI_N \). So

\[
I(\theta) = P^T \Lambda^{-1} P \\
= \frac{1}{c} \left[ \begin{array}{c} \sum_{n=1}^{N} \log(Y_n + \theta_2)^2 \sum_{n=1}^{N} \theta_1 \frac{\log(Y_n + \theta_2)}{Y_n + \theta_2} \\
\sum_{n=1}^{N} \theta_1 \frac{\log(Y_n + \theta_2)}{Y_n + \theta_2} \sum_{n=1}^{N} \theta_1^2 \frac{\log(Y_n + \theta_2)}{Y_n + \theta_2} \end{array} \right].
\]

where 0 above is the null matrix with appropriate dimensions from the context. It is apparent that the derivation is much simpler than that in [20]. The advantage of (2.14) will be more prominent as the dimension of the parameter increases.
2.5 MDL Complexity Comparison

As discussed in Section 2.2, the two model selection criteria of FIA and NML differ only in the model complexity measure $C_{\text{FIA}}$ and $C_{\text{NML}}$. With the formula of Fisher information derived in Section 2.4, the computation of $C_{\text{NML}}$ is usually much more demanding than that of $C_{\text{FIA}}$. Under certain conditions [11], it turns out that $C_{\text{FIA}}$ and $C_{\text{NML}}$ are closely related. In the present section, we compare these two complexity measures for a particular model to gain further insight into their relationship.

In demonstrating the relationship between $C_{\text{FIA}}$ and $C_{\text{NML}}$, we consider the MDL complexity of a simple model with multinomial distribution (for more comprehensive discussing, please refer to [32]). The data of this model is multinomial $C$-tuple random vector $X|\Theta \sim \text{Mult}_C(n, \theta_1, \theta_2, \ldots, \theta_C)$ with $\theta = [\theta_1, \theta_2, \ldots, \theta_{C-1}]^T$ as the parameter. We derive the analytical expression of the complexity for this model using both FIA and NML criteria. The results are compared either as the function of sample size or as the function of number of categories.

2.5.1 FIA Complexity

The FIA complexity measure in (2.1) is restated as

$$C_{\text{FIA}} \triangleq \frac{k}{2} \log \frac{n}{2\pi} + \log \int_{\Omega} \sqrt{|I(\theta)|} d\theta.$$ 

For the multinomial model, the dimension of the parameter $k = C - 1$. $\Omega = \{(\theta_1, \theta_2, \ldots, \theta_{C-1}) : \theta_c \geq 0 \ \forall c, \ \sum_{c=1}^{C-1} \theta_c \leq 1\}$. Using the formula (2.9) of Fisher information
matrix with multinomial distribution, we have

\[
I(\theta) = \frac{\partial (\theta_1, \ldots, \theta_C)}{\partial (\theta_1, \ldots, \theta_{C-1})}^T \cdot \text{diag}(\theta_1^{-1}, \ldots, \theta_C^{-1}) \cdot \frac{\partial (\theta_1, \ldots, \theta_C)}{\partial (\theta_1, \ldots, \theta_{C-1})}
\]

\[
= \begin{bmatrix} I_{C-1} & -1_{(C-1)\times 1} \end{bmatrix} \cdot \begin{bmatrix} A & 0 \\ 0 & \theta_C^{-1} \end{bmatrix} \cdot \begin{bmatrix} I_{C-1} \\ -1_{1\times(C-1)} \end{bmatrix}
\]

\[
= A + 1_{(C-1)\times(C-1)} \cdot \theta_C^{-1},
\]

where \(1_{n \times m}\) is \(n \times m\) matrix with all elements equaling to one, and \(A = \text{diag}(\theta_1^{-1}, \ldots, \theta_{C-1}^{-1})\).

Then, we have

\[
|I(\theta)| = \begin{vmatrix}
\frac{1}{\theta_1} + \frac{1}{\theta_C} & \frac{1}{\theta_C} & \cdots & \frac{1}{\theta_C} \\
\frac{1}{\theta_C} & \frac{1}{\theta_2} + \frac{1}{\theta_C} & \cdots & \frac{1}{\theta_C} \\
0 & \frac{1}{\theta_C} & \cdots & \frac{1}{\theta_C} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{1}{\theta_C} & \frac{1}{\theta_C} & \cdots & \frac{1}{\theta_C} + \frac{1}{\theta_C} \\
\frac{1}{\theta_1} & 0 & \cdots & 0 & \frac{-1}{\theta_{C-1}} \\
0 & \frac{1}{\theta_C} & \cdots & 0 & \frac{-1}{\theta_{C-1}} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & \frac{1}{\theta_{C-2}} & \frac{-1}{\theta_{C-1}} \\
\frac{1}{\theta_C} & \frac{1}{\theta_C} & \cdots & \frac{1}{\theta_C} & \frac{1}{\theta_{C-1}} + \frac{1}{\theta_C}
\end{vmatrix}
\]

\[
= \begin{vmatrix}
B & -1_{(C-2)\times 1} \cdot \frac{1}{\theta_{C-1}} \\
1_{1\times(C-2)} \cdot \frac{1}{\theta_C} & \frac{1}{\theta_{C-1}} + \frac{1}{\theta_C}
\end{vmatrix}
\]

\[
= |B| \cdot \begin{vmatrix}
\frac{1}{\theta_{C-1}} + \frac{1}{\theta_C} - 1_{1\times(C-2)} \cdot \frac{1}{\theta_C} \cdot B^{-1} \cdot -1_{(C-2)\times 1} \cdot \frac{1}{\theta_{C-1}}
\end{vmatrix}
\]

\[
= \prod_{c=1}^{C} \frac{1}{\theta_c}.
\]
Using Dirichlet integration, we have
\[
\int_{\Omega} \sqrt{|I(\theta)|} d\theta = \int_{\Omega} \prod_{c=1}^{C} \theta_c^{-1/2} d\theta = \frac{\Gamma(1/2)^C}{\Gamma(C/2)}.
\]
Finally, the FIA complexity of the model is
\[
C_{\text{FIA}} = \frac{C - 1}{2} \log \frac{n}{2\pi} + \log \left( \frac{\pi^{C/2}}{\Gamma(C/2)} \right).
\]
(2.15)
The same result is also discussed by Rissanen [11] whereas Kontkanen [32] has more general extension.

In the equation (2.15), it is clear that \(C_{\text{FIA}}\) is no longer a linear function of the dimension of the parameter (i.e., \(k = C - 1\)). In contrast, for AIC or BIC, model complexity is measured as a linear function of the dimension of the model parameter. More specifically, because of the second term in (2.15), it is not a monotonic increasing function of the number of categories \(C\) for fixed \(n\) either. Using Stirling’s approximation, we find that the maximum complexity occurs if \(C\) is approximately equal to \(n\).

### 2.5.2 NML Complexity

If considering models in the sample space, the NML complexity is defined as
\[
C_{\text{NML}} \triangleq \log \int_{\hat{\theta}(x) \in \Omega} f_{X|\hat{\theta}}(x|\hat{\theta}(x)) dx,
\]
where \(\hat{\theta}(x)\) is MLE of the parameter \(\theta\) from the sample data \(x\). To obtain the exact expression for the \(C_{\text{NML}}\), we need the analytical solution of \(\hat{\theta}(x)\), which is an optimization problem. From the monotonicity of the \(\log(\cdot)\) function, the optimal estimation of the likelihood function is equivalent to that of the logarithm likelihood
function, which makes the analysis much easier if the objective function consists of multiplicative terms. The likelihood function of this model is the following.

\[
f_{X|\Theta}(x|\theta) = \left( \frac{n}{x_1, \ldots, x_C} \right)^C \prod_{c=1}^C \theta_{c}^{x_c},
\]

\[
\log f_{X|\Theta}(x|\theta) = \log \left( \frac{n}{x_1, \ldots, x_C} \right) + \sum_{c=1}^C x_c \log \theta_c.
\]

Since

\[
\frac{d^2 x_c \log \theta_c}{d\theta_c^2} = \frac{-x_c}{\theta_c^2} \leq 0,
\]

function \(x_c \log \theta_c\) of \(\theta_c\) is convex (Readers are referred to Appendix A for the characteristic properties of the convex function). \(\log f_{X|\Theta}(x|\theta)\) is the summation of convex function, which makes it also convex over the convex set \(\Omega\). With the constraint of \(\sum_{c=1}^C \theta_c = 1\), using a Lagrange multiplier, we would maximize the function

\[
\log f_{X|\Theta}(x|\theta) - \lambda \sum_{c=1}^C \theta_c
\]

over the region where \(\theta_c\) are nonnegative. By taking the first partial derivative versus \(\theta_c\) to be zero, we have

\[
\frac{x_c}{\hat{\theta}_c} = \lambda, \quad \forall c \in \{1, 2, \ldots, C\}.
\]

By considering \(\sum_{c=1}^C \hat{\theta}_c = 1\) and \(\sum_{c=1}^C x_c = n\), we have \(\lambda = n\). So

\[
\hat{\theta}_c = \frac{x_c}{n}, \quad \forall c \in \{1, 2, \ldots, C\}.
\]

If any \(x_c = 0\) we could eliminate that term from the summation and get the same result. We then have

\[
C_{NML} = \log \left( \sum_{0 \leq x_c \leq n \atop x_1 + x_2 + \ldots + x_C = n} \left( \frac{n}{x_1, \ldots, x_C} \right)^C \prod_{c=1}^C \left( \frac{x_c}{n} \right)^{x_c} \right). \quad (2.16)
\]
We calculate $C_{\text{NML}}$ in the above equation by considering all the possible $(n+C-1\choose C-1)$ data patterns in the sample space for the fixed number of categories $C$ and sample size $n$. For each data pattern we need to compute a multinomial coefficient and the multiplication of $C$ terms. There exists an efficient recursive algorithm using standard combinatorics [32]. Even so, (2.16) is still much computational heavier than (2.15).

2.5.3 The Comparison

After deriving the analytical expression for $C_{\text{FIA}}$ and $C_{\text{NML}}$, we compare them either as the function of sample size or as the function of the number of categories. The results are presented in Figure 2.2 through Figure 2.4.

Figure 2.2: Complexity as function of sample size.
Complexity of Multinomial Distribution with $N = 50$

Figure 2.3: Complexity as function of category.

Figure 2.4: Relative complexity difference with different categories.
The complexity measures are compared in the Figure 2.2 where the solid line represents $C_{\text{NML}}$ and dashed line stands for $C_{\text{FIA}}$. In this figure, we consider the models with 5 categories. The models differ in the sample size which ranges from 1 to 150. As could be observed, the two complexity measures are strikingly close to each other, with $C_{\text{NML}}$ being slightly greater than $C_{\text{FIA}}$ across all sample size. And both assume the shape of logarithm function of $n$, which is also indicated by their analytical expression.

In Figure 2.3, we plot the two complexity measures as the function of category, given a fixed sample size 50. Here, the x-axis is the number of categories instead of sample size as in Figure 2.2. It indicates that $C_{\text{FIA}}$ and $C_{\text{NML}}$ again follow closely one another. The nonlinearity of the MDL complexity curve is apparently due to the functional form of the model. Also plotted in the figure is the complexity measure of BIC. Note that the BIC curve captures good portion of $C_{\text{FIA}}$ and $C_{\text{NML}}$ curves at least for smaller category numbers. This confirm the assertion that $C_{\text{FIA}}$ is reducing to the complexity penalty of BIC as the sample size becomes large.

Considering two, three, four and five categories, we also compare the relative complexity difference with changing sample size in Figure 2.4. The relative complexity difference is defined to be $2(C_{\text{NML}} - C_{\text{FIA}})/(C_{\text{NML}} + C_{\text{FIA}})$, which measures how relatively two complexities differ. Such a measurement is used to compare the difference of complexities for the models with different category number. The curves in the figure suggest that the two complexity measures are asymptotically converge as the sample size approaching infinity. It also shows a bigger relative difference as the category increases from two to five.
In conclusion, the above results indicate that $C_{\text{FIA}}$ provides a relatively good
approximation to $C_{\text{NML}}$ for the multinomial model.

2.6 Example Application

To illustrate the advantage of using MDL in model selection, we compare the
model selection performance of retention models [28, 29] by MLE, BIC, CV, and FIA
criteria. For the retention models, the data sample $X_k|[a, b]^T \sim \text{Bin}(n, f(a, b, t_k))$,
where we choose sample size $n=20$, the independent variable $t_k=1, 2, 4, 8, 16$ and

$$f(a, b, t) = \begin{cases} 
1/(1 + t^a) & \text{(M1)} \\
1/(1 + a + bt) & \text{(M2)} \\
t^{-b}e^{-at} & \text{(M3)} 
\end{cases}$$

with the range of exponential parameter to be $[0, 10]$ and $[0, 100]$ otherwise.

Using (2.10), we first compute the $C_{\text{FIA}}$ of each model. The FIA complexity of
each model is 1.2361, 1.5479, and 1.7675. M1 is the simplest model with only scalar
parameter. M2 and M3 both have two-dimensional parameters, their complexity dif-
ference (i.e., 0.2196) is due to their functional form. After computing the complexity
measure, we generate data samples from the models. For each model, we generate
1,000 random parameter values sampled from Jeffreys prior over the sample space
where the formula for Fisher information is also applied. For each parameter which
indexes one particular distribution, we generate 100 random data samples. At last,
we fit all models to each of the 100,000 data samples and obtain judgment measures
for each criterion.

The model selection result is presented in Table 2.2. It consists of four matrices
corresponding to MLE, BIC, CV, and FIA. In each matrix, the element with row $i$ and
Table 2.2: The model recovery rates of three retention models.

<table>
<thead>
<tr>
<th>Selection Method/</th>
<th>Data Generating Model ($C_{\text{FIA}}$)</th>
<th>M1(1.2361)</th>
<th>M2(1.5479)</th>
<th>M3(1.7675)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Fitting Model</td>
<td>MLE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M1</td>
<td>22%</td>
<td>11%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>M2</td>
<td>41%</td>
<td>88%</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>M3</td>
<td>37%</td>
<td>1%</td>
<td>96%</td>
</tr>
<tr>
<td></td>
<td>BIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M1</td>
<td>91%</td>
<td>55%</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>M2</td>
<td>4%</td>
<td>44%</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>M3</td>
<td>5%</td>
<td>1%</td>
<td>88%</td>
</tr>
<tr>
<td></td>
<td>CV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M1</td>
<td>52%</td>
<td>40%</td>
<td>7%</td>
</tr>
<tr>
<td></td>
<td>M2</td>
<td>28%</td>
<td>53%</td>
<td>19%</td>
</tr>
<tr>
<td></td>
<td>M3</td>
<td>20%</td>
<td>7%</td>
<td>74%</td>
</tr>
<tr>
<td></td>
<td>FIA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M1</td>
<td>83%</td>
<td>37%</td>
<td>7%</td>
</tr>
<tr>
<td></td>
<td>M2</td>
<td>11%</td>
<td>62%</td>
<td>6%</td>
</tr>
<tr>
<td></td>
<td>M3</td>
<td>6%</td>
<td>1%</td>
<td>87%</td>
</tr>
</tbody>
</table>

column $j$ represents the percentage that we choose model $i$ from the data generated by model $j$ under that criterion. Hence the bigger the diagonal element, the better the model selection criterion. Therefore, we might use the trace of the matrix as a rough measure of the adequacy for the corresponding model selection criterion.

