OPTIMAL BAYESIAN TRANSFER LEARNING FOR CLASSIFICATION AND REGRESSION

A Dissertation

by

ALIREZA KARBALAYGHAREH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2019

Major Subject: Electrical Engineering

ABSTRACT

Machine learning methods and algorithms working under the assumption of identically and independently distributed (i.i.d.) data cannot be applicable when dealing with massive data collected from different sources or by various technologies, where heterogeneity of data is inevitable. In such scenarios where we are far from simple homogeneous and uni-modal distributions, we should address the data heterogeneity in a smart way in order to take the best advantages of data coming from different sources. In this dissertation we study two main sources of data heterogeneity, time and domain. We address the time by modeling the dynamics of data and the domain difference by transfer learning.

Gene expression data have been used for many years for phenotype classification, for instance, classification of healthy versus cancerous tissues or classification of various types of diseases. The traditional methods use static gene expression data measured in one time point. We propose to take into account the dynamics of gene interactions through time, which can be governed by gene regulatory networks (GRN), and design the classifiers using gene expression trajectories instead of static data. Thanks to recent advanced sequencing technologies such as single-cell, we are now able to look inside a single cell and capture the dynamics of gene expressions. As a result, we design optimal classifiers using single-cell gene expression trajectories, whose dynamics are modeled via Boolean networks with perturbation (BNp). We solve this problem using both expectation maximization (EM) and Bayesian framework and show the great efficacy of these methods over classification via bulk RNA-Seq data.

Transfer learning (TL) has recently attracted significant research attention, as it simultaneously learns from different source domains, which have plenty of labeled data, and transfers the relevant knowledge to the target domain with limited labeled data to improve the prediction performance. We study transfer learning with a novel Bayesian viewpoint. Transfer learning appears where we do not have enough data in our target domain to train the machine learning algorithms well but have good amount of data in other relevant source domains. The probability distributions of the source and target domains might be totally different but they share some knowledge underlying the similar tasks between the domains and are related to each other in some sense. The ultimate goal of transfer learning is to find the amount of relatedness between the domains and then transfer the amount of knowledge to the target domain which can help improve the classification task in the data-poor target domain. Negative transfer is the most vital issue in transfer learning and happens when the TL algorithm is not able to detect that the source domain is not related to the target domain for a specific task. For addressing all these issues with a solid theoretical backbone, we propose a novel transfer learning method based on a Bayesian framework. We propose a Bayesian transfer learning framework, where the source and target domains are related through the joint prior distribution of the model parameters. The modeling of joint prior densities enables better understanding of the transferability between domains. Using such an idea, we propose optimal Bayesian transfer learning (OBTL) for both continuous and count data as well as optimal Bayesian transfer regression (OBTR), which are able to optimally transfer the relevant knowledge from a data-rich source domain to a data-poor target domain, whereby improving the classification accuracy in the target domain with limited data.

DEDICATION

I gratefully dedicate this dissertation to my wife, Fatemeh.

ACKNOWLEDGMENTS

I would like to sincerely thank my Ph.D. advisor, Dr. Edward R. Dougherty, for offering me continuous help, insightful guidance, and ceaseless support and motivation throughout my Ph.D. research. I learned from him how to see the important science and machine learning problems from the lenses of mathematics and optimization. I also would like to appreciate my co-advisors, Dr. Ulisses Braga-Neto and Dr. Xiaoning Qian, for their constructive comments and kind support and also Dr. Krishna Narayanan and Dr. Anirban Bhattacharya for serving as a member of my committee. Finally, I should especially thank my supportive family and my warmhearted wife, whose unwavering encouragement and support, even from hundreds of miles away, were undoubtedly the main reasons of my success in this journey.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Dr. Edward R. Dougherty, Dr. Ulisses Braga-Neto, Dr. Xiaoning Qian, and Dr. Krishna Narayanan of the Department of Electrical and Computer Engineering and Dr. Anirban Bhattacharya of the Department of Statistics at Texas A&M University. All work for the dissertation was completed by the student, under the advisement of Dr. Edward R. Dougherty of the Department of Electrical and Computer Engineering.

Funding Sources

My work was supported by the National Science Foundation, through NSF awards CCF-1320884 and CCF-1553281.

NOMENCLATURE

TL	Transfer Learning
DA	Domain adaptation
OBTL	Optimal Bayesian Transfer Learning
OBTR	Optimal Bayesian Transfer Regression
GRN	Gene Regulatory Network
MCMC	Markov Chain Monte Carlo
HMC	Hamiltonian Monte Carlo
BN	Boolean Network
BNp	Boolean Network with perturbation
PBN	Probabilistic Boolean Network
POBDS	Partially-Observed Boolean Dynamical Systems
ML	Maximum Likelihood
EM	Expectation Maximization
NGS	Next Generation Sequencing
MSE	Mean Square Error
MMSE	Minimum Mean Square Error
RNA-Seq	Ribonucleic Acid Sequencing
i.i.d.	identically and independently distributed

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Machine learning has become one of the most important and challenging research topics today with immense number of applications such as genomics or computer vision, which owes its popularity to the emergence of different types of data which can now be easily acquired thanks to the growth of technology. Being exposed to different types of data coming from various measurement technologies lead to great amount of heterogeneity in data, which had not been experienced so far. As a result, we would more likely be very far away from the identically and independently distributed (i.i.d.) assumption, upon which many machine learning methods have been built. In this dissertation, we have addressed two types of heterogeneity in data: heterogeneity due to time evolution and heterogeneity due to domain difference. First, we address the time heterogeneity via modeling the dynamics of data. Second, we solve heterogeneity due to domain difference via transfer learning.

In chapter 2, we thoroughly study the classification of gene expression trajectories using the partial knowledge of gene regulatory networks (GRN). Traditionally, gene expression data from different sources of measurements, such as microarray and RNA-Seq, have been used for phenotype classification, that is, the classification of cancer versus normal or the classification of different types of diseases. According to the fact that the genes in our cells interact with each other and actually evolve over time via a GRN, we propose to model their dynamics and classify phenotypes using gene expression trajectories (time-series) which contain more information than the static data which are a snapshot of gene expressions in one time point. We first use partially known GRNs and model the gene expression state trajectories using Boolean Networks with perturbation (BNp), where a gene has two values: 1 for On and 0 for Off. We study the classification error under different mutated networks and attractor cycles and show that the mutations which change the attractor structures more lead to lower classification error. We then generalize it to real gene expression trajectories by assuming an observation model on top of state dynamics. Thanks to recent single-cell sequencing technologies, we are now able to sequence the gene expressions inside every cell at each time point and generate meaningful gene expression trajectory data which can reflect the dynamics of gene regulatory networks. We design a classifier using single-cell gene expression trajectories and show that it can significantly outperform the classifiers which use static gene expression data derived from bulk gene expression technologies like RNA-Seq. The reason is that in bulk RNA-Seq, the expression values of genes are averaged over the population of cells and the dynamics cannot be captured. For the single-cell gene expression trajectories, we propose an expectation maximization (EM) algorithm with closed-from updates which can efficiently estimate the unknown parameters of the model using the observed trajectories. We finally study the single-cell gene expression trajectories in a Bayesian framework and propose intrinsically Bayesian robust classifier for the trajectories, where instead of estimating the unknown parameters, we assume they belong to an uncertainty class governed by a prior distribution. We evaluate all these methods on different important gene regulatory networks including p53 and cell cycle networks and demonstrate their efficacy in accurately classifying different phenotypes.

In chapter 3, we study transfer learning with a novel Bayesian viewpoint. Transfer learning appears where we do not have enough data in our target domain to train the machine learning algorithms well but have good amount of data in other relevant source domains. The probability distributions of the source and target domains might be totally different but they share some knowledge underlying the similar tasks between the domains and are related to each other in some sense. The ultimate goal of transfer learning is to find the amount of relatedness between the domains and then transfer the amount of knowledge to the target domain which can help improve the classification task in the data-poor target domain. Domain adaptation (DA) is a method to address transfer learning, where the source and target data are mapped to a common domain in which they follow a similar distribution. However, domain adaptation is not a very clever way of addressing transfer learning in that it can only work well when the two domains are highly related and can easily fail if there are huge distribution difference between the source and target domains. Indeed, many existing transfer learning and domain adaptation methods fail to answer questions regarding the relatedness and transferability between the source and target domains. As a result, they are very prone to *negative transfer*, which is a phenomenon in which transfer learning fails to transfer helpful knowledge to the target domain and consequently deteriorates the performance compared to the target-only training without transferring any knowledge. Negative transfer is the most vital issue in transfer learning and happens when the algorithm is not able to detect that the source domain is not related to the target domain for a specific task. For solving all these issues, we propose a novel transfer learning method based on a Bayesian framework, which is the scope of chapter 3. We propose a Bayesian transfer learning framework, where the source and target domains are related through the joint prior distribution of the model parameters. The modeling of joint prior densities enables better understanding of the transferability between domains. We first consider continuous data under a Gaussian model. We define a joint Wishart distribution for the precision matrices of the Gaussian feature-label distributions in the source and target domains to act like a bridge that transfers the useful information of the source domain to help classification in the target domain by improving the target posteriors. Using several theorems from multivariate statistics, the posteriors and posterior predictive densities are derived in closed forms in terms of hypergeometric functions of matrix argument, leading to our novel closed-form and fast Optimal Bayesian Transfer Learning (OBTL) classifier. Then we generalize the OBTL for the regression problem and similarly propose Optimal Bayesian Transfer Regression (OBTR). Finally, we extend our transfer learning idea to count data with the aim of cancer classification using the next generation sequencing data such as RNA-Seq. In this case, for addressing over-dispersion in RNA-Seq data, we use Negative Binomial (NB) and define joint priors for the parameters of the source and target domain. We learn the posteriors using Hamiltonian Monte Carlo (HMC) algorithm and then define the optimal transfer learning classifier for the count data. We evaluate the performance of our proposed transfer learning methods for classification and regression using both image and cancer datasets and show that they are able to optimally transfer the relevant knowledge from the source to the target domain and consequently improve the performance of data-poor target domains.

2. CLASSIFICATION OF GENE EXPRESSION TRAJECTORIES USING THE KNOWLEDGE OF GENE REGULATORY NETWORKS[*]

2.1 Classification of Gene State Trajectories

2.1.1 Overview

Gene-expression-based phenotype classification is used for disease diagnosis and prognosis relating to treatment strategies. In this section we study classification based on sequential measurements of multiple genes using gene regulatory network (GRN) modeling. We assume there are two networks, original (healthy) and mutated (cancerous), and observations consist of trajectories of network states. The problem is to classify an observation trajectory as coming from either the original or mutated network. GRNs are modeled via probabilistic Boolean networks (PBN), which incorporate stochasticity at both the gene and network levels. Mutation affects the regulatory logic. Classification is based upon observing a trajectory of states of some given length. We characterize the Bayes classifier and find the Bayes error for a general PBN and the special case of a single Boolean network affected by random perturbations (BNp). The Bayes error is related to network sensitivity, meaning the extent of alteration in the steady-state distribution of the original network owing to mutation. Using standard methods to calculate steady-state distributions is cumbersome and sometimes impossible, so we provide an efficient algorithm and approximations. Extensive simulations are performed to study the effects of various factors, including approximation accuracy. We apply the classification procedure to a p53 BNp and a mammalian cell cycle PBN.

[*]Reprinted with permission from “Classification of Single-Cell Gene Expression Trajectories from Incomplete and Noisy Data” by A. Karbalayghareh, U. Braga-Neto, and E. Dougherty 2019. IEEE/ACM Transactions on Computational Biology and Bioinformatics 16, no. 1, 193-207, Copyright 2019 IEEE.

[*]Reprinted with permission from “Intrinsically Bayesian robust classifier for single-cell gene expression trajectories in gene regulatory networks” by A. Karbalayghareh, U. Braga-Neto, and E. Dougherty 2018. BMC Systems Biology 12, no. 3, 1-10, Copyright 2018 BMC.

[*]Reprinted with permission from “Classification of State Trajectories in Gene Regulatory Networks” by A. Karbalayghareh, U. Braga-Neto, J. Hua, and E. Dougherty 2018. IEEE/ACM Transactions on Computational Biology and Bioinformatics 15, no. 1, 68-82, Copyright 2018 IEEE.

2.1.2 Introduction

Gene-expression-based phenotype classification was among the first applications proposed for high-throughput expression measurements, starting with DNA microarrays, the aim being disease diagnosis or prognosis relating to treatment strategies [11-14]. Owing to feature selection and error estimation in extremely high-dimensional spaces with limited sample data, accurate classification and error estimation have proven to be difficult, even with the advent of RNA-seq data [15-20]. Measurement noise in high-throughput data and heterogeneity across samples and patients increase the challenge. One proposed approach is to use groups of genes as features, such as merging genes in signaling pathways. This can help avoid redundant information contained in selected genes, for instance, selecting several genes in a pathway that are regulated by a single master gene [21]. The approach is to jointly analyze the expression levels of genes related by functionality, which can be obtained via transcriptome analysis [22-24], GO annotations [25], or other sources. Several methods have been proposed to measure the activity of a particular pathway: mean or median [26], first principle component [24], using a subset of genes in the pathway [27], and combining log-likelihood ratios of genes in the pathway [28].

While the aforementioned methods take advantage of multiple gene activity, in the end they all rely on single measurements and therefore do not take advantage of the regulatory information in trajectory data. In this section we consider classification based on sequential measurements of multiple genes. The problem is modeled via gene regulatory networks (GRNs). There are two networks, an original and one having undergone mutation, and observations consist of trajectories of network states, the classification problem being to classify an observation trajectory as coming from either the original or mutated network.

Before describing the mathematical setting, we note that traditionally it has been difficult to collect time-course gene-expression data for cancer, not only because cancer is known for its heterogeneity, but also because the cells are not synchronized. Thus, traditionally, for quality time-course expression data, one has to purify and synchronize the whole cell population, which is very challenging and cannot be achieved in a routine manner [29]. However, with the breakthrough in single-cell profiling, the problem can be overcome by profiling individual cells using RNA-Seq or quantitative PCR [30]. The individual cells can be captured via standard methods, such as flow cytometry, glass capillaries, or laser [31], and be measured at various time points. For example, in [32], between 49 to 77 cells have been collected at each time for 4 total time points and a software, Monocle, has been built to extract various gene-expression trajectories of individual cells. The authors have shown that key gene-expression transition sequences can be observed based on the trajectories. The method, according to the authors, can also be applied to time-course data collected via quantitative PCR. Since the regulation dynamic can provide a wide range of information not readily available from existing medical tests, driven by the need for personalized treatment, one would envision that in the future such procedures might be commercialized to help physicians make better diagnoses and choose the best treatments.

We model GRNs via probabilistic Boolean networks (PBNs) [33]. These characterize regulatory relations over discrete steps, which need not be time but instead can be related to gene state transitions such as in the cell cycle. They incorporate stochasticity at the gene level by allowing random gene perturbation and at the network level by consisting of multiple Boolean networks randomly selected based upon the activity of latent variables outside the network. While for simplicity we assume binary values, corresponding to a gene expressing or not expressing, the general PBN model makes no such assumption and the results of this work extend directly to multi-valued genes. PBNs have been used extensively for the study of optimal intervention based on control over time [34] and a one-time targeted alteration of the regulatory functions [35].

We assume that the mutated network has arisen from a mutation affecting the regulatory logic, and classification is based upon observing a trajectory of states of some given length. In this section, we characterize the Bayes classifier and find the Bayes error for a general PBN and the special case of a single Boolean network (BNp) affected by random perturbations. It will be seen that longer trajectories lower the Bayes error. We also see that if the two networks are similar and share some same attractor cycles, the longer trajectories are required to achieve a desired Bayes error close to zero, but when they are totally dissimilar, even the short trajectories can result in a lower Bayes error. As is commonly assumed (although not always mentioned), we suppose that classification is in the steady-state. Owing to perturbation, the Markov chain corresponding to a PBN is irreducible and hence possesses a steady-state distribution for the states.

We relate the Bayes error to the sensitivity of the network to mutation, where by sensitivity we mean quantification of the alteration in the steady-state distribution of the original network owing to mutation [36]. We define a trajectory-based notion of sensitivity suitable to the classification problem. Lack of sensitivity is good for cell survival because long-run wild-type state probabilities are not significantly altered, but this makes classification more difficult. If a function mutation significantly reduces the number of common attractor states in the two networks, then the Bayes error will be low. However, if the function mutation does not change the attractor structures of the original network, we cannot expect to have a Bayes error around zero, unless we have access to very long trajectories. The steady-state probabilities of attractors in PBNs have been derived in [37] with good approximations.

Using standard methods to calculate steady-state distributions is cumbersome and sometimes impossible due to the high computation time required for inverting large transition probability matrices (TPMs). We provide an efficient algorithm to help ease the computation; nevertheless, the computational burden is still prohibitive when the trajectory length is long, especially because we want to do simulations over large numbers of networks. Therefore, we provide approximations when the gene perturbation probability is small, a common assumption. We provide extensive simulations to study the effects of various factors, including the goodness of the approximations. Finally we apply the classification procedure to a p53 BNp and a mammalian cell cycle PBN.

2.1.3 Background

For a binary Boolean network (BN) on n genes, a truth table gives the functional relationships between the genes [38]. Each gene value $x_i \in \{0,1\}$, for $i = 1, \cdots, n$, at time $k+1$ is determined by the values of some predictor genes at time k via a Boolean function $f_i : \{0,1\}^n \to \{0,1\}$ in the truth table. In practice, f_i is a function of small number of genes, K_i, which is called input degree of the gene x_i in the network. Given a truth table, a gene network can be represented as a graph with vertices representing genes and edges representing regulations. Given an initial state, a BN will eventually reach a set of states, called an *attractor cycle*, through which it will cycle endlessly. Each initial state corresponds to a unique attractor cycle, and the set of initial states leading to a specific attractor cycle is known as the *basin of attraction* (BOA) of the attractor cycle.

2.1.3.1 Boolean Networks with perturbation (BNp)

For BNps, perturbation is introduced with a probability p by which the current state of the network can be randomly changed. Implicitly, we assume that there is an independent identically distributed (i.i.d.) random perturbation vector at each time k, denoted by $\mathbf{n}_k \in \{0,1\}^n$, where the ith gene flips at time k if the ith component of $\mathbf{n}_k$ is equal to 1. Therefore, the dynamical model of the states can be expressed as

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) \oplus \mathbf{n}_{k+1}, \qquad k = 0, 1, 2, \cdots, \tag{2.1}$$

where $\mathbf{x}_k = [x_1(k), x_2(k), \cdots, x_n(k)]^T$ is a binary state vector, called a *gene activity profile* (GAP), at time k, in which $x_i(k)$ indicates the expression level of the ith gene at time k (either 0 or 1); $\mathbf{f}(\mathbf{x}_k) = [f_1, f_2, \cdots, f_n]^T : \{0,1\}^n \to \{0,1\}^n$ is the vector of the network functions, in which f_i shows the expression level of the ith gene at time $k+1$ when the system lies in the state $\mathbf{x}_k$ at time k; $\mathbf{n}_k = [n_1(k), n_2(k), \cdots, n_n(k)]^T$ is the perturbation vector at time k, in which $n_1(k), n_2(k), \cdots, n_n(k)$ are i.i.d. Bernoulli random variables for every k with the parameter $p = P(n_i(k) = 1)$; and $\oplus$ is component-wise modulo 2 addition. The existence of perturbation makes the corresponding Markov chain of a BNp irreducible. Hence, the network possesses a steady-state distribution π describing its long-run behavior. A BNp inherits the attractor structure from the original BN without perturbation, the difference being that a random perturbation can cause a BNp to jump out of an attractor cycle, perhaps then transitioning to a different attractor cycle. If p is sufficiently small, π will reflect the attractor structure within the original network. We can derive the transition probability matrix (TPM) if we know the truth table and the perturbation probability for a BNp. As a result, the steady-state distribution π can be computed as well.

2.1.3.2 Probabilistic Boolean Network (PBN)

The network function in a PBN is not fixed and changes over time. We first consider *context-sensitive* PBNs, in which the current function governing the network will be changed if a switch ξ is on ($\xi = 1$), which has the probability q. When the switch is on, the new network function will be selected among L functions $\{\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \cdots, \mathbf{f}^{(L)}\}$ with corresponding probabilities $\{c_1, c_2, \cdots, c_L\}$. Then we analyze *instantaneously random* PBNs, which are a special case of context-sensitive PBNs with $q = 1$, meaning that the network functions are changing at each time point independently with the selection probabilities $\{c_1, c_2, \cdots, c_L\}$. The PBN dynamics are defined as

$$\mathbf{x}_{k+1} = \mathbf{f}_{k+1}(\mathbf{x}_k) \oplus \mathbf{n}_{k+1}, \qquad k = 0, 1, 2, \cdots, \tag{2.2}$$

where $\mathbf{x}_k$ and $\mathbf{n}_k$ denote the state and perturbation vectors, as defined in (2.1). Here, $\mathbf{f}_{k+1}$ is the network function at time $k+1$, which is randomly picked from the context set $\{\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \cdots, \mathbf{f}^{(L)}\}$, with the aforementioned probabilities, at each time $k+1$. A BNp is a PBN possessing a single context, that is, $L = 1$. When $L = 1$, $\mathbf{f}_{k+1} = \mathbf{f}^{(1)}$ in (2.2) is a constant function over time and (2.2) turns to the BNp definition in (2.1).

In this section, we address the following problem. Suppose the truth table governing the original BNp (or PBN) is altered to yield a *mutated* BNp (PBN), keeping p unchanged, and we are given an observed trajectory of the states in the steady-state. Our goal is to classify this arbitrary trajectory as belonging to the original or mutated BNp (PBN). We will obtain the Bayes classifier for this problem and find the corresponding Bayes error for the classifier.

2.1.4 Methods

2.1.4.1 Bayes Classifier

• Context-sensitive PBN:

We first consider the general case of context-sensitive PBNs. Let $\mathbf{x}_k$ denote an $n \times 1$ binary state vector at time k. Let $\bar{\mathbf{x}}_k = 1 + \sum_{i=1}^{n} x_i(k) \times 2^{i-1}$ denote the index of $\mathbf{x}_k$ at time k, such that

$\bar{\mathbf{x}}_k \in \{1, 2, \cdots, 2^n\}$; for instance, $(0, 0, ..., 0)^T \to 1$ and $(1, 1, ..., 1)^T \to 2^n$. In a context-sensitive PBN the state of the corresponding Markov chain at each time consists of both a GAP and a context, and is of the form $(\mathbf{X}_k, \mathbf{F}_k)$. Thus, the TPM has dimension $2^n \times L$ by $2^n \times L$. Its entries are the probabilities $P(\mathbf{X}_{k+1} = \mathbf{x}_{k+1}, \mathbf{F}_{k+1} = \mathbf{f}_{k+1} | \mathbf{X}_k = \mathbf{x}_k, \mathbf{F}_k = \mathbf{f}_k)$. In our classification problem we only have access to the GAP observations in the steady-state. As a result, we will use a hidden Markov model (HMM) to compute the probability of a given GAP trajectory in the steady-state.