Let us first look at the model recovery rates using MLE. From the first column of the first matrix, we observe that, for the data coming from M1, only 22% of them are attributed to M1 but 41% and 37% of them are wrongly attributed to M2 and M3 respectively. And the more complex the model is, the bigger the diagonal element corresponding to the data generated by that model. This confirms that MLE will normally bias the more complex model. In the model selection result of BIC below,
such a bias is clearly rectified. But for the data from M2, BIC incorrectly chooses M1 most. CV generally selects the correct model most, but the advantage is not so prominent comparing with the result of FIA because the diagonal elements of the FIA matrix are approximately 10% bigger than those using CV criterion. From all the model recovery rates listed in the table, we conclude that FIA has the best performance among all the model selection criteria in our application example.

2.7 Summary and Conclusion

Model selection can proceed confidently when complexity, a well justified intrinsic property of the model, is available. In this chapter, we solve the problem of calculating FIA complexity measure by deriving an elegant formula to calculate Fisher information. The formula greatly simplifies the computation and can be applied to any model with multinomial distribution or independent normal distribution. It is warned that although the formula is straightforward to use, it should not be applied blindly. For special cases, it might be more efficient to use simpler formula. For non-independent normal models, the intermediate result of derivation can be used to calculate Fisher information albeit without the independent assumption. We also show that FIA complexity measure provides a good approximation to NML complexity measure, at least in the context of a multinomial model. Ideally, we would prefer NML to FIA. But at most time in practice, the parameter space has a much lower dimension than the sample space; thus, FIA is more computationally feasible than NML. We further illustrate the adequacy of FIA by comparing its performance with other model selection criteria with the selection of retention models.
CHAPTER 3

Componential Analysis

Another focus of this research is on decomposing model complexity to the contribution of each parameter. This chapter first discusses the motivation and significance of componential analysis in the domain of mathematical modeling. Illustrative simple examples are then provided to facilitate the understanding of the issue. Both global and local approaches to componential analysis are proposed after the formal definition of componential analysis. The simulation results on categorization and retention models are also presented.

3.1 Motivation and Significance

In mathematical modeling, we wish to identify lawful patterns of phenomena and developing theories that provide a concise explanation of the nature. As a link between experimental data and physical process, mathematical modeling provides a precise approach by inferring structural and functional properties of a physical process from observed data in explicitly mathematical expression. Put another way, theoretical principles that guide the model’s development could be instantiated by the quantitative statistical models. Therefore, the behavior of the model must originate from the theoretical ideas that motivate its implementation.
In the meantime, finding the critical influential aspect of the model motivates componential analysis. Determining the cause of a model’s performance is a substantial issue in mathematical modeling. However, it is often difficult to identify what aspects or characteristics of the model are critical in explaining physical processes. While the mathematical models are indexed by a vector of parameter, the theoretical principles and model parameters can not always be wrapped into one-to-one correspondence. Hence, identifying the parameter’s contribution to model’s behavior is a challenging problem. Similarly, model parameter faithfulness is defined as the role played by parameter to a model’s ability to fit arbitrary data. The investigation of the parameter faithfulness is also the goal of the current study.

A careful examination of the progress of model selection criterion also motivates componential analysis. Previously, Akaike Information Criterion (AIC) [4] and Bayesian Information Criterion (BIC) [5] have been proposed to solve model selection problem. As asymptotic approximation, both criteria resolve only the dimension of the parameter, but ignore the relative importance of each component of the parameter. Cross Validation (CV) [6] somehow takes into account the effect of elemental grandness. But how and how well it does this is not clear. Succeeding to the above criterion, Minimum Description Length principle (MDL) [11, 12, 7, 8, 9, 10, 33] takes advantage of the information in the individual component of the parameter to make more accurate decision. Componential analysis might be the tool to dig out that information from the model.

Componential analysis addresses the issue of quantification of parameter importance, which is one of the fundamental problems in mathematical modeling yet to be fully understood. Even if we could determine the simplest model that provides
an excellent description of observed data by a certain model selection criterion, the
critical aspect of the model that makes the model fit the data so well is often elusive.
Componential analysis is a statistical method to address the issue of evaluating pa-
rameter importance. Yielding a measure of parameter importance is the final product
of componential analysis.

The result of componential analysis could be used as a guideline in dealing with
practical problems. For example, in a grid search of maximum likelihood estimation,
we could choose the grid precision of certain parameter according to its importancy.
Identifying the importancy of each parameter also gives an instruction of selecting
relative few parameters to make the model more precise and simpler. In the following,
we further illustrate componential analysis by simple examples.

3.2 Simple Examples

The basic ideas of componential analysis are illustrated using tutorial models. The
models are rather artificial and the example is chosen for purely pedagogical reasons
in order to demonstrate as clearly as possible the problem of componential analysis.

As the first example, let’s consider the toy model shown in upper left panel of
Figure 3.1. The first toy model could generate only four data ensembles which are
presented by small circles. Intuitively, model complexity or flexibility has to do with
a model’s intrinsic ability to fit or generate diverse ensembles of data. In the first toy
model, all the data ensembles are created by varying one parameter of $\theta_1$. Therefore,
we could define

$$I_{\theta_1} = 1, \quad I_{\theta_2} = 0,$$

where $I_{\theta_1}$ is the importance index of $\theta_1$ and $I_{\theta_2}$ is the importance index of $\theta_2$. 

43
In the second toy model, there is one more data ensemble generated by varying parameter $\theta_2$. However, we still have

$$I_{\theta_1} > I_{\theta_2},$$

since $\theta_2$ accounts for two data ensembles which are less than four data ensembles accounted by $\theta_1$.

Next, let’s look at the third toy example in lower left panel of Figure 3.1. More specifically, we may quantify the importancy of each parameter as follows. There are totally eight data ensembles for this model. By fixing $\theta_1$, we lost $8 - 2 = 6$ data ensembles. Similarly, we lost $8 - 4 = 4$ data ensembles by fixing $\theta_2$. After normalizing,
we could define
\[ I_{\theta_1} = \frac{6}{6 + 4} = 0.6, \]
and
\[ I_{\theta_2} = \frac{4}{6 + 4} = 0.4. \]

The above procedure of componential analysis does not depend upon how the parameters are labeled. As shown in the last toy example in lower right panel of Figure 3.1, the data ensembles are not evenly distributed along the x-axis in contrast to the third toy model. The result of importancy of \( \theta_1 \) and \( \theta_2 \) is exactly the same by the similar procedure as in the third toy example. Therefore, componential analysis by counting just data ensembles is reparameterization invariant.

However, the above examples do not fully capture what componential analysis is about for at least two reasons. First, real models are defined in continuous parameter space instead of discrete in the above toy models. Secondly, the number of data ensembles is not finite or countable, but uncountably infinite. Hence, we can not simply count the data ensembles. These are the challenges in componential analysis and much more sophisticated mathematical and statistical methods are needed. For this purpose, we could utilize a highly efficient calculating formula for the Fisher information [33]. With this very useful statistical tool, various algorithms to find the importancy index are proposed. The empirical simulation results on mathematical models using proposed algorithms are also obtained.

### 3.3 The Componential Analysis Problem

Before investigating all the machinery directly applicable to componential analysis, we start with the formal definition of componential analysis.
3.3.1 Formal Definition of the Problem

In componential analysis, we are interested in quantifying the overall contribution of each of the model’s parameters to the mathematical model’s ability to fit arbitrary patterns of data. Mathematical model here refers to a parametric family of probability distribution function $f_{X|\Theta}(x|\theta)$ embedded in the space of distribution as a Riemannian manifold [1] with $x \in \mathcal{X}$ and $\theta \in \Omega$. $\mathcal{X}$ and $\Omega$ are the sets in sample space and the parameter space respectively [2]. Each point in $\Omega$ indexes a distribution in the model. The sample space or parameter space could be an Euclidean space with arbitrary dimensions.

The component in terms of componential analysis refers to the set of individual element in the parameter vector. Loosely speaking, componential analysis is the analytical method to evaluate the contribution of each parameter component to the model’s ability to fit diverse patterns of data, which is termed model complexity and will be described in full technical detail in Section 3.4. The contribution of the parameter to model complexity can be quantified as the importancy index of the parameter. The goal of componential analysis is to develop a justified measure of this index. Before we define the index of the importancy for each component, investigating the required characteristic attributions for those index is important. At a minimal level, the index should satisfy the following two properties, which can be regarded as axioms.

1. Proportionality

The importancy should be a relative concept, which entails we can only have the relative contribution of each component of the parameter. Thus, the importancy
index should be a percentage value; that is,

\[ I_\Phi \in [0, 1]. \tag{3.1} \]

The value \( I_\Phi \) means component \( \Phi \) accounts for \( I_\Phi \) of the model complexity out of 100 percent. Clearly, we have \( I_{\Theta} = 1 \) where \( \Theta \) denotes the whole parameter.

2. **Additivity**

The importancy should be an additive quantity. The additive property here signifies that if \( \Phi \) and \( \Psi \) are two different disjoint components of the parameter with importancy index \( I_\Phi \) and \( I_\Psi \) each, then the importancy index of \( \Phi \) and \( \Psi \) is

\[ I_{\Phi + \Psi} = I_\Phi + I_\Psi. \tag{3.2} \]

This property assumes that each component of the parameter should be mutually independent in the sense of contribution to the model complexity.

Given these properties, the importancy index of any component could be obtained as follows.

With the model parameter \( \Theta = (\Theta_1, \ldots, \Theta_K) \), the problem to find the importancy index of any component \( \Phi \) boils down to determine the importancy index \( I_{\Theta_i} \) of each coordinate \( \Theta_i \) because

\[ I_\Phi = \sum_{\Theta_i \in \Phi} I_{\Theta_i}. \tag{3.3} \]

according to property (3.2).

In summary, componential analysis is to evaluating the importancy index for the parameter in a parametric family of probability distribution. The analysis is sufficiently new; it has no direct precedent in statistics. Consequently, it would be
useful to review similar problems to generate feasible solution ideas and to gain new insights into the problem. Next, we will discuss closely related variable importance problems.

### 3.3.2 Variable Importance Problems

Although the variable importance and componential analysis address two different problems, the more scrutinizing on the ideas behind variable importance might shed new lights on componential analysis. One answer to variable importance problem is the semi-partial correlation coefficient and partial correlation coefficient [34, pp. 69–75]. Also, in dealing with categorization and regression problems, Breiman [35] proposed a method to define variable importance for the decision tree models. In the following, the definition of variable importance in each circumstance will be examined in detail.

**Semi-partial/Partial Correlation Coefficient**

The concept of semi-partial/partial correlation coefficient originates from the analysis of the linear regression models with multiple independent variables. In multiple regression model, there always exists linearity assumption; the estimation of the dependent variable $Y$ from a set of independent variable $\{X_i\}_{i=1}^I$ can be expressed as

$$\hat{Y} = \sum_{i=0}^I B_i X_i,$$

where $B_i$ is the regression coefficient.

Measuring the contribution of each independent variable to the multiple correlation is an important problems in multiple regression, where the multiple correlation $R$ denotes the measure of association between variables $Y$ and $\hat{Y}$. As one of such measure, semi-partial correlation coefficient is defined as following.
Definition 3.3.1 Let $R$ be the multiple correlation and $r_{Yi}$ be the correlation coefficient of $Y$ and $X_i$. Then the contribution measure of $X_i$ to the multiple correlation is defined as

$$sr_i \triangleq \sqrt{R^2 - r_{Yi}^2}.$$  

Intuitively, it is convenient to use Ballantine in Figure 3.2 to understand the meaning of this coefficient. The figure shows the variance of three variables with circles of unit area. The overlapping area of two circles represents the association of variables as the square of correlation coefficient. The total area of $Y$ covered by $X_1$ and $X_2$ represents the proportion of variance of $Y$ accounted for by the two variables, which is equal to $R^2$. The area represented by $sr_i^2$ is the portion of $Y$ overlapped uniquely by the independent variable $X_i$ which equals the increase in the squared multiple correlation when the variable is added to the regression equation.

An alternative way to describe each independent variable’s participation in determining $R^2$ is given by the partial correlation coefficient. The partial correlation coefficient also describes how much of the $Y$ variance is uniquely estimated by a

Figure 3.2: The Venn diagram for $X_1$ and $X_2$ with $Y$. 


certain independent variable. It is defined as

\[ pr_i \triangleq \sqrt{\frac{R^2 - r_{Y_i}^2}{1 - r_{Y_i}^2}}. \]

The squared partial correlation is the proportion of variance not associated with one independent variable but associated with the other. As illustrated in Figure 3.2,

\[ pr_i^2 = \frac{s_{r_i}^2}{s_{r_i}^2 + e}. \]

So \( pr_i^2 \) can be viewed as a transformation of the semi-partial correlation coefficient.

Semi-partial/partial correlation coefficients essentially take correlation as a measure of association between two variables and define the contribution of each independent variable to the multiple correlation accordingly. The goal of componential analysis is to investigate how and how much each component of the parameter contributes to the complexity of the model. Apparently, they are similar in the sense that both try to decompose positive scalar quantity into many facets of the cause.

However, subtle differences exist between each other. Firstly, \( R \) is practically data-dependent, as \( R \) could always be expressed in terms of correlation coefficients among variables, which are estimated from the data sample. Model complexity is data-independent; hence, it is an intrinsic property of the model. Secondly, unlike componential analysis, semi-partial/partial correlation analysis only consider the linear regression model. The linear regression model represented by \( \hat{Y} \) is the best approximation to \( Y \) by the criterion of the sum of the square error which is goodness of fit measure. It is known that least square estimation may not yield an optimal prediction for future phenomena because it is just one of many fit measures (e.g, mean squared error is another one). Therefore, the resulting fitting model may not
yield the best generalizability. In contrary, componential analysis is a more general technique, which could apply to the nonlinear models.

Despite the above differences, there are still several similarities between them. First of all, the quantity to be decomposed in semi-partial/partial correlation case is the multiple correlation $R$. The multiple correlation is not directly related to model complexity in componential analysis but may have subtle connections. The multiple correlation measures how much variance of dependent variable $Y$ accounted for by the optimal span of the dependent variables $\{X_i\}$. In a sense, $R$ is related to model complexity; the more complex the fitted model is, the larger the $R$ is. It means that the model complexity of the linear regression model represented by $\hat{Y}$ and multiple correlation $R$ have monotonic increasing relationship.

The second similarity between semi-partial/partial correlation and componential analysis lies in their duality facet of the cause. In semi-partial/partial correlation case, each factor is represented by independent variable $X_i$. And for componential analysis, the participation of each component of the parameter is investigated. For the linear regression model, it is observed that one can always switch the roles of $B_i$ and $X_i$ without affecting the model fit since $\hat{Y} = \sum B_i X_i$. If we regard $X_i$ as a random variable, we could treat $B_i$ as the parameter. If we regard $B_i$ as random variable, we could treat $X_i$ as the parameter. But $B_0$ is not associated with any independent variable in the linear regression model. One direct remedy is to add another pseudo variable $X_0$, which is constant with the value 1.

Furthermore, both semi-partial/partial correlation and componential analysis try to find the *unique* contribution of each one of the causes, which have interaction among each other. For example, in the case of semi-partial/partial correlation shown
in Figure 3.2, the overlap area between the independent variables represents their interaction. The unique contribution is difficult to define because it is always conditioned upon a given set of other facets. This is one of the challenges of componential analysis. Besides semi-partial/partial correlation coefficient, the other method of defining variable importance is under the context of decision tree models.

**Variable Importance in Decision Tree Models**

Classification And Regression Trees (CART) [35] are non-parametric decision models designed to solve categorization and regression problems. They both make sequential optimal decisions, which revolve around the following three issues:

1. The selection of the splits. Among a set of candidate splits \( S \), we need to choose the best one by certain *goodness of split* decision criterion.

2. The judgment of when to stop splitting. The critical issue is to grow a right sized tree, which strikes the best compromise between tree complexity and decision accuracy.

3. The designation of a value to every terminal node. Category number in the case of classification or the predicted response value in the case of regression is assigned to each leaf of the tree.

Since the procedure to build the tree model is a sequence of the one-step optimizations instead of the overall optimization, CART is not a *optimal* but a *honest* tree. Constructing a good decision tool useful in practice is the purpose of this model. An important criterion for a good decision procedure is that it not only produces accurate predictions, but also provides insight and understanding into the predictive structure
of the data. Specifically, besides its robust performance in decision making, CART assumes the ability to estimate the relative importance of the model variables.

The variables in the decision tree model are represented by the measurement vector and independent vector in classification and regression tree, respectively. In the case of classification tree, variable could be \textit{numerical} or \textit{categorical}. For a regression tree, the variables are called \textit{independent variables} or \textit{predictor variables}. As the definition of variable importance carries over entirely from classification tree to regression tree, we will focus on the case of classification tree from now on.

In constructing classification tree models, each node is represented by a set of data cases while an optimal split from a set of candidates is chosen to partition the node in the data space. The candidate splits are generated by the set of standard questions. The standard question is defined in such a way that every split is generated by one question, and each question depends on the value of only a \textit{single} variable.

The fundamental idea of choosing split is to select the one that makes the data in each of the descendant node “purer” than the data in the parent node. The measurement of impurity of node \(t\) is essentially the risk in terms of the decision theory. For the whole tree \(T\), the impurity is defined as the weight sum of the impurity of leaves of the tree weighted by the probability of any case goes to that leaf.

The measurement of the purity in classification model is realized by the impurity function \(i(t)\) of \(\{p(i|t)\}\), where \(p(i|t)\) is the proportion of the cases in node \(t\) belongs to class \(i\). The implementation of \(i(t)\) could simply be a \textit{entropy function}

\[
i(t) = -\sum_i p(i|t) \log p(i|t),
\]

53
or a *Gini index*

\[ i(t) = - \sum_{i \neq j} p(i|t)p(j|t). \]

With the impurity measure \( i(t) \), the *goodness of split* measure for split \( s \) could be defined as

\[ \Delta i(s, t) \triangleq i(t) - P_L \cdot i(t_L) - P_R \cdot i(t_R), \]

where \( t_L \) and \( t_R \) are the left and right sub-nodes generated by \( s \) with proportion of \( P_L \) and \( P_R \) of the cases from \( t \). The optimal split \( s^* \) is chosen such that the decrease in impurity is maximized among all the splits in set \( S \);

\[ \Delta i(s^*, t) = \max_{s \in S} \Delta i(s, t). \]

Let \( S_m \) be the set of all splits on variable \( x_m \) and \( S_m^c \) be the set of splits complementary to \( S_m \). The surrogate split \( \tilde{s}_m \in S_m \cup \tilde{S}_m \) on variable \( x_m \) is defined as

\[ \tilde{s}_m \triangleq \text{argmax}_{s_m \in S_m \cup \tilde{S}_m} p(t_L \cap t'_L) + p(t_R \cap t'_R), \]

where \( t'_L \) and \( t'_R \) are the sub-nodes generated by the split \( s_m \). Finally, the measure of the importance of variable \( x_m \) is defined as

\[ M(x_m) \triangleq \sum_{t \in T} \Delta i(\tilde{s}_m, t)p(t), \quad (3.4) \]

where \( T \) is the optimal subtree selected by the cross-validation or test sample procedure and \( p(t) \) is denoted as the probability of any case sent to node \( t \).

In comparison to the models in componential analysis, the decision tree models are non-parametric in nature and the importancy is concerned with the measurement of the independent variables of the model. The definition of the variable importance depends on the tree structure of the model; decreasing in impurity is the goal while
surrogate split is selected. Moreover, the procedure is suboptimal and plain analytical derivation is not applicable, because no model equations are defined or needed. On the other hand, for componential analysis, the parametric family of distribution has been investigated. The models have explicit structures in terms of the mathematical equations, where the importance of the parameter, instead of variable, on model complexity is required.

Although the variable importance in decision tree models and componential analysis address two different problems, the ideas to solve the problems could be the same. The essential idea of variable importance in decision tree models is to use the decrease in impurity of the model by the split on the variable to measure the importancy of that variable. Correspondently, we might think of the decrease in the Riemannian volume measure as the index for the importance of the parameter by fixing that parameter at certain point in the space of distribution. Explicitly, if $I_{\theta_i}$ is the importancy index of coordinate $\Theta_i$ in model $M$, it is reasonable to assume that

$$I_{\theta_i} \propto \min_{\theta_i} C - C_i, \quad (3.5)$$

where $C$ is the Riemannian volume of the whole model and $C_i$ is the Riemannian volume of the whole sub-model by fixing $\theta_i$. Geometrically, Riemannian volume is directly related to the concept of model complexity, which is the topic of the following section.

3.4 Complexity from Model Selection Criteria

Componential analysis is on decomposing model’s ability to fit arbitrary patterns of data to the contribution of each parameter. Such an ability is quantified as model
complexity. The concept of model complexity originates from model selection problems and could be derived from model selection criterion. Accordingly, we provide a brief overview of several model selection criteria. Various model selection criteria cause the diversity of model complexity measure.

As the intrinsic property of the model, model complexity characterizes the inherent flexibility of the model, which makes the analysis data independent. Close inspection of the model reveals three discernible facets of model complexity: the number of parameters, the parameter range, and the functional form. Next, we will review several model selection criteria and their induced complexity measures.

Introduced by Akaike in his seminal paper [4], Akaike Information Criterion (AIC) is developed using expected log likelihood function as the risk of decision and finding its asymptotic approximation. It is defined as

\[
\text{AIC} = -\log f_{X|\theta}(x|\hat{\theta}) + k,
\]

where \(\log(\cdot)\) is the natural logarithm function of base \(e\) and \(k\) is the dimension of the parameter. Similar to AIC, the Bayesian Information Criterion (BIC), which is a modification of maximum likelihood estimator by taking the first two order asymptotic of Bayes estimators under a special class of priors, is defined as

\[
\text{BIC} \triangleq -\log f_{X|\theta}(x|\hat{\theta}) + \frac{k}{2} \log(n),
\]

where \(n\) is the sample size.

The first term of AIC/BIC represents a goodness of fit measure whereas the second term represents a complexity measure. From the AIC/BIC viewpoint, model complexity is measured by the number of parameters. Consequently, each of the model’s parameter is assumed to contribute equally to overall complexity. This is a
unrealistic assumption as different parameter normally contribute differently to the model complexity. Obviously, componental analysis is solved in both cases, in which the parameter contributes equally to the model complexity.

Claimed to be the direct successor of AIC, Information COMPlexity criterion (ICOMP) [36] is formulated by considering the information theoretic complexity measure of the model. It defines the complexity as a measure of the degree of interdependency between the whole system and a simple enumerative composition of its subsystems or parts. For $X \sim \mathcal{N}_p(\mu, \sigma)$, $\mu \in \mathbb{R}^p$ and $\sigma \in \mathbb{R}^{p \times p}$. Using the estimated inverse Fisher information matrix, the information measure of complexity for model selection is formulated as

$$\text{ICOMP}(\text{IFIM}) \triangleq - \log f(x|\hat{\theta}) + C_{\text{ICOMP}}(\hat{I}^{-1}(\hat{\theta})), \quad (3.8)$$

where $C_{\text{ICOMP}}$ is defined as

$$C_{\text{ICOMP}}(\sigma) \triangleq \frac{p}{2} \log \left( \frac{\text{tr}(\sigma)}{p} \right) - \frac{1}{2} \log |\sigma|$$

$$= \frac{p}{2} \log \left( \frac{\frac{1}{p} \sum_{i=1}^{p} \lambda_i}{\prod_{i=1}^{p} \lambda_i^{\frac{1}{p}}} \right)$$

with $\{ \lambda_i \}$ as the set of eigenvalues of $\sigma$. However, the complexity measure of ICOMP depends on parameterization, which does not satisfy the requirement that model complexity should be the intrinsic property of the model.

Ye [37] also proposes the effective number of parameters or Generalized Degrees of Freedom (GDF), which can be used as a measure of the complexity of a general modeling procedure. For any general modeling procedure $\mathcal{M}$ represented by the mapping $\mathcal{M} : Y \rightarrow \hat{\mu}$, where $Y \sim \mathcal{N}(\mu, \sigma^2 I)$ with $\sigma$ assumed to be known and $\mu \in \mathbb{R}^n$, the GDF for the modeling process $\mathcal{M}$ was given by $D(\mathcal{M}) = \sum_{i=1}^{n} h_i^M(\mu)$,
where

\[ h_i^M(\mu) \triangleq \frac{\partial E_\mu[\hat{\mu}_i(Y)]}{\partial u_i} \]  \hspace{1cm} (3.9)

\[ = \lim_{\delta \to 0} E_\mu \left[ \frac{\hat{\mu}_i(Y + \delta e_i) - \hat{\mu}_i(Y)}{\delta} \right]. \]

However, the validity of generalization from the concept of degree of freedom to the general modeling procedure is not entirely justified.

With the mean minimum distance defined as the best fit averaged over all the points in the sample space, Dunn [38] uses it as the complexity measure of the algebraic model \( M \) represented by the ordered triple \( (P, F, Q) \), where \( P \subseteq \mathbb{R}^m \) is called the parameter domain, \( Q \subseteq \mathbb{R}^n \) is called outcome space, and \( F: \mathbb{R}^m \to \mathbb{R}^n \) is a deterministic mapping. \( R = F(P) \) is the prediction range of the model. As the model complexity is the model's propensity to fit arbitrary patterns of data, Dunn considers the best fitting point \( \hat{r} \in R \) to arbitrary point \( q \in Q \). Without the paradigm of probability, he uses least squared error (LSE) as the best fit measure, which can be interpreted as the distance between \( q \) and \( R \).

\[ \hat{E}_M(q) \triangleq \min_{r \in R} \|q - r\|^2 = \|q - \hat{r}\|^2. \]

Then, the mean minimum distance is defined as the best fit averaged over all the points in \( Q \).

\[ \overline{E}_M \triangleq \frac{\int_Q \hat{E}_M(q) dq}{\int_Q dq}. \]  \hspace{1cm} (3.10)

In order to evaluate \( \overline{E}_M \), \( Q \) must be bounded. To make the complexity measure independent of measurement scale, the scaled mean minimum distance is defined as

\[ \overline{S}_M \triangleq \int_{[0,1]^n} \hat{E}_M(q) dq. \]
Although he has the correct intuition that model complexity is closely related to the sum-of-all-the-best-fit, the uncertainty is missing in Dunn’s models.

From the geometric perspective, Myung et al. [39] observe that the geometry structure of probability distributions provides a framework, from which we could define model complexity in a more accurate and complete way. Based on the MDL criterion by Fisher information [2] approximation, Myung et al. go on to define the model complexity from the geometric perspective as follows

\[ C_{\text{FIA}} \triangleq \frac{k}{2} \log \frac{n}{2\pi} + \log \int_{\Omega} \sqrt{|I(\theta)|} d\theta, \quad (3.11) \]

where \( k \) is the dimension of the parameter, \( n \) is the sample size, \( \Omega \) is the range of the parameter, and \( I(\theta) \) is the Fisher information matrix as defined in (2.2). The geometry approach is more elegant and general, where insights into the relationship between the geometric framework and Bayesian inference can also be provided [18].

As the most recent formulation of MDL, Normalized Maximum-Likelihood distribution (NML) also provides a sophisticated model complexity measure. From coding perspective, Rissanen [12] shows the strong optimality of NML as a universal code for the data with the aid of a model. The code here refers to the probability distribution \( p \) of a random quantity. The code length is justified as \( \log(1/p) \) from Shannon’s information theory. Consequently, the NML complexity is then defined as follows

\[ C_{\text{NML}} \triangleq \log \int_{\hat{\theta}(x) \in \Omega} f_{X|\Theta}(x|\hat{\theta}(x)) dx. \quad (3.12) \]

This measure captures the full scope of model complexity, thereby representing a better solution to the long-standing problem of quantifying model complexity. Since \( C_{\text{FIA}} \) can be obtained as a Taylor series expansion of \( C_{\text{NML}} \) under the assumption of large sample sizes. Therefore, \( C_{\text{FIA}} \) is a partial solution to that problem as the higher
order contributions of complexity are ignored in its complexity measure. We have verified that $C_{FIA}$ provides a pretty good approximation to $C_{NML}$ in the context of saturated multinomial model [33]. After reviewing model complexity measure, we go on to propose possible approaches to solve the componential analysis problem. Both global and local approaches to componential analysis are proposed with mathematical gory detail in the following part.

### 3.5 Global Approaches

For a given model complexity measure, global approach is to evaluate importance index induced by the complexity measure which is already a global property. For example, the method motivated by decision tree mentioned in (3.5) is a global approach.

Before the global approach is presented, some notation is defined to facilitate the explanation. Let $\mathcal{M} = \{f(x|\theta) : x \in \mathcal{X}, \theta \in \Omega\}$ be the model that we are interested in and it satisfies some regularity condition. Without loss of generality, let us assume that parameter $\Theta$ consists of two components; $\Theta = (\Phi, \Psi)$. Denote $\mathcal{M}_\phi = \{f_\phi(x|\psi) : (\phi, \psi) \in \Omega\}$ which is the model derived from $\mathcal{M}$ by fixing $\Phi = \phi$. $C_\phi$ is denoted as the complexity measure of the model $\mathcal{M}_\phi$. $\mathcal{M}_\psi$ and $C_\psi$ are defined in the similar way. Suppose we want the index $I_\Phi$ which measures the importance of the component $\Phi$ on the model complexity.

#### 3.5.1 Best Point Method

To find the importance index of $\Phi, \Psi$ is regarded as nuisance parameter. The index $I_\Phi$ is calculated from the complexity $C_\psi$ of $\mathcal{M}_\psi$ which is the model from $\mathcal{M}$
with $\Psi$ fixed at some “best” point $\psi$; that is,

$$I_\Phi \propto h(C_\psi),$$

where $h$ is some reasonable monotonic function. How to quantify the term best, and after that, how to find the best $\psi$ and function $h$ are the critical issues to be resolved.

### 3.5.2 Average Method

Instead of trying to find one best $\psi$, complexity measure $C_\psi$ is calculated for the model $M_\psi$ at every $\psi$. Then, the index $I_\Phi$ is defined as the average of those complexity measure $C_\psi$. Therefore,

$$I_\Phi \propto E[h(C_\psi)].$$

How to find the prior distribution of $\Psi$ to get the best average $E[h(C_\psi)]$ is of critical importance. Reference prior might be the solution, which is defined as the following:

On each model $M_\psi$, first find the Jeffreys prior of $\Phi$. Then using this prior to integrate out parameter $\phi$ of the model $M$. The resulting marginal distribution forms a new model $M_{new}$. The Jeffreys’ prior on this new model is the reference prior.

Similar to the above method, we could also proceed to find a new model $M_{new}$ with $\Phi$ as the parameter. Then $I_\Phi$ is defined to be the function of complexity $C_{new}$ of the model $M_{new}$; that is,

$$I_\Phi \propto h(C_{new}).$$

$M_{new}$ is constructed as the following: For each $\phi$, the Jeffreys prior $J_\phi(\psi)$ on the model $M_\phi$ is obtained first. Then using this prior, $M_{new}$ is constructed by integrating out $\psi$ from the distributions in $M$. More explicitly, for any $f(x|\phi) \in M_{new}$, there exists
\[ f(x|\theta) \in \mathcal{M}, \text{ such that} \]
\[ f(x|\phi) = \int_{\psi} f(x|\theta) J_\phi(\psi) d\psi. \]

### 3.6 Local Approaches

Without borrowing certain model complexity measure explicitly, local approaches investigate how the complexity is evaluated at each point in the model. Thereafter, importancy index is defined locally. The global measure could also be induced by the procedure combining average and integration.