The TPM in this case is a $2^n \times L$ by $2^n \times L$ matrix, whose entries are defined by

$$\begin{aligned} P(\mathbf{X}_{k+1} = \mathbf{x}_{k+1}, \mathbf{f}_{k+1} = \mathbf{f}^{(i)} | \mathbf{X}_k = \mathbf{x}_k, \mathbf{f}_k = \mathbf{f}^{(j)}) = \\ P(\mathbf{X}_{k+1} = \mathbf{x}_{k+1} | \mathbf{X}_k = \mathbf{x}_k, \mathbf{f}_{k+1} = \mathbf{f}^{(i)}) \\ \times P(\mathbf{f}_{k+1} = \mathbf{f}^{(i)} | \mathbf{f}_k = \mathbf{f}^{(j)}). \end{aligned} \tag{2.3}$$

From (2.2) we have

$$\begin{aligned} P(\mathbf{X}_{k+1} = \mathbf{x}_{k+1} | \mathbf{X}_k = \mathbf{x}_k, \mathbf{f}_{k+1} = \mathbf{f}^{(i)}) = \\ p^{d(\mathbf{x}_{k+1}, \mathbf{f}^{(i)}(\mathbf{x}_k))} (1-p)^{n - d(\mathbf{x}_{k+1}, \mathbf{f}^{(i)}(\mathbf{x}_k))}, \end{aligned} \tag{2.4}$$

where $d(\mathbf{x}_{k+1}, \mathbf{f}^{(i)}(\mathbf{x}_k))$ is the Hamming distance between two binary vectors $\mathbf{x}_{k+1}$ and $\mathbf{f}^{(i)}(\mathbf{x}_k)$. We know that $\{\mathbf{f}_k\}$ itself forms an irreducible Markov chain, and consequently possesses a steady-state distribution. If the current function is $\mathbf{f}^{(i)}$, then the probability that the function stays fixed at the next time is $1 - q + qc_i$, which is the addition of the probabilities of two exclusive events: the network switch is not called for (probability $1 - q$), or the switch is called for and the current network is selected again (probability qc_i). If the current function is not $\mathbf{f}^{(i)}$, then the probability that $\mathbf{f}^{(i)}$ will be the network function at the next time point is qc_i, which is the probability that the switch is called for and the function $\mathbf{f}^{(i)}$ is selected. Therefore, we can write,

$$P(\mathbf{f}_{k+1} = \mathbf{f}^{(i)} | \mathbf{f}_k = \mathbf{f}^{(j)}) = 1_{[i=j]}(1 - q + qc_i) + 1_{[i \neq j]}(qc_i), \tag{2.5}$$

for $i, j = 1, \cdots, L$. Consequently, from (2.3)-(2.5), the TPM has the following entries,

$$\begin{aligned} P(\mathbf{X}_{k+1} = \mathbf{x}_{k+1}, \mathbf{f}_{k+1} = \mathbf{f}^{(i)} | \mathbf{X}_k = \mathbf{x}_k, \mathbf{f}_k = \mathbf{f}^{(j)}) = \\ \{1_{[i=j]}(1 - q + qc_i) + 1_{[i \neq j]}(qc_i)\} \\ \times p^{d(\mathbf{x}_{k+1}, \mathbf{f}^{(i)}(\mathbf{x}_k))}(1-p)^{n-d(\mathbf{x}_{k+1}, \mathbf{f}^{(i)}(\mathbf{x}_k))}. \end{aligned} \tag{2.6}$$

Now we aim to compute the probability of a given GAP trajectory in a specific PBN in the steady-state. Suppose $\mathcal{X} = [\mathbf{x}_s, \mathbf{x}_{s+1}, \cdots, \mathbf{x}_{s+m-1}]$ is an observed trajectory of length m in the steady-state, where s is a time point in the steady-state, and $\mathcal{F} = [\mathbf{f}_{s+1}, \cdots, \mathbf{f}_{s+m-1}]$ is composed of the hidden network functions from time $s+1$ to $s+m-1$. The joint probability of $\mathcal{X}$ and $\mathcal{F}$ is

$$P(\mathcal{X}, \mathcal{F}) = P(\mathcal{X}|\mathcal{F})P(\mathcal{F}), \tag{2.7}$$

where $P(\mathcal{X}|\mathcal{F})$ and $P(\mathcal{F})$ can be factorized as

$$P(\mathcal{X}|\mathcal{F}) = P(\mathbf{x}_s) \prod_{k=1}^{m-1} P(\mathbf{x}_{s+k}|\mathbf{x}_{s+k-1}, \mathbf{f}_{s+k}) \tag{2.8}$$

$$P(\mathcal{F}) = P(\mathbf{f}_{s+1}) \prod_{k=2}^{m-1} P(\mathbf{f}_{s+k}|\mathbf{f}_{s+k-1}). \tag{2.9}$$

In order to derive the probability of trajectory, $P(\mathcal{X})$, we should marginalize the joint PMF of (2.7) over $\mathcal{F}$,

$$P(\mathcal{X}) = \sum_{\mathcal{F}} P(\mathcal{X}, \mathcal{F}), \tag{2.10}$$

which can be efficiently computed using the forward update in the structure of the HMM due to the factorization in (2.8) and (2.9). We define the vectors α_k, for $k = s+1, \cdots, s+m-1$, with the i-th entry,

$$\alpha_{s+1}(i) = P(\mathbf{x}_s)P(\mathbf{f}_{s+1} = \mathbf{f}^{(i)})P(\mathbf{x}_{s+1}|\mathbf{x}_s, \mathbf{f}_{s+1} = \mathbf{f}^{(i)}),$$
$$\alpha_{s+k+1}(i) = P(\mathbf{x}_{s+k+1}|\mathbf{x}_{s+k}, \mathbf{f}_{s+k+1} = \mathbf{f}^{(i)}) \times \sum_{j=1}^{L} \alpha_{s+k}(j) P(\mathbf{f}_{s+k+1} = \mathbf{f}^{(i)}|\mathbf{f}_{s+k} = \mathbf{f}^{(j)}), \quad (2.11)$$

where $k = 1, \cdots, m-2$ and $i = 1, \cdots, L$.

Proposition 1: The steady-state distribution of the network functions is equal to their selection probabilities, that is, $P(\mathbf{f}_k = \mathbf{f}^{(i)}) = c_i$, where $k \to \infty$.

Proof: From the total probability rule,

$$P(\mathbf{f}_{k+1} = \mathbf{f}^{(i)}) = P(\mathbf{f}_{k+1} = \mathbf{f}^{(i)}|\mathbf{f}_k = \mathbf{f}^{(i)})P(\mathbf{f}_k = \mathbf{f}^{(i)}) + P(\mathbf{f}_{k+1} = \mathbf{f}^{(i)}|\mathbf{f}_k \neq \mathbf{f}^{(i)})P(\mathbf{f}_k \neq \mathbf{f}^{(i)}). \quad (2.12)$$

As mentioned before, $\{\mathbf{f}_k\}$ is an irreducible Markov chain and has a steady-state distribution, that is, $P(\mathbf{f}_{k+1} = \mathbf{f}^{(i)}) = P(\mathbf{f}_k = \mathbf{f}^{(i)})$ when $k \to \infty$; using this and (2.5) in (2.12) leads to $P(\mathbf{f}_k = \mathbf{f}^{(i)}) = c_i$, where $k \to \infty$.

According to Proposition 1, $P(\mathbf{f}_{s+1} = \mathbf{f}^{(i)}) = c_i$ in (2.11). Furthermore, $P(\mathbf{x}_s) = \pi_{\bar{\mathbf{x}}_s}$ in (2.11) is the steady-state distribution of $\mathbf{x}_s$, which can be computed from the TPM in (2.6). Hence, we can rewrite (2.11) as

$$\alpha_{s+1}(i) = \pi_{\bar{\mathbf{x}}_s} c_i P(\mathbf{x}_{s+1}|\mathbf{x}_s, \mathbf{f}_{s+1} = \mathbf{f}^{(i)}),$$
$$\alpha_{s+k+1}(i) = P(\mathbf{x}_{s+k+1}|\mathbf{x}_{s+k}, \mathbf{f}_{s+k+1} = \mathbf{f}^{(i)}) \times \sum_{j=1}^{L} \alpha_{s+k}(j) P(\mathbf{f}_{s+k+1} = \mathbf{f}^{(i)}|\mathbf{f}_{s+k} = \mathbf{f}^{(j)}). \quad (2.13)$$

Using (2.4), (2.5), and (2.13), we can compute α_{s+m-1}. Finally, the probability of the trajectory $\mathcal{X}$

can be computed by summing the entries of α_{s+m-1} as

$$P(\mathcal{X}) = \sum_{i=1}^{L} \alpha_{s+m-1}(i). \tag{2.14}$$

Suppose $\pi_{\bar{\mathbf{x}}_s}$ and $\tilde{\pi}_{\bar{\mathbf{x}}_s}$ are the steady-state probabilities of being in state $\mathbf{x}_s$ in the original and mutated PBNs, respectively, $\{\mathbf{f}^{(1)}, \cdots, \mathbf{f}^{(L)}\}$ and $\{\tilde{\mathbf{f}}^{(1)}, \cdots, \tilde{\mathbf{f}}^{(L)}\}$ are the L constituent network functions of the original and mutated PBNs, respectively, and $\{c_1, c_2, \cdots, c_L\}$ and $\{\tilde{c}_1, \tilde{c}_2, \cdots, \tilde{c}_L\}$ are the corresponding function selection probabilities. As a result, the probability of the trajectory $\mathcal{X}$ in the original network, $P(\mathcal{X}|PBN_{original})$, can be computed from (2.14). We can also compute the probability of the trajectory $\mathcal{X}$ in the mutated network, $P(\mathcal{X}|PBN_{mutated})$, using (2.14), but we should use $\tilde{\pi}_{\bar{\mathbf{x}}_s}$, $\tilde{\mathbf{f}}^{(i)}$, $\tilde{c}_i$, and $\tilde{q}$ instead of $\pi_{\bar{\mathbf{x}}_s}$, $\mathbf{f}^{(i)}$, c_i and q, respectively, in (2.1)-(2.14).

Let p_0 and p_1 be the prior probabilities of the original and mutated PBNs, respectively. Then the posterior probabilities of the classes after observing the trajectory $\mathcal{X}$ are

$$\eta(\mathcal{X}) = P(PBN_{mut.}|\mathcal{X}) = \frac{p_1 P(\mathcal{X}|PBN_{mut.})}{P(\mathcal{X})}, \tag{2.15}$$

$$1 - \eta(\mathcal{X}) = P(PBN_{orig.}|\mathcal{X}) = \frac{p_0 P(\mathcal{X}|PBN_{orig.})}{P(\mathcal{X})}. \tag{2.16}$$

The Bayes classifier is given by

$$\psi^{\star}(\mathcal{X}) = \begin{cases} 1, & \eta(\mathcal{X}) \geq 1 - \eta(\mathcal{X}) \\ 0, & \eta(\mathcal{X}) < 1 - \eta(\mathcal{X}) \end{cases}$$

$$= \begin{cases} 1, & p_1 P(\mathcal{X}|PBN_{mut.}) \geq p_0 P(\mathcal{X}|PBN_{orig.}) \\ 0, & p_1 P(\mathcal{X}|PBN_{mut.}) < p_0 P(\mathcal{X}|PBN_{orig.}) \end{cases}. \tag{2.17}$$

In (2.17), the classes 0 and 1 denote the original and mutated PBNs, respectively. If we assume the classes are equally likely, $p_0 = p_1 = \frac{1}{2}$, then p_0 and p_1 can be dropped from both sides of the inequality in (2.17).

• Instantaneously random PBN:

Now we consider the case $q = 1$, which means that the network functions are changing at each time point. The probability of selecting $\mathbf{f}^{(i)}$ at time $k+1$ is independent of the previous network function at time k and is equal to c_i. We can see this fact from (2.5):

$$P(\mathbf{f}_{k+1} = \mathbf{f}^{(i)} | \mathbf{f}_k = \mathbf{f}^{(j)}) = P(\mathbf{f}_{k+1} = \mathbf{f}^{(i)}) = c_i. \tag{2.18}$$

Furthermore, the TPM in (2.6) becomes

$$\begin{aligned} &P(\mathbf{X}_{k+1} = \mathbf{x}_{k+1}, \mathbf{f}_{k+1} = \mathbf{f}^{(i)} | \mathbf{X}_k = \mathbf{x}_k, \mathbf{f}_k = \mathbf{f}^{(j)}) = \\ &\qquad c_i p^{d(\mathbf{x}_{k+1}, \mathbf{f}^{(i)}(\mathbf{x}_k))} (1-p)^{n-d(\mathbf{x}_{k+1}, \mathbf{f}^{(i)}(\mathbf{x}_k))} \\ &= P(\mathbf{X}_{k+1} = \mathbf{x}_{k+1}, \mathbf{f}_{k+1} = \mathbf{f}^{(i)} | \mathbf{X}_k = \mathbf{x}_k). \end{aligned} \tag{2.19}$$

As a result, the TPM of the GAP can be achieved by marginalizing (2.19) over $\mathbf{f}^{(i)}$ as

$$\begin{aligned} &P(\mathbf{X}_{k+1} = \mathbf{x}_{k+1} | \mathbf{X}_k = \mathbf{x}_k) = \\ &\sum_{i=1}^{L} c_i p^{d(\mathbf{x}_{k+1}, \mathbf{f}^{(i)}(\mathbf{x}_k))} (1-p)^{n-d(\mathbf{x}_{k+1}, \mathbf{f}^{(i)}(\mathbf{x}_k))}. \end{aligned} \tag{2.20}$$

Note that in this case, $P(\mathcal{F})$ in (2.9) is factorized as (based on independence)

$$P(\mathcal{F}) = \prod_{k=1}^{m-1} P(\mathbf{f}_{s+k}). \tag{2.21}$$

From (2.7), (2.8), (2.10), and (2.21), we have

$$P(\mathcal{X}) = \pi_{\bar{\mathbf{x}}_s} \prod_{k=0}^{m-2} P(\mathbf{X}_{s+k+1} = \mathbf{x}_{s+k+1} | \mathbf{X}_{s+k} = \mathbf{x}_{s+k}). \tag{2.22}$$

Therefore, from (2.20) and (2.22), the probabilities of the steady-state GAP trajectory $\mathcal{X}$ in the original and mutated instantaneously random PBNs are

$$P(\mathcal{X}|PBN_{original}) = \pi_{\bar{\mathbf{x}}_s} \prod_{k=0}^{m-2} \left\{ \sum_{i=1}^{L} c_i p^{d(\mathbf{x}_{s+k+1}, \mathbf{f}^{(i)}(\mathbf{x}_{s+k}))} (1-p)^{n-d(\mathbf{x}_{s+k+1}, \mathbf{f}^{(i)}(\mathbf{x}_{s+k}))} \right\}, \quad (2.23)$$

$$P(\mathcal{X}|PBN_{mutated}) = \tilde{\pi}_{\bar{\mathbf{x}}_s} \prod_{k=0}^{m-2} \left\{ \sum_{i=1}^{L} \tilde{c}_i p^{d(\mathbf{x}_{s+k+1}, \tilde{\mathbf{f}}^{(i)}(\mathbf{x}_{s+k}))} (1-p)^{n-d(\mathbf{x}_{s+k+1}, \tilde{\mathbf{f}}^{(i)}(\mathbf{x}_{s+k}))} \right\}. \quad (2.24)$$

The Bayes classifier is the same as in (2.17).

• BNp:

According to (2.1), the TPM of a BNp is a $2^n \times 2^n$ matrix with the following entries.

$$P(\mathbf{X}_{k+1} = \mathbf{x}_{k+1} | \mathbf{X}_k = \mathbf{x}_k) = p^{d(\mathbf{x}_{k+1}, \mathbf{f}(\mathbf{x}_k))} (1-p)^{n-d(\mathbf{x}_{k+1}, \mathbf{f}(\mathbf{x}_k))}. \quad (2.25)$$

We can see that the TPM in (2.25) can also be achieved from the TPM of the GAP in the instantaneously random PBN (2.20) by letting $L = 1$, $c_1 = 1$, and $\mathbf{f}^{(1)} = \mathbf{f}$. The probability of $\mathcal{X}$ in the BNp is

$$P(\mathcal{X}) = \pi_{\bar{\mathbf{x}}_s} \prod_{k=0}^{m-2} P(\mathbf{X}_{s+k+1} = \mathbf{x}_{s+k+1} | \mathbf{X}_{s+k} = \mathbf{x}_{s+k}). \quad (2.26)$$

Using (2.25) and (2.26), the probability of $\mathcal{X}$ in the original and mutated BNps can be written as

$$\begin{aligned} P(\mathcal{X}|BNp_{original}) &= \pi_{\bar{\mathbf{x}}_s} p^{\sum_{k=0}^{m-2} d(\mathbf{x}_{s+k+1}, \mathbf{f}(\mathbf{x}_{s+k}))} \\ &\quad \times (1-p)^{n(m-1) - \sum_{k=0}^{m-2} d(\mathbf{x}_{s+k+1}, \mathbf{f}(\mathbf{x}_{s+k}))}, \end{aligned} \quad (2.27)$$

$$\begin{aligned} P(\mathcal{X}|BNp_{mutated}) &= \tilde{\pi}_{\bar{\mathbf{x}}_s} p^{\sum_{k=0}^{m-2} d(\mathbf{x}_{s+k+1}, \tilde{\mathbf{f}}(\mathbf{x}_{s+k}))} \\ &\quad \times (1-p)^{n(m-1) - \sum_{k=0}^{m-2} d(\mathbf{x}_{s+k+1}, \tilde{\mathbf{f}}(\mathbf{x}_{s+k}))}, \end{aligned} \quad (2.28)$$

where $\mathbf{f}$ and $\tilde{\mathbf{f}}$ are the network functions of the original and mutated BNps, respectively. Similar to

(2.17), the Bayes classifier is

$$\psi^{\star}(\mathcal{X}) = \begin{cases} 1, & p_1 P(\mathcal{X}|BNp_{mut.}) \geq p_0 P(\mathcal{X}|BNp_{orig.}) \\ 0, & p_1 P(\mathcal{X}|BNp_{mut.}) < p_0 P(\mathcal{X}|BNp_{orig.}) \end{cases}, \tag{2.29}$$

where the classes 0 and 1 denote the original and mutated BNps, respectively.

The GAP steady-state distribution $\pi = [\pi_1, \pi_2, \cdots, \pi_{2^n}]$ can be easily calculated using the fact that $\pi = \pi P$ and $\sum_{i=1}^{2^n} \pi_i = 1$, where P is the GAP TPM of the original instantaneously random PBN and original BNp, whose entries are respectively computed using (2.20) and (2.25). More specifically, $\pi = \pi P$ can be written in the form $\pi(I - P) = 0$. We know $I - P$ is not a full-rank matrix and that one out of 2^n linear equations depends on the others. Therefore, we remove one column (say the last column) of the matrix $I - P$ and replace it by an all-one column, which adds the normalization constraint, $\sum_{i=1}^{2^n} \pi_i = 1$, to the set of the linear equations $\pi = \pi P$. Calling the resultant matrix Q, we have

$$\pi = [0, 0, \cdots, 0, 1]Q^{-1}. \tag{2.30}$$

We know from the Markov chain properties that Q is a full-rank matrix and has an inverse. We should also note that although computing the steady-state distribution using (2.30) is exact, it may not be the most efficient method in terms of the computation time. As a result, we may need to use the approximate and faster algorithms, like power methods, to compute π in very large networks. For the mutated PBN, $\tilde{\pi}$ can be similarly derived. However, as we will study the behavior of the average Bayes error over many random networks and many random mutations, we will provide an algorithm to compute $\tilde{\pi}$ very efficiently without a need for matrix inversion like in (2.30).

2.1.4.2 Bayes Error and Long-run Sensitivity

This section provides the Bayes error for the previously derived Bayes classifiers. It uses the fact that the Bayes error can be expressed via the posterior probabilities by

$$\epsilon^{\star} = E\left[\min\{\eta(\mathcal{X}), 1 - \eta(\mathcal{X})\}\right]. \tag{2.31}$$

• Bayes error for BNps:

From (2.15), (2.16) (replacing PBNs with BNps), and (2.31), and with the assumption of equally likely BNps, i. e., $p_0 = p_1 = \frac{1}{2}$, the Bayes error can be written as

$$\epsilon^\star = \frac{1}{2}\sum_{\mathcal{X}} \min\left\{P(\mathcal{X}|BNp_{orig.}),\ P(\mathcal{X}|BNp_{mut.})\right\}, \tag{2.32}$$

where $P(\mathcal{X}|BNp_{original})$ and $P(\mathcal{X}|BNp_{mutated})$ are given in (2.27) and (2.28), and the summation is over all possible 2^{mn} trajectories of the length m. For long trajectories (large m) and big networks (large n), 2^{mn} is huge, and it is impossible to compute the exact Bayes error using (2.32). We will consider approximation to reduce the complexity of computing the Bayes error.

We begin by writing the Bayes error in terms of the *trajectory long-run sensitivity*, $\omega_m(\mathbf{f}, \tilde{\mathbf{f}})$, which we define as the total absolute change in the steady-state probability masses of trajectories of length m resulting from changing $\mathbf{f}$ to $\tilde{\mathbf{f}}$. Using the equality

$$\min\{a, b\} = \frac{a+b}{2} - \frac{|a-b|}{2}, \tag{2.33}$$

in conjunction with (2.32) and the fact that $\sum_{\mathcal{X}} P(\mathcal{X}|BNp_{original}) = \sum_{\mathcal{X}} P(\mathcal{X}|BNp_{mutated}) = 1$, the Bayes error can be expressed as

$$\epsilon^\star = \frac{1}{2}\left[1 - \omega_m(\mathbf{f}, \tilde{\mathbf{f}})\right], \tag{2.34}$$

where

$$\omega_m(\mathbf{f}, \tilde{\mathbf{f}}) = \frac{1}{2}\sum_{\mathcal{X}} \left|P(\mathcal{X}|BNp_{orig.}) - P(\mathcal{X}|BNp_{mut.})\right|. \tag{2.35}$$

Note that $0 \leq \omega_m(\mathbf{f}, \tilde{\mathbf{f}}) \leq 1$ and $\omega_m(\mathbf{f}, \tilde{\mathbf{f}})$ is also called Kolmogorov's variational distance. According to (2.34), the difficulty of classification is inversely related to the sensitivity, with $\epsilon^\star \approx 0.5$ when $\omega_m(\mathbf{f}, \tilde{\mathbf{f}}) \approx 0$, and $\epsilon^\star \approx 0$ when $\omega_m(\mathbf{f}, \tilde{\mathbf{f}}) \approx 1$. The more sensitive a network is to mutation, the easier it is to classify.

The main challenge in calculating the sensitivity in (2.35) is summation over the 2^{mn} trajectories $\mathcal{X}$. In most cases, the perturbation probability p is small. Assuming p is sufficiently small, we can reduce the trajectory space to achieve a good approximation for the sensitivity with a feasible computational complexity. To this end, we consider the trajectories in which there is at most one gene perturbation. The Bayes error in (2.32) and the long-run sensitivity in (2.35) are some functions of the probabilities given in (2.27) and (2.28), and those probabilities have a term p to the power of $\sum_{k=0}^{m-2} d(\mathbf{x}_{s+k+1}, \mathbf{f}(\mathbf{x}_{s+k}))$ and $\sum_{k=0}^{m-2} d(\mathbf{x}_{s+k+1}, \tilde{\mathbf{f}}(\mathbf{x}_{s+k}))$ respectively in the original and mutated BNps. As a result, when p is small enough, we can only consider the trajectories for which those Hamming distances are equal to 0 or 1, and terms with higher powers of p are negligible and a good approximation results by ignoring them. Using this fact, we define the reduced space of trajectories of length m in the original BNp by

$$R_0 = \left\{ \mathcal{X} \;\middle|\; \sum_{k=0}^{m-2} d(\mathbf{x}_{s+k+1}, \mathbf{f}(\mathbf{x}_{s+k})) = 0 \right\}, \tag{2.36}$$

$$R_1 = \left\{ \mathcal{X} \;\middle|\; \sum_{k=0}^{m-2} d(\mathbf{x}_{s+k+1}, \mathbf{f}(\mathbf{x}_{s+k})) = 1 \right\}, \tag{2.37}$$

and $R = R_0 \cup R_1$, where $\mathcal{X} = [\mathbf{x}_s, \mathbf{x}_{s+1}, \cdots, \mathbf{x}_{s+m-1}]$. Since in the Boolean networks, there is only one directed edge between any two states, starting from state $\mathbf{x}_s$ there is only one trajectory of length m. Thus, the cardinality of R_0 is $|R_0| = 2^n$. Since there are $m-1$ state transitions, and in each transition there are n positions to apply one gene perturbation, $|R_1| = n(m-1)2^n$. Since R_0 and R_1 are disjoint, $|R| = |R_0| + |R_1| = 2^n(nm - n + 1)$. We analogously define $\tilde{R}_0$ and $\tilde{R}_1$ for the mutated BNp by using $\tilde{\mathbf{f}}$ instead of $\mathbf{f}$ in (2.36) and (2.37), and define $\tilde{R} = \tilde{R}_0 \cup \tilde{R}_1$, for which $|\tilde{R}| = |R| = 2^n(nm - n + 1)$. If $p \approx 0$, then $P(\mathcal{X}|BNp_{original}) \approx 0$ for $\mathcal{X} \notin R$ and $P(\mathcal{X}|BNp_{mutated}) \approx 0$ for $\mathcal{X} \notin \tilde{R}$. Due to the minimum function in the Bayes error in (2.32) and the absolute value function in the sensitivity in (2.35), only trajectories in $R \cup \tilde{R}$ play a non-negligible role in determining the Bayes error and sensitivity. Moreover $|R \cup \tilde{R}| \leq 2^{n+1}(nm - n + 1)$, which is much less than 2^{mn} for large m. Since the summands in (2.32) and (2.35) have positive values, we have the following lower bounds for the Bayes error and long-run sensitivity:

$$\epsilon^\star \geq \frac{1}{2} \sum_{\mathcal{X} \in R \cup \tilde{R}} \min\left\{ P(\mathcal{X}|BNp_{orig.}),\ P(\mathcal{X}|BNp_{mut.}) \right\}, \tag{2.38}$$

$$\omega_m(\mathbf{f}, \tilde{\mathbf{f}}) \geq \frac{1}{2} \sum_{\mathcal{X} \in R \cup \tilde{R}} \left| P(\mathcal{X}|BNp_{orig.}) - P(\mathcal{X}|BNp_{mut.}) \right|. \tag{2.39}$$

From (2.34) and (2.39), an upper bound for the Bayes error can be derived as

$$\epsilon^\star \leq \frac{1}{2} \left[1 - \frac{1}{2} \sum_{\mathcal{X} \in R \cup \tilde{R}} \left| P(\mathcal{X}|BNp_{orig.}) - P(\mathcal{X}|BNp_{mut.}) \right| \right]. \tag{2.40}$$

The tightness of the lower and upper bounds in (2.38) and (2.40) depends on how small p is. As $p \to 0$, these bounds converge to the Bayes error. However, when p increases, the gap between the bounds grows. A tight approximation requires that p be sufficiently small. We will examine this with simulations.

• Bayes error in instantaneously random PBNs:

From (2.15), (2.16), and (2.31), and assuming equally likely PBNs, i. e., $p_0 = p_1 = \frac{1}{2}$, the Bayes error in classifying the two PBNs (original and mutated) is

$$\epsilon^\star = \frac{1}{2} \sum_{\mathcal{X}} \min\left\{ P(\mathcal{X}|PBN_{orig.}),\ P(\mathcal{X}|PBN_{mut.}) \right\}. \tag{2.41}$$

Again, this summation is over 2^{mn} possible trajectories. To reduce computational complexity, we can analogously define R_0 and R_1 as for BNps; however, for PBNs, the corresponding reduction is insufficient on account of context switching. Thus, we must reduce even further and only consider R_0, in which case, owing to context switching, $|R_0| \leq L^{m-1} \times 2^n$, where L is the number of constituent BNs in the PBN. Based on (2.23) and (2.24), which are for the instantaneously random

PBNs, we restrict our computation to

$$R_0 = \left\{ \mathcal{X} \;\middle|\; \prod_{k=0}^{m-2} \left\{ \sum_{i=1}^{L} c_i \mathbf{1}_{[\mathbf{x}_{s+k+1} = \mathbf{f}^{(i)}(\mathbf{x}_{s+k})]} \right\} \neq 0 \right\}, \tag{2.42}$$

$$\tilde{R}_0 = \left\{ \mathcal{X} \;\middle|\; \prod_{k=0}^{m-2} \left\{ \sum_{i=1}^{L} \tilde{c}_i \mathbf{1}_{[(\mathbf{x}_{s+k+1} = \tilde{\mathbf{f}}^{(i)}(\mathbf{x}_{s+k})]} \right\} \neq 0 \right\}, \tag{2.43}$$

for the original and mutated PBNs, under the assumption that $p \approx 0$. The Bayes error lower and upper bounds, similar to (2.38) and (2.40), are

$$\epsilon^{\star} \geq \frac{1}{2} \sum_{\mathcal{X} \in R_0 \cup \tilde{R}_0} \min \left\{ P(\mathcal{X} | PBN_{orig.}),\; P(\mathcal{X} | PBN_{mut.}) \right\}, \tag{2.44}$$

$$\epsilon^{\star} \leq \frac{1}{2} \left[1 - \frac{1}{2} \sum_{\mathcal{X} \in R_0 \cup \tilde{R}_0} \left| P(\mathcal{X} | PBN_{orig.}) - P(\mathcal{X} | PBN_{mut.}) \right| \right], \tag{2.45}$$

respectively. Simulations will demonstrate the tightness of the bounds.