More explicitly, since \( \Theta_i \) will have a different contribution to model complexity at different points in the model manifold \( \mathcal{M} \), the global index quantity \( I_{\Theta_i} \) will be the aggregation of local importancy index in \( \mathcal{M} \), which is denoted as \( I_{\Theta_i}(\theta) \). Again, \( I_{\Theta_i}(\theta) \) is the local importancy index of coordinate \( \Theta_i \) around the point \( \Theta = \theta \) in \( \mathcal{M} \).

The standard differential geometric approach is to approximate the neighborhood around \( \theta \) by its tangent space which is a vector space with Fisher information \( I(\theta) \) as the metric tensor. Furthermore, the reparameterization invariant volume element measure on this space is
\[ \sqrt{|I(\theta)|} d\theta. \quad (3.13) \]

From this point on, there are two roads to proceed: sub-model method and eigenvalue method. Both approaches are set forth to evaluate the local importancy index first.

#### 3.6.1 Sub-model Method

For the sub-model approach, in considering (3.13), it is natural to assume that
\[ I_{\Theta_i}(\theta) \propto \sqrt{|I_i(\theta)|}, \]
where \( I_i(\theta) \) is the Fisher information of the model with all the parameter fixed at \( \Theta = \theta \) except the \( i \)th coordinate. According to equation (3.1), The local importancy index is defined as

\[
I_{\theta_i}(\theta) \triangleq \frac{\sqrt{|I_i(\theta)|}}{\sum_j \sqrt{|I_j(\theta)|}} = \frac{\sqrt{I_i(\theta)}}{\sum_j \sqrt{I_j(\theta)}}.
\] (3.14)

As \( I_i(\theta) \) is a scalar, its determinant equals to itself. Let \( J(\theta) \) be the Jeffreys prior for the whole model, then the importancy index of \( \Theta_i \) is

\[
J_{\Theta_i} \triangleq \int_\Omega J(\theta) I_{\theta_i}(\theta) d\theta.
\] (3.15)

With \( I(\theta) \) as the Fisher information of the whole model, we know that Jeffreys prior \( J(\theta) = \frac{\sqrt{|I(\theta)|}}{\int_\Omega \sqrt{|I(\theta)|} d\theta} \). Substitute \( J(\theta) \) and (3.14) into (3.15), we have

\[
J_{\Theta_i} = \frac{\int_\Omega \frac{\sqrt{|I(\theta)| I_{\theta_i}(\theta)}}{\sum_j \sqrt{I_j(\theta)}} d\theta}{\int_\Omega \sqrt{|I(\theta)|} d\theta}.
\] (3.16)

After we know \( J_{\Theta_i} \), we could get \( J_\Phi \) for any component \( \Phi \) using (3.3).

### 3.6.2 Eigenvalue Method

The eigenvalue approach is motivated by the eigenvalue decomposition of Fisher information matrix. Because the Fisher information matrix \( I(\theta) \) is symmetric, there always exists an orthogonal matrix \( D = [D_1, \ldots, D_K] \) and a diagonal matrix \( \Lambda \), such that

\[
I(\theta) = D\Lambda D^T,
\]

where the set of diagonal elements \( \{\lambda_i\} \) of \( \Lambda \) is the set of eigenvalues of \( I(\theta) \) and \( D_i \) is the eigenvector corresponding to \( \lambda_i \). As \( I(\theta) \) is a covariance matrix which is positive definite, \( \lambda_i \) is positive. Therefore,

\[
\sqrt{|I(\theta)|} = \prod_i \lambda_i^{\frac{1}{2}}.
\]
which could be interpreted that the contribution to the model complexity along eigenvector $D_i$ is $\lambda_i^{1/2}$. As $D$ is orthogonal, we could imagine a hyper-cube in $K$ dimensional Euclidean space, whose $i$th edge has length $\lambda_i^{1/2}$. We may present this hyper-cube by a vector $V = [\lambda_1^{1/2}, \ldots, \lambda_K^{1/2}]^T$ in the coordinate system with $\{D_i\}$ as its bases. Projecting the contributions back to the normal coordinate system with bases $\{e_i\}$ and adding them up by their absolute value, we have vector $U = D'V = [U_1, \ldots, U_K]^T$ where $D'$ comes from $D$ with each element replaced by its absolute value. If we assume that

$$I_{\theta_i}(\theta) \propto U_i,$$

we could define the local importancy index

$$I_{\theta_i}(\theta) \triangleq \frac{U_i}{\sum_j U_j}. \quad (3.17)$$

Now, similarly as the approach in Section 3.6.1, we could aggregate $I_{\theta_i}(\theta)$ to get $I_{\Theta_i}$ by the appropriate prior for the whole model. Next, we present the simulation result of the method motivated by the local approaches and the decision tree.

### 3.7 Simulation Results

Among all the algorithms presented above, the most comparable methods are sub-model method and eigenvalue method. We choose the GCM model [25] to test these methods and compare their simulation results. For this model, the probability of choosing category $c$ in response of stimulus $n$ is given by

$$p_{n,c} = \frac{\sum_{m \in C_c} s_{nm}}{\sum_q \sum_{p \in C_q} s_{np}},$$

where $C_c$ is the set of all indexes of the prototype stimuli in category $c$ and

$$s_{ij} = \exp \left( -s \cdot \left( \sum_{t=1}^{K-1} w_t |x_{it} - x_{jt}|^{r} \right)^{1/r} \right),$$

64
where \( x_{it} \) is the \( t \)th component of multidimensional stimulus \( i \) and \( \sum_{t=1}^{K-1} w_t = 1 \).

<table>
<thead>
<tr>
<th>method</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
<th>( w_4 )</th>
<th>( w_5 )</th>
<th>( s )</th>
<th>( r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub-model</td>
<td>0.3142</td>
<td>0.2313</td>
<td>0.1601</td>
<td>0.1319</td>
<td>0.1365</td>
<td>0.0107</td>
<td>0.0153</td>
</tr>
<tr>
<td>eigenvalue</td>
<td>0.2755</td>
<td>0.2152</td>
<td>0.1699</td>
<td>0.1507</td>
<td>0.1527</td>
<td>0.0147</td>
<td>0.0212</td>
</tr>
</tbody>
</table>

Table 3.1: The importance index for each component of the GCM model.

The results of the methods applying to the GCM model are listed in Table 3.1. The two rows in the table are for the sub-model and eigenvalue approaches presented in Section 3.6.1 and Section 3.6.2 respectively. The importance index of five weight parameter \( w_i \), sensitivity parameter \( s \), and Minkowski metric parameter \( r \) are compared. From the computer simulating data, it is observed that the ordinal importance of each parameter is shown to be coincidental for both approaches. For example, parameter \( w_1 \) assumes 0.3142 importance by the sub-model method whereas 0.2755 by the eigenvalue method. Both are the largest among all the parameters.

We also make the simulation using the method motivated by the decision tree model. The results are obtained from a simple method motivated by the idea from the definition of variable importance in decision tree models. If \( I_{\theta_i}(\theta) \) is the local importance index of coordinate \( \Theta_i \) around the point indexed by parameter \( \Theta = \theta \in \mathbb{R}^K \) in model \( M \), it is straightforward to assume that

\[
I_{\theta_i}(\theta) \propto \sqrt{|I(\theta)|} - \sqrt{|I_i(\theta)|}, \quad (3.18)
\]

where \( I_i(\theta) \) is the Fisher information of the model with only \( i \)th coordinate of the parameter fixed at \( \Theta_i = \theta_i \). We use retention model to test this idea. For the retention models, the data sample \( X_k|[a, b]^T \sim \text{Bin}(n, f(a, b, t_k)) \), where we choose the sample...
size $n = 20$, the independent variable $t_k = 1, 2, 4, 8, 16$, and

$$f(a, b, t) = \begin{cases} \frac{1}{a + t^b} & \text{(M1)} \\ \frac{1}{1 + a + bt} & \text{(M2)} \\ t^{-b} e^{-at} & \text{(M3)} \end{cases}$$

where $a \in [5, 10]$ and $b \in [5, 10]$.

By normalizing the quantity in (3.18), the simulation result of local importance index for the example model M1 as the functions of $a$ and $b$ is listed in Figure 3.3. The global importance index of each parameter through averaging is also calculated and the results are given in the table 3.2.

![Figure 3.3: Local parameter importance index of parameter $a$ and $b$ for M1.](image)

The above index is sensitive to the length of parameter interval. For example, in model M2, if we extend the range of parameter $b$ to be $[5, 20]$, then the importance index of $b$ will increase from 0.3484 to 0.6356.
<table>
<thead>
<tr>
<th>Model</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.5506</td>
<td>0.6516</td>
<td>0.0021</td>
</tr>
<tr>
<td>b</td>
<td>0.4494</td>
<td>0.3484</td>
<td>0.9979</td>
</tr>
</tbody>
</table>

Table 3.2: The importance index for each component of retention models.

3.8 Summary

This chapter introduces componential analysis. The problem of componential analysis is to investigate how and how much each parameter affects mathematical model’s ability to fit arbitrary patterns of data. Yielding a measure of parameter importance is the final product of componential analysis. The problem of componential analysis is formalized and prior approaches to the similar problems are examined. Various model complexity measures derived from model selection criteria are also reviewed. Several approaches to define and estimate importance index of each parameter are proposed, where the promising results are also presented.
CHAPTER 4

Bandwidth Efficient Nonsystematic Turbo Codes

This chapter presents a class of channel coding schemes that adopt a nonsys-
tematic turbo codes and a bandwidth efficient quadrature amplitude modulation to
achieve high-performance communication over noisy channel with low signal-to-noise
ratio. We use nonsystematic constituent code to expand our searching domain, and
apply puncturing and a 16QAM to improve the bandwidth efficiency. We also present
the performance of such a coding scheme.

4.1 Introduction

The issue of error detection and correction is critical for efficient and reliable in-
formation processing systems. Error control coding has great practical importance in
implementing reliable data transmission over noisy communication links, data stor-
age media such as random access memory and compact discs, and other applications
where the integrity of data is important. With the received data, error detection
methods enable one to determine whether the data has been corrupted during trans-
mission. Although error detection methods does not tell us where the transmitted
data is corrupted, error correction scheme allows us to recover the original information
from the contaminated noisy signal.
As an important component of communication system, channel coding embodies an error detection or an error correction code applying to digital signals before transmission. Common channel coding schemes include parity-check codes, cyclic codes, BCH codes, and convolutional codes. Besides the practical importance, channel coding has sound theoretical backup. Noisy channel coding theorem developed by Shannon [16] in 1948 proved that there exists a coding scheme by which the transmission rate will reliably approach an asymptotic limit termed \textit{channel capacity}. Searching for such good channel codes has been the main effort of many coding theorists since then.

Of all practical error correction methods known to date, turbo codes, introduced by Berrou et al. [17] in 1993, come closest to approaching the theoretical limit of maximum information transfer rate over a noisy channel. Turbo codes are a class of high-performance error correction codes achieving maximal information transfer over a bandwidth-limited communication link in the presence of noise. To further improve the transmission spectral efficiency, and to expand our searching domain, we have designed and implemented a computer simulation system to test bandwidth efficient nonsystematic turbo coding scheme. The system diagram is shown in Figure 4.1.

In the later section of the chapter, we discuss the blocks in Figure 4.1 in detail. Section 4.2 reviews nonsystematic binary turbo codes. Section 4.3 focuses on 16QAM Gray mapping and puncturing pattern. We also explain how to get the input from the received channel symbol to the binary turbo decoder. In Section 4.4, we discuss the process of using independent Gaussian assumption to find the threshold of such coding schemes. In the last section, we show simulation results.
4.2 Binary Nonsystematic Turbo Codes

A general turbo code consists of two or more constituent codes connected by interleaver in parallel [40]. In normal turbo codes, the output from one of the constituent codes is the binary source without coding. Such turbo codes are typically called systematic turbo codes. Nonsystematic turbo codes expand the searching domain by dropping such an assumption and use general convolutional codes. The convolutional codes cooperate by exchanging extrinsic information. The block diagram of a nonsystematic binary turbo encoder is shown in Figure 4.2.

The interleaver permutes the bit sequence to distribute the burst noise over a long time period, making the error easy to correct at each time instance. After traveling through Additive White Gaussian Noise (AWGN) channel, the outputs of each constituent code $Y_1$ and $Y_2$ will be transformed to normally distributed random variable $\Lambda_1$ and $\Lambda_2$:

$$\Lambda_i = Y_i + E,$$
where the error $E \sim \mathcal{N}(0, \sigma^2)$ with variance $\sigma^2$ for $i = 1, 2$. The block diagram of a turbo decoder for a nonsystematic turbo code is shown in Figure 4.3.

$L_{ij}$ in the figure is the extrinsic information from constituent code $i$ to constituent code $j$. The measure is typically Logarithm Likelihood Ratio (LLR). BCJR [41] algorithm can be implemented as a Maximum A Posteriori (MAP) decoder to decode each constituent code. Such Bayesian statistics estimator calculates posterior probability
measure from the prior probability measure. In the binary turbo decoder, the posterior LLR from one constituent code is used as prior LLR to the other constituent code. The process is iteratively continued and the decoding is stopped after some convergence criteria or a certain number of iterations.

The choice of the constituent code is essential for the performance of the turbo codes. Typically, recursive systematic convolutional codes are used as constituent codes for turbo codes. It has been shown [40, 42] that there exist good nonsystematic recursive convolutional codes that can be used as the constituent codes for turbo codes. In this study, one of the low-complexity high-performance nonsystematic recursive convolutional codes that we found is shown in Figure 4.4.

![Figure 4.4: The nonsystematic turbo encoder.](image)

4.3 Bandwidth Efficient Scheme

To improve the transmission spectral efficiency, Le Goff et al. [43] introduced a bit interleaved turbo coded modulation scheme that combines binary turbo codes with a bandwidth efficient modulation. In this study, we use the similar pragmatic
method also mentioned by Viterbi [44]. The binary turbo encoder output is punctured by a certain puncturing pattern. Then, we use 16QAM Gray mapping to map the punctured block of binary digits to 16QAM symbols. At receiver, we first convert the received bandwidth efficient modulation to equivalent Binary Phase-Shift Keying (BPSK) symbols as explained in Section 4.3.3, and then decode using a binary turbo decoder.

4.3.1 Puncturing Patterns

The purpose of puncturing is to increase the rate of turbo codes by periodically deleting selected bits. Different code rate is achieved by changing the percentage of the punctured bits. By puncturing two bits in a group of six bits, the rate of the original codes will increase from one third to one half. We have verified that, for the same rate, different puncturing patterns also affect the performance. Several of puncturing patterns are shown in Figure 4.5 among many others that we have tried. The upper three and lower three patterns all drop the bits at the same location. The permutation of the bits $U_i, i \in \{1, 2, 3, 4\}$ relates to the Gray mappings explained in the following.