2.1.4.3 Markov Chain Perturbation Theory and Multiple Function Mutations

The steady-state distribution $\tilde{\pi}$ of the mutated BNp governed by $\tilde{\mathbf{f}}$ can be computed similarly to π using (2.30) by replacing Q by $\tilde{Q}$; however, computational savings can be had by using Markov Chain perturbation theory to derive $\tilde{\pi}$ directly from π and the TPMs of the original and mutated BNps, P and $\tilde{P}$. Keep in mind that we are referring to function perturbations as mutations and we will state the original Markov Chain perturbation theory in terms of mutations so that it is consistent with our terminology. A *rank-one mutation* (perturbation) has the TPM $\tilde{P} = P + ab^T$, where a and b are two arbitrary column vectors satisfying $b^T e = 0$, where e is an all-one column vector.

Theorem [35]: Consider a rank-one mutation for which $\tilde{P} = P + ab^T$ and let π (a row vector) and $Z = [I - P + e\pi]^{-1}$ be the steady-state distribution and the fundamental matrix of the original Markov chain, respectively. Then, the steady-state distribution and the fundamental matrix of the mutated Markov chain are respectively given by

$$\tilde{\pi} = \pi + \frac{\pi a}{1 - b^T Z a} b^T Z, \tag{2.46}$$

$$\tilde{Z} = \left[I - \frac{\pi a}{1 - b^T Z a} e b^T Z \right] \left[Z + \frac{Z a b^T Z}{1 - b^T Z a} \right]. \tag{2.47}$$

One special case of a rank-one mutation is a mutation in only one state, which changes only one row of the TPM. In this case, $a = e_k$ and e_k is a vector with 1 in the k-th entry and 0s in the other entries. We consider a commonly used 1-bit function mutation in which the output of only one gene is flipped in the transition from a specific state and the other outputs are kept unchanged. If $\mathbf{x}^\star$ is the state in which the output of the $i^\star$-th gene is mutated (flipped), then we can write $\tilde{\mathbf{f}}(\mathbf{x}^\star) = \mathbf{f}(\mathbf{x}^\star) \oplus e_{i^\star}$, and $\tilde{\mathbf{f}}(\mathbf{x}) = \mathbf{f}(\mathbf{x})$ for $\mathbf{x} \neq \mathbf{x}^\star$. Since there is only a change in $\bar{\mathbf{x}}^\star$-th row of the TPM P, $a = e_{\bar{\mathbf{x}}^\star}$. Furthermore, b^T can easily be computed by subtracting the $\bar{\mathbf{x}}^\star$-th rows of the original and mutated TPMs, respectively, P and $\tilde{P}$.

More complicated function mutations can be considered by extending to multiple 1-bit mutations. Indeed, all the complex function mutations can be viewed as several 1-bit mutations taking place successively. As a result, the steady-state distribution can be again obtained from the results of the Markov chain perturbation theory in a recursive manner. When we have multiple 1-bit function mutations, we split them into several 1-bit function mutations for which we can use (2.46) to compute the steady-state distribution. For the second 1-bit function mutation, we update π and Z using (2.46) and (2.47) and similarly compute the steady-state distribution and so forth.

Algorithm 1 shows how to compute the Bayes error, given the original BNp and mutated BNp after applying multiple 1-bit function mutations. Using this algorithm, we only need one matrix inversion for the original BNp. For the mutated BNps, we can update π and Z based on the algorithm, without a need for matrix inversion. This can considerably reduce the complexity, especially when we want to study the effect of many multiple 1-bit function mutations to obtain the average Bayes error over many randomly generated BNps. This helps more in the case of large networks. With n genes, the dimension of the matrices to be inverted is $2^n \times 2^n$.

Algorithm 1 Computing the Bayes error for multiple 1-bit function mutations in BNps

```
1:  procedure
2:      Initialize the number of the genes: $n$
3:      Initialize the length of trajectory: $m$
4:      Initialize the gene perturbation probability: $p$
5:      Initialize the number of the 1-bit function mutations: $n_{mut}$
6:      Initialize the position of the $k$-th function perturbation: $(\mathbf{x}^\star(k), i^\star(k))$, where $\bar{\mathbf{x}}^\star(k) \in \{1, 2, \cdots 2^n\}$ and $i^\star(k) \in \{1, 2, \cdots, n\}$ for $k = 1, \cdots, n_{mut}$.
7:      Initialize the original BN: $\mathbf{f}$ and save it: $\mathbf{f}_o \leftarrow \mathbf{f}$
8:      Initialize the perturbed BN: $\tilde{\mathbf{f}} \leftarrow \mathbf{f}$
9:      Compute the TPM of the original BNp using (2.25): $P$
10:     Compute the SS distribution of the original BNp: $\pi \leftarrow [0, \cdots, 1]Q^{-1}$
11:     Compute the fundamental matrix of the original BNp: $Z \leftarrow [I - P + e\pi]^{-1}$
12:     Compute the reduced trajectory set: $R$
13:     for $k = 1 : n_{mut}$ do
14:         $\tilde{\mathbf{f}}(\mathbf{x}^\star(k)) \leftarrow \mathbf{f}(\mathbf{x}^\star(k)) \oplus e_{i^\star(k)}$
15:         Compute $\bar{\mathbf{x}}^\star(k)$-th row of $\tilde{P}$, that is, $\tilde{P}(\bar{\mathbf{x}}^\star(k), :)$,        from (2.25) (use $\tilde{\mathbf{f}}$ instead of $\mathbf{f}$)
16:         $a \leftarrow e_{\bar{\mathbf{x}}^\star(k)}$
17:         $b^T \leftarrow \tilde{P}(\bar{\mathbf{x}}^\star(k), :) - P(\bar{\mathbf{x}}^\star(k), :)$
18:         $\tilde{\pi} \leftarrow \pi + \frac{\pi a}{1 - b^T Z a} b^T Z$
19:         if $k == n_{mut}$ then
20:             • Compute the reduced trajectory set: $\tilde{R}$
21:             • Compute the Bayes error's bounds from        (2.38) and (2.40).
22:             • break
23:         else
24:             $Z \leftarrow \left[I - \frac{\pi a}{1 - b^T Z a} e b^T Z\right] \left[Z + \frac{Z a b^T Z}{1 - b^T Z a}\right]$
25:             $\mathbf{f} \leftarrow \tilde{\mathbf{f}}$
26:             $P \leftarrow \tilde{P}$
27:             $\pi \leftarrow \tilde{\pi}$
28:         end if
29:     end for
30: end procedure
```

As mentioned in Algorithm 1, $(\mathbf{x}^{\star}(k), i^{\star}(k))$ shows the position of the k-th 1-bit function mutation for $k = 1, \cdots, n_{mut}$, where $\mathbf{x}^{\star}(k)$ is the state in which the output of the gene $i^{\star}(k)$ is flipped. As there are $n2^n$ choices for a 1-bit mutation, the number of all possible n_{mut} 1-bit function mutations is $C(n2^n, n_{mut})$. Considering all possible function mutations is computationally impossible for even a moderate n and n_{mut}. Hence, in simulating a great number of networks, we generate a few random 1-bit mutations of length n_{mut} and average the Bayes error over these random mutations.

2.1.4.4 Studying the Bayes error and sensitivity of BNps when $p \approx 0$

We reduced the BNp trajectory space from cardinality 2^{mn} to at most $2^{n+1}(nm-n+1)$ by only considering trajectories in $R_0 \cup R_1$ and $\tilde{R}_0 \cup \tilde{R}_1$, where R_0 ($\tilde{R}_0$) and R_1 ($\tilde{R}_1$) were respectively the trajectory spaces with no and only one gene perturbations; however, when n is large, this can still be too time-consuming for simulations involving many random networks. If we assume $p \approx 0$ in such cases, we can only consider trajectories in R_0 and $\tilde{R}_0$. Under this assumption, the computation time of the Bayes error will be very fast and we only need to find the attractor states, since the non-attractor states have zero steady-state probabilities when $p \approx 0$. Recall that $|R_0| = |\tilde{R}_0| = 2^n$.

Proposition 2: The sensitivity and Bayes error possess the limits

$$\lim_{p \to 0} \omega_m(\mathbf{f}, \tilde{\mathbf{f}}) = \frac{1}{2} \left\{ \sum_{i \in A \setminus C_m} \pi_i + \sum_{i \in B \setminus C_m} \tilde{\pi}_i + \sum_{i \in C_m} |\pi_i - \tilde{\pi}_i| \right\}, \tag{2.48}$$

$$\lim_{p \to 0} \epsilon^{\star} = \frac{1}{2} \sum_{i \in C_m} \min\{\pi_i, \tilde{\pi}_i\}, \tag{2.49}$$

where A and B are the attractor states of the original and mutated BNps, respectively, and C_m is the set of common attractor states of the two BNps from which there exists an identical trajectory of length $m-1$ in the two graphs,

$$C_m = \left\{ \bar{\mathbf{x}}_s \;\middle|\; \mathbf{x}_s \in A \cap B, \quad \mathcal{X} \in R_0 \cap \tilde{R}_0 \right\}. \tag{2.50}$$

Proof: If $\mathbf{x}_s$ is not an attractor state, then $\pi_{\bar{\mathbf{x}}_s} \to 0$ as $p \to 0$. Since $p \to 0$, from (2.27), (2.28), and (2.36),

$$P(\mathcal{X}|BNp_{orig.}) = \mathbf{1}_{[\mathcal{X}\in R_0,\ \mathbf{x}_s\in A]}\ \pi_{\bar{\mathbf{x}}_s}, \tag{2.51}$$

$$P(\mathcal{X}|BNp_{mut.}) = \mathbf{1}_{[\mathcal{X}\in \tilde{R}_0,\ \mathbf{x}_s\in B]}\ \tilde{\pi}_{\bar{\mathbf{x}}_s}, \tag{2.52}$$

where $\mathbf{1}_{[A]}$ is the indicator function whose value is 1 when A is true and is 0 otherwise. From (2.32), (2.51), and (2.52),

$$\begin{aligned} \epsilon^\star &= \frac{1}{2}\sum_{\mathcal{X}} \min\left\{\mathbf{1}_{[\mathcal{X}\in R_0,\ \mathbf{x}_s\in A]}\ \pi_{\bar{\mathbf{x}}_s},\ \mathbf{1}_{[\mathcal{X}\in \tilde{R}_0,\ \mathbf{x}_s\in B]}\ \tilde{\pi}_{\bar{\mathbf{x}}_s}\right\} \\ &= \frac{1}{2}\sum_{\mathcal{X}} \mathbf{1}_{[\mathcal{X}\in R_0\cap\tilde{R}_0,\ \mathbf{x}_s\in A\cap B]}\ \min\left\{\pi_{\bar{\mathbf{x}}_s},\ \tilde{\pi}_{\bar{\mathbf{x}}_s}\right\}. \end{aligned} \tag{2.53}$$

Using (2.53) and the fact that there is only one trajectory in $R_0 \cap \tilde{R}_0$ starting from $\mathbf{x}_s$, we have

$$\epsilon^\star = \frac{1}{2}\sum_{\bar{\mathbf{x}}_s\in C_m} \min\left\{\pi_{\bar{\mathbf{x}}_s},\ \tilde{\pi}_{\bar{\mathbf{x}}_s}\right\} = \frac{1}{2}\sum_{i\in C_m} \min\left\{\pi_i,\ \tilde{\pi}_i\right\}, \tag{2.54}$$

which finishes the proof of (2.49). Similarly, from (2.35), (2.51), and (2.52),

$$\begin{aligned} \omega_m(\mathbf{f},\tilde{\mathbf{f}}) &= \frac{1}{2}\sum_{\mathcal{X}} \left|\mathbf{1}_{[\mathcal{X}\in R_0,\ \mathbf{x}_s\in A]}\ \pi_{\bar{\mathbf{x}}_s} - \mathbf{1}_{[\mathcal{X}\in \tilde{R}_0,\ \mathbf{x}_s\in B]}\ \tilde{\pi}_{\bar{\mathbf{x}}_s}\right| \\ &= \frac{1}{2}\left\{\sum_{i\in A\setminus C_m}\pi_i + \sum_{i\in B\setminus C_m}\tilde{\pi}_i + \sum_{i\in C_m}|\pi_i - \tilde{\pi}_i|\right\}, \end{aligned} \tag{2.55}$$

which finishes the proof of (2.48). ■

For finding the set C_m, we first use an efficient algorithm to find the attractor states in the two BNps. Then, after determining a common attractor state $\mathbf{x}_s$, we check if there is a trajectory in $R_0 \cap \tilde{R}_0$ whose starting state is $\mathbf{x}_s$. If there is, then $\mathbf{x}_s \in C_m$; otherwise, $\mathbf{x}_s \notin C_m$. The efficient algorithm to find the set C_m is very fast.

A key understanding is that the Bayes error, for a fixed p, is a function of two factors:

Function mutation: It affects both $\tilde{\pi}_i$ and C_m. A strong mutation can ruin the attractor structures of the original network and, as a result, the number of the common attractors in the two BNps and the size of the set C_m will be decreased, resulting in a lower Bayes error. A weak mutation barely affects the attractor cycles in the original BNp, resulting in $\epsilon^\star \approx 0.5$. The strongest mutation is one in which there is no common attractor state in the two BNps, so $C_m = \oslash$, $\omega_m = 1$, and $\epsilon^\star = 0$.

The trajectory length m: It affects the Bayes error by affecting the size of C_m. For a given original and mutated BNp, the size of C_m is a non-increasing function of m. If m increases, it will be harder to find a common trajectory of length $m - 1$ starting from the common attractor states of the two BNps. Therefore, we expect the Bayes error decreases by the increase of m and tends to zero for sufficiently large m.

2.1.5 Simulation Results

2.1.5.1 Synthetic BNps

• Single BNp:

Consider a Boolean network function $\mathbf{f}$ for a BNp with $n = 4$ genes that has been generated randomly (with probability 0.5 for each gene to be 0 or 1), its truth table being given in Table 2.1 (a). This BNp has three attractor cycles: $13 \rightarrow 13$, $5 \rightarrow 9 \rightarrow 5$, and $10 \rightarrow 14 \rightarrow 15 \rightarrow 10$. We consider one 1-bit function mutation in only one state. For example, we choose $\bar{\mathbf{x}}^\star = 5$ (which is in the attractor cycles of the original network) and $i^\star = 1$ (first gene to be perturbed). The mutated BNp has three attractor cycles: $13 \rightarrow 13$, $10 \rightarrow 14 \rightarrow 15 \rightarrow 10$, and $1 \rightarrow 6 \rightarrow 9 \rightarrow 5 \rightarrow 1$. With this mutation, one of the attractor cycles of the original BNp has changed and the other two have been kept unchanged.

Fig. 2.1 (a) represents the Bayes error $\epsilon^\star$ in (2.32) versus the gene perturbation probability p for different values of m. For a fixed p, the Bayes error decreases as m increases. However, for a fixed m, the Bayes error as a function of p does not have a unique behavior. For instance, for $m = 2$ to $m = 5$, the Bayes error is a monotone increasing function in terms of p, but for $m = 6$, it is not a monotone function, in such a way that it first decreases and then increases as p grows from

Table 2.1: (a) Truth table of the original and mutated Boolean functions $\mathbf{f}$ and $\tilde{\mathbf{f}}$, (b) The Bayes error for all the 1-bit function perturbations. $m = 4$ and $p = 0.01$. Reprinted with permission from [1], ©2018 IEEE.

(a)

$\bar{\mathbf{x}}$	$\mathbf{x}^T$	$\mathbf{f}^T(\mathbf{x})$	$\tilde{\mathbf{f}}^T(\mathbf{x})$
1	0000	0101	0101
2	0001	0100	0100
3	0010	0000	0000
4	0011	0100	0100
5	0100	1000	0000
6	0101	1000	1000
7	0110	0010	0010
8	0111	0100	0100
9	1000	0100	0100
10	1001	1101	1101
11	1010	1110	1110
12	1011	0011	0011
13	1100	1100	1100
14	1101	1110	1110
15	1110	1001	1001
16	1111	1101	1101

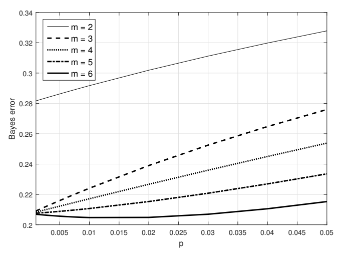
(b)

$\bar{\mathbf{x}}^\star$	$i^\star = 1$	$i^\star = 2$	$i^\star = 3$	$i^\star = 4$
1	0.4036	0.4825	0.4824	0.4839
2	0.4815	0.4969	0.4957	0.4673
3	0.4919	0.4919	0.4462	0.4917
4	0.4831	0.4972	0.4961	0.4976
5	0.2171	0.2624	0.2497	0.2497
6	0.3751	0.3938	0.3702	0.3704
7	0.4474	0.4574	0.4933	0.4900
8	0.4981	0.4997	0.4996	0.4997
9	0.2616	0.2216	0.2114	0.2425
10	0.3221	0.3078	0.3241	0.3203
11	0.4427	0.4412	0.4575	0.4926
12	0.4752	0.4975	0.4975	0.4966
13	0.3822	0.3835	0.3993	0.3986
14	0.3240	0.3247	0.3214	0.3094
15	0.3219	0.2920	0.3225	0.3213
16	0.4691	0.4956	0.4705	0.4737

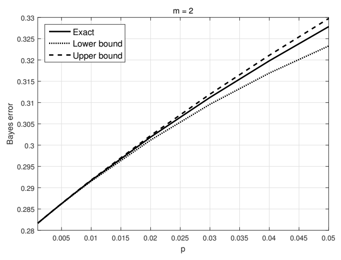
0.001 to 0.05.

The lower and upper bounds of the Bayes error in (2.38) and (2.40) are depicted in Figs. 2.1 (b), (c), and (d) for $m = 2$, $m = 4$, and $m = 6$, respectively. These figures show that when p is small enough, the bounds are tight. For a given m, these bounds become loose with an increase of p, but the value of p after which the bounds are not tight depends on m. For a given p, both the lower and upper bounds become loose with increasing m, the reason being that as m grows, we are disregarding more trajectories by only considering the effective trajectories in R and $\tilde{R}$. Therefore, as m grows, p should shrink to zero for these bounds to be tight and provide a good approximation of the exact Bayes error; if p does not shrink, then the bounds will become loose.

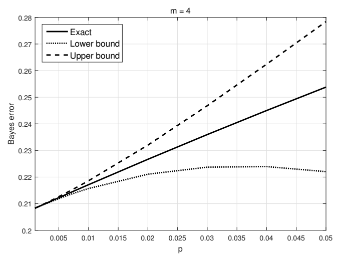
Figs. 2.1 (e) and (f) plot the Bayes error and its lower and upper bounds versus m for two different values of p. Note that the exact Bayes error has been computed up to $m = 6$, because for larger m we cannot compute it due to an exponential complexity with respect to m. Fig. 2.1



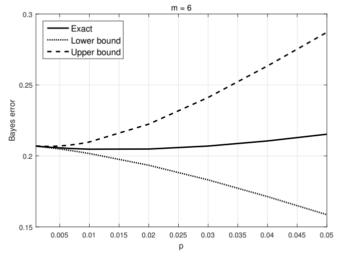
(a)



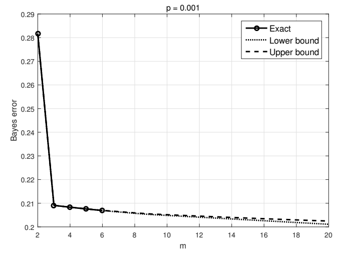
(b)



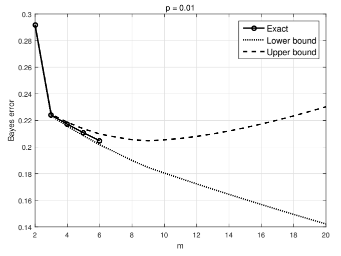
(c)



(d)



(e)



(f)

Figure 2.1: (a): Bayes error vs. p. (b), (c), (d): Bayes error lower and upper bounds vs. p respectively for $m = 2$, $m = 4$, and $m = 6$. (e), (f): Bayes error lower and upper bounds vs. m respectively for $p = 0.001$ and $p = 0.01$. Reprinted with permission from [1], ©2018 IEEE.

(e) shows the results for $p = 10^{-3}$, from which we see that the Bayes error's lower and upper bounds are sandwiched and converge to the exact Bayes error for all $m \leq 20$, because $p = 10^{-3}$ is considered small enough for this range of m. Fig. 2.1 (f) represents the results for $p = 0.01$. Contrary to the previous case, the bounds are getting loose with increasing m, since $p = 0.01$ cannot be considered small enough. These two figures demonstrate that the Bayes error is a non-increasing function of m and its reduction rate depends on the value of p. Larger p tends to give a higher reduction rate of the Bayes error for larger m, since when p is very small and there are common attractor cycles in the original and mutated BNps (as this case), the reduction rate of the Bayes error will be very slow, but when p is a bit larger, the higher perturbation frequency can more readily get the network out of attractor cycles and thereby lead to a greater decrease in the Bayes error.

Finally, we are interested in the Bayes error over all 1-bit function mutations in the different states. Table 2.1 (b) summarizes these results when $m = 4$ and $p = 0.01$, using (2.32). We see that mutations in the attractor states lead to lower Bayes error, which is expected. The lowest Bayes error is 0.2114, which is related to flipping the gene $i^\star = 3$ at the state $\bar{\mathbf{x}}^\star = 9$, which is also an attractor state. In general, mutations that ruin all the attractor cycles of the original BNp can lead to Bayes error near zero. In the examples of Fig. 2.1, $\bar{\mathbf{x}}^\star = 5$ (attractor state) and $i^\star = 1$, which, based on the table, leads to Bayes error 0.2171. As mentioned, this function mutation changes only one attractor cycle of the original BNp and leaves the other two unchanged. The results of Table 2.1 (b) are for 1-bit function mutations in the specific states. In general, arbitrary mutations can change the outputs of many states, thereby resulting in greater reduction of the Bayes error.

• Average Bayes error over many random networks;

Fig. 2.2 shows average Bayes errors for n_{mut} random 1-bit function mutations to the Boolean function $\mathbf{f}$, for $n_{mut} = 1, 2, \cdots, 100$, over 1000 randomly generated BNps (averaged over both random function mutations and random BNps) versus n_{mut} for $n = m = 4$, $n = m = 6$, $n = m = 8$, and $n = m = 10$, assuming $p = 10^{-3}$ for all scenarios. Since p is very small, we have used the limiting results of the Bayes error in (2.49). All averages decrease as the number of 1-bit mutations (n_{mut}) increases. We have used Algorithm 1 for calculating the Bayes error over 1000 random BNps, and 100 random mutations for each each BNp and n_{mut}.

Fig. 2.3 shows the average Bayes error lower bound for different trajectory lengths, $m = 2, 3, \cdots, 10$, given a fixed number of function mutations $n_{mut} = 10$ and $p = 0.01$, and using Algorithm 1 for calculating the Bayes error over 1000 random BNps. The averages decrease with increasing m: longer trajectories lead to smaller Bayes errors. Note that for larger networks (larger n), for a given n_{mut}, m and p, there is a higher average Bayes error lower bound. For instance, when $n_{mut} = m = 10$ and $p = 0.01$, Fig. 2.3 shows that the average Bayes error is approximately 0.068, 0.145, and 0.26 for $n = 4$, $n = 6$, and $n = 8$, respectively.

2.1.5.2 Real Gene regulatory Networks

We consider two GRNs for which we already know the wild-type and mutated networks. For a BNp, we analyze a p53 network; for a PBN, we consider a mammalian cell-cycle PBN.