4.3.2 Gray Mappings

Having the punctured group of bits, we map them to high order symbols by 16QAM Gray mappings. The rate of the codes after mapping is two bits per channel use. We find the number of all the possible unique 16QAM Gray mappings as follows. First, we ignore symmetry. As the constellation is two-dimensional, we consider each dimension separately. For one-dimension Gray mapping of two bits, let’s consider the ring in Figure 4.6.
We can construct different one-dimension Gray mapping by starting from any point and roaming around the ring clockwise or counterclockwise. Therefore, there are 8 such Gray mappings in one-dimension including symmetry. Because the resulting constellation is the direct product of each dimension, we will have 64 two-dimensional 16QAM Gray mappings by ignoring symmetry.

Let’s then consider symmetry in the Gray mapping. The set of Gray mapping with same symmetry contains the mappings obtained by all possible flipping and rotation
transformations. Consequently, there are 8 different symmetries for a unique mapping shown in Figure 4.7. And the sets of symmetry mappings of a unique mapping are disjoint, otherwise the two mappings are not unique up to symmetry. Further more, if we assume \( P(u_i = 0) = P(u_i = 1) \), we could get pairwise equivalent mapping by exchanging each bit from 1 to 0 or from 0 to 1. As the result, we have \( 64/8/2=4 \) unique mappings up to symmetry. Several unique gray mappings already exist in the literature [43, 45]. These gray mappings may also give good performance combined with a certain constituent code and puncturing pattern. As an example, Figure 4.8 shows one of the unique mappings.

![Figure 4.7: The symmetry Gray mapping patterns.](image)

4.3.3 Logarithm Likelihood Ratio

Under independent Gaussian assumption, we use posterior LLR to estimate the corresponding BPSK channel output and use it as the input to the binary turbo decoder. Suppose \( (x, y) \) is the received 16QAM symbol, where \( x \) is the in-phase and \( y \) is the quadrature component, and \( u_i, i \in \{1, 2, 3, 4\} \) are the corresponding binary digits at time instance \( k \). If assuming that \( P(u_i = 0) = P(u_i = 1) \), we could calculate
the posterior LLR as follows.

\[
\Lambda(u_i) = \log \frac{P(u_i = 1|x, y)}{P(u_i = 0|x, y)} = \log \frac{p(x, y|u_i = 1)}{\sum_{s \in S^-} p(x, y|s)} \\
= \log \frac{\sum_{s \in S^+} p(x|s)p(y|s)}{\sum_{s \in S^-} p(x|s)p(y|s)} = \log \sum_{s \in S^+} \exp \left( \frac{(x - x_s)^2 + (y - y_s)^2}{-N_0} \right),
\]

where \( S^+_i \triangleq \{16\text{QAM symbol}| u_i = 1\} \) and \( S^-_i \triangleq \{16\text{QAM symbol}| u_i = 0\} \). \((x_s, y_s)\) is the coordinate of the symbol \(s\), and \(N_0\) is the one-side power spectral density of channel noise.
Binary turbo decoder assumes that the input to the other constituent decoder has variance approximately $N_0/2$. If BPSK modulation with unit bit energy is used, the posterior LLR of the bits equals $4y/N_0$ [46], where $y$ is the channel output of the bit. Therefore, multiplying the LLR by a factor of $N_0/4$, we get the AWGN channel output of BPSK modulated bits. The LLR for each bits $u_i$ for $i = 1, 2, 3, 4$ is computed first. Equivalent $y$ is calculated by multiplying the LLR’s by $N_0/4$.

4.4 Threshold Analysis

Threshold is a certain signal-to-noise ratio after which the performance of the codes will dramatically improve. It is an appropriate measure to evaluate codes. El Gamal et al. [47] have introduced Gaussian approximation to find the threshold of systematic binary turbo codes. Gaussian approximation assumes the extrinsic information, that is, the posterior LLR, as a random variable with normal distribution. We made the same assumption when we used the posterior LLR as the input to the binary turbo decoder. For the nonsystematic turbo codes, the two constituent codes are not symmetric; hence, the threshold is defined on pairs of constituent codes. For the bandwidth efficient turbo coding scheme, we add a subsystem to calculate the LLR to convert channel symbol output to binary turbo decoder input.

4.5 Performance

By computer simulation system implementing the coding scheme shown in Figure 4.1, the performance is evaluated by the bit error rate and the frame error rate shown in Figure 4.9.
Figure 4.9: Performance of the bandwidth efficient nonsystematic turbo codes.

The bandwidth efficient nonsystematic turbo codes utilizes the nonsystematic encoder in Figure 4.4, the puncturing pattern 6 in Figure 4.5, and the Gray mapping in Figure 4.8. The binary turbo decode makes 10 iterations and the input data frame length is 4096. One important observation from Figure 4.9 is that both bit error rate and frame error rate are low although the code rate is close to channel capacity.

4.6 Conclusion

This chapter presents our efforts in searching a channel coding scheme to achieve the asymptotically error free communication and the information transmission rate of the channel capacity. Such a coding scheme has been proved its existence by Shannon’s well-known information theory in 1948. Based on the exciting breakthrough
of turbo codes introduced by Berrou et al. in 1993, our scheme combines nonsys-
tematic turbo codes and quadrature amplitude modulation. We used nonsystematic
turbo codes to extend our searching domain and 16QAM to further improve the
the information transmission rate. To reach our goal, we implemented a complete
communication simulation system. In particular, we designed and implemented a
turbo decoder that can decode turbo codes with any asymmetric, nonsystematic, and
nonlinear constituent codes. With the computer simulation infrastructure, we tested
various of 16QAM modulations and puncturing patterns to be used with different non-
systematic constituent codes. We also analyzed the turbo-coded modulation using
the Gaussian approximation model. By simulating and comparing various heuristics,
the empirical results indicate that our coding scheme has good performance at low
SNR and high rate.
CHAPTER 5

Bitmap Index Compression with an Integrated Reorganization

Designing efficient bitmap schemes for storage and retrieval of massive scientific data is a challenging problem. With integrated data reorganization methods, this study improves bitmap index performance by a factor of 3 to 15. Our first step is to explore the speciality of Gray code ordering. After analyzing the algorithm, we find that Gray code ordering favors the first few columns of the bitmap. Furthermore, the number of the runs in the column could increase exponentially if the bits are uniformly distributed. Next, we arrange the columns of bitmap according to the bitmap itself and the query history. By the bitmap index itself, we design a novel integrated reorganization algorithm that maximally extends the longest run to facilitate the later run length encoding. Regarding the query history, we model the query access patterns by several statistical distributions and move the more frequently accessed columns ahead. Results on empirical data sets show the performance of the bitmap index has significantly improved in terms of the compressed bitmap size and the efficiency of the queries. In essence, our technique can be applied to both equality-encoded and range-encoded bitmap indices or even more general circumstances of bit matrices.
5.1 Introduction

Large volumes of data have been generated in scientific experiments including biology, high-energy physics, astrophysics, and climate modeling. Querying such huge amounts of data is becoming increasingly difficult. Being an effective way to store the synopsis of the original data, bitmap index \cite{48, 49, 50, 51} is a particularly promising strategy for accessing these types of data efficiently. However, the size of bitmap index is still large. Run length encoding and its lossless compression variants have been applied to further compress the bitmap indices \cite{52}. Along with Word-Aligned Hybrid code (WAH) \cite{53}, Gray code ordering \cite{54} of the data tuples has already been shown to greatly boost the compression ratio \cite{55}. While Gray code ordering only arranges the bitmap rowwise, we advance it by manipulating the synopsis columnwise based on the property of the bitmap itself and the query history to further improve the bitmap index performance.

For both conventional and scientific databases, the number of tuples in the database will far exceed the number of attributes for the tuple. Every tuple in the database is represented by one row of the bitmap index whereas each column of the bitmap is generated by classifying every tuple attribute into a few categories. Thus, the bitmap index generally has much higher order of rows than the number of columns. Such an asymmetric property of the bitmaps has at least three implications.

First, Gray code ordering is effective only for rowwise permutation but not for columnwise permutation. The optimality of Gray code ordering is based on the assumption that the data set fully fills the Hamming space corresponding to the particular Gray code. But this is far from true if we apply Gray code ordering to the bitmap columnwise. For example, suppose the bitmap has 9 rows and 3
columns. By rowwise, there will be 9 points in the Hamming space with cardinality $2^3 = 8$. However, if we take it columnwise, there will be only 3 points in the Hamming space with size $2^9 = 512$. If the data set is becoming more and more sparse in the whole Hamming space, Gray code ordering will quickly lose its effectiveness. Previous experiment results have already revealed such a trend [55].

Secondly, run length encoding should scan the data columnwise because rowwise scan will guarantee breaks in the short run. This implies that we should apply the columnwise permutation before the rowwise permutation if they are two separate steps. This sequence should be more effective than the other way around because there will only be a few edge points affected by columnwise permutation. However, as discussed later, the greedy columnwise permutation algorithm seamlessly combines the columnwise permutation with Gray code ordering; thus, the sequence of columnwise and rowwise for the integrated reorganization algorithm is irrelevant.

Lastly, the overhead of keeping the order information of rowwise permutation will be formidable since the number of tuples in the database could be astronomical. On the contrary, the number of columns for regular bitmap is up to the order of thousands, which suggests that it is feasible to track the columnwise permutation. Consequently, column reordering will not hurt the query performance.

In this study, the bitmap index is reorganized by reordering its smaller data units. The difference between our approach and the previous Gray code ordering is in the view of different data units; we want to manipulate the bitmap in the unit of both columns and rows whereas Gray code ordering works specifically in the unit of the rows or data tuples. Integrating both columnwise and rowwise permutations, the big picture of our scheme is outlined in Figure 5.1. We will thoroughly discuss each block
of Figure 5.1 in later sections.

The remainder of this chapter is organized as follows. In the next section, we introduce the bitmap files, Gray code ordering, and word-aligned hybrid compression algorithm. Section 5.3 begins with the discussion of the column reordering problem. We then analyze the peculiarity of Gray code ordering. Based on such a peculiarity, we design columnwise permutation methods by both bitmap index itself and query history models. The results of the columnwise permutation on empirical data set are presented in Section 5.4. Finally, Section 5.5 concludes the study and suggests directions for future research.

5.2 Preliminaries and Related Work

This section will serve as a literature review on the bitmap index, the prior data reorganization method, and the specific compression technique used for the bitmap index.
5.2.1 The Bitmap Index

Bitmap index used in database indexing is a special kind of bit matrix [56]. Each binary row vector in the bitmap represents one tuple in the database. It is usually generated by quantizing the attributes of the tuples. The quantization process proceeds in two steps. First, many categories are produced by limiting the possible values of each attribute. Next, the tuple data are encoded according to the category that its attribute belongs to.

Equality encoding bitmap and range encoding bitmap are two types of bitmaps suitable for point query and range query respectively. The difference between them is the way to encode the quantized data set. The attribute value must fall into a certain category among all the categories for that attribute. For equality encoding bitmap, we put a 1 in that category and 0s for the others. For range encoding bitmap, we put 1s in that category and all the later categories for that attribute. The last category of the attribute will be removed for range encoding as it will be a redundant constant column with only 1s.

As an example, suppose we have tuples with two attributes and each attribute is quantized into three categories. Also assume that one data tuple has the values in the first category of attribute 1 and the second category of attribute 2. The equality encoding bitmap and the range encoding bitmap for that tuple are listed in Table 5.1.

It is obvious that equality encoding bitmap and range encoding bitmap essentially contain the same information about the synopsis of the data. If we have either one, we could generate the other one without any additional information. Readers are referred to Appendix B for more details on the conversion.
5.2.2 Gray Code Ordering

One effective way to rowwise reorganize the bitmap index for run length encoding is Gray code ordering. A Gray code is a coding scheme of binary numbers such that the change between adjacent figures is minimized to one bit [57, 58, 59]. For instance, $(000, 001, 011, 010, 110, 111, 101, 100)$ is a 3-bit Gray code. An $n$-bit Gray code corresponds to a Hamiltonian cycle on an $n$-dimensional Hamming space. From another perspective, it is a kind of Space-Filling Curve (SFC) in Hamming space, where a space-filling curve is a mapping from a one-dimensional set to a multi-dimensional set [60]. Gray code is analogous to a binary version of Hilbert SFC, because both are optimal in minimizing the changes between adjacent points [61, 62].

It is worth noting that Gray code is not unique. After a certain transformation, such as the cyclic shift of the entries or permutation of the bits, the code is still Gray code. One way to construct a $(n + 1)$-bit Gray code from a $n$-bit Gray code is to apply the reflection technique [55] described below.

1. Let $(s_1, s_2, \ldots, s_n)$ be a Gray code.

2. First write it forwards and then append the same code writing it backwards. That is $(s_1, s_2, \ldots, s_n, s_n, \ldots, s_2, s_1)$.
3. Append 0 at the beginning of the first \( n \) numbers and 1 at the beginning of the last \( n \) numbers.

Because of the reflection process, the constructed code is called a *binary-reflected Gray code*. We use the term *fundamental Gray code* to name the binary-reflected Gray code recursively generated using \((0, 1)\) as the initial 1-bit Gray code.

Essentially, every \( n \)-bit Gray code imposes a order relation on \( n \)-dimensional Hamming space making it a ordered set. With respect to fundamental Gray code, the ordering of a set of numbers is referred to as *Gray code ordering*. In effect, applying Gray code ordering with columnwise permutation is equivalent to ordering the data by the order relation dictated by one Gray code among many other kinds of Gray codes. However, an efficient Gray code ordering algorithm has been designed [54] by reversing the fundamental Gray code generation procedure. The recursive ordering algorithm [54] is described below.

1. Sorting all bits in the first column thus dividing all the rows into two parts—the first part with all 0s in the first column while the second part with all 1s.

2. Applying Gray code ordering algorithm to the first part beginning from the next column.

3. Applying *REVERSE* Gray code ordering algorithm to the second part beginning from the next column.

The reverse Gray code ordering refers to the ordering of the binary strings based on the order relation induced by the Gray code which is the reverse of fundamental Gray code. It is this reverse Gray code ordering that makes the process different from the ordinary numerical sorting and helps to make the runs longer as discussed later. After
Gray code ordering, the bitmap index could be actually compressed by certain run length encoding techniques.

5.2.3 Word-Aligned Hybrid Compression

Pure run length encoding is not a good strategy for compressing bitmap index because of its accessing inefficiency. Word-Aligned Hybrid code (WAH) is a special run length compression technique tailored to compress the bitmap index because the compressed bitmaps could be efficiently used by the queries without decompression [63, 64].

By mixing run length encoding and direct storage, WAH compression algorithm partitions the data into segments according to memory word length and divides the segments into fill word and literal word. For instance, if the word length is $p$ bits, every column is partitioned to many $(p - 1)$-bit segments. If a segment is filled with one run of uniform bits, the most significant bit of the compressed data word is set to be 1 to indicate that the word is a fill word. The next bit of the word is made the same as the run. Continue to scan and count the number of consecutive segments that are filled with the same bits. The remaining bits of the fill word are used to store the number of segments counted. On the other hand, if a segment is a mixture of both 0 and 1, mark the most significant bit of encoded word 0 to indicate that this word is a literal word and copy the segment to the rest of its bits directly.

Table 5.2 provides a simple example assuming the word length is 8 bits. The first row is a column from the original bitmap, which starts with 18 0s, followed by 6 1s, and ends with 4 0s. The second row partitions the column into 4 segments, each of which has 7 bits. The first two segments with uniform 0s will be encoded into a fill
word whereas the next two segments are symbolized by two literal words because of
mixing 0s and 1s. Row 3 lists the hex representation of its WAH encoding.

<table>
<thead>
<tr>
<th>Original Bits</th>
<th>18\times0, 6\times1, 4\times0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-bit Segments</td>
<td>(7\times0)\times2, (4\times0,3\times1), (3\times1,4\times0)</td>
</tr>
<tr>
<td>WAH in Hex</td>
<td>82 07 70</td>
</tr>
</tbody>
</table>

Table 5.2: A WAH coding example.

Run length encoding achieves compression by replacing repeated runs of 0s and
1s with a single instance of the symbol and a run count. Unlike traditional run length
encoding, WAH puts more emphasis on the longer runs. Moreover, the runs must
be longer than approximately two word lengths to be effective. We will discuss its
implications further in the next section.

5.3 Columnwise Permutation

Orthogonal to the prior rowwise Gray code ordering technique, our columnwise
permutation method is along the other dimension of the bitmap. After discussing
column reordering problem, we investigate the peculiarity of Gray code ordering.
Accordingly, we propose columnwise permutation schemes based on the bitmap index
itself and then also the query history pattern.

5.3.1 The Column Reordering Problem

Generally, an order is a relation among points in a certain set. With the order
relation defined, the set will be called *ordered set*. Numerical/lexicographic order is
the most commonly used order. Nevertheless, such order renders jumps between some
adjacent points. For example, there is a two-digit difference between 19 and 20. Gray code ordering avoids such jumps for binary strings. However, as discussed in Section 5.1, Gray code ordering is not suitable for the column reordering of the bitmap. We want to find an appropriate order relation on the set of Hamming space corresponding to the columns such that the performance of the bitmap will be improved after Gray code ordering and WAH encoding.

From another point of view, reordering is a special kind of transformation. Mathematically, it could be represented as the following permutation function

\[ f: \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\}, \]

where \( n \) is the data unit size. The goal of this study is to find permutation function \( f \) applied to the bitmap index columnwise thus the bitmap is compressed more. It is worth noting that \( f \) is 1–1 and onto. This implies that the permutation is a reversible transformation. Furthermore, the small number of column number \( n \) will ensure that \( f^{-1} \) is easy to implement for empirical data applications. Hence, the overhead of the columnwise permutation will not outrun the benefit of improved compression.

As the preliminary investigation, we have constructed both deterministic permutation functions and a family of random ones. The deterministic permutation function arranges the columns ascendingly or descendingly by certain criterion calculated for each column. Let \( f \) and \( g \) be the ascending and descending permutation functions, respectively. Obviously, we have

\[ f(i) = n + 1 - g(i) \]

for \( i = 1, 2, \ldots, n \).
The family of random permutation functions are constructed by recursively applying permutation function to itself. So the set of permutation functions is \( \{h^{(j)}\}_{j \in \mathbb{N}} \), where \( h \) is the pre-chosen random permutation function and \( h^{(j)} \) is the mapping applying \( h \) by \( j \) times. Clearly, \( \{h^{(j)}\} \) is finite because the number of different permutation of order \( n \) is finite. Hence \( h^{(j)} \) will eventually run in a circle. However, as \( n! \) is such a huge number for our problem, we have not encountered such a circle for many families of random permutation functions with a sufficiently large size. Finding the best one out of \( n! \) possible permutation functions is one object of this research. To reach our goal, we need to dig deeper into Gray code ordering technique.

5.3.2 Peculiarity of Gray Code Ordering

As described in Section 5.2, Gray code ordering algorithm is a process of Divide-and-Conquer; it first divides the rows of the bitmap into two smaller parts by sorting/clustering the bits in the first column, then each of the two parts is recursively sorted. Note that the ordering of the rows in the smaller part will not break the uniform segment of bits in the first column because the permutation is restricted to that smaller part and all bits in the first column are the same for that part.

Apparently, such a process is not restricted to binary data. We could also apply similar Divide-and-Conquer process to the ternary, quaternary, or data with any bases. However, it is the reflecting or reversing that makes Gray code ordering be especially tuned for run length encoding by making the runs longer. To illustrate this point more clearly, the process of Gray code ordering algorithm is presented graphically in Figure 5.2.
In Figure 5.2, each bar represents one column in the bitmap. Shown are the first 6 columns of the bitmap. The white portion of the bar represents the continued sequence of 0s whereas the black portion represents the continued sequence of 1s. As mentioned earlier, Gray code ordering is essentially a numerical ordering with reflecting or reversing. Depicted graphically, the reflecting or reversing operation is to flip a certain segment with two portions: one white and one black. The part (a) in the figure demonstrates the ordering before flipping or simply the ordinary numerical sorting whereas the part (b) in the figure depicts the result of flipping or the outcome of Gray code ordering. It is easy to see that many runs concatenate to form longer
runs after flipping. This explains why Gray code ordering is more effective than the numerical/lexicographic ordering.

Although the flip extends the runs for Gray code ordering, the number of runs in each column still increases very quickly as we go further along the column dimension of the bitmap. For example, the first column has 2 runs, the second has 3 runs, the third has 5 runs, the forth has 9 runs, the fifth has 17 runs, and so on. In fact, it is easy to prove that the number of runs in column \( n \) will be \( 2^n + 1 \) starting from \( n = 0 \) and assuming that each sorting will produce non-trivial runs of 0s and 1s. The direct consequence of the exponential increasing of runs is that the average run length will decrease quickly. This has an important implication that Gray code ordering favors the first few columns in terms of the assistance to WAH encoding. This is transformed to the compressed column size or the query time.

To quantitively analyze such a bias of Gray code ordering, suppose the bitmap has \( m \) rows and the word length is \( p \) for WAH encoding. The average run length in column \( n \) will be \( m/(2^n + 1) \). As mentioned in Section 5.2.3, WAH coding will be effective only for the runs with length greater than about two word lengths. Therefore,

\[
\frac{m}{2^n + 1} \geq 2(p - 1).
\]

We have

\[
 n \leq \log_2 \left( \frac{m}{2(p - 1)} - 1 \right).
\]  \quad (5.1)

For an empirical data set, the number of rows \( m = 112,361 \) and the code length \( p = 32 \). By plugging these figures into (5.1), the number of affected columns by Gray code ordering is calculated to be \( n \leq 10 \), or the first ten columns.
Note that the above analysis is the worst case scenario; $2^n + 1$ is the upper bound of the number of runs in column $n$ after Gray code ordering. For an empirical data set, we will often encounter the sorting with only 0s or 1s. Such cases will be more common when the difference between the number of 0s and number of 1s gets bigger in the bitmap. For instance, there will be much larger number of 0s than 1s in equality encoding bitmap. Consequently, with 0s and 1s not uniformly distributed in the bitmap, the affected columns by Gray code ordering will be generally larger than the estimation calculated by (5.1).

5.3.3 Reordering by Bitmap Data

Since Gray code ordering helps the early columns by making less number of runs, it is intuitively ‘feasible’ to use such power of making longer runs to cope with columns that are harder to compress. In other words, we could put the most difficult column at the beginning and order the remaining columns by how difficult the column is to compress.

We quantify how hard the column is to compress by an index of compressibility. The compressibility measure should be calculated by the characteristic of each column itself. For run length encoding, the number of runs in the column will directly affect the compressed size of that column. However, the number of runs will not be a good compressibility metric because Gray code ordering will affect the runs in the column. The other prominent property invariant under the row permutation is the number of 1s in that column. Based on such property, the compressibility of column $i$ could be characterized by the following number

$$A_i \triangleq |m/2 - n_i|, \quad (5.2)$$
where $n_i$ is the number of 1s in that column and $m$ is the total number of rows in the bitmap. This definition is motivated by the observation that the more even the number of 0s and 1s in the column, the harder it will be compressed by run length encoding. Therefore, the bigger the number $A_i$, the easier it is to compress.

However, the columnwise permutation based solely on the ascending order of compressibility will generally not improve compression. At first glance, this result is rather counterintuitive. After we take a closer look at Figure 5.2, we observe that the average run length after Gray code ordering will decrease exponentially if the bits are uniformly distributed, although it makes the first few difficult columns compress well. Note that compressibility index in (5.2) is the measure of the uniformness of the 0s or 1s; the lower the compressibility, the smaller the difference between the number of 0s and 1s in the column, hence the more uniformity of the 0s or 1s. Otherwise stated, the difficult columns drain the clustering power of Gray code ordering so fast that the benefit of well compressing such columns is comparatively small. Based on such an observation, we should also consider putting easily compressed columns at beginning such that the clustering power of Gray code ordering is extended as far as possible to the later columns.

Furthermore, the reordering criterion in (5.2) with the grandness of columns is not appropriate. As shown in Figure 5.2, the ordering permutes the rows restricted in many segments, but not in the whole column. Consequently, we should make our decision of permuting the columns by the smaller segments that each Gray code ordering permutes on. Therefore, combined with Gray code ordering, the framework of *integrated reorganization algorithm* is proposed in this study as
1. Starting from the first column as the current column. Swapping the best one among all the rest of the columns according to a certain column choosing criterion with the current column.

2. Sorting all bits in the current column thus dividing all the rows into two parts—the first part with all 0s in the column whereas the second part with all 1s.

3. Applying Gray code ordering combined with columnwise permutation algorithm to the first part beginning from the next column.

4. Applying reverse Gray code ordering combined with columnwise permutation algorithm to the second part beginning from the next column.

Similar to Gray code ordering algorithm described in Section 5.2.2, the integrated reorganization algorithm is still recursive in nature but has crucial step of columnwise permutation.

As discussed before, the reordering criterion should be based on the permuting segments instead of the whole column. Being the direct descendant of (5.2), the column choosing criterion could be

$$B_i \triangleq \sum_j \left| \frac{m_{i,j}}{2} - n_{i,j} \right|, \quad (5.3)$$

where the summation index $j$ is over all the segments that each ordering of 0s and 1s takes place. Although theoretically feasible, the criterion (5.3) is hard to implement\(^2\). This is because the number of the segments increases exponentially as shown in Figure 5.2. In the experiments using (5.3), it is convenient to track the boundary positions of the segment only in the first few columns.

\(^2\)The summation in (5.3) could be further refined with weighted sum. However, this will make the criterion even more complicated.
To overcome such restriction, let’s investigate the saving of WAH encoding as the function of the run length. The saving $S$ here is defined as the reduction of the storage in bits after WAH encoding. From the description of WAH in Section 5.2.3, it is straightforward that $S$ is approximately the linear function of run length $l$; that is,

$$S(l) \propto l.$$ 

Such a relation implies that the longer the runs, the more effective the WAH encoding. By this speciality of WAH encoding, we could permute the columns to maximally extend the longest run in the columns of the bitmap index. Thus, the better columnwise permutation criterion is

$$C_i \triangleq \left\lfloor \frac{m_i^*}{2} - n_i^* \right\rfloor,$$  \hspace{1cm} (5.4)

where $m_i^*$ is the length of the longest segment in Gray code ordering and $n_i^*$ is the number of 1s in that segment. Unlike (5.3), which requires storing $2^i$ segment separating points for column $i$, criterion (5.4) only needs keeping two ending points of the longest segment for all the columns.

### 5.3.4 Reordering by Query History

Besides the intrinsic information from the bitmap itself, we could also permute the columns using the extrinsic information of query access patterns. The accumulated query history contains knowledge about how often each column of the bitmap index is accessed by the users. We exploit such knowledge by utilizing the peculiarity of Gray code ordering.

After Gray code ordering, each column of a bitmap will generally have increasing number of runs or compressed column size as we go further in the column dimension.
As a result, Gray code ordering will favor the first columns of the bitmap in terms of compressing. Therefore, it will be effective to put commonly queried columns at the beginning. This is orthogonal to rowwise Gray code ordering. After we integrate these two, the performance of the bitmap index will be improved.

To investigate the performance of columnwise permutation based on the query history, we quantify the query history with the *query weight vector*  

\[ q = [q_1, q_2, \ldots, q_n] \in [0, 1]^n. \]

With this query weight vector, the *performance metric* could be defined by the following weighted sum of compressed size of each column  

\[ M = \frac{n \sum_{i=1}^{n} q_i c_i}{\sum_{i=1}^{n} q_i}, \quad (5.5) \]

where \( c_i \) is the compressed size of column \( i \) after column reordering and Gray code ordering. The coefficient is chosen such that the value is comparable to the size of the compressed bitmap.

We could generate each component of the query weight vector with a uniform distribution if without any prior information about the query. Let \( D = [D_1, D_2, \ldots, D_n] \) be the query weight vector modeled with a uniform distribution. We have  

\[ D_i \sim U(0, 1) \quad (5.6) \]

for \( i = 1, 2, \ldots, n. \)

However, the real world query history is usually not uniformly distributed, but rather has a skew distribution. Putting it differently, a few columns are queried very often while many others are rarely accessed. Choosing appropriate distribution to
describe this phenomenon is the model selection problem in which we should consider both the fit of the data and the model complexity [33]. To investigate the effectiveness of columnwise permutation based on query history, we will consider four models: power law, Zipf’s law, and two log-linear distributions. All these models can characterize such a circumstance that the probability of occurrence of certain events starts high and tapers off.

Denote $E$ as the query weight vector modeled with power law. The probability density function of its $i$th component $E_i$ is

$$ f_{E_i}(x) = cx^{c-1} $$

with parameter $c > 0$, $x \in (0, 1)$. The plot of (5.7) with $c = 0.5$ is illustrated in Figure 5.3. Obviously, $f_{E_i}(x)$ will go infinity as $x$ takes limit to 0 when $c < 1$.

Described in Appendix C, the classic case of Zipf’s law is a $1/n$ function. Given a set of Zipfian distributed frequencies, sorted from the most common to the least common, the second most common frequency will occur 1/2 as often as the first one. The third most common frequency will occur 1/3 as often as the first one. The $n$th most common frequency will occur 1/n as often as the first one. With $F$ denoted as the query weight vector modeled with Zipf’s law, we have

$$ F_i = \frac{1}{i}. $$

Also derived in Appendix C are a family of log linear distributions. As the example of modeling the query history, we will consider the first two order log linear distributions in (C.2) and (C.3). Let $G$ and $H$ be the query weight vectors modeled by the first order log linear distribution and second order log linear distribution,
The Query History Models

Figure 5.3: The query history models.

respectively. We have their probability density functions

\[ f_{G_i}(x) = \frac{c}{x + (e^c - 1)^{-1}} \]  \hspace{1cm} (5.9)

and

\[ f_{H_i}(x) = \frac{e^c}{((e^c - 1)x + 1)^2} \]  \hspace{1cm} (5.10)

where the parameter \( c > 0 \) and \( x \in (0, 1) \) for both distributions. The plots of (5.9) and (5.10) with \( c = 1 \) are also displayed in Figure 5.3. Different from the power law in (5.7), they are bounded when \( x = 0 \).
5.4 Results and Discussions

In this section, we present our empirical work\(^3\) on the effects of column reordering, the bias of Gray code ordering, and the performance of columnwise permutation through both the bitmap index itself and the query history models.

5.4.1 The Effects of Column Reordering

To investigate the effects of column reordering, we have tried 10,000 random permutation functions to randomly arrange the columns of the bitmap by the experiment sketched in Figure 5.1. The results on the equality encoding bitmap are presented in Figure 5.4 and Figure 5.5. The output of the experiments is the file size of the compressed bitmap with both row and column reordering. In Figure 5.4, we have listed all the compressed file sizes in a monotonic increasing order; x-axis is the permutation index number, whereas y-axis is the compressed file size. For 10,000 random permutations, the largest compressed file size is 702,620 bytes and the smallest is 542,508 bytes with an average file size of 606,490 bytes. The distribution information of those compressed file sizes is shown by the histogram in Figure 5.5.