• p53 BNp:

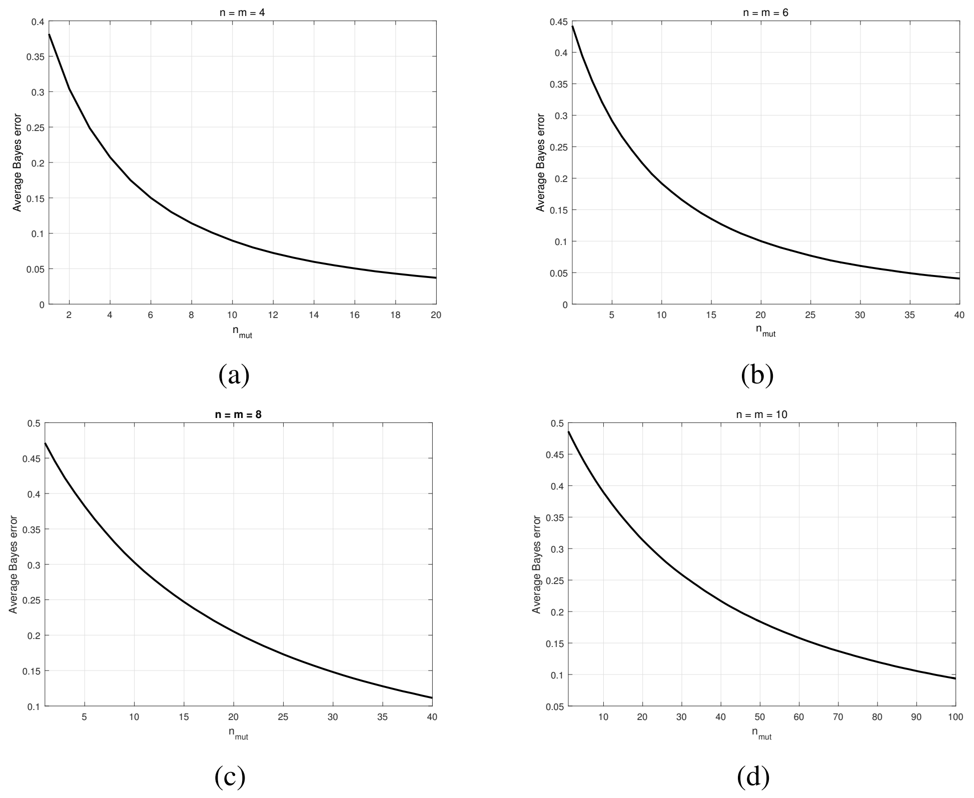


Figure 2.2: $p = 10^{-3}$. Average Bayes error (averaged over 1000 random BNps) versus n_{mut} for (a): $n = m = 4$, (b): $n = m = 6$, (c): $n = m = 8$, and (d): $n = m = 10$. Reprinted with permission from [1], ©2018 IEEE.

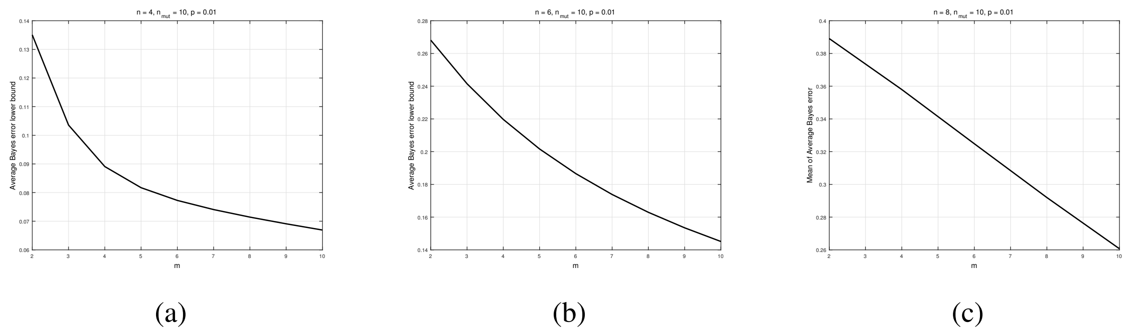


Figure 2.3: $p = 0.01$ and $n_{mut} = 10$. Average Bayes error lower bound (averaged over 1000 random BNps) versus m for (a): $n = 4$, (b): $n = 6$, and (c): $n = 8$. Reprinted with permission from [1], ©2018 IEEE.

We use the wild-type BNp of the p53 network whose GRN is depicted in Fig. 2.4, adapted from [7]. This GRN has four genes and its regulating functions are defined in Table 2.2. According to [39], cancerous BNps result when either gene $p53$ is deactivated ($f_2 = 0$ in Table 2.2) or gene

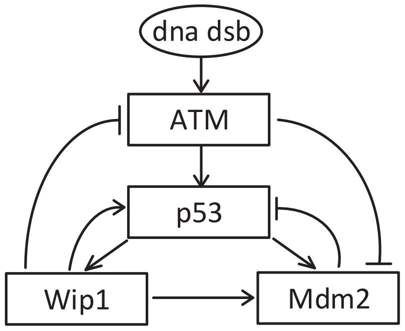


Figure 2.4: p53 gene regulatory network. Reprinted with permission from [1], ©2018 IEEE.

$Mdm2$ is activated ($f_4 = 1$ in Table 2.2) permanently. Hence, we have computed the Bayes error when one of the classes is the wild-type and another is one of the aforementioned mutated networks. DNA_{dsb} in Fig. 2.4 is a Boolean signal that indicates the presence of a double strand break. We assume a DNA damage situation in which $DNA_{dsb} = 1$. Figure 2.5 (a) represents the wild-type BNp, while Figs. 2.5 (b) and 2.5 (c) are related to the mutated BNps with $f_2 = 0$ and $f_4 = 1$, respectively. The numbers inside the nodes show the indices of the binary states. The wild-type BNp has an attractor cycle consisting of seven states, $9, 13, 15, 7, 8, 4, 2$, while the mutated BNps in Figs. 2.5 (b) and (c) have a single state attractor 9 and 10, respectively. The BNps in (a) and (c) have no common attractor states. Although the BNp in (a) has one common attractor state (9) with the BNp in (c), there is no common trajectory of any length m in them. Thus, we expect that as $p \to 0$, the Bayes error tends to zero. The simulation results in Fig. 2.6 confirm this fact.

Figure 2.6 (a) shows the Bayes error $\epsilon^\star$ versus p for several values of m when the mutated network has the p53 gene deactivated, calculated from (2.48). Note that $\epsilon^\star$ is an increasing function in terms of p. This is an anticipated phenomena, for when p increases, it becomes harder to distinguish between the original and mutated networks because the genes are randomly flipped more often and do not let us see the true structure of the networks easily. In other words, when we have larger p, the two TPMs of the original and mutated networks become more similar to

Table 2.2: Definitions of Boolean functions in wild-type p53 BNp. Reprinted with permission from [1], ©2018 IEEE.

Order	Gene	Regulating function
x_1	ATM	$f_1 = \overline{Wip1} \wedge (ATM \vee dna_{dsb})$
x_2	$p53$	$f_2 = \overline{Mdm2} \wedge (ATM \vee Wip1)$
x_3	$Wip1$	$f_3 = p53$
x_4	$Mdm2$	$f_4 = \overline{ATM} \wedge (p53 \vee Wip1)$

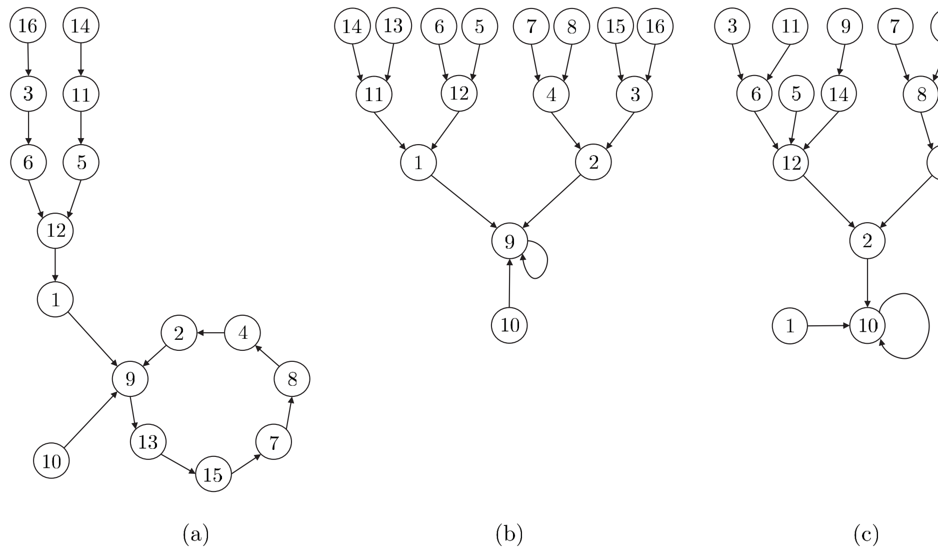
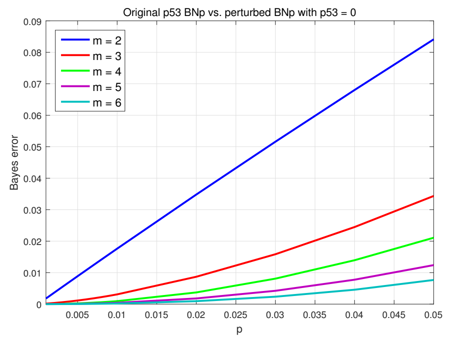
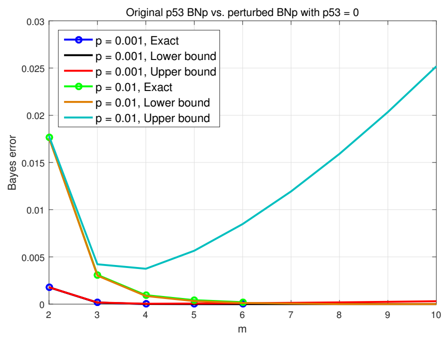


Figure 2.5: (a) Wild-type (original) p53 BNp. (b) Mutated (perturbed) BNp with p53 = 0. (c) Mutated (perturbed) BNp with MDM2 = 1. Reprinted with permission from [1], ©2018 IEEE.

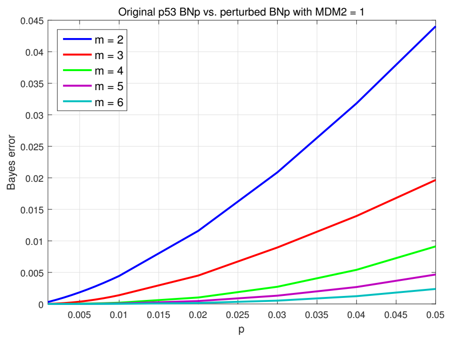
each other, and as a result, identifying the real BNps will be hard, leading to a larger Bayes error. Regardless of m, as $p \to 0$, $\epsilon^\star \to 0$, which is expected since there is no common trajectory of length $m \geq 2$ in the two BNps of Fig. 2.5 (a) and (b). In Fig. 2.6 (a), for $p \leq 0.01$ and $m \geq 3$, we have $\epsilon^\star \approx 0$. Fig. 2.6 (b) depicts the Bayes error and its lower and upper bounds versus m for $p = 0.001$ and $p = 0.01$. The bounds have been calculated using (2.38) and (2.40). The exact Bayes error has only been computed up to $m = 6$. It can be seen from 2.6 (b) that the Bayes error is decreasing with increasing m. Fig. 2.6 (b) shows that the bounds are tight when p is



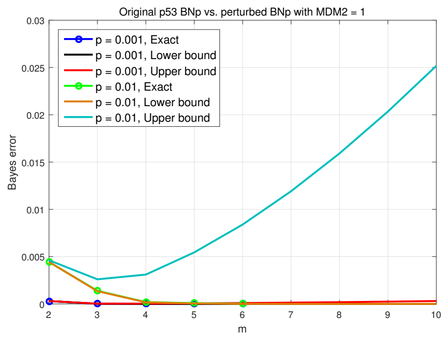
(a)



(b)



(c)



(d)

Figure 2.6: (a) and (b): Bayes error versus p and m. First class is the wild-type p53 BNp, and the second is mutated BNp with deactivated gene p53. (c) and (d): Bayes error versus p and m. First class is the wild-type p53 BNp, and the second is mutated BNp with activated gene MDM2. Reprinted with permission from [1], ©2018 IEEE.

sufficiently small; indeed, for $p = 0.001$ they are roughly equal. Moreover, in 2.6 (b) the upper bound becomes looser with increase of m, the reason being that $p = 0.01$ cannot be considered small enough for larger m. Figs. 2.6 (c) and (d) show analogous results when the mutated network is the p53 network with activated Mdm2, the difference being that the Bayes error is even less, for any p and m, than those in Figs. 2.6 (a) and (b), since in this case, the wild-type and mutated BNps do not share a common attractor state. We conclude from the results in Fig. 2.6 that the wild-type and either mutated BNp are well classifiable, the Bayes errors being close to zero.

• Instantaneously random Mammalian Cell-Cycle PBN:

We use the wild-type mammalian cell-cycle PBN, whose GRN, adapted from [7], is depicted in Fig. 2.7. This GRN consists of ten genes and the regulating functions are defined in Table 2.3. The value of gene CycD is determined by some extracellular signals and is assumed to be 0 or 1

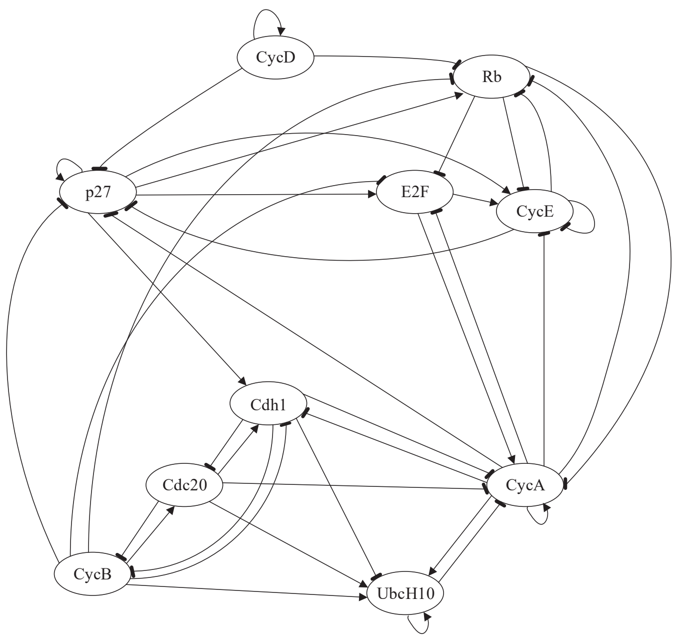


Figure 2.7: Mammalian cell-cycle gene regulatory network. Reprinted with permission from [1], ©2018 IEEE.

with the probability of 0.5 for each case. This fact enables us to define the two constituent BNps of this PBN. The first BNp has CycD always off (i. e., $f_1 = 0$ in Table 2.3) and the second BNp has CycD always on (i. e., $f_1 = 1$ in Table 2.3). Each BNp has the selection probability of 0.5 , that is, $c_1 = c_2 = 0.5$. As for the mutated PBN, according to [7], p27 is a key gene in the cell-cycle network whose absence can lead to a cancerous network. Therefore, for the mutated PBN, we permanently set the value of p27 to zero in both the constituent BNps, that is, $f_3 = 0$ in Table 2.3 for the mutated PBN.

Regarding classification of these PBNs, the Bayes error lower and upper bounds in (2.44) and (2.45) are depicted in Fig. 2.8 for $p = 10^{-3}$ and $p = 10^{-4}$. Computing the exact Bayes error using (2.41) is impossible due to an intractable computational cost. Even when $m = 5$, there are $2^{50} \approx 10^{15}$ trajectories. Furthermore, in using the bounds in (2.44) and (2.45), we are assuming sufficiently small p. The bounds in Fig. 2.8 can be used to capture the behavior of the exact Bayes error for $p = 10^{-3}$ and $p = 10^{-4}$, the latter being tighter. Since the Bayes error is a decreasing function of m, for sufficiently large m the wild-type and cancerous PBNs are classifiable with a desired low Bayes error.

Table 2.3: Definitions of Boolean functions for the wild-type mammalian cell-cycle PBN with 10 genes. Reprinted with permission from [1], ©2018 IEEE.

Order	Gene	Regulating function
x_1	$CycD$	$f_1 =$ Extracellular signals
x_2	Rb	$f_2 = (\overline{CycD} \wedge \overline{CycE} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{CycD} \wedge \overline{CycB})$
x_3	$p27$	$f_3 = (\overline{CycD} \wedge \overline{CycE} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{(CycE \wedge CycA)} \wedge \overline{CycD} \wedge \overline{CycB})$
x_4	$E2F$	$f_4 = (\overline{Rb} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{Rb} \wedge \overline{CycB})$
x_5	$CycE$	$f_5 = (E2F \wedge \overline{Rb})$
x_6	$CycA$	$f_6 = (E2F \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{(Cdh1 \wedge UbcH10)}) \vee (CycA \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{(Cdh1 \wedge UbcH10)})$
x_7	$Cdc20$	$f_7 = CycB$
x_8	$Cdh1$	$f_8 = (\overline{CycA} \wedge \overline{CycB}) \vee Cdc20 \vee (p27 \wedge \overline{CycB})$
x_9	$UbcH10$	$f_9 = \overline{Cdh1} \vee (Cdh1 \wedge UbcH10 \wedge (Cdc20 \vee CycA \vee CycB))$
x_{10}	$CycB$	$f_{10} = (\overline{Cdc20} \wedge \overline{Cdh1})$

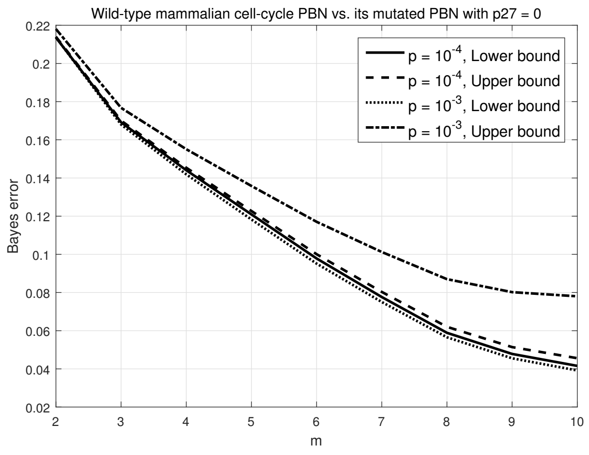


Figure 2.8: Lower and upper bounds of the Bayes error versus m. The two classes are the wild-type and mutated (with p27 = 0) mammalian cell-cycle PBNs. Reprinted with permission from [1], ©2018 IEEE.

2.1.6 Conclusion

This section characterized the classification of trajectories observed in m successive states in the steady-state to an original (wild-type) or mutated GRN using the PBN and BNp frameworks, in particular deriving the Bayes classifier and the Bayes error. To circumvent computational complexity, we proposed an effective and reduced trajectory space when the gene perturbation probability

p is small and found lower and upper bounds for the Bayes error, which are acceptably tight when p is sufficiently small. The procedure was applied to classify trajectories for both synthetic and real BNps and PBNs, including computing the Bayes errors.

2.2 Classification of Single-Cell Gene Expression Trajectories

2.2.1 Overview

In this section, we study the classification of gene-expression trajectories coming from two classes, healthy and mutated (cancerous) using Boolean networks with perturbation (BNps) to model the dynamics of each class at the state level. Each class has its own BNp, which is partially known based on gene pathways. We employ a Gaussian model at the observation level to show the expression values of the genes given the hidden binary states at each time point. We use expectation maximization (EM) to learn the BNps and the unknown model parameters, derive closed-form updates for the parameters, and propose a learning algorithm. After learning, a plug-in Bayes classifier is used to classify unlabeled trajectories, which can have missing data. Measuring gene expressions at different times yields trajectories only when measurements come from a single cell. In multiple-cell scenarios, the expression values are averages over many cells with possibly different states. Via the central-limit theorem, we propose another model for expression data in multiple-cell scenarios. Simulations demonstrate that single-cell trajectory data can outperform multiple-cell average expression data relative to classification error, especially in high-noise situations. We also consider data generated via a mammalian cell-cycle network, both the wild-type and with a common mutation affecting the gene p27.

2.2.2 Introduction

In the previous section we characterized the Bayes classifier and Bayes error for classification of steady-state trajectories observed in successive states in an original (wild-type) or mutated gene regulatory network (GRN) modeled via probabilistic Boolean networks (PBNs) [1]. In the present section we consider classification when the networks are only partially known and the training data consist of labeled trajectories from an original and mutated network modeled as Boolean networks with perturbation (BNp), which is a special case of a PBN, observed indirectly through noise [2,40]. The overall model is called a partially-observed Boolean dynamical system (POBDS) [41].

Owing to heterogeneity across samples and patients, it has long been recognized that it can be beneficial to use groups of genes as features. This can help avoid redundant information contained in selected genes, for instance, several genes in a pathway regulated by a single master gene [21]. The approach is to jointly analyze the expression levels of genes related by functionality, which can be obtained via transcriptome analysis [22-24], GO annotations [25], or other sources. Several methods have been proposed to measure the activity of a particular pathway: mean or median [26], first principle component [24], using a subset of genes in the pathway [27], and combining log-likelihood ratios of genes in the pathway [28]. Although these methods utilize multiple-gene features, they still rely on single measurements and do not take advantage of regulatory information in trajectory data.

Single-cell gene expression has recently become popular, as it is able to reveal the expressions of genes in many different cells in parallel in a single experiment, instead of bulk gene expression methods like conventional RNA-Seq in which the reported expression level of a gene is actually an average over cells with different states and possibly different types [42]. As a result, it has been utilized and proven to be a very effective alternative of bulk expression methods in various research studies. For instance, [43] demonstrated single-cell RNA-Seq (scRNA-Seq) as an effective strategy for classification of sensory neuron types. [44] used scRNA-Seq data to classify low quality cells. A massively parallel single-cell RNA profiling was used in [45] to classify retinal bipolar cells, where 15 previously known and two novel types were identified. Authors in [46] proposed a nonnegative matrix factorization (NMF) method as a robust unsupervised learning of cell subtypes from single-cell gene expression data. [47] proposed a method for unsupervised clustering of single-cell epigenetic data using single-cell ATAC-seq data.

Single-cell expression measurements have enabled generating and using time-series data and discovering the regulatory information of genes, since bulk expression measurements, like RNA-Seq or microarrays, destroy crucial information by averaging signals from individual cells together [42]. However, lower amounts of mRNA in individual cells cause experimental issues which lead to dropout events [48], such that expressions of some genes are missed in some cells. Accordingly, in this section, we also consider missing values of genes in order to better reflect the real data. [49]

proposed a differential expression method using single-cell RNA-Seq time-series data for recovery of potential cell types from complex mixtures of multiple cell types. BNP-Seq, proposed in [50], is a Bayesian nonparametric differential expression analysis of count data, which might be beneficial if applied to single-cell RNA-Seq data to discover differentially expressed genes. Furthermore, [51] presented a probabilistic model with a Bayesian inference scheme to analyze single-cell time-series data, which was used for pseudotime estimation. Single-cell gene expression time-series measurements have also been employed to infer gene regulatory networks; for instance, single-cell expression measurements at four time points of blood development were used in [52] to synthesize a Boolean network model for 20 related transcription factors.

In this section, gene regulation is modeled via BNps, in which states are binary vectors, and 1 and 0 represent On and Off, respectively (Binary representation is chosen because it models switch-like gene behavior and because it makes computation tractable, but the theory is directly extendable to any number of expression levels.) We consider a Gaussian observation model, in which the expression level of each gene given its state (hidden) follows a normal density with some unknown mean and variance. We observe the Gaussian expression values of n genes in m consecutive time points; however, to take account of missing data, at each time point there is a probability, p_{miss}, of not observing the expression of a gene. After observation of such trajectories, we estimate the unknown network parameters as well as the unknown network connections, which are partially known. For maximum likelihood estimation and inference, we use the Expectation Maximization (EM) approach to estimate the continuous parameters of the networks. We then plug in the estimated parameters and the inferred networks to the Bayes classifier. We study the effects of the different parameters on the average classification error over many random networks using trajectory data of different length and missing probability.

When gene-expression values are measured from tissues containing many cells, with genes not synchronized, a gene may be in different states at any time across the cell sample. Expression data derived from a multiple-cell scenario is approximated by average expression values across all states. To treat multiple-cell averaging, we consider averaged expression data and use a static model that does not take into account the dynamics of the networks. We compare the classification errors using trajectory data (single-cell) and averaged data (multiple-cell) in the simulation part and show that trajectories outperform averaged data if the trajectory length is sufficient, even with missing data.

2.2.3 Preliminaries

For a Boolean network (BN) on n genes, a truth table gives the functional relationships between the genes [38]. Each gene value $x_i \in \{0,1\}$, for $i = 1, \cdots, n$, at time $k+1$ is determined by the values of some predictor genes at time k via a Boolean function $f_i : \{0,1\}^n \to \{0,1\}$ in the truth table. In practice, f_i is a function of a small number of genes, K_i, called the *in-degree* of the gene x_i in the network. The in-degree of the network is the maximum of K_i's, that is, $K = \max_{i=1,\cdots,n} K_i$. A gene network can be represented as a graph with vertices representing genes and edges representing regulations. There is a state diagram of 2^n states corresponding to the truth table of the BN, representing the dynamics of the network. Given an initial state, a BN will eventually reach a set of states, called an *attractor cycle*, through which it will cycle endlessly. Each initial state corresponds to a unique attractor cycle, and the set of initial states leading to a specific attractor cycle is known as the *basin of attraction* (BOA) of the attractor cycle.

2.2.3.1 State Model

We allow stochasticity in our state model by using BNps instead of deterministic BNs. For BNps, perturbation is introduced with a probability p by which the state of the network can be randomly changed at any time. Implicitly, we assume that there is an independent identically distributed (i.i.d.) random perturbation vector at each time k, denoted by $\mathbf{n}_k \in \{0,1\}^n$, such that the i-th gene flips at time k if the i-th component of $\mathbf{n}_k$ is equal to 1. Therefore, the dynamical model of the states can be expressed as

$$\mathbf{X}_{k+1} = \mathbf{f}(\mathbf{X}_k) \oplus \mathbf{n}_{k+1}, \qquad k = 0, 1, 2, \cdots, \tag{2.56}$$

where $\mathbf{X}_k = [x_1(k), x_2(k), \cdots, x_n(k)]^T$ is a binary state vector, called a *gene activity profile* (GAP), at time k, in which $x_i(k)$ indicates the expression level of the i-th gene at time k (either 0 or 1); $\mathbf{f} = [f_1, f_2, \cdots, f_n]^T : \{0,1\}^n \to \{0,1\}^n$ is the vector of the network functions, in which f_i shows the expression level of the i-th gene at time $k+1$ when the system lies in the state $\mathbf{X}_k$ at time k; $\mathbf{n}_k = [n_1(k), n_2(k), \cdots, n_n(k)]^T$ is the perturbation vector at time k, in which $n_1(k), n_2(k), \cdots, n_n(k)$ are i.i.d. Bernoulli random variables for every k with the parameter $p = P(n_i(k) = 1)$ for every $i = 1, \cdots, n$; and $\oplus$ is component-wise modulo 2 addition.

The existence of perturbation makes the corresponding Markov chain of a BNp irreducible. Hence, the network possesses a steady-state distribution π describing its long-run behavior. A BNp inherits the attractor structure from the original BN without perturbation, the difference being that a random perturbation can cause a BNp to jump out of an attractor cycle, perhaps then transitioning to a different attractor cycle. If p is sufficiently small, π will reflect the attractor structure within the original network. We can derive the transition probability matrix (TPM) if we know the truth table and the perturbation probability of a BNp. As a result, the steady-state distribution π can be computed as well.

We assume that the networks are partially known, perhaps from biological pathway knowledge or previous partial inference, and the missing model parameters are estimated from the new trajectory data. One could, in principle, assume that nothing is known about the BNs except the genes and depend entirely on the data, but we are generally interested in using prior knowledge to facilitate classifier design.

Since we do a supervised classification between two classes, healthy and a specific mutated phenotype, the mutated genes of that mutated phenotype determine the desired BN. For example, we have considered a specific phenotype in which the gene p27 is mutated. As a result, we are required to use a pathway and a BN which involves the gene p27; one such BN is the known cell-cycle BN which we have utilized as our healthy class and its mutated version of knocked out p27 as the mutated class. In general, if we are interested in a supervised classification between a healthy class and a mutated class in which some genes are mutated, we need to use a BN which involves those mutated genes.

2.2.3.2 Observation Model

Our model for gene expression is the partially-observed Boolean dynamical system (POBDS) [41], which is a special case of a hidden Markov model (HMM). Having defined the state transition model as a BNp, we now define the observation model given the hidden states by assuming that the expression level of each gene at any time comes from a Gaussian distribution whose mean value is specified by that gene's binary state value, which is hidden. In other words, depending on whether a gene is active or not, its expression value comes from two Gaussian distributions with two different means. The observation model for the j-th gene at time k is

$$p\left(y_j(k)|x_j(k)\right) \sim \mathcal{N}\left(\lambda + \delta_j x_j(k), \sigma^2\right), \quad j = 1, 2, \cdots, n, \tag{2.57}$$

where $x_j(k)$ is the hidden binary state (0 or 1) of the j-th gene at time k, and $y_j(k)$ is the observed expression value of the j-th gene at time k. The variance σ^2 is constant, but the mean varies over time, as the value of $x_j(k)$ is changing according to the state dynamics (2.56). This shows that when the j-th gene is off (suppressed) and on (expressed), its observed expression values come from Gaussian distributions with the means of λ and $\lambda + \delta_j$, respectively, and with the same variance σ^2. In (2.57), λ is the baseline expression level of the genes, which depends on the sequencing technology, and δ_j is the activation coefficient of the j-th gene, which determines the level of the expression for the j-th gene when it is on. Although we can proceed with arbitrary values of δ_j for different genes, for the sake of simplicity we assume the same activation coefficient for all the genes, that is, $\delta_j = \delta$ for $j = 1, \cdots, n$.

We denote the expression values of all n genes at time k by the vector $\mathbf{Y}_k = [y_1(k), \cdots, y_n(k)]^T$. If we assume that, at any time point k, the expression value of each gene given its binary state is independent of the expressions of other genes given their corresponding binary states, we can write,

$$\mathbf{Y}_k = \lambda \mathbf{1}_n + \delta \mathbf{X}_k + \epsilon, \quad k = 1, 2, \cdots, \tag{2.58}$$

where $\mathbf{1}_n$ is an $n \times 1$ all-one vector and $\epsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ is a $n \times 1$ multivariate Gaussian random variable of zero mean and diagonal covariance matrix (I_n is $n \times n$ identity matrix) showing the variability across the samples. The state vector $\mathbf{X}_k$ in (2.58) is hidden (not observed), and the conditional distribution of $\mathbf{Y}_k$ given $\mathbf{X}_k$ is

$$p(\mathbf{Y}_k|\mathbf{X}_k) \sim \mathcal{N}(\lambda \mathbf{1}_n + \delta \mathbf{X}_k, \sigma^2 I_n), \quad k = 1, 2, \cdots . \tag{2.59}$$

Note that there are two types of variability: intra-subject and inter-subject. Intra-subject variability is sometimes called within-subject variability and refers to the variability of the samples in one subject, for example, the variability seen in the expression values of the genes in one individual at different times. Subject in this context means either cell, organism, or individual. Inter-subject variability is sometimes called between-subject variability and represents the variability among the samples of different subjects. For example, the variability seen in the expression values of the genes in different individuals refers to inter-subject variability. For avoiding confusion, we use the term "inter-cell variability" to refer to the variability across different cells of an individual, which will be used in Section 4 for the analysis of the multiple-cell scenario. In this section, we do single-cell analysis and do not deal with inter-cell variability, since each individual has a single cell to be used for expression measurements. We should note that in (2.58), ϵ accounts for both the inter- and intra-subject variability.

2.2.4 Classification of Trajectories with Missing Data in Single-cell Scenarios

Assume there are two BNps corresponding to the healthy and mutated (cancerous) classes, each having n genes, and we partially know the networks but do not know the model parameters p, λ, δ, and σ^2. The healthy and mutated networks may have distinct model parameter values. Using D observed trajectories, $\mathbb{Y} = \{\mathcal{Y}^{(1)}, \mathcal{Y}^{(2)}, \cdots, \mathcal{Y}^{(D)}\}$, we infer the unknown parameters and connections of each network. $\mathbb{Y}$ may be incomplete, meaning that there may be missing data. Without missing data, each trajectory $\mathcal{Y}^{(d)}$, for $d = 1, \cdots, D$, has the expression values of the n genes in m consecutive time points. However, if each gene at each time point has the probability

p_{miss} of being missed, then each observed trajectory has the form $\mathcal{Y}^{(d)} = \left[\mathbf{Y}_{i_1}^{(d)}, \cdots, \mathbf{Y}_{i_{m(d)}}^{(d)}\right]$, where $T_{obs}^{(d)} = \{i_1, \cdots, i_{m(d)}\}$ is the set of time points at which at least one gene is observed.

For the maximum likelihood (ML) problem, the search space consists of both discrete and continuous parts. The space of network functions is discrete and that of the parameters is continuous. Suppose $\mathbf{F} = \{\mathbf{f}^1, \mathbf{f}^2, \cdots, \mathbf{f}^M\}$ is the uncertainty set of M network functions containing the unknown true network function in (2.56). We wish to infer the true network function using the observation data. In each class, healthy or mutated, we assume an uncertainty set of network functions $\mathbf{F}$. Although there are many biologically confirmed gene pathways from which Boolean networks can be constructed, we are likely to be uncertain about some regulations and interactions between some genes. In such cases, we can form an uncertainty class $\mathbf{F}$ of network functions, each being a possible network function which would be inferred from the observed data. For instance, assume we know that the gene A regulates the gene B but are not sure about the type of regulation, that is, activator or suppressor. As such, in our uncertainty class of network functions we let $\mathbf{f}^1$ and $\mathbf{f}^2$ be the network functions for the cases that regulator A to B is activator and suppressor, respectively. In a similar way, we can consider any kind of uncertainty in the network structure, and the true network is inferred from the observed trajectories.

Suppose the model parameters are defined as the vector $\theta = [p, \lambda, \delta, \sigma^2]^T$. For any given network function $\mathbf{f}^i$, $i = 1, \cdots, M$, we employ the EM algorithm to find the optimal parameters θ by

$$\hat{\theta}_i = \underset{\theta}{\operatorname{argmax}}\, p(\mathbb{Y}|\mathbf{f}^i, \theta), \tag{2.60}$$

where $p(\mathbb{Y}|\mathbf{f}^i, \theta)$ is the likelihood of the observation trajectory set $\mathbb{Y}$ given that the network function is $\mathbf{f}^i$ and the parameter is θ. The ML inferred network function and estimated parameters are then derived as

$$(\hat{\mathbf{f}}, \hat{\theta}) = \arg \max_{(\mathbf{f},\theta)\in\{(\mathbf{f}^1,\hat{\theta}_1),\cdots,(\mathbf{f}^M,\hat{\theta}_M)\}} p(\mathbb{Y}|\mathbf{f}, \theta). \tag{2.61}$$

2.2.4.1 EM algorithm for finding θ

In (2.60), the network function is given, and we are supposed to find the ML estimation for θ. To ease notation, suppose the network function in (2.60) is denoted by $\mathbf{f}$. As there are hidden states in the model, we employ the EM algorithm to estimate the parameters. The EM algorithm can be described simply as repeating the following steps until convergence:

1- E-step: $Q(\theta, \theta^{(s)}) = \sum_{\mathbb{X}} \log[p(\mathbb{X}, \mathbb{Y}|\theta)] P(\mathbb{X}|\mathbb{Y}, \theta^{(s)})$,

2- M-step: $\theta^{(s+1)} = \mathrm{argmax}_\theta\, Q(\theta, \theta^{(s)})$,

where $\mathbb{X} = \{\mathcal{X}^{(1)}, \cdots, \mathcal{X}^{(D)}\}$ contains the hidden state trajectories corresponding to D observed trajectories, such that $\mathcal{X}^{(d)} = [\mathbf{X}_1^{(d)}, \cdots, \mathbf{X}_m^{(d)}]$ are the hidden states of the d-th trajectory from time 1 to m.

• E-step:

Since the D trajectory observations are i.i.d., we can write the joint log-likelihood of $\mathbb{X}$ and $\mathbb{Y}$ as

$$\log[p(\mathbb{X}, \mathbb{Y}|\theta)] = \sum_{d=1}^{D} \log[p(\mathcal{X}^{(d)}, \mathcal{Y}^{(d)}|\theta)]. \tag{2.62}$$

The joint likelihood of each d-th observed trajectory $\mathcal{Y}^{(d)}$ and its corresponding hidden trajectory $\mathcal{X}^{(d)}$ can be factored as

$$p(\mathcal{X}^{(d)}, \mathcal{Y}^{(d)}|\theta) = P(\mathbf{X}_1^{(d)}) \prod_{k=1}^{m-1} P(\mathbf{X}_{k+1}^{(d)}|\mathbf{X}_k^{(d)}) \prod_{k \in T_{obs}^{(d)}} p(\mathbf{Y}_k^{(d)}|\mathbf{X}_k^{(d)}), \tag{2.63}$$

where, under the usual biological assumption of steady-state observations, $P(\mathbf{X}_1^{(d)})$ is the steady-state probability of state $\mathbf{X}_1^{(d)}$. Let $\mathbf{x}^i$ denote the $n \times 1$ binary vector of the state i, for $i = 1, \cdots, 2^n$. For example, if $n = 4$, then $\mathbf{x}^1 = [0,0,0,0]^T$ and $\mathbf{x}^{10} = [1,0,1,1]^T$. Denote the steady-state distribution by the 1×2^n vector $\pi = [\pi_1, \cdots, \pi_{2^n}]$, where π_i is the steady-state probability of being in the i-th state. Then $P(\mathbf{X}_1^{(d)} = \mathbf{x}^i) = \pi_i$ for any $d = 1, \cdots, D$ and $i = 1, \cdots, 2^n$. Should we drop the steady-state assumption, then $P(\mathbf{X}_1^{(d)})$ is an arbitrary distribution to be estimated from the observed data. The second term in (2.63) is the probability of transitioning from state $\mathbf{X}_k^{(d)}$ at time k to state $\mathbf{X}_{k+1}^{(d)}$ at time $k+1$, which using (2.56) can be written as

$$P(\mathbf{X}_{k+1}^{(d)}|\mathbf{X}_k^{(d)}) = p^{\mathbf{d}(\mathbf{X}_{k+1}^{(d)},\mathbf{f}(\mathbf{X}_k^{(d)}))}(1-p)^{n-\mathbf{d}(\mathbf{X}_{k+1}^{(d)},\mathbf{f}(\mathbf{X}_k^{(d)}))}, \tag{2.64}$$

where $\mathbf{d}(\mathbf{X}_{k+1}^{(d)}, \mathbf{f}(\mathbf{X}_k^{(d)}))$ denotes the Hamming distance between the two binary vectors $\mathbf{X}_{k+1}^{(d)}$ and $\mathbf{f}(\mathbf{X}_k^{(d)})$. The probabilities in (2.64) are the entries in the transition probability matrix. The third term in (2.63), $p(\mathbf{Y}_k^{(d)}|\mathbf{X}_k^{(d)})$, is the likelihood of the gene-expression vector $\mathbf{Y}_k^{(d)}$ at time k given its corresponding hidden state vector $\mathbf{X}_k^{(d)}$. In the absence of missing data, $\mathbf{Y}_k^{(d)}$ contains the expression values of all n genes, but with missing data some expression values may not appear in $\mathbf{Y}_k^{(d)}$. Let $G_k^{(d)}$ denote the set of genes whose expressions have been observed at time k of the d-th trajectory. Then from (2.59), we have,

$$p(\mathbf{Y}_k^{(d)}|\mathbf{X}_k^{(d)}) = \prod_{j \in G_k^{(d)}} (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left[\frac{-\left(y_j^{(d)}(k) - \lambda - \delta x_j^{(d)}(k)\right)^2}{2\sigma^2}\right]. \tag{2.65}$$

Using (2.62)-(2.65), the joint log-likelihood can be written as in (2.66).

$$\begin{aligned}
&\log[p(\mathbb{X}, \mathbb{Y}|\theta)] = \\
&\sum_{d=1}^{D}\left\{\log P(\mathbf{X}_1^{(d)}) + \sum_{k=1}^{m-1}\left[\mathbf{d}(\mathbf{X}_{k+1}^{(d)}, \mathbf{f}(\mathbf{X}_k^{(d)}))\log p + [n - \mathbf{d}(\mathbf{X}_{k+1}^{(d)}, \mathbf{f}(\mathbf{X}_k^{(d)}))]\log(1-p)\right]\right. \\
&\left. + \sum_{k \in T_{obs}^{(d)}} \sum_{j \in G_k^{(d)}}\left[-\frac{1}{2}\log 2\pi\sigma^2 - \frac{\left(y_j^{(d)}(k) - \lambda - \delta x_j^{(d)}(k)\right)^2}{2\sigma^2}\right]\right\}.
\end{aligned} \tag{2.66}$$

Now we can compute $Q(\theta, \theta^s)$. After some straightforward simplifications and dropping the constant parts, $Q(\theta, \theta^s)$ can be derived as in (2.67),

$$\begin{aligned} Q(\theta, \theta^{(s)}) &= \sum_{d=1}^{D} \sum_{i=1}^{2^n} \log(\pi_i) \mathbf{\Pi}_i^{(d,s)}(1) \\ &+ \sum_{d=1}^{D} \sum_{k=1}^{m-1} \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \left[\mathbf{d}(\mathbf{x}^j, \mathbf{f}(\mathbf{x}^i)) \log p + [n - \mathbf{d}(\mathbf{x}^j, \mathbf{f}(\mathbf{x}^i))] \log(1-p) \right] \mathbf{\Xi}_{i,j}^{(d,s)}(k) \\ &+ \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{i=1}^{2^n} \sum_{j \in G_k^{(d)}} \left[-\frac{1}{2} \log \sigma^2 - \frac{\left(y_j^{(d)}(k) - \lambda - \delta \mathbf{x}_j^i \right)^2}{2\sigma^2} \right] \mathbf{\Pi}_i^{(d,s)}(k), \end{aligned} \tag{2.67}$$

where,

$$\mathbf{\Pi}_i^{(d,s)}(k) = P(\mathbf{X}_k^{(d)} = \mathbf{x}^i | \mathcal{Y}^{(d)}, \theta^{(s)}), \tag{2.68}$$

for any $i = 1, \cdots, 2^n$, $k = 1, \cdots, m$, and $d = 1, \cdots, D$, is the posterior probability of the state i at time k after the observation of the d-th trajectory, and given the parameter vector $\theta^{(s)}$. Furthermore, in (2.67),

$$\mathbf{\Xi}_{i,j}^{(d,s)}(k) = P(\mathbf{X}_k^{(d)} = \mathbf{x}^i, \mathbf{X}_{k+1}^{(d)} = \mathbf{x}^j | \mathcal{Y}^{(d)}, \theta^{(s)}), \tag{2.69}$$

for any $i, j = 1, \cdots, 2^n$, $k = 1, \cdots, m-1$, and $d = 1, \cdots, D$, is the posterior probability of two consecutive states being i and j, respectively, at times k and $k+1$ after the observation of the d-th trajectory and given the parameter vector $\theta^{(s)}$.

• M-step:

Having derived $Q(\theta, \theta^{(s)})$, we address the second step of the EM method, which is the maximization of $Q(\theta, \theta^{(s)})$. We take the derivative of $Q(\theta, \theta^{(s)})$ with respect to θ. The derivatives of $Q(\theta, \theta^{(s)})$ with respect to p, λ, δ, and σ^2 are

$$\frac{\partial Q}{\partial p} = \sum_{d=1}^{D} \sum_{\mathbf{i}=1}^{2^n} \frac{\pi_i'}{\pi_i} \mathbf{\Pi}_i^{(d,s)}(1) + \sum_{d=1}^{D} \sum_{k=1}^{m-1} \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \left[\frac{\mathbf{d}(\mathbf{x}^j, \mathbf{f}(\mathbf{x}^i))}{p(1-p)} - \frac{n}{1-p} \right] \mathbf{\Xi}_{i,j}^{(d,s)}(k), \tag{2.70}$$

$$\frac{\partial Q}{\partial \lambda} = \frac{1}{\sigma^2} \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{i=1}^{2^n} \sum_{j \in G_k^{(d)}} \left(y_j^{(d)}(k) - \lambda - \delta \mathbf{x}_j^i \right) \mathbf{\Pi}_i^{(d,s)}(k), \tag{2.71}$$

$$\frac{\partial Q}{\partial \delta} = \frac{1}{\sigma^2} \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{i=1}^{2^n} \sum_{j \in G_k^{(d)}} \mathbf{x}_j^i \left(y_j^{(d)}(k) - \lambda - \delta \mathbf{x}_j^i \right) \mathbf{\Pi}_i^{(d,s)}(k), \tag{2.72}$$

$$\frac{\partial Q}{\partial \sigma^2} = \frac{-1}{2\sigma^2} \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{i=1}^{2^n} \sum_{j \in G_k^{(d)}} \left[1 - \frac{\left(y_j^{(d)}(k) - \lambda - \delta \mathbf{x}_j^i \right)^2}{\sigma^2} \right] \mathbf{\Pi}_i^{(d,s)}(k), \tag{2.73}$$

respectively. $\pi_i^{'}$ in (2.70) is the derivative of the steady-state distribution of the state i with respect to p, that is, $\pi_i^{'} = \frac{\partial \pi_i}{\partial p}$. To find π', we start with the fact that the steady-state distribution $\pi = [\pi_1, \pi_2, \cdots, \pi_{2^n}]$ satisfies

$$\begin{aligned} \pi &= \pi A, & (2.74) \\ \sum_{i=1}^{2^n} \pi_i &= 1, & (2.75) \end{aligned}$$

where A is the TPM with the corresponding entries (2.64),

$$A_{i,j} = p^{\mathbf{d}(\mathbf{x}^j, \mathbf{f}(\mathbf{x}^i))} (1-p)^{n - \mathbf{d}(\mathbf{x}^j, \mathbf{f}(\mathbf{x}^i))}. \tag{2.76}$$

Taking the derivative of both sides in (2.74) and (2.75) with respect to p yields

$$\begin{aligned} \pi'(I - A) &= \pi A', & (2.77) \\ \sum_{i=1}^{2^n} \pi_i' &= 0, & (2.78) \end{aligned}$$

where A' is the derivative of the TPM with respect to p and, using (2.76), can be written in terms of A as

$$A'_{i,j} = \left(\frac{\mathbf{d}(\mathbf{x}^j, \mathbf{f}(\mathbf{x}^i)) - np}{p(1-p)} \right) A_{i,j}. \tag{2.79}$$

π' is easily found from the linear equations (2.77) and (2.78).

Given the derivatives in (2.71), (2.72), and (2.73), thanks to the specific form of the Gaussian distribution, we can derive closed-formed solutions for λ, δ, and σ^2 by setting the derivatives equal to zero. Such closed-form solutions considerably reduce the complexity of the EM algorithm because they eliminate the iterative computations required for the algorithms like gradient descent in every M-step. Define ρ_1 and ρ_2 by

$$\begin{aligned}\rho_1 &= \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{i=1}^{2^n} \sum_{j \in G_k^{(d)}} y_j^{(d)}(k) \mathbf{\Pi}_i^{(d,s)}(k) \\ &= \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{j \in G_k^{(d)}} y_j^{(d)}(k), \end{aligned} \tag{2.80}$$

$$\rho_2 = \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{i=1}^{2^n} \sum_{j \in G_k^{(d)}} \mathbf{\Pi}_i^{(d,s)}(k) = \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{j \in G_k^{(d)}} 1, \tag{2.81}$$

where we have used the fact that the summation of the posterior probabilities of the states is one, $\sum_{i=1}^{2^n} \mathbf{\Pi}_i^{d,s}(k) = 1$, for any d and k and s. Define ρ_3 and ρ_4 by

$$\rho_3 = \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{i=1}^{2^n} \sum_{j \in G_k^{(d)}} \mathbf{x}_j^i \mathbf{\Pi}_i^{(d,s)}(k), \tag{2.82}$$

$$\rho_4 = \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{i=1}^{2^n} \sum_{j \in G_k^{(d)}} \mathbf{x}_j^i y_j^{(d)}(k) \mathbf{\Pi}_i^{(d,s)}(k). \tag{2.83}$$

Setting the derivatives in (2.71) and (2.72) equal to zero yields two linear equations for λ and δ:

$$\rho_2 \lambda + \rho_3 \delta = \rho_1, \quad \rho_3 \lambda + \rho_3 \delta = \rho_4. \tag{2.84}$$

If $\rho_3 \neq 0$, then

$$\lambda^{(s+1)} = \frac{\rho_1 - \rho_4}{\rho_2 - \rho_3}, \tag{2.85}$$

$$\delta^{(s+1)} = \frac{\rho_2 \rho_4 - \rho_1 \rho_3}{\rho_2 \rho_3 - \rho_3^2}. \tag{2.86}$$

If $\rho_3 = 0$, then $\rho_4 = 0$, $\lambda^{(s+1)} = \frac{\rho_1}{\rho_2}$, but δ cannot be found. In this case, we define $\delta^{(s+1)} = \delta^{(s)}$.

Now define ρ_5 by

$$\rho_5 = \sum_{d=1}^{D} \sum_{k \in T_{obs}^{(d)}} \sum_{i=1}^{2^n} \sum_{j \in G_k^{(d)}} \left(y_j^{(d)}(k) - \lambda^{(s+1)} - \delta^{(s+1)} \mathbf{x}_j^i \right)^2 \mathbf{\Pi}_i^{(d,s)}(k). \tag{2.87}$$

Setting the derivative in (2.73) equal to zero gives the following solution for σ^2:

$$\sigma^{2(s+1)} = \frac{\rho_5}{\rho_2}. \tag{2.88}$$

To find a closed-form solution for p note that the derivative of Q with respect to p in (2.70) consists of two terms. In the first term, π and π' are not explicit functions of p, which means that we are unable to derive a closed-form solution for p by setting the derivative equal to zero. From simulations, we found that the second term in (2.70) plays a much more important role than the first term, that is, has a much larger value. Hence, a good approximation results from omitting the first term in (2.70) and setting the second to zero, which gives the following approximate closed-form solution for p (we tested its accuracy and it can correctly estimate the real p):

$$p^{(s+1)} = \frac{\rho_6}{nD(m-1)}, \tag{2.89}$$

where ρ_6 is defined as

$$\rho_6 = \sum_{d=1}^{D} \sum_{k=1}^{m-1} \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \mathbf{d}(\mathbf{x}^j, \mathbf{f}(\mathbf{x}^i)) \Xi_{i,j}^{(d,s)}(k). \tag{2.90}$$

Note that in deriving (2.89) we have used the fact that $\sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \Xi_{i,j}^{(d,s)}(k) = 1$ for any k, d, and s, since $\Xi_{i,j}^{(d,s)}(k)$ is the posterior probability of two consecutive states being i and j at times k and $k+1$, respectively.