From the result in Figure 5.4, we observe that the difference between the maximum and the minimum file size is more than 20%. This suggests that the performance of Gray code sorting algorithm is significantly affected by the order in which we process the columns. Thus, finding a good ordering of columns will be a feasible way to improve the bitmap performance. Also shown by the bell shape histogram in Figure 5.5, the compressed file size is approximately normally distributed around

\(^3\)As the example, these typical results come from the image databases with 112361 images. The images are collected from a commercial CD-ROM and 64-dimensional color histograms are transformed with Singular Value Decomposition (SVD) as the feature vectors.
average value of 606,490 with standard deviation 22,808. Figure 5.6 presents the confirming results for the other set of data.

5.4.2 The Bias of Gray Code Ordering

This section provides the empirical results on the peculiarity of Gray code ordering discussed in Section 5.3.2. Gray code ordering clusters the same bits by permuting the rows of the bitmap. The resulting runs of uniform bit will help WAH to compress. To examine how this row permutation affects the columns of the bitmap, we compute the runs before and after Gray code ordering for each column in the bitmap and present the results in Figure 5.7. The x-axis represents the column index number whereas the y-axis lists number of runs and compressed size of each column. Hence, the figure
Figure 5.5: The histogram of the file sizes.

illustrates the distribution of runs among all the columns and the compressed size of each column.

There are six subplots in the figure arranged in a three-by-two array. The three subplots on the left side are the results for equality encoding bitmap whereas the three subplots on the right side are for the same bitmap but with range encoding. The subplots in the first and second rows contain information about the bitmap before and after Gray code ordering, respectively. The third row depicts additional column reordering by the criterion in (5.2).

In any of the subplots, it is confirmed that the number of runs will be approximately proportional to the size of the compressed column. However, the major finding is obtained by comparing the first two rows of the subplots. It is obvious that the
peaks shift to the right after Gray code ordering, as shown in the subplots of the second row. This confirms the analysis in Section 5.3.2 that Gray code ordering favors first columns in terms of number of runs, which is transformed to the compressed column size or the query time.

To further verify the analysis in Section 5.3.2, that Gray code ordering is mostly effective on the first 10–20 columns of the bitmap if the bits are nearly uniformly distributed, we have ordered columns by how asymmetric the number of 0s or 1s in that column is, and put the columns with almost equal numbers of 0s and 1s at the beginning. The number of runs with such additional columnwise permutation is drawn by the subplots in the third row of Figure 5.7. It is observed that there are steep increases around the 20th column. This supports the result in (5.1); it is calculated that $n \leq 10$ for the worst case scenario. The gradual decrease of runs for the later columns is because there are almost all 0s or 1s in those columns. Figure 5.8
5.4.3 Performance of Reordering by Bitmap Data

To apply the methods of bitmap reorganization discussed in Section 5.3.3, we design three deterministic permutation functions according to the column reordering

Figure 5.7: The distribution of runs & compressed column size.

presents the similar results of the bias of Gray code ordering with the other set of data.
criteria (5.2), (5.3), and (5.4). Criterion (5.3) only tracks the first 20 columns of the bitmap because of its exponentially increasing computational expense. The reordering techniques are applied to the same bitmap data sets as the previous experiment. To validate the proposed method, we observe the decrease in the size of the bitmap index. The result of the file sizes is listed in Table 5.3.
<table>
<thead>
<tr>
<th>Methods</th>
<th>Equality Encoding</th>
<th>Range Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Gray Code Ordering</td>
<td>1397004</td>
<td>923068</td>
</tr>
<tr>
<td>With Gray Code Ordering</td>
<td>683664</td>
<td>380344</td>
</tr>
<tr>
<td>Integrated Permutation by $A_i$</td>
<td>762972</td>
<td>398304</td>
</tr>
<tr>
<td>Integrated Permutation by $B_i$</td>
<td>661704</td>
<td>380248</td>
</tr>
<tr>
<td>Integrated Permutation by $C_i$</td>
<td>475188</td>
<td>285044</td>
</tr>
</tbody>
</table>

Table 5.3: The performance of five methods compared by file size (bytes).

Table 5.3 reveals the effectiveness of the column reordering criterion presented in (5.4) for both equality encoding bitmap and range encoding bitmap. In this table, the first column gives the types of compressed bitmap files whereas the next two columns list the sizes of the files for equality-encoded and range-encoded bitmap indices respectively. The table provides the size of the files compressed by WAH algorithm. Five rows of data correspond to five types of compression techniques: without Gray code ordering, with Gray code ordering, with columnwise permutation by $A_i$ in (5.2), with columnwise permutation by $B_i$ in (5.3), and with columnwise permutation by $C_i$ in (5.4).

From the data presented in the table, as expected, we observe that the compression of the bitmap index significantly improves after Gray code ordering. While the permutation using criterion $A_i$ does not improve the compression, the improvement by criterion $C_i$ is significant. For the equality encoding bitmap, with the columnwise encoding $C_i$, there is more than a 30% improvement (from 683,664 to 475,188) in spite of the fact that it has already been improved more than 2 times by Gray code.

\(^4\)To simplify the notation, we will use the numbers in Table 5.3 to refer to the file size and the corresponding method.
ordering (from 1,397,004 to 683,664). The similar improvement is also obtained by $C_i$ for the range encoding bitmap.

The result in the third row indicates that it is not enough to permute columns of the bitmap solely on the compressibility of that column. As discussed in Section 5.3.2, the runs will increase exponentially if the 0s or 1s are uniformly distributed. This is also attested by the experiment results of the third row in the Figure 5.7—the runs increase sharply after the first few columns. The promising results of columnwise permutation on equality encoding bitmap together with the results for range encoding bitmap by $C_i$ show that extending the clustering power of Gray code ordering is also important in designing the columnwise permutation functions.

The result of 475,188 from $C_i$ on equality encoding bitmap is much better than 542,508, which is the best outcome found from 10,000 random permutation functions shown in Figure 5.4. With the data presented in Figure 5.5, if we model the file sizes by a random variable of normal distribution with mean 606,490 and standard deviation 22,808, the probability of randomly getting a better permutation function than 475,188 is only $4.2852 \times 10^{-9}$. Table 5.4 reveals the same conclusion for the performance of reordering by bitmap data from the other set of data.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Equality Encoding</th>
<th>Range Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Gray Code Ordering</td>
<td>834856</td>
<td>611544</td>
</tr>
<tr>
<td>With Gray Code Ordering</td>
<td>215584</td>
<td>137372</td>
</tr>
<tr>
<td>Integrated Permutation by $A_i$</td>
<td>236276</td>
<td>148952</td>
</tr>
<tr>
<td>Integrated Permutation by $B_i$</td>
<td>216580</td>
<td>137380</td>
</tr>
<tr>
<td>Integrated Permutation by $C_i$</td>
<td>198000</td>
<td>128724</td>
</tr>
</tbody>
</table>

Table 5.4: The performance of reordering by bitmap data.
5.4.4 Performance of Reordering by Query History

To confirm the columnwise permutation method using query history patterns introduced in Section 5.3.4, we calculate the performance metric in (5.5) for five query history models and list the results of equality encoding bitmap in Table 5.5.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Original</th>
<th>With Permutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform $D_i$</td>
<td>166210</td>
<td>135671</td>
</tr>
<tr>
<td>Power $E_i$</td>
<td>169379</td>
<td>114330</td>
</tr>
<tr>
<td>Zipf $F_i$</td>
<td>156575</td>
<td>48133</td>
</tr>
<tr>
<td>LL1 $G_i$</td>
<td>177987</td>
<td>119145</td>
</tr>
<tr>
<td>LL2 $H_i$</td>
<td>161856</td>
<td>105570</td>
</tr>
</tbody>
</table>

Table 5.5: The performance metric by five query history models.

Five rows of data in Table 5.5 represent five query history models: uniform, power, Zipf, the first order log linear, and the second order log linear. The corresponding mathematical representations are $D_i$ in (5.6), $E_i$ in (5.7), $F_i$ in (5.8), $G_i$ in (5.9), and $H_i$ in (5.10), respectively. Same as the models illustrated in Figure 5.3, the parameter $c$ is set to 0.5 for the power model and 1 for the other two log linear models. The second column in the table represents the performance metric without column reordering whereas the numbers in the third column are with columnwise permutation.

The data in Table 5.5 indicate 18%, 33%, 69%, 33%, 35% improvements on performance in respect to no columnwise permutation case for the five models. The uniform model has the least improvement of 18% whereas Zipf model improves the most with 69%. This suggests that the performance of columnwise permutation is
greatly improved if the query history has a skew distribution. It is also confirmed by
the two log linear models; the first order log linear model with less improvement of
33% than the second order log linear model with 35% is less asymmetrical than the
second order log linear model as shown in Figure 5.3.

5.5 Summary and Future Work

In this study, we improve bitmap index performance through integrated data
reorganization approaches that manipulate the data both columnwise and rowwise.
Along the column dimension of the bitmap, our method is orthogonal to the prior
rowwise Gray code ordering and can be applied with it together. One of the major
findings is that Gray code ordering algorithm favors the first few columns in terms
of number of runs. The results of the confirming experiment on this peculiarity of
Gray code ordering are also provided. Accordingly, we have permuted the columns of
the bitmap by both the bitmap data and the query history. By bitmap index itself,
we design an integrated reorganization algorithm that maximally extends the longest
run to facilitate the later run length encoding. From the query history, we model
the query access patterns by several models and move the more frequently queried
columns ahead. After we combine these techniques with Gray code ordering, the
experiment results on empirical data sets show the performance of the bitmap index
has significantly improved in terms of the compressed bitmap size. Our technique
performs well for both equality encoding bitmap and range encoding bitmap, and it
can be applied to the general binary matrices.

Future research can proceed in several directions. As mentioned earlier, the
present Gray code ordering is based on the fundamental Gray code. There are many
other Gray codes, and columnwise permutation essentially makes use of one another Gray code. Whether we can find some of them to bias more or less than the fundamental Gray code is another interesting research project. Based on the framework of integrated reorganization algorithm proposed here, we could also jointly consider the relative number of 0s and 1s in the bitmap to find integrated permutation criterion optimal for the particular bitmap. Besides the properties of the bitmap itself, selecting the appropriate model for the query history from the empirical use of the bitmap is also important.
CHAPTER 6

Conclusion

As a powerful tool, mathematical modeling is crucial in an efficient and effective information processing system. The purpose of this dissertation study is to investigate both theoretical and empirical studies on mathematical modeling. The theoretical studies include MDL model selection and componential analysis. The empirical studies include bandwidth efficient nonsystematic turbo codes and bitmap index compression through an integrated reorganization.

At the core of the progress in science is the problem of deciding among a set of competing models from the data. The recent development of model selection criterion has been focused on minimum description length principle, including FIA and NML. In applying FIA criterion to the selection of parametric family of distributions, one of the major obstacles is to calculate Fisher information. A general formula of Fisher information is provided in this study to overcome the barrier. The formula eliminating the need for numerical expectation or the second derivative of the likelihood function; thus, it greatly simplifies the computation and can be applied to any model with multinomial distribution or independent normal distribution. Also presented in this study is the application of the formula for the models of categorization, information integration, retention, and psychophysics. Furthermore, our investigation in
the NML criterion realizes that NML demands much more expensive computation
than FIA. This is because NML and FIA require integration in the sample space
and the parameter space respectively, while the sample space has a much higher di-
mension than the parameter space for most practical applications. Our investigation
indicates that FIA provides a good approximation to NML for a multinomial model.
Finally, we present the application of MDL for the model selection in the selection of
the retention models. We demonstrate the advantage of MDL through a comparison
between the model selection performance of MDL and several other commonly used
selection criteria. It is concluded that the derived formula for Fisher information
could be applied in scientific enterprises other than MDL model selection as well.

Componential analysis is the other theoretical respect of the current research.
Along this line of research, we formalize the problem of componential analysis and
explore various methods to define and estimate importance index of each param-
ter. Our first step is to introduce several key concepts. For example, to simplify the
inquiry in scientific research, a model is qualitatively characterized by finite factors.
Each factor is then quantitatively represented by the parameter of the model. The
problem of componential analysis is to investigate how and how much each param-
eter affects mathematical model’s ability to fit arbitrary patterns of data. Such an
ability is quantified as the measure of model complexity. The notion of model com-
plexity originates from the observation that choosing competing models based solely
on the goodness of fit can result in the selection of an unnecessarily complex model
that overfits the data. The contribution of the parameter to model complexity can
be quantified as the importance index of the parameter. The current study exam-
ines prior approaches to the similar problems and reviews several known measures
of model complexity; then, the importancy index is proposed globally or locally. A global approach is to evaluate importancy index by choosing a certain model complexity measure. As the model complexity is a global property of the model, the derived importancy index is a global attribute of the model. On the other hand, a local approach is to investigate how the complexity is evaluated at each point in the model. The importancy index is then defined locally without explicitly borrowing certain model complexity measure. The global measure could also be induced by the procedure combining average by integration of the local measure. Using a highly efficient calculating method for the Fisher information presented before, this study tests the proposed approaches and presents the preliminary results.

From an empirical perspective, coding is the direct implementation of mathematical modeling. Channel coding and source coding are the practical applications of two important concepts in the information theory: channel capacity and entropy. This study investigates these concepts in two particular cases respectively: bandwidth efficient nonsystematic turbo codes and bitmap index compression through an integrated reorganization.

One of the goals of this study is to search a channel coding scheme that can achieve the asymptotically error free communication and the information transmission rate of the channel capacity. Shannon’s well-known information theory in 1948 has proved the existence of such a coding scheme. Combining turbo codes and quadrature amplitude modulation, our scheme is based on the exciting breakthrough of turbo codes introduced by Berrou et al. in 1993. We used nonsystematic turbo codes to extend our searching domain and 16QAM to further improve the information transmission rate. To fulfill our goal, we implemented a complete communication
simulation system. Particularly, we designed a turbo decoder that can decode turbo codes with any asymmetric, nonsystematic, and nonlinear constituent codes. With the computer simulation infrastructure, we tested various 16QAM modulations and puncturing patterns to be used with different non-systematic constituent codes. We also analyzed the turbo-coded modulation using the Gaussian approximation model. The result of simulating and comparing various heuristics indicates that our coding scheme has good performance at low SNR and high rate.

Another achievement of this study is to improve bitmap index performance through integrated data reorganization approaches that manipulate the data both columnwise and rowwise. Along the column dimension of the bitmap, our method is orthogonal to the prior rowwise Gray code ordering and can be applied with it together. One of the major findings is that Gray code ordering algorithm favors the first few columns in terms of number of runs. The current study provides the confirming experiment result on this peculiarity of Gray code ordering. Accordingly, we have permuted the columns of the bitmap by both the bitmap data and the query history. By bitmap index itself, we design an integrated reorganization algorithm that maximally extends the longest run to facilitate the later run length encoding. From the query history, we model the query access patterns by several models and move the more frequently queried columns ahead. After we combine these techniques with Gray code ordering, the experiment results on empirical data sets show the performance of the bitmap index has significantly improved in terms of the compressed bitmap size. Our technique performs well for both equality encoding bitmap and range encoding bitmap, and it can be applied to the general binary matrices.
Overall, the current study investigates mathematical modeling from both theoretical and empirical standpoints. The promising results expand our view and provide potential directions for future research adventures. Launching a successful practice takes a mixture of considerations and efforts from all directions. There are still impending mystery of mathematical modeling that is waiting for being uncovered the veil.
APPENDIX A

Characteristic Properties of Convex Function

Let \( X \) be a convex set in Euclidean space. The real-valued function \( f: X \rightarrow \mathbb{R} \) is a convex function if and only if any of the following four characteristic properties is satisfied.