• Computing the posterior probabilities of the states:

As our model is an HMM, the posterior probabilities in (2.68) and (2.69) can be efficiently computed using the forward-backward algorithm, whose complexity is linear in m. From Bayes rule, we know that the posterior probabilities of states such as those in (2.68) and (2.69) can be

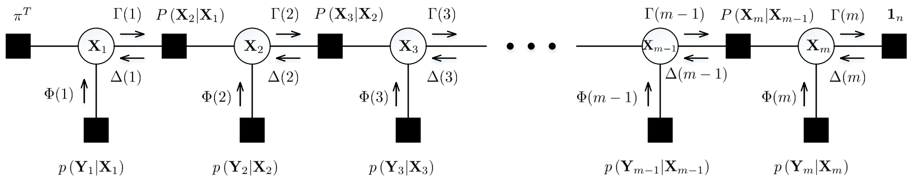


Figure 2.9: Factor graph of the HMM. Reprinted with permission from [2], ©2019 IEEE.

computed using the joint distribution of the states $\mathcal{X}$ and observation trajectories $\mathcal{Y}$, which are factored as in (2.63). A joint distribution (2.63) for an HMM can be represented by a factor graph as in Fig. 2.9 [53]. In this figure, the circles show the state variable from time 1 to m; the factor nodes (black) between the state nodes denote the transition probabilities between two consecutive states; the factor nodes under the state nodes represent the likelihood of each observation given its corresponding state. The rightmost factor node is an all-one vector and the leftmost factor node is the initial distribution of the states, which based on our assumption is the steady-state distribution. The message-passing is done through the graph by defining so-called forward, Γ, and backward parameters, Δ. It can be shown that the posteriors in (2.68) and (2.69) can be derived as [53]

$$\mathbf{\Pi}_i^{(d,s)}(k) = \frac{\Gamma_i^{(d,s)}(k)\Delta_i^{(d,s)}(k)}{\sum_{r=1}^{2^n}\Gamma_r^{(d,s)}(k)\Delta_r^{(d,s)}(k)}, \tag{2.91}$$

$$\mathbf{\Xi}_{i,j}^{(d,s)}(k) = \frac{\Gamma_i^{(d,s)}(k)A_{i,j}^{(s)}\Delta_j^{(d,s)}(k+1)\Phi_j^{(d,s)}(k+1)}{\sum_{r=1}^{2^n}\Gamma_r^{(d,s)}(m)}, \tag{2.92}$$

where $A_{i,j}^{(s)} = P(\mathbf{X}_{k+1} = \mathbf{x}^j|\mathbf{X}_k = \mathbf{x}^i, \theta = \theta^{(s)})$ is the transition matrix defined in (2.76), and $\Phi^{(d,s)}(k)$ is a $2^n \times 1$ vector at time k, whose j-th entry is defined as $\Phi_j^{(d,s)}(k) = p(\mathbf{Y}_k^{(d)}|\mathbf{X}_k^{(d)} = \mathbf{x}^j, \theta = \theta^{(s)})$, which can be computed using (2.65). Furthermore, $\Gamma_i^{(d,s)}(k)$ and $\Delta_i^{(d,s)}(k)$ are respectively the forward and backward parameters, defined and recursively computed by

$$\begin{aligned}
\Gamma_i^{(d,s)}(k) &= p(\mathbf{Y}_1^{(d)}, \cdots, \mathbf{Y}_k^{(d)}, \mathbf{X}_k^{(d)} = \mathbf{x}^i | \theta^{(s)}), \\
\Gamma_i^{(d,s)}(1) &= \pi_i^{(s)} \Phi_i^{(d,s)}(1), \\
\Gamma_j^{(d,s)}(k+1) &= \Phi_j^{(d,s)}(k+1) \sum_{i=1}^{2^n} \Gamma_i^{(d,s)}(k) A_{i,j}^{(s)}, \qquad (2.93)
\end{aligned}$$

and

$$\begin{aligned}
\Delta_i^{(d,s)}(k) &= p(\mathbf{Y}_{k+1}^{(d)}, \cdots, \mathbf{Y}_m^{(d)} | \mathbf{X}_k^{(d)} = \mathbf{x}^i, \theta^{(s)}), \\
\Delta_i^{(d,s)}(m) &= 1, \\
\Delta_i^{(d,s)}(k) &= \sum_{j=1}^{2^n} \Delta_j^{(d,s)}(k+1) A_{i,j}^{(s)} \Phi_j^{(d,s)}(k+1), \qquad (2.94)
\end{aligned}$$

for any $k = 1, \cdots, m-1$.

Define the vectors $\Gamma^{(d,s)}(k) = [\Gamma_1^{(d,s)}(k), \cdots, \Gamma_{2^n}^{(d,s)}(k)]^T$ and $\Delta^{(d,s)}(k) = [\Delta_1^{(d,s)}(k), \cdots, \Delta_{2^n}^{(d,s)}(k)]^T$. From (2.93) and (2.94), we have the following recursions in vector-matrix form:

$$\begin{aligned}
\Gamma^{(d,s)}(1) &= {\pi^{(s)}}^T \circ \Phi^{(d,s)}(1), \\
\Gamma^{(d,s)}(k+1) &= \left[{A^{(s)}}^T \Gamma^{(d,s)}(k) \right] \circ \Phi^{(d,s)}(k+1), \qquad (2.95)
\end{aligned}$$

and

$$\begin{aligned}
\Delta^{(d,s)}(m) &= \mathbf{1}_{2^n}, \\
\Delta^{(d,s)}(k) &= A^{(s)} \left[\Delta^{(d,s)}(k+1) \circ \Phi^{(d,s)}(k+1) \right], \qquad (2.96)
\end{aligned}$$

where $\mathbf{1}_{2^n}$ is the all-one column vector of length 2^n and $\circ$ denotes the Hadamard product (or component-wise product). The superscript T denotes transpose. Now suppose that $\mathbf{\Pi}^{(d,s)}(k)$ and $\Xi^{(d,s)}(k)$ are respectively a $2^n \times 1$ vector and $2^n \times 2^n$ matrix whose entries are given in (2.91) and

(2.92). Then,

$$\mathbf{\Pi}^{(d,s)}(k) = \frac{\Gamma^{(d,s)}(k) \circ \Delta^{(d,s)}(k)}{\| \Gamma^{(d,s)}(k) \circ \Delta^{(d,s)}(k) \|_1}, \quad k = 1, \cdots, m, \tag{2.97}$$

$$\mathbf{\Xi}^{(d,s)}(k) = \frac{\left[\Gamma^{(d,s)}(k)\Delta^{(d,s)}(k+1)^T\right] \circ A^{(s)} \circ \mathbf{\Phi}^{(d,s)}(k+1)}{\| \Gamma^{(d,s)}(m) \|_1}, \tag{2.98}$$

$k = 1, \cdots, m-1$, where $\mathbf{\Phi}^{(d,s)}(k)$ is the $2^n \times 2^n$ matrix

$$\mathbf{\Phi}^{(d,s)}(k) = \left[\Phi^{(d,s)}(k), \cdots, \Phi^{(d,s)}(k)\right]^T. \tag{2.99}$$

In the case of missing data, if there is a time point k at which no expression of the n genes is observed ($k \notin T_{obs}$), then $p(\mathbf{Y}_k|\mathbf{X}_k = \mathbf{x}^i) = 1$ for any $i = 1, \cdots, 2^n$, and $\Phi(k) = \mathbf{1}_{2^n}$ in Fig. 2.9, as well as in all the equations (2.91)-(2.99); however, if at least one gene expression is observed at time k ($k \in T_{obs}$), then $p(\mathbf{Y}_k|\mathbf{X}_k)$, and thus $\Phi_j(k) = p(\mathbf{Y}_k|\mathbf{X}_k = \mathbf{x}^j)$, is computed from (2.65).

2.2.4.2 *Learning Algorithm*

In the previous subsection, we demonstrated that, given the network function, we can estimate the parameters by the EM method. Let $\hat{\theta}_i$ denote the estimated parameter vector, defined in (2.60) and derived via the EM algorithm when $\mathbf{f} = \mathbf{f}^i$ for $i = 1, \cdots, M$. The final estimates for both the network function, $\hat{\mathbf{f}}$, and the parameter vector, $\hat{\theta}$, can be determined from (2.61). Let $l(\mathbb{Y}|\mathbf{f}, \theta) = \log p(\mathbb{Y}|\mathbf{f}, \theta)$ be the log-likelihood of the observed trajectories. Since all D observations are independent,

$$l(\mathbb{Y}|\mathbf{f}, \theta) = \sum_{d=1}^{D} \log p(\mathcal{Y}^{(d)}|\mathbf{f}, \theta). \tag{2.100}$$

For $d = 1, \cdots, D$, $p(\mathcal{Y}^{(d)}|\mathbf{f}, \theta)$ is derived by marginalizing the joint distribution of the states and observations over the states as

$$p(\mathcal{Y}^{(d)}|\mathbf{f}, \theta) = \sum_{\mathcal{X}^{(d)}} p(\mathcal{X}^{(d)}, \mathcal{Y}^{(d)}|\mathbf{f}, \theta), \tag{2.101}$$

where $p(\mathcal{X}^{(d)}, \mathcal{Y}^{(d)}|\mathbf{f}, \theta)$ is given by (2.63) with the factor-graph shown in Fig. 2.9. Computing $p(\mathcal{Y}^{(d)}|\mathbf{f}, \theta)$ only requires the forward parameter Γ and recursions up to time m. In fact, the desired likelihood for the d-th trajectory is the summation of the entries of $\Gamma^{(d)}(m)$, that is,

$$p(\mathcal{Y}^{(d)}|\mathbf{f}, \theta) = \parallel \Gamma^{(d)}(m) \parallel_1, \tag{2.102}$$

where $\Gamma^{(d)}(m)$ can be computed in the same way as in (2.95), assuming the parameter vector θ and the network function $\mathbf{f}$. From (2.100) and (2.102), we can write

$$l(\mathbb{Y}|\mathbf{f}, \theta) = \sum_{d=1}^{D} \log \parallel \Gamma^{(d)}(m) \parallel_1 . \tag{2.103}$$

Then, according to (2.61), the final estimate for the network function and parameters is given by

$$(\hat{\mathbf{f}}, \hat{\theta}) = \arg \max_{(\mathbf{f},\theta)\in\{(\mathbf{f}^1,\hat{\theta}_1),\cdots,(\mathbf{f}^M,\hat{\theta}_M)\}} l(\mathbb{Y}|\mathbf{f}, \theta). \tag{2.104}$$

We summarize the algorithm for learning the network function and model parameters in Algorithm 2.

2.2.4.3 *Plug-In Bayes Classifier*

Let labels 0 and 1 refer to the healthy and mutated classes, respectively, let $\mathbb{Y}_0$ and $\mathbb{Y}_1$ denote the respective training trajectory sets, and let $\mathbf{F}_0$ and $\mathbf{F}_1$ denote the respective uncertain network function sets for the two classes. We apply Algorithm 2 to both classes, with corresponding $\mathbb{Y}$ and $\mathbf{F}$, to derive the learned network functions $\hat{\mathbf{f}}_0$ and $\hat{\mathbf{f}}_1$, and the estimated parameters $\hat{\theta}_0$ and $\hat{\theta}_1$, for the healthy and mutated classes, respectively. These are plugged into the Bayes classifier. For any new trajectory $\mathcal{Y} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_m]$ of n genes with any arbitrary length m and possibly missing data, the classifier is defined by

$$\psi_D(\mathcal{Y}) = \begin{cases} 1, & \hat{c}_1 p(\mathcal{Y}|\hat{\mathbf{f}}_1, \hat{\theta}_1) \geq \hat{c}_0 p(\mathcal{Y}|\hat{\mathbf{f}}_0, \hat{\theta}_0) \\ 0, & \hat{c}_1 p(\mathcal{Y}|\hat{\mathbf{f}}_1, \hat{\theta}_1) < \hat{c}_0 p(\mathcal{Y}|\hat{\mathbf{f}}_0, \hat{\theta}_0) \end{cases}, \tag{2.105}$$

Algorithm 2 Learning Algorithm

```
1: Inputs: the number of genes $n$, the length of trajectories $m$, the set of $D$ observations
   $\mathbb{Y} = \{\mathcal{Y}^{(1)}, \mathcal{Y}^{(2)}, \cdots, \mathcal{Y}^{(D)}\}$, the set of uncertain network functions $\mathbf{F} = \{\mathbf{f}^1, \cdots, \mathbf{f}^M\}$,
   and a convergence threshold, $\tau$, for the EM.
2: Outputs: $\hat{\mathbf{f}}$ and $\hat{\theta}$, the estimated network function and model parameters $\theta = [p, \lambda, \delta, \sigma^2]^T$.
3: procedure
4:     for $i = 1$ to $M$ do
5:         • $s = 0$
6:         • $\mathbf{f} = \mathbf{f}^i$
7:         • Initialize $\theta^{(0)} = 0$
8:         • Randomly initialize $\theta^{(1)}$
9:         while $\| \theta^{(s+1)} - \theta^{(s)} \| > \tau$ do
10:            • $s = s + 1$
11:            E-step:
12:            • Compute $\mathbf{\Pi}^{(d,s)}(k)$ for any $k = 1, \cdots, m$,
                 and $d = 1, \cdots, D$ via (2.97).
13:            • Compute $\Xi^{(d,s)}(k)$ for any $k = 1, \cdots, m - 1$,
                 and $d = 1, \cdots, D$ via (2.98).
14:            M-step:
15:            • Compute $\rho_j$ for $j = 1, \cdots, 6$, via (2.80),
                 (2.81), (2.82), (2.83), (2.87), and (2.90).
16:            • Compute $\lambda^{(s+1)}$ via (2.85).
17:            • Compute $\delta^{(s+1)}$ via (2.86).
18:            • Compute $\sigma^{2^{(s+1)}}$ via (2.88).
19:            • Compute $p^{(s+1)}$ via (2.89).
20:            • $\theta^{(s+1)} = \left[p^{(s+1)}, \lambda^{(s+1)}, \delta^{(s+1)}, \sigma^{2^{(s+1)}}\right]^T$
21:        end while
22:        • $\hat{\theta}_i = \theta^{(s+1)}$
23:    end for
24:    • Get $\hat{\mathbf{f}}$ and $\hat{\theta}$ via (2.103) and (2.104).
25: end procedure
```

where $\hat{c}_0$ and $\hat{c}_1$ are the estimated values of the prior probabilities of the healthy and mutated classes, respectively. These can be estimated by the number of training trajectories for each class divided by the total number of training trajectories D; however, this estimate is unreliable for small samples [54]. Often there are substantial data regarding the proportions of healthy and pathological phenotypes – for instance, false negative rates resulting from preliminary testing such as mammography and needle biopsies, so that excellent estimates of $\hat{c}_0$ and $\hat{c}_1$ are available for genomic classification following prelimineary testing. We assume the equiprobable case, $\hat{c}_0 = \hat{c}_1 = \frac{1}{2}$, which makes classification most challenging.

In the simulations we generate an equal number of training trajectories for each class, $|\mathbb{Y}_0| = |\mathbb{Y}_1| = \frac{D}{2}$. The likelihoods $p(\mathcal{Y}|\hat{\mathbf{f}}_0, \hat{\theta}_0)$ and $p(\mathcal{Y}|\hat{\mathbf{f}}_1, \hat{\theta}_1)$ in (2.105) can be computed, as in (2.102), by

$$p(\mathcal{Y}|\hat{\mathbf{f}}_i, \hat{\theta}_i) = \| \Gamma^{(i)}(m) \|_1, \quad i = 0, 1, \tag{2.106}$$

where $\Gamma^{(i)}(m)$ can be computed by forward computations, as in (2.95), by

$$\begin{aligned} \Gamma^{(i)}(1) &= {\pi^{(i)}}^T \circ \Phi^{(i)}(1), \\ \Gamma^{(i)}(k+1) &= \left[{A^{(i)}}^T \Gamma^{(i)}(k) \right] \circ \Phi^{(i)}(k+1), \end{aligned} \tag{2.107}$$

where $\pi^{(i)}$, $A^{(i)}$, and $\Phi^{(i)}$ are computed for the class $i = 0, 1$, assuming $\mathbf{f} = \hat{\mathbf{f}}_i$ and $\theta = \hat{\theta}_i$.

2.2.5 Classification of Averaged Steady-State Expression Data in Multiple-Cell Scenarios

In the absence of single-cell technology, when measuring expressions in a nonsynchronized multiple-cell setting, at each time point the measured expression value of each gene is an average over 2^n different states. The underlying state model for the evolution of the genes in each cell is the same BNp model (2.56); however, since the observations are static expression data in the steady-state and not trajectories, we use the state model (2.56) only for calculating the steady-state distribution π.

2.2.5.1 Observation Model

Suppose that the expression values of the genes in every individual are measured from a tissue consisting of N cells. As mentioned previously, the variability existing in the different cells of an individual is inter-cell variability. According to (2.59), in every cell c of an individual, the expression values of the n genes, $\mathbf{Y}^{(c)} = [y_1^{(c)}, \cdots, y_n^{(c)}]^T$, given the state $\mathbf{X} = [x_1, \cdots, x_n]^T$ in the steady-state, follow a Gaussian model:

$$p(\mathbf{Y}^{(c)}|\mathbf{X}) \sim \mathcal{N}\left(\lambda\mathbf{1}_n + \delta\mathbf{X}, \sigma_{ic}^2 I_n\right), \quad c = 1, \cdots, N, \tag{2.108}$$

where σ_{ic}^2 denotes the inter-cell variability across the different cells of an individual. In the multiple-cell scenario we do not observe expression values of the genes in every single cell but only observe the expression averaged over N cells, namely,

$$S_N = \frac{\sum_{c=1}^{N} \mathbf{Y}^{(c)}}{N}. \tag{2.109}$$

We can obtain the distribution of $\mathbf{Y}^{(c)}$ from (2.108) by marginalizing over the states $\mathbf{X}$, which have the steady-state distribution π. Doing so, the distribution of $\mathbf{Y}^{(c)}$ for any c is

$$p(\mathbf{Y}^{(c)}) = \sum_{i=1}^{2^n} p(\mathbf{Y}^{(c)}|\mathbf{X} = \mathbf{x}^i)\pi_i, \tag{2.110}$$

which is a Gaussian mixture distribution with 2^n components. Since the $\mathbf{Y}^{(c)}$'s are independent for different cells and have the same mean and covariance matrix, we can use the central-limit theorem to approximate the distribution of S_N. The mean and covariance matrix of $\mathbf{Y}^{(c)}$ are given by

$$\begin{aligned} \mu = \mathrm{E}\left[\mathbf{Y}^{(c)}\right] = \mathrm{E}\left[\mathrm{E}\left[\mathbf{Y}^{(c)}|\mathbf{X}\right]\right] = \\ \mathrm{E}\left[\lambda 1_n + \delta\mathbf{X}\right] = \lambda\mathbf{1}_n + \delta\sum_{i=1}^{2^n}\mathbf{x}^i\pi_i, \end{aligned} \tag{2.111}$$

$$\begin{aligned} \Sigma_{\mathbf{Y}} &= \mathrm{cov}\left(\mathbf{Y}^{(c)}\right) = \mathrm{cov}\left(\mathrm{E}(\mathbf{Y}^{(c)}|\mathbf{X})\right) + \mathrm{E}(\mathrm{cov}\left(\mathbf{Y}^{(c)}|\mathbf{X}\right)) \\ &= \mathrm{cov}(\lambda\mathbf{1}_n + \delta\mathbf{X}) + \mathrm{E}(\sigma_{ic}^2 I_n) = \delta^2\Sigma_{\mathbf{X}} + \sigma_{ic}^2 I_n, \end{aligned} \tag{2.112}$$

where $\Sigma_{\mathbf{X}}$ is the covariance matrix of the states in the steady-state,

$$\begin{aligned}\Sigma_{\mathbf{X}} = \mathrm{cov}(\mathbf{X}) = \mathrm{E}(\mathbf{X}\mathbf{X}^T) - \mathrm{E}(\mathbf{X})\mathrm{E}(\mathbf{X})^T = \\ \sum_{i=1}^{2^n} \mathbf{x}^i {\mathbf{x}^i}^T \pi_i - \left(\sum_{i=1}^{2^n} \mathbf{x}^i \pi_i\right)\left(\sum_{i=1}^{2^n} \mathbf{x}^i \pi_i\right)^T .\end{aligned} \tag{2.113}$$

According to the central-limit theorem, when N is large (having many cells), the distribution of S_N converges to the multivariate normal

$$S_N \sim \mathcal{N}\left(\mu, \frac{\Sigma_{\mathbf{Y}}}{N}\right), \tag{2.114}$$

where μ and $\Sigma_{\mathbf{Y}}$ are given in (2.111) and (2.112).

Since S_N is the averaged expression values of the genes over many cells in only one individual and our samples come from different individuals, we should also take into account the inter-subject variability between different individuals. As a result, in the multiple-cell scenario, the expression values of the n genes, denoted by the vector $\mathbf{Z} = [z_1, \cdots, z_n]^T$, can be modeled by

$$\mathbf{Z} = S_N + \epsilon, \tag{2.115}$$

where ϵ provides the inter-subject variability. Since in the single-cell scenario we assumed that $\epsilon \sim N(0, \sigma^2 I_n)$, we assume the same here. Since S_N and ϵ are multivariate Gaussian and independent of each other, $\mathbf{Z}$ has a multivariate Gaussian distribution,

$$p(\mathbf{Z}) \sim \mathcal{N}\left(\mu, \frac{\Sigma_{\mathbf{Y}}}{N} + \sigma^2 I_n\right). \tag{2.116}$$

Usually there are many cells (large N) in the tissues from which the expressions are measured. N is typically large in practice, for instance, according to [55], there are millions of cells in bulk RNA-Seq experiments. Hence, the first part of the covariance matrix of $\mathbf{Z}$, that is, $\frac{\Sigma_{\mathbf{Y}}}{N}$, has negligible entries, and we can well approximate the distribution of $\mathbf{Z}$ by

$$p(\mathbf{Z}|\mu, \sigma^2) \sim \mathcal{N}\left(\mu = \lambda \mathbf{1}_n + \delta \sum_{i=1}^{2^n} \mathbf{x}^i \pi_i, \sigma^2 I_n\right). \tag{2.117}$$

2.2.5.2 Plug-In Bayes Classifier

To use the Bayes plug-in classifier, we need to estimate the parameters using the training data, but since the class-conditional densities in (2.117) are Gaussian and σ^2 may be different in the two classes, the Bayes plug-in classifier is quadratic discriminant analysis (QDA). Hence, we need only estimate μ and σ^2 in (2.117), not λ, δ, and π. This reduces the complexity because we skip the cumbersome optimization problem of finding p (for estimating π), which would have been done for each of M possible network functions $\mathbf{f}^{(i)}$, for $i = 1, \cdots, M$, for each class. Moreover, we do not even need to partially know the network functions in the classifier, which is beneficial when we have no knowledge of network structures. In other words, although μ in (2.117) is a function of λ, δ, p, and $\mathbf{f}$ (two last determine π), we do not need to estimate them to estimate μ, which we can directly estimate from the observed data.

The ML estimates of μ and σ^2 in (2.117) from observed data $\mathbb{Z} = [\mathbf{Z}^{(1)}, \cdots, \mathbf{Z}^{(D)}]$ are

$$\hat{\mu} = \frac{\sum_{d=1}^{D} \mathbf{Z}^{(d)}}{D}, \tag{2.118}$$

$$\hat{\sigma}^2 = \frac{\sum_{d=1}^{D} \parallel \mathbf{Z}^{(d)} - \hat{\mu} \parallel_2^2}{nD}, \tag{2.119}$$

respectively. The log-likelihood (after dropping the constant parts) of any expression vector $\mathbf{Z}$ from (2.117) is

$$l(\mathbf{Z}|\mu, \sigma^2) = -\frac{n}{2} \log \sigma^2 - \frac{\parallel \mathbf{Z} - \mu \parallel_2^2}{2\sigma^2}. \tag{2.120}$$

Let $\mathbb{Z}_0$ and $\mathbb{Z}_1$ denote the training data sets of the healthy and mutated classes, respectively, the total number of data points being D. Suppose $\hat{\mu}_i$ and $\hat{\sigma}_i^2$ are the estimated values (using (2.118) and (2.119)) for the class $i = 0, 1$ using $\mathbb{Z}_i$. We assume $\hat{c}_0 = \hat{c}_1 = \frac{1}{2}$ and $| \mathbb{Z}_0 |=| \mathbb{Z}_1 |= \frac{D}{2}$. The

Bayes plug-in classifier (QDA) for $\mathbf{Z} = [z_1, \cdots, z_n]^T$ is

$$\psi_D(\mathbf{Z}) = \begin{cases} 1, & l(\mathbf{Z}|\hat{\mu}_1, \hat{\sigma}_1^2) \geq l(\mathbf{Z}|\hat{\mu}_0, \hat{\sigma}_0^2) \\ 0, & l(\mathbf{Z}|\hat{\mu}_1, \hat{\sigma}_1^2) < l(\mathbf{Z}|\hat{\mu}_0, \hat{\sigma}_0^2) \end{cases}, \tag{2.121}$$

where $l(\mathbf{Z}|\hat{\mu}_i, \hat{\sigma}_i^2)$ can be computed from (2.120) for each class $i = 0, 1$.

2.2.5.3 Classification Difficulty

We wish to quantify classification difficulty relative to attractor structure. From (2.117), the expression values of the n genes in classes 0 and 1 are modeled as

$$p(\mathbf{Z}^{(j)}) \sim \mathcal{N}\left(\mu_j, \sigma_j^2 I_n\right), \quad j = 0, 1. \tag{2.122}$$

If we assume that λ and δ are the same in the two classes, then

$$\mu_j = \lambda 1_n + \delta \sum_{i=1}^{2^n} \mathbf{x}^i \pi_i^{(j)} = \lambda 1_n + \delta X {\pi^{(j)}}^T, \quad j = 0, 1, \tag{2.123}$$

where $X = [\mathbf{x}^1, \cdots, \mathbf{x}^{2^n}]$ is the $n \times 2^n$ binary matrix representing the binary states, its i-th column being the i-th binary state.

In the Gaussian settings, the means and covariance matrices affect classification error. We focus on the distance

$$\begin{aligned} \xi = (\mu_0 - \mu_1)^T(\mu_0 - \mu_1) = \\ \delta^2 \left(\pi^{(0)} - \pi^{(1)}\right) X^T X \left(\pi^{(0)} - \pi^{(1)}\right)^T \end{aligned} \tag{2.124}$$

between the means because this distance is directly relatable to the attractors. If the perturbation probability p is very small and each class has only one attractor cycle, then the steady-state distributions are accurately approximated by

$$\pi_i^{(j)} = \frac{1}{|\mathcal{A}_j|} 1(i \in \mathcal{A}_j), \quad j = 0, 1, \quad i = 1, \cdots, 2^n, \tag{2.125}$$

where $\mathcal{A}_j$ is the set of the attractor states of class j, $1(.)$ is the indicator function (equals to 1 if its argument is true and equals to 0 otherwise), and $|\mathcal{A}_j| \in \{1, \cdots, 2^n\}$ is the attractor length for class j [37].

For any attractor lengths we find networks having minimum ($\xi = 0$) and maximum distances, where a network is identified with its attractor states because according to the preceding equation ξ depends only on these. Letting $\Sigma(\mathcal{A}_j)$ denote the sum of the values in $\mathcal{A}_j$, if $|\mathcal{A}_0| = |\mathcal{A}_1|$ and $\Sigma(\mathcal{A}_0) = \Sigma(\mathcal{A}_1)$, then $\xi = 0$. We represent minimum ξ and maximum ξ cases by $(\mathcal{A}_0^f, \mathcal{A}_1^f)$ (f for failure) and $(\mathcal{A}_0^o, \mathcal{A}_1^o)$ (o for optimal), respectively, where

$$(\mathcal{A}_0^f, \mathcal{A}_1^f) = \left\{ \mathcal{A}_0, \mathcal{A}_1 : \left(\pi^{(0)} - \pi^{(1)}\right) X^T X \left(\pi^{(0)} - \pi^{(1)}\right)^T = 0 \right\}, \tag{2.126}$$

$$(\mathcal{A}_0^o, \mathcal{A}_1^o) = \underset{\mathcal{A}_0, \mathcal{A}_1}{\arg\max} \left(\pi^{(0)} - \pi^{(1)}\right) X^T X \left(\pi^{(0)} - \pi^{(1)}\right)^T. \tag{2.127}$$

Letting $l_0 = |\mathcal{A}_0|$ and $l_1 = |\mathcal{A}_1|$ denote the lengths, the numbers of sets $\mathcal{A}_0$ and $\mathcal{A}_1$ are $\binom{2^n}{l_0}$ and $\binom{2^n}{l_1}$, respectively. Note that there are $(l_0 - 1)!$ and $(l_1 - 1)!$ cyclic permutations of each set, which lead to different attarctors, but we do not consider them since they all have the same average and thus are equivalent in the current sense. Hence, the size of the search space for solving (2.126) and (2.127) is $\binom{2^n}{l_0}\binom{2^n}{l_1}$. In general, the solutions are not unique.

2.2.6 Simulation Results and Discussion

As this is the first work which studies supervised classification of single-cell gene expression trajectories under the framework of Boolean networks with perturbations, there is not a similar trajectory-based method to compare it with. However, we compare the performance of our proposed trajectory-based classifier, using single-cell gene expression trajectories as its input, with that of a multiple-cell averaging classifier which uses bulk gene expression data, like RNA-Seq or microarray, as its input. We have set the model parameters to do a fair comparison between these two methods. The comparisons show a clear advantage of the first method, especially in high-noise scenarios.

2.2.6.1 Some Specific Networks

Let $\mathcal{A} = \{a_1, a_2, \cdots, a_l\}$ be the set of attractor states with the length l, in which order matters, such that the attractor cycle is $a_1 \to a_2 \to \cdots \to a_l \to a_1$. We consider three specific cases as examples and compare the two methods of classification, single-cell trajectories and multiple-cell averaging. In all the cases, we assume $n = 4$, $p = 0.001$, $p_{miss} = 0$, $\lambda = 10$, and $\delta = 30$.

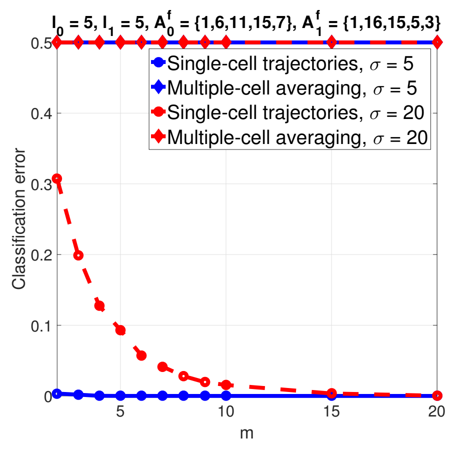
Case 1: Suppose the failure case with $l_0 = 5$ and $l_1 = 5$. We choose $\mathcal{A}_0 = \{1, 6, 11, 15, 7\}$ and $\mathcal{A}_1 = \{1, 16, 15, 5, 3\}$. Since $|\mathcal{A}_0| = |\mathcal{A}_1|$ and $\Sigma(\mathcal{A}_0) = \Sigma(\mathcal{A}_1)$, $\xi = 0$. Hence, we expect that averaging cannot perform well. Figure 2.10a represents the classification error of the two methods, single-cell trajectories and multiple-cell averaging, for $\sigma = 5$ and $\sigma = 20$. It can be seen that for any value of σ, averaging has the maximum classification error 0.5. However, the single-cell trajectory method has the error 0 (perfect classification for all m) for $\sigma = 5$, and a decreasing error as a function of m for $\sigma = 20$.

Case 2: Figures 2.10b, 2.10c, 2.10d, and 2.10e, which show results for optimal attractor sets derived from (2.127), relate to $l_0 = l_1 = 1$ (single attractors), $l_0 = l_1 = 3$, $l_0 = l_1 = 5$, and $l_0 = 2, l_1 = 3$, respectively. There are many solutions to (2.127). We only select the first solution to show the results, the selected optimal attractor sets $\mathcal{A}_0$ and $\mathcal{A}_1$ being written at the top of each figure. For example, in Fig. 2.10b ($l_0 = l_1 = 1$), $\mathcal{A}_0 = \{1\}$ and $\mathcal{A}_1 = \{16\}$, meaning that the attractor cycles in class 0 and class 1 are $[0, 0, 0, 0]^T \to [0, 0, 0, 0]^T$ and $[1, 1, 1, 1]^T \to [1, 1, 1, 1]^T$, respectively. For the low noise levels ($\sigma = 5$), both methods yield zero classification error. For $\sigma = 10$, the trajectory method still has zero error for every m, but averaging has nonzero error, even though its difference with zero is slight. For $\sigma = 20$, the trajectory method has nonzero error for small m, but it is still much less than the error of the averaging method. No matter the size of σ, the error of the trajectory method will converge to 0 for sufficiently large m. In sum, the trajectory method is more robust realtive to the noise level than the averaging method.

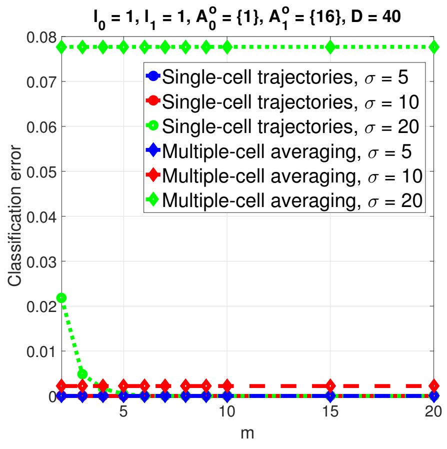
Case 3: The only scenario in which averaging can work better than the trajectory method is when there are similar trajectories in the attractor cycles of the two classes. In such situations, the trajectory method may make a mistake in classifying short trajectories, while the averaging method may be able to classify better. For example, consider $l_0 = l_1 = 5$ with $\mathcal{A}_0 = \{9, 5, 3, 2, 1\}$ and $\mathcal{A}_1 = \{9, 5, 3, 2, 16\}$. The trajectory $9 \to 5 \to 3 \to 2$ exists in the both attractor cycles. As a result, we expect that for short observed trajectories, the trajectory method will perform poorly. Figure 2.10f shows the results in this case for $\sigma = 2, 5, 20$. For $\sigma = 2$, the averaging method yields zero error but the trajectory method has nonzero error for $m \leq 4$. For $\sigma = 5$, the averaging method still has better performance than the trajectory method with $m \leq 3$, but for $\sigma = 20$, the trajectory method outperforms averaging for any m. This again shows that the averaging method cannot work in high noise regimes, whereas the trajectory method can result in a very low error if the observed trajectories are long enough.

2.2.6.2 Random Synthetic Networks

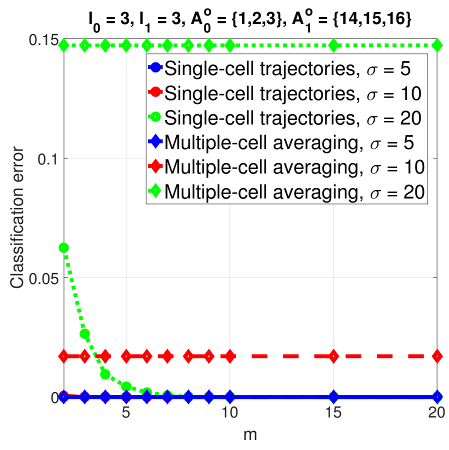
To evaluate performance on random synthetic networks, we randomly generate 500 Boolean networks for each case of $n = 4, 6, 8$ genes and consider a maximum in-degree of $K = 2$, meaning that each gene has 1 or 2 randomly assigned predictors. If the i-th gene has in-degree K_i, its 2^{K_i} outputs are selected from a Bernoulli distribution with parameter $p_{bias} = 0.5$. A single-bit mutation is applied to all healthy networks to obtain the corresponding 500 mutated networks. Specifically, we randomly pick a gene, say gene i, and randomly flip the value of one of its 2^{K_i} outputs, $0 \to 1$ and $1 \to 0$. This mutation changes the output of 2^{n-K_i} states in the truth table of the healthy BN. We restrict the generated healthy and mutated BNs to have a single attractor cycle, with the minimum length of the two attractor cycles being L. The simulations will demonstrate that L is an important parameter in determining the sufficient trajectory length in low-noise scenarios. In all simulations, we use the same parameter values, p, λ, δ, σ^2, for both the healthy and mutated



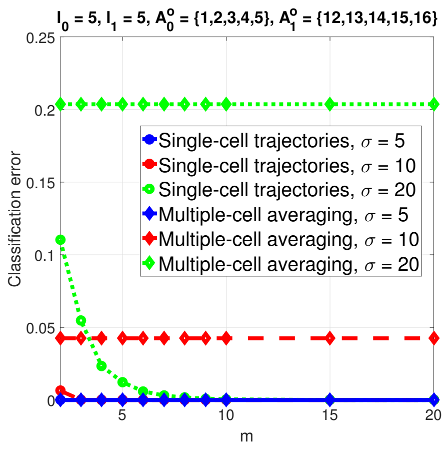
(a)



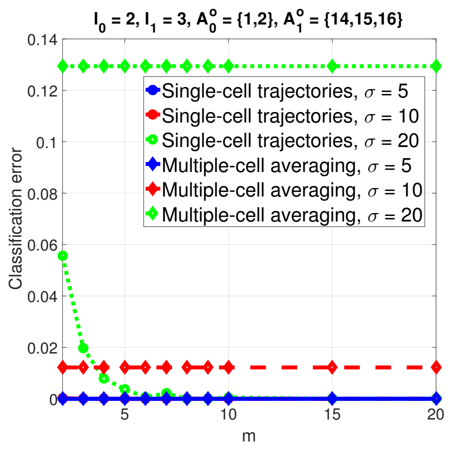
(b)



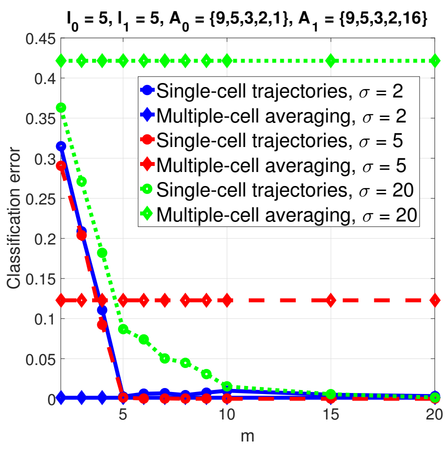
(c)



(d)



(e)



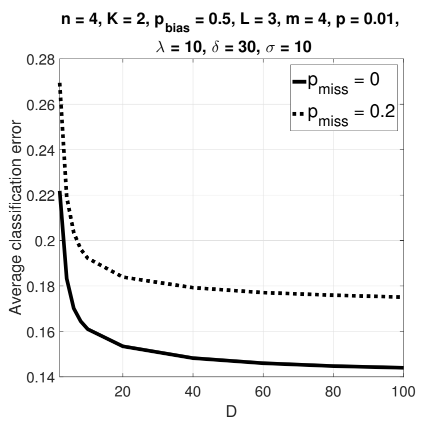
(f)

Figure 2.10: Classification error of single-cell trajectory method versus m. The classification error of the multiple-cell averaging method is also included in the plots for comparison. Reprinted with permission from [2], ©2019 IEEE.

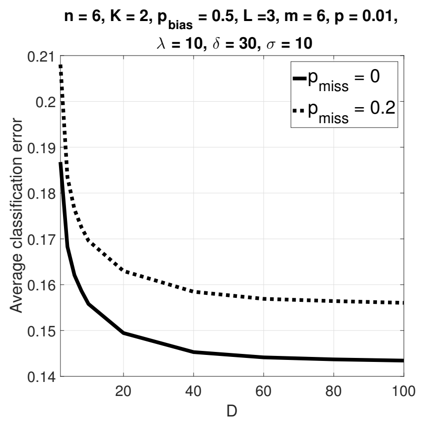
networks: $\lambda = 10$, $\delta = 30$, and $M = 2$. We use three different values for the observation noise level: $\sigma = 5$ (low noise), $\sigma = 10$ (medium noise), and $\sigma = 20$ (high noise).

Figure 2.11, in which $\sigma = 10$, shows average classification error versus D, the total number of training trajectories for both the healthy and mutated BNs, for different numbers of genes n, trajectory length m, minimum attractor length L, gene perturbation probability p, and gene missing probability p_{miss}. As expected, missing observations deteriorate classifier performance in all cases. Average error decreases with more training trajectories and converges to a fixed value when D becomes large enough. The value of D required for a converged error rate depends on n, m, and p_{miss}. For a given network size, having larger m and lower p_{miss} can speed up estimation of the network parameters, so that smaller D is required to achieve the converged error. Furthermore, for larger networks, the required value of D increases. Note that we have assumed that we partially know the networks, so that the search space is limited to M functions. Hence, the error curves have fairly fast convergence realtive to D. In the absence of network knowledge, more training data would be required to correctly learn the networks and convergence would be slower.

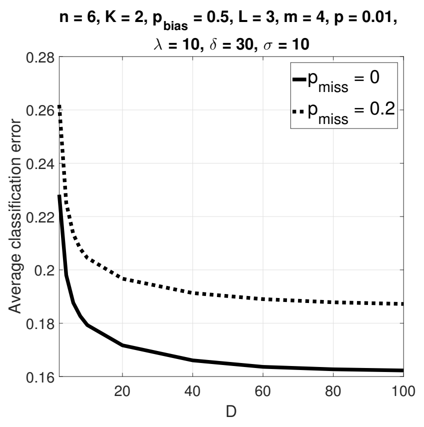
Figure 2.12 demonstrates the behavior of the average classification error versus m, where based on the results in Fig. 2.11, we assume $D = 40$ training trajectories to achieve the converged error. We set $p_{miss} = 0$ in Fig. 2.12. Figures 2.12a and 2.12b show the error for $n = 4$ gene networks when $p = 0.001$ and $p = 0.01$, respectively, assuming different values of $L = 2, 4$ and $\sigma = 5, 20$. Figures 2.12c and 2.12d present the error for $n = 6$ gene networks when $p = 0.001$ and $p = 0.02$, respectively, assuming different values of $L = 4, 6$ and $\sigma = 5, 20$. In all figures for every value of m and L, the error increases with increasing σ. Moreover, the error curves are always monotonically decreasing in terms of m. There is a special case in which the error gets fixed after some m. This is when p is close to 0 and σ is small. In such conditions, the sufficient m to achieve the least possible error is $L + 1$ because when $p \approx 0$ the BNps tend to BNs, which are deterministic, meaning that the observations occur only in the attractor states and circulate inside the attractor cycles. In such a case, the maximum length of a trajectory that can help distinguish the two networks is $L+1$, where L is the minimum length of the attractor cycles in the two networks. When p is considerable, there



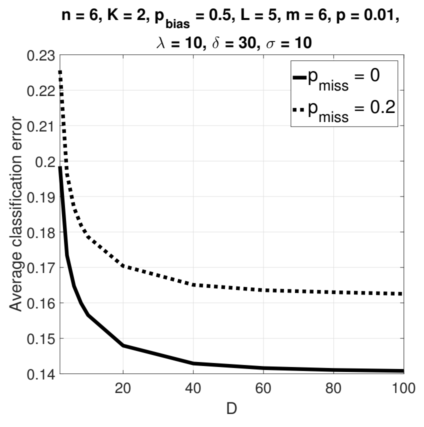
(a)



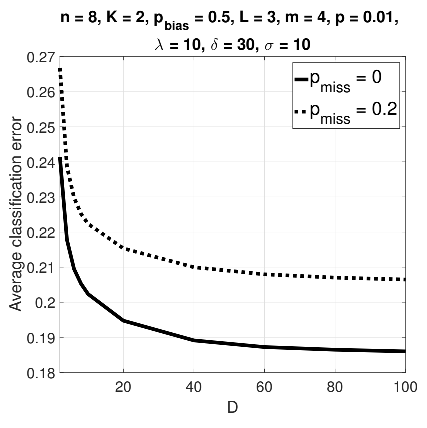
(b)



(c)

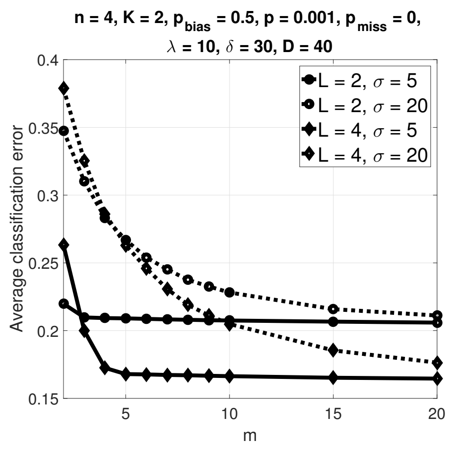


(d)

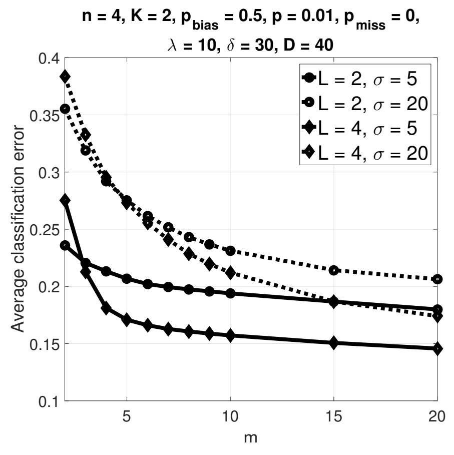


(e)

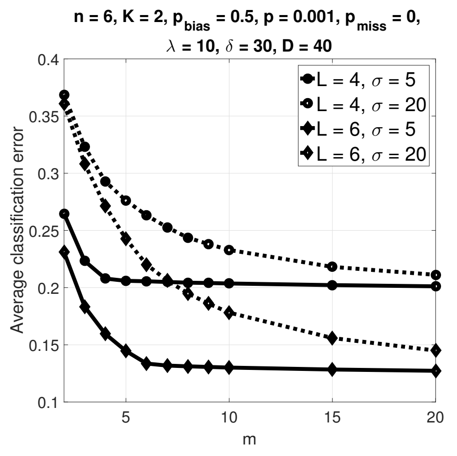
Figure 2.11: Average classification error of the trajectory classifier over 500 synthetic BNs versus D with $K = 2$ and $p_{bias} = 0.5$. Parameter values are $p = 0.01$, $\lambda = 10$, $\delta = 30$, $\sigma = 10$, (a) $n = 4$, $L = 3$, $m = 4$, (b) $n = 4$, $L = 5$, $m = 6$, (c) $n = 6$, $L = 3$, $m = 4$, (d) $n = 6$, $L = 5$, $m = 6$, (e) $n = 8$, $L = 3$, $m = 4$. Reprinted with permission from [2], ©2019 IEEE.



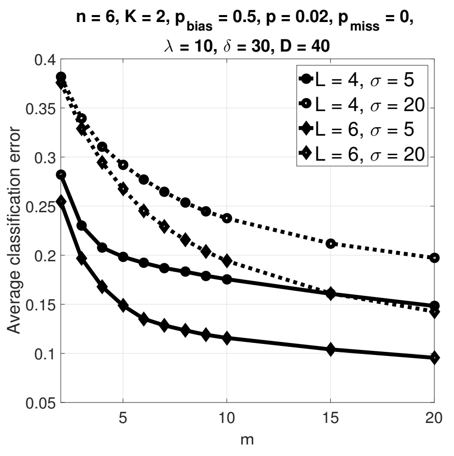
(a)



(b)



(c)

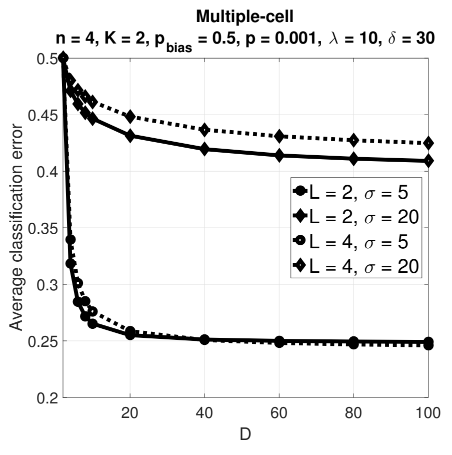


(d)

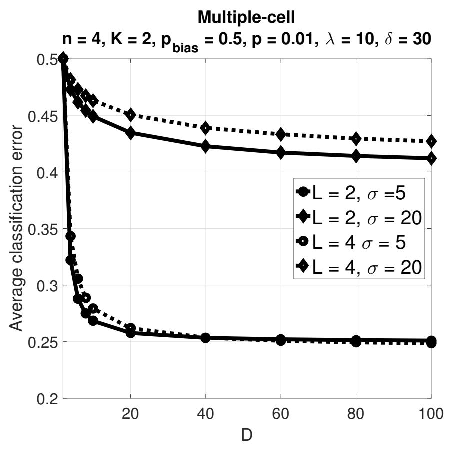
Figure 2.12: Average classification error of the trajectory classifier over 500 synthetic BNs versus m with $K = 2$ and $p_{bias} = 0.5$. Parameter values are $p_{miss} = 0$, $\lambda = 10$, $\delta = 30$, $D = 40$, (a) $n = 4$, $p = 0.001$, (b) $n = 4$, $p = 0.01$, (c) $n = 6$, $p = 0.001$, (d) $n = 6$, $p = 0.02$. Reprinted with permission from [2], ©2019 IEEE.

is a nonnegligible probability of jumping states, so that longer trajectories can help. In Figs. 2.12a and 2.12c, where $p = 0.001$, when $\sigma = 5$, the error curves flatten out after $m = L + 1 = 3, 5, 7$, corresponding to $L = 2, 4, 6$, respectively. In Figs. 2.12a and 2.12c, in which observation noise is high, $\sigma = 20$, the error curves still converge to a constant value, but the convergence is much slower and longer trajectories are required ($> L + 1$). In Figs. 2.12b and 2.12d, where $p = 0.01, 0.02$, respectively, the error curves are permanently decreasing with increasing m and do not converge to a fixed value, even in the low-noise cases.

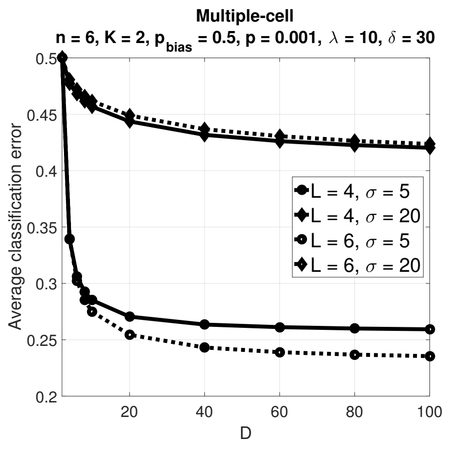
Figure 2.13 depicts the average errors versus D in multiple-cell scenarios, where there is no



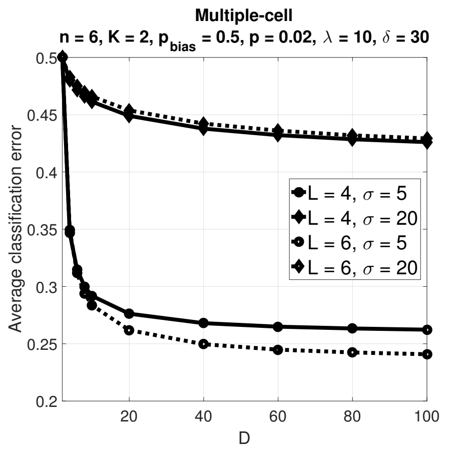
(a)



(b)



(c)

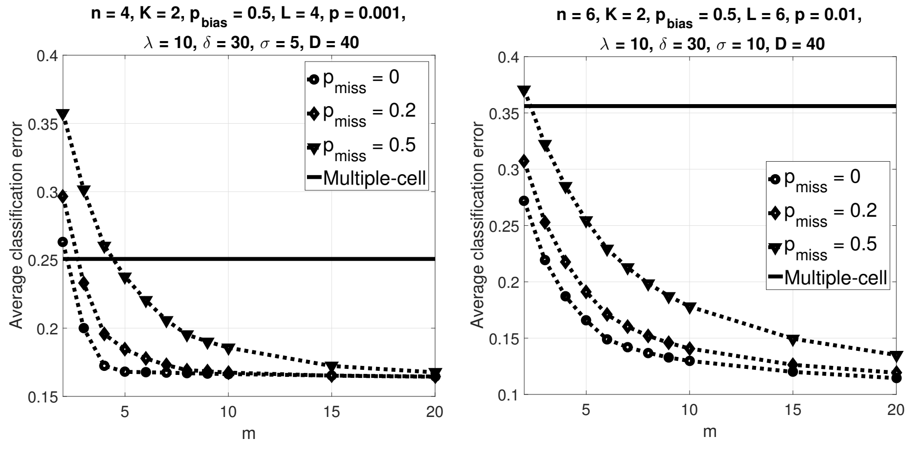


(d)

Figure 2.13: Average classification error of the multiple-cell classifier over 500 synthetic BNs versus D with $K = 2$ and $p_{bias} = 0.5$. Parameter values are $\lambda = 10$, $\delta = 30$, (a) $n = 4$, $p = 0.001$, (b) $n = 4$, $p = 0.01$, (c) $n = 6$, $p = 0.001$, (d) $n = 6$, $p = 0.02$. Reprinted with permission from [2], ©2019 IEEE.

trajectory data but only averaged expression data. Figures 2.13a and 2.13b show the error curves in 4-gene networks for $p = 0.001$ and $p = 0.01$, respectively, and different values of $L = 2, 4$ and $\sigma = 5, 20$. Figures 2.13c and 2.13d show similar results in 6-gene networks for $p = 0.001$ and $p = 0.02$, respectively, and different values of $L = 4, 6$ and $\sigma = 5, 20$. The error is higher in high-noise cases and it decreases to converge to a fixed value. The convergence rate depends on σ. When σ is low (high), the convergence is fast (slow).