1. \[
\frac{(x_2-x_3)f(x_1)+(x_3-x_1)f(x_2)+(x_1-x_2)f(x_3)}{(x_3-x_2)(x_1-x_3)(x_2-x_1)} \geq 0, \quad \forall \ (x_1, x_2, x_3) \in X^3 = (a, b)^3,
\]

2. \( f \) has monotonically increasing one-side derivatives on \( X = (a, b) \),

3. \( f \in C(X) \) and \( f\left(\frac{x_1+x_2}{2}\right) \leq \frac{f(x_1)+f(x_2)}{2}, \quad \forall \ (x_1, x_2) \in X^2, \)

4. \( f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2), \quad \forall \ (x_1, x_2) \in X^2, \forall \lambda \in (0, 1). \)

**Proof:** Proceed by four steps: \( 1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 1. \)

\((1 \Rightarrow 2)\) From \( f \), we have its difference quotient [65].

\[
\phi(x_1, x_2) \triangleq \frac{f(x_1) - f(x_2)}{x_1 - x_2}, \quad \forall \ (x_1, x_2) \in (a, b)^2. \tag{A.1}
\]

Clearly, \( \phi \geq 0 \) if and only if \( f \) is monotonic increasing. Since \( \phi(x_1, x_2) \) is invariant in permuting its arguments, let’s fixed one and consider the difference quotient on the other.

\[
\psi(x_1, x_2, x_3) \triangleq \frac{\phi(x_1, x_3) - \phi(x_2, x_3)}{x_1 - x_2}, \quad \forall \ (x_1, x_2, x_3) \in (a, b)^3. \tag{A.2}
\]
With (A.1) and (A.2), we have
\[
\psi(x_1, x_2, x_3) = \frac{(x_2 - x_3)f(x_1) + (x_3 - x_1)f(x_2) + (x_1 - x_2)f(x_3)}{(x_3 - x_2)(x_1 - x_3)(x_2 - x_1)}.
\]

Similar as \( \phi \), \( \psi(x_1, x_2, x_3) \) is invariant in permuting its arguments. Since \( \psi \geq 0 \) by assumption, \( \phi \) is monotonic increasing on one of its argument by keeping the other fixed.

Now, \( \forall x_3 \in (a, b) \) choose \( a < x_1 < x_3 < x_2 < b \). We have \( \phi(x_1, x_3) \leq \phi(x_2, x_3) \). As \( x_1 \to x_3 \), \( \phi(x_1, x_3) \) is increasing and is upper bounded by \( \phi(x_2, x_3) \). By the least upper bound property of \( \mathbb{R} \), \( \phi(x_1, x_3) \to f'(x_3-) \). Similarly, \( f'(x_3+) \) exists and \( \phi(x_2, x_3) \to f'(x_3+) \) as \( x_2 \to x_3 \). By making \( x_1 \to x_3 \) and \( x_2 \to x_3 \) on both sides of \( \phi(x_1, x_3) \leq \phi(x_2, x_3) \), we have
\[
f'(x_3-) \leq f'(x_3+). \tag{A.3}
\]

Next, choose \( a < x_1 < x_3 < x_4 < x_2 < b \) to consider one-side derivatives on different points. We have \( \phi(x_1, x_3) \leq \phi(x_1, x_4) \leq \phi(x_2, x_4) \). Taking \( x_3 \to x_1 \) and \( x_4 \to x_2 \) on the first and last terms, respectively, we have
\[
f'(x_1+) \leq f'(x_2-). \tag{A.4}
\]

In conclusion, the property 2 follows from (A.3) and (A.4).

\((2 \Rightarrow 3) \) As \( f \) has one-side derivatives, \( f \) is continuous or \( f \in \mathcal{C}(X) \).

Since the property 3 does not change under permutation of the subscripts, we could assume that \( x_1 < x_2 \). According to the generalized mean-value theorem for functions with one-sided derivatives and the monotonically increasing one-side derivatives of \( f \), we can find \( \xi, \eta \) with \( x_1 < \xi < \frac{x_1 + x_2}{2} < \eta < x_2 \) such that
\[
f(\xi-) \leq \phi(x_1, \frac{x_1 + x_2}{2}) \leq f(\xi+) \leq f(\eta-) \leq \phi(x_2, \frac{x_1 + x_2}{2}) \leq f(\eta+).
\]
Then, property 3 follows from \(\phi(x_1, \frac{x_1 + x_2}{2}) \leq \phi(x_2, \frac{x_1 + x_2}{2})\).

(3 \Rightarrow 4) \forall \lambda \in (0, 1), construct a sequence \(\{\lambda_n\}\) as following:

\[
\begin{cases}
\lambda_1 = \frac{1}{2}, \\
\lambda_{n+1} = \lambda_n + I_n \cdot \frac{1}{2^n},
\end{cases}
\text{where } I_n = \begin{cases}
-1, & \lambda_n \geq \lambda \\
1, & \lambda_n < \lambda
\end{cases}
\text{ and } n \in \mathbb{N}.
\]

Claim: \(|\lambda_n - \lambda| \leq \frac{1}{2^n}\). Prove by induction:

(I) \therefore \lambda \in (0, 1) \implies \lambda_1 - \lambda \in (-.5, .5) \implies |\lambda_1 - \lambda| < \frac{1}{2}.

(II) suppose \(|\lambda_n - \lambda| \leq \frac{1}{2^n}\)

(i) If \(\lambda_n \geq \lambda \implies \lambda_n - \lambda \in [0, \frac{1}{2^n}] \implies \lambda_n - \lambda \in [-\frac{1}{2^n}, \frac{1}{2^n}] \implies \lambda_n + I_n \cdot \frac{1}{2^n} - \lambda \in [-\frac{1}{2^{n+1}}, \frac{1}{2^{n+1}}] \implies |\lambda_{n+1} - \lambda| \leq \frac{1}{2^{n+1}}.

(ii) Similarly for \(\lambda_n < \lambda\), we also have \(|\lambda_{n+1} - \lambda| \leq \frac{1}{2^{n+1}}\).

By induction, we have \(|\lambda_n - \lambda| \leq \frac{1}{2^n}\). Taking \(n \to \infty\) \implies |\lambda_n - \lambda| \to 0 \implies \lambda_n \to \lambda.

From the construction of \(\{\lambda_n\}\), we have

\[
\lambda_n = \frac{1}{2} + \sum_{k=1}^{n-1} I_k \cdot \frac{1}{2^{k+1}} = \frac{1}{2n} + \sum_{k=1}^{n-1} (1 + I_k) \cdot \frac{1}{2^{k+1}},
\]

\[
1 - \lambda_n = \frac{1}{2} - \sum_{k=1}^{n-1} I_k \cdot \frac{1}{2^{k+1}} = \frac{1}{2n} + \sum_{k=1}^{n-1} (1 - I_k) \cdot \frac{1}{2^{k+1}}.
\]

Therefore,

\[
f(\lambda_n x_1 + (1 - \lambda_n)x_2) = f\left(\frac{x_1 + x_2}{2^n} + \sum_{k=1}^{n-1} (1 + I_k) \cdot \frac{x_1}{2^{k+1}} + (1 - I_k) \cdot \frac{x_2}{2^{k+1}}\right)
\]

\[
= f\left(\frac{x_1 + x_2}{2^n} + \sum_{k=2}^{n-1} ((1 + I_k)x_1 + (1 - I_k)x_2) \frac{1}{2^{k+1}}\right) + (1 + I_1) \frac{x_1}{2} + (1 - I_1) \frac{x_2}{2^n}\]

\[
\leq \frac{1}{2} f\left(\frac{x_1 + x_2}{2^{n-1}} + \sum_{k=2}^{n-1} ((1 + I_k)x_1 + (1 - I_k)x_2) \frac{1}{2^k}\right) + \frac{1}{2} f\left((1 + I_1) \frac{x_1}{2} + (1 - I_1) \frac{x_2}{2}\right)
\]

118
\[
\begin{align*}
\frac{1}{2} & f \left( \frac{x_1 + x_2}{2^{n-1}} + \left[ \sum_{k=3}^{n-1} \left( (1 + I_k)x_1 + (1 - I_k)x_2 \right) \frac{1}{2^k} \right] + (1 + I_2) \frac{x_1}{2^2} + (1 - I_2) \frac{x_2}{2^2} \right) \\
& + (1 + I_1) \frac{f(x_1)}{2^2} + (1 - I_1) \frac{f(x_2)}{2^2} \\
\leq & \frac{1}{2^n} f \left( \frac{x_1 + x_2}{2} + \sum_{k=3}^{n-1} \left( (1 + I_k)x_1 + (1 - I_k)x_2 \right) \frac{1}{2^{k-1}} \right) \\
& + \sum_{k=1}^2 (1 + I_k) \frac{f(x_1)}{2^{k+1}} + (1 - I_k) \frac{f(x_2)}{2^{k+1}} \\
& \vdots \\
\leq & \left( \frac{1}{2^n} + \sum_{k=1}^{n-1} (1 + I_k) \frac{1}{2^{k+1}} \right) f(x_1) + \left( \frac{1}{2^n} + \sum_{k=1}^{n-1} (1 - I_k) \frac{1}{2^{k+1}} \right) f(x_2) \\
= & \lambda_n f(x_1) + (1 - \lambda_n) f(x_2).
\end{align*}
\]

Taking \( \lambda_n \to \lambda \) on both sides of \( f(\lambda_n x_1 + (1 - \lambda_n)x_2) \leq \lambda_n f(x_1) + (1 - \lambda_n)f(x_2) \), we have

\[
f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2),
\]

because \( f \) is continuous.

**\( 4 \Rightarrow 1 \)** Since both property 1 and property 4 do not change under the permutation of the subscripts, we could let \( x_3 = \lambda x_1 + (1 - \lambda)x_2 \in (x_1, x_2) \subset (a, b) \). Hence,

\[
\lambda = \frac{x_3 - x_2}{x_1 - x_2}.
\]

Plugging this into (A.5), we have

\[
\frac{(x_2 - x_3)f(x_1) + (x_3 - x_1)f(x_2) + (x_1 - x_2)f(x_3)}{(x_3 - x_2)(x_1 - x_3)(x_2 - x_1)} \geq 0.
\]

\[\square\]
APPENDIX B

Conversion Bitmaps

Equality encoding bitmap and range encoding bitmap basically synopsize the same information about the data. We could convert bitmap from one to the other without any additional information. The conversion is accomplished by finding the boundary position of the attributes.

From the equality encoding bitmap, we could use the maximum column number of the $i$th one as the boundary between $i$th attribute and $(i + 1)$th attribute. For example, if the maximum column number of the first 1 in every row of the bitmap index is 8, the first attribute has 8 columns, whereas the second attribute starts with column 9. This rule of finding the bitmap structure assumes that the columns at both ends of the attribute contain 1s in them.

To generate an equality encoding bitmap from the range encoding bitmap, we could consider the transitions between the adjacent bits. If we find the bits change from 1 to 0, there will guarantee an attribute boundary between those two bits. Unfortunately, this condition is only necessary but not sufficient. In other words, a certain position being an attribute boundary does not ensure we will find the 1 to 0 transition at that position in all the bitmap index data. If the 1 to 0 transition at that position does not exist, there will be a miss of the attribute boundary. Algorithmically,
such a miss could not be avoided. Let’s further investigate the possibility of missing the attribute boundaries.

Assume the bitmap index has $m$ rows, $a$ attributes, and $b$ categories for each attribute. Under the condition that each category occurs equally, it is straightforward to find the missing probability as

$$\Pr(\text{miss}) = (a - 1) \left(1 - \left(1 - \frac{1}{b}\right)^{2}\right)^m.$$ 

Suppose $a = 100$ and $b = 8$. Table B.1 lists the missing probability with several values of $m$.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$\Pr(\text{miss})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$4.9516 \times 10^{-5}$</td>
</tr>
<tr>
<td>100</td>
<td>$9.6998 \times 10^{-62}$</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
</tr>
</tbody>
</table>

Table B.1: $\Pr(\text{miss})$ as the function of the number of rows.

This shows that when we have 1,000 tuples in the bitmap, the probability of missing will be smaller than the smallest double precision number. Moreover, the practical bitmap index has much higher number of tuples than 1,000. Consequently, the missing probability will be virtually zero for an empirical bitmap index.
Zipf’s Law and Log Linear Distributions

Zipf’s law describes a phenomenon that a few events occur very often whereas many others occur rarely. A great amount of Internet data suggest that Web access follows Zipf’s law. Mathematically, Zipf’s law is closely related to the zeta distribution. Denote \( P_n \) as the frequency of the \( n \)th ranked event. The zeta distribution is defined as 
\[
P_n = \frac{1}{n^s \zeta(s)},
\]
where \( n \in \mathbb{N} \), parameter \( s > 1 \), and \( \zeta(s) \triangleq \sum_{n=0}^{\infty} n^{-s} \) is the Riemann’s zeta function. Zipf’s law could be looked upon as a special case of zeta distribution with the support as the truncated subset of \( \mathbb{N} \) and \( s \approx 1 \). Therefore, the approximation of Zipf’s law could be formulated as \( P_n = \frac{c}{n} \), where \( n \in \{1, \ldots, N\} \) for certain integer \( N \) and \( c \) is a normalization constant.

Zipf’s law describes a discrete probability distribution. Its probability function follows a straight line when drew in a plot with logarithmic scale on both axes. To extend logarithmic linear property to the continuous distribution, we have the definition of the log linear distributions.

**Definition C.0.1** The log linear distributions are a family of continuous distributions describing the occurrence frequency of certain events. The probability density function of the distribution will be presented as a straight line segment in the plot with both axes in logarithm scale.
To model the query frequency with the log linear distribution, we need its mathematical expression. Denote $x$ as the frequency variable and $f(x)$ as the probability density function of the log linear distribution. The range of $x$ is $[0, 1]$ and the logarithm of this range is the negative real line. After we apply the linear transformation on $x$, the plot of log linear distribution on double-logarithmic plot is in Figure C.1.

![Figure C.1: The log linear distribution.](image)

Then we have

$$f(x) = \left( \frac{e^{b-a}}{(e^{b-a} - 1)x + 1} \right)^{\frac{k-l}{b-a}} \cdot e^l$$

with $x \in [0, 1]$, $b > a$, $k > 0$, and $l < 0$. Accordingly,

$$\int_0^1 f(x) dx = e^{k-l} \int_0^1 ((e^{b-a} - 1)x + 1)^{\frac{k-l}{b-a}} dx \cdot e^l. \quad (C.1)$$

If $k - l \neq b - a$,

$$(C.1) = \left. \frac{e^{k-l}((e^{b-a} - 1)x + 1)^{\frac{1-k}{b-a}+1}}{(l-k)\frac{b-a+1}{b-a+1}(e^{b-a} - 1)} \cdot e^l \right|_0^1$$

$$= \frac{e^{b-a+l} - e^k}{(l-k)\frac{b-a+1}{b-a+1}(e^{b-a} - 1)}$$

$$= 1.$$
\[ e^k = \frac{e^{b-a} - 1}{b-a} k + e^{b-a+l} - \frac{b-a+l}{b-a} (e^{b-a} - 1). \]

Now we have a family of log linear distributions. Specifically, if \( b-a+l = 0 \), we have

\[ f(x) = \frac{e^{b-a}}{((e^{b-a} - 1)x + 1)^2}. \quad (C.2) \]

On the other hand, if we assume that \( k - l = b - a \) in (C.1), we have another family of the first order log linear distributions:

\[ f(x) = \frac{(b-a)^{-1}}{x + (e^{b-a} - 1)^{-1}}. \quad (C.3) \]