Figure 2.14 shows average classification error versus m for different values of $p_{miss} = 0$ (no missing), $p_{miss} = 0.2$ (low missing probability), and $p_{miss} = 0.5$ (high missing probability). For



(a) (b)

Figure 2.14: Average classification error of the trajectory classifier and multiple-cell classifier over 500 synthetic BNs versus m with $K = 2$ and $p_{bias} = 0.5$. Parameter values are $\lambda = 10$, $\delta = 30$, $D = 40$, (a) $n = 4$, $L = 4$, $p = 0.001$, $\sigma = 5$, (b) $n = 6$, $L = 6$, $p = 0.01$, $\sigma = 10$. Reprinted with permission from [2], ©2019 IEEE.

the sake of comparison, we have also included in Fig. 2.14 the error of the multiple-cell scenarios for the same parameter values. Figure 2.14a shows the results for 4-gene networks when $L = 4$, $p = 0.001$, $\sigma = 5$, and $D = 40$. In this figure, smaller p_{miss} always yields a lower error rate for every value of m, but the differences decrease as m grows. The salient point of Fig. 2.14a is that one can always get a lower error rate by using the single-cell trajectory data, even with missing data, than by using the multiple-cell averaged data. In the case of Fig. 2.14a, the trajectory data with $p_{miss} = 0, 0.2, 0.5$ has lower error than multiple-cell data when $m \geq 3$, $m \geq 3$, $m \geq 5$, respectively. We previously mentioned that when $p \approx 0$ and σ is small, the error curves flatten out after $m = L + 1$; Fig. 2.14a shows that that is true when there are no missing data ($p_{miss} = 0$). With missing data, convergence is slow and longer trajectories are required to reach the converged error. Figure 2.14b shows similar results for 6-gene networks when $L = 6$, $p = 0.01$, $\sigma = 10$, and $D = 40$. Again, classification using trajectory data, even with high probability of missing data, can considerably lower the error rate as opposed to using multiple-cell averaged data.

2.2.6.3 *Real Network: Mammalian Cell-Cycle BN*

For an illustration using a real network, we use the wild-type mammalian cell-cycle BN, whose GRN [56] is shown in Fig. 2.15. This GRN has ten genes. The regulating functions are defined in Table 2.4 [56]. According to [56], one mutated situation is that the gene p27 is always off and cannot be activated. As a result, we derive the healthy BN from Table 2.4 and for the mutated/cancerous BN we put the value of p27 in Table 2.4 to zero, that is, $f_3 = 0$. This means that in the cancerous scenario the value of p27 does not obey the regulating functions and is always zero. The gene CycD is determined by extracellular signals. As we do not know the value of CycD, we have $M = 2$ candidate network functions for each of the healthy and mutated networks, which are corresponding to $f_1 = 0$ and $f_1 = 1$ in Table 2.4. If $f_1 = 0$, then the healthy and mutated networks have the singleton attractor cycles $\mathcal{A}_0 = \{389\}$ and $\mathcal{A}_1 = \{261\}$, respectively. If $f_1 = 1$, both the healthy and mutated networks have the same attractor cycle $\mathcal{A}_0 = \mathcal{A}_1 = \{516, 524, 527, 583, 613, 629, 561\}$. Consequently, the multiple-cell averaging method cannot classify the two networks when $f_1 = 1$. In the simulations, we assume that the trajectory expression data are generated from the networks with $f_1 = 0$ in both the healthy and mutated networks.

Figures 2.16a and 2.16b show the classification error of the trajectories of length $m = 6$ with $p = 0.05$ and two values of $p_{miss} = 0, 0.2$ versus D for low-noise ($\sigma = 5$) and high-noise ($\sigma = 20$) scenarios, respectively. A higher probability of missing data deteriorates classifier performance, as does higher observation noise σ. Convergence of the error curves is faster for lower σ. The classification error of the healthy and mutated mammalian cell-cycle networks when using averaged expression data in the multiple-cell scenario is shown in Figs. 2.16c and 2.16d for $\sigma = 5, 10, 20$, when $p = 0.01$ and $p = 0.05$, respectively. In Figs. 2.16c and 2.16d, the classifier based on the multiple-cell expression data can only work well in the low-noise scenarios and is very susceptible to observation noise. Convergence of the error curves versus D gets slower as σ increases. For a given σ, the error of the multiple-cell classifier decreases with decreasing p, the reason being that larger p makes the steady-state distributions of the healthy and mutated cell-cycle BNs similar to

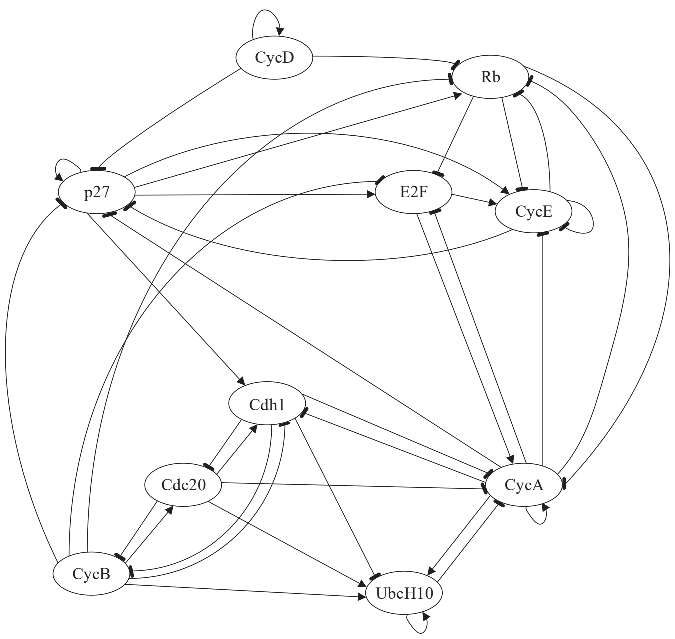


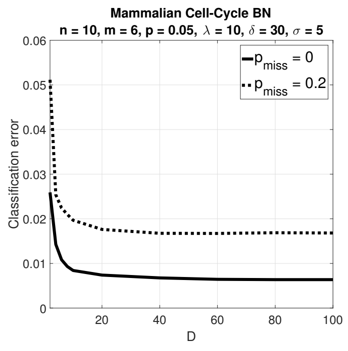
Figure 2.15: Mammalian cell-cycle gene regulatory network. Reprinted with permission from [2], ©2019 IEEE.

Table 2.4: Definitions of Boolean functions for the wild-type mammalian cell-cycle BN with 10 genes. Reprinted with permission from [2], ©2019 IEEE.

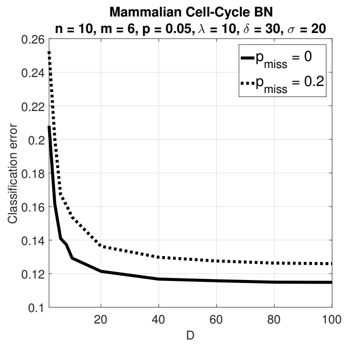
Order	Gene	Regulating function
x_1	$CycD$	$f_1 =$ Extracellular signals
x_2	Rb	$f_2 = (\overline{CycD} \wedge \overline{CycE} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{CycD} \wedge \overline{CycB})$
x_3	$p27$	$f_3 = (\overline{CycD} \wedge \overline{CycE} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{(CycE \wedge CycA)} \wedge \overline{CycD} \wedge \overline{CycB})$
x_4	$E2F$	$f_4 = (\overline{Rb} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{Rb} \wedge \overline{CycB})$
x_5	$CycE$	$f_5 = (E2F \wedge \overline{Rb})$
x_6	$CycA$	$f_6 = (E2F \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{(Cdh1 \wedge UbcH10)}) \vee (CycA \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{(Cdh1 \wedge UbcH10)})$
x_7	$Cdc20$	$f_7 = CycB$
x_8	$Cdh1$	$f_8 = (\overline{CycA} \wedge \overline{CycB}) \vee Cdc20 \vee (p27 \wedge \overline{CycB})$
x_9	$UbcH10$	$f_9 = \overline{Cdh1} \vee (Cdh1 \wedge UbcH10 \wedge (Cdc20 \vee CycA \vee CycB))$
x_{10}	$CycB$	$f_{10} = (\overline{Cdc20} \wedge \overline{Cdh1})$

each other, so that the means of two normal distributions for the two classes get closer to each other, leading to a larger error.

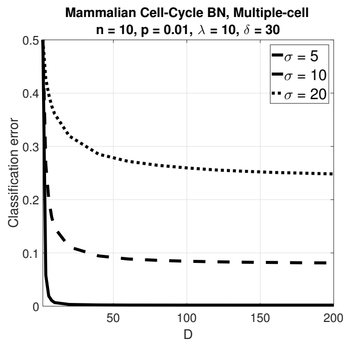
Figures 2.16e and 2.16f present the error versus m in the mammalian cell-cycle networks for



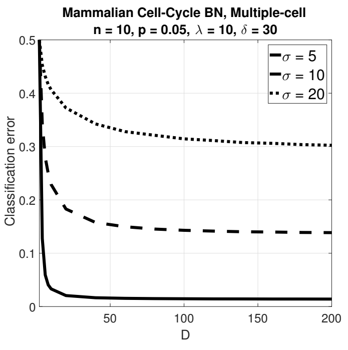
(a)



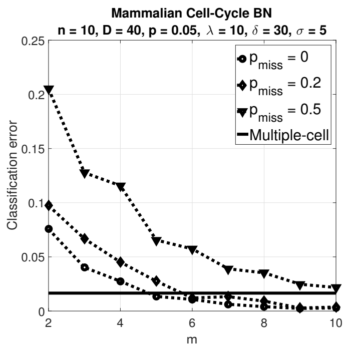
(b)



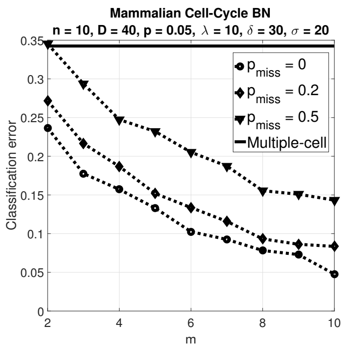
(c)



(d)



(e)



(f)

Figure 2.16: Classification errors of the trajectory and multiple-cell classifiers in the mammalian cell-cycle BN. The fixed parameters are $n = 10$, $\lambda = 10$, $\delta = 30$. (a) Classification error of the trajectory classifier versus D. The parameters are $m = 6$, $p = 0.05$, $\sigma = 5$, (b) Classification error of the trajectory classifier versus D. The parameters are $m = 6$, $p = 0.05$, $\sigma = 20$, (c) Classification error of the multiple-cell classifier versus D. The parameter is $p = 0.01$, (d) Classification error of the multiple-cell classifier versus D. The parameter is $p = 0.05$, (e) Classification error of the trajectory and multiple-cell classifiers versus m. The parameters are $D = 40$, $p = 0.05$, $\sigma = 5$, (f) Classification error of the trajectory and multiple-cell classifiers versus m. The parameters are $D = 40$, $p = 0.05$, $\sigma = 20$. Reprinted with permission from [2], ©2019 IEEE.

$p = 0.05$, $D = 40$ and different values of $p_{miss} = 0, 0.2, 0.5$ when $\sigma = 5$ and $\sigma = 20$, respectively. Similar to the synthetic networks, the error curves have a decreasing trend as m increases. In Figs.

2.16e and 2.16f, the error of the multiple-cell classifier is shown for the same parameter values. In the low-noise scenario of Fig. 2.16e, the error of the multiple-cell classifier is better than that of the trajectory classifier when m is small, the extent depending on the probability of missing data in the trajectory data; however, for longer trajectories (larger m), the error of the trajectory classifier is less. We observe in Fig. 2.16f that the performance of the multiple-cell classifier is very bad in the high-noise scenario, the error of the multiple-cell classifier being much greater than that of the trajectory classifier for every value of m and even for high p_{miss}.

2.2.7 Conclusion

This section studied classification of gene-expression trajectories coming from two classes, healthy and mutated (cancerous) using Boolean networks with perturbation (BNps) to model the dynamics of each class at the state level, meaning that each class has its own BNp, which we partially know based on gene pathways. We employed a Gaussian model at the observation level to show the expression values of the genes given the hidden states at each time point. We used the expectation maximization (EM) methodology to learn the BNps and the unknown model parameters, derived closed-form updates for the parameters, and proposed a learning algorithm. After learning, a plug-in Bayes classifier was used to classify the unlabeled trajectories. The effect of missing data was also considered.

From the biological perspective, measuring gene expressions at different times yields trajectories only when the measurements come from a single cell. In multiple-cell scenarios, the expression values of the genes are averages over many cells with possibly different states. Using the central-limit theorem, we proposed another model for expression data in multiple-cell scenarios. Using simulations, it was demonstrated that single-cell trajectory data can outperform multiple-cell average expression data in terms of the classification error, especially in high-noise situations.

2.3 Intrinsically Robust Bayesian Classification of Gene Expression Trajectories

2.3.1 Overview

In this section we assume a partially known Gaussian observation model belonging to an uncertainty class of models. We derive the intrinsically Bayesian robust classifier to discriminate between wild-type and mutated networks based on expression trajectories. The classifier minimizes the expected error across the uncertainty class relative to the prior distribution. We test it using a mammalian cell-cycle model, discriminating between the normal network and one in which gene p27 is mutated, thereby producing a cancerous phenotype. Tests examine all model aspects, including trajectory length, perturbation probability, and the hyperparameters governing the prior distribution over the uncertainty class.

2.3.2 Introduction

Genes have interactions with each other, which can determine how they are behaving over time and define the dynamics of gene regulatory networks (GRNs). One way of showing the dynamics of GRNs over discrete time points is Boolean networks with perturbation (BNp) [33]. A BNp is a Markov chain, in which the state of a gene (0 for off and 1 for on) at the current time is a function of the states of its predictor genes at the previous time plus a small random Boolean noise.

Suppose we have single-cell measurements sampled with a sufficient rate to detect regulatory timing. In effect, this would mean that classification would be done on data reflecting an underlying gene regulatory network. In section 2.1 we proposed a classifier to classify the state trajectories of the two classes: wild-type and mutated, each having its own BNp. We derived the Bayes classifier and computed the Bayes error. We analyzed the effects of the length of the trajectories, perturbation probability, and different mutations on the Bayes error.

In section 2.2 we assumed an observation model on the state dynamics of the BNps, from which the expression values of the genes are obtained. As the parameters of the model were all unknown and the network functions were partially known, we proposed an expectation-maximization (EM)-based algorithm to estimate these parameters and functions, and then plugged them into the Bayes classifier. In 2.2 we assumed that the expression trajectory data come from single-cell measurements and compared that with a multiple-cell scenario, in which instead of trajectories we have the averaged expressions of all genes over all cells, which translates to an average over all states.

In this section, we extend the single-cell trajectory classification to the Bayesian framework. We propose the intrinsically Bayesian robust (IBR) classifier for the trajectorires [3]. The IBR classifier is a specific type of the obtimal Bayesian classifier (OBC), first introduced in [57, 58] for the classification of static data. In fact, the difference between the OBC and IBR classifiers is that in the OBC the expectation of the class-conditional densities is taken over the posteriors of the parameters to obtain the effective class-conditional densities, whereas in the IBR classifier the expectation is taken over the priors. The IBR/OBC concept has been applied to linear and morphological filtering [59, 60], and IBR Kalman filtering [61]. Regarding the prior distributions, prior construction methods using the pathway knowledge have been studied in the literature, such as [62, 63].

Here we apply the IBR classifier to the classification of trajectories. As opposed to section 2.2, where we estimated the parameters, here we assume that the parameters belong to an uncertainty class governed by a prior distribution. We assume that there are two classes: wild-type ($S = 0$) and mutated ($S = 1$). We introduce a Bayesian version of the partially observed Boolean dynamical system (POBDS), proposed in [41], as the observation model. We use a beta prior distribution for the prior probability of the class $S = 0$ and also for the gene perturbation probability. Since the observation model given the states is Gaussian, we employ the normal-gamma distribution as the prior distribution of the mean and precision (inverse of variance) of the Gaussian model.

In the simulation part, we employ a mammalian cell-cycle gene regulatory network [56] consisting of 10 genes as the BNp for the class $S = 0$ and its mutated version as the BNp for the class $S = 1$. We analyze the effects of all the hyperparameters of the model and the length of the observed trajectories on the classification error. The proposed classifier is computationally efficient because we use the sum-product method to reduce the complexity to $m \times 2^n$, where m is the length of the observed trajectory and n is the number of genes in the network. Being linearly dependent on time points, m, makes the classifier very fast even for longer trajectories.

2.3.3 Methods

In a Boolean network (BN) for n genes, each gene value $x_i \in \{0,1\}$, for $i = 1, \cdots, n$, at time $k+1$ is determined by the values of some predictor genes at time k via a Boolean function $f_i : \{0,1\}^n \to \{0,1\}$. In practice, f_i is a function of a small number of genes, K_i, called the *in-degree* of the gene x_i in the network. The in-degree of the network is $K = \max_{i=1,\cdots,n} K_i$. A gene network can be represented as a graph with vertices representing genes and edges representing regulations. There is a state diagram of 2^n states corresponding to the truth table of the BN, representing the dynamics of the network. Given an initial state, a BN will eventually reach a set of states, called an *attractor cycle*, through which it will cycle endlessly. Each initial state corresponds to a unique attractor cycle, and the set of initial states leading to a specific attractor cycle is known as the *basin of attraction* (BOA) of the attractor cycle.

2.3.3.1 State Model

We allow stochasticity in our state model by using Boolean networks with perturbation (BNps) instead of deterministic BNs. For BNps, perturbation is introduced with a probability p_k by which the state of a gene in the network can be randomly flipped at time k. We assume that there is an independent identically distributed (i.i.d.) random perturbation vector at each time k, denoted by $\mathbf{n}_k \in \{0,1\}^n$, such that the i-th gene flips at time k if the i-th component of $\mathbf{n}_k$ is equal to 1. Therefore, the dynamical model can be expressed as

$$\mathbf{X}_{k+1} = \mathbf{f}(\mathbf{X}_k) \oplus \mathbf{n}_{k+1}, \qquad k = 0, 1, 2, \cdots, \tag{2.128}$$

where $\mathbf{X}_k = [x_1(k), x_2(k), \cdots, x_n(k)]^T$ is a binary state vector, called a *gene activity profile* (GAP), at time k, in which $x_i(k)$ indicates the expression level of the i-th gene at time k (either 0 or 1); $\mathbf{f} = [f_1, f_2, \cdots, f_n]^T : \{0,1\}^n \to \{0,1\}^n$ is the vector of the network functions, in which f_i shows the expression level of the i-th gene at time $k+1$ when the system lies in the state $\mathbf{X}_k$ at time k; $\mathbf{n}_k = [n_1(k), n_2(k), \cdots, n_n(k)]^T$ is the perturbation vector at time k, in which

$n_1(k), n_2(k), \cdots, n_n(k)$ are i.i.d. Bernoulli random variables for every k with the parameter $p_k = P(n_i(k) = 1)$ for $i = 1, \cdots, n$; and $\oplus$ is component-wise modulo 2 addition.

The existence of perturbation makes the corresponding Markov chain of a BNp irreducible. Hence, the network possesses a steady-state distribution π describing its long-run behavior. If p_k is sufficiently small, π will reflect the attractor structure within the original network. We can derive the transition probability matrix (TPM) if we know the truth table and the perturbation probability of a BNp. As a result, the steady-state distribution π can be computed as well.

• Prior for state parameter: We assume that we know the underlying Boolean networks for both the wild-type and mutated classes, and the only uncertain parameter at the state level is the perturbation probability p_k. Since $0 < p_k < 1$, we can employ the beta prior for p_k, for all $k = 1, 2, \cdots$, as

$$g(p_k) \sim \text{Beta}(a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p_k^{a-1}(1-p_k)^{b-1}, \tag{2.129}$$

where a and b are known parameters. Since in reality p_k is close to zero, we can choose a and b in such a way that this fact is satisfied. To this end, we can use the mean and variance of p_k:

$$\text{E}[p_k] = \frac{a}{a+b}, \quad \text{Var}[p_k] = \frac{ab}{(a+b)^2(a+b+1)}. \tag{2.130}$$

2.3.3.2 Observation Model

We define a Bayesian partially-observed Boolean dynamical system (BPOBDS) as the model for the gene expression data. In this model, we assume that the gene expressions come from Gaussian distributions whose parameters are governed by prior distributions whose parameters (the hyperparameters of the observations) are a function of the hidden Boolean states. If $y_j(k)$ is the expression value of the j-th gene at time k, then the observation model is

$$p\left(y_j(k) | \theta_j(k), \lambda_j(k)\right) \sim \mathcal{N}(\theta_j(k), \lambda_j(k)^{-1}), \tag{2.131}$$

for $j = 1, 2, \cdots, n$ and $k = 1, 2, \cdots$, where $\theta_j(k)$ and $\lambda_j(k)$ denote the mean and precision, respectively, of the Gaussian distribution.

• Priors for observation parameters:

We employ the well-known normal-gamma prior distribution for $\theta_j(k)$ and $\lambda_j(k)$:

$$\begin{aligned} p(\lambda_j(k)) &\sim \text{Gamma}(\alpha_0, \beta_0), \\ p(\theta_j(k)|\lambda_j(k), x_j(k)) &\sim \mathcal{N}\left(\mu_j(k), (\kappa_0\lambda_j(k))^{-1}\right), \\ \text{where } \mu_j(k) &= \mu_0 + \delta_0 x_j(k), \end{aligned} \tag{2.132}$$

where α_0, β_0, κ_0, μ_0, and δ_0 are known positive hyperparameters, and $x_j(k)$ is the hidden Boolean state of gene j at time k. The intuition behind the prior (2.132) is that when gene j at time k is on or off, that is, $x_j(k) = 1$ or 0, the hyper-mean of the expression for that gene is $\mu_j(k) = \mu_0 + \delta_0$ or $\mu_j(k) = \mu_0$, respectively, at time k. In (2.132), μ_0 is the baseline expression level and δ_0 is the expression coefficient. The hyperparameters α_0, β_0, and κ_0 determine the level of uncertainty, by which we can control the variance of the outputs. We assume the same values of hyperparameters for all genes at all times.

2.3.3.3 IBR Classifier

If one knows the feature-label distribution, then the error of any classifier can be found and an optimal (Bayes) classifier minimizes classifier error. If the feature-label distribution is unknown but belongs to an uncertainty class Θ of feature-label distributions, then we desire a classifier to minimize the expected error over the uncertainty class. Given a classifier ψ, from the perspective of mean-square error (MSE), the best error estimate minimizes the MSE between its true error (a function of parameter θ) and an error estimate. This Bayesian minimum-mean-square-error (MMSE) estimate is given by the expected true error, $\widehat{\varepsilon}(\psi) = \mathrm{E}_\theta[\varepsilon(\psi, \theta)]$, where $\varepsilon(\psi, \theta)$ is the error of ψ on the feature-label distribution parameterized by θ and the expectation is taken relative to the prior distribution $\pi(\theta)$ [64].

An IBR classifier minimizes the Bayesian MMSE estimate. If $\psi(\mathbf{x}) = 0$ if $\mathbf{x} \in R_0$ and

$\psi(\mathbf{x}) = 1$ if $\mathbf{x} \in R_1$, where $\mathbf{x}$ is a multidimensional vector of data, and R_0 and R_1 partition the sample space, then [57]

$$\widehat{\varepsilon}(\psi) = \mathrm{E}_\pi[c] \int_{R_1} f_\Theta(\mathbf{x}|0)\, d\mathbf{x} + (1 - \mathrm{E}_\pi[c]) \int_{R_0} f_\Theta(\mathbf{x}|1)\, d\mathbf{x},$$

where

$$f_\Theta(\mathbf{x}|y) = \int_{\boldsymbol{\Theta}_y} f_{\theta_y}(\mathbf{x}|y)\, \pi(\theta_y)\, d\theta_y$$

is the *effective class-conditional density* for class y, $\boldsymbol{\Theta}_y$ being the space for θ_y, $f_{\theta_y}(\mathbf{x}|y)$ is the class-conditional density, and c is the prior probability of the class 0. The IBR classifier is given by [57]

$$\psi_{\mathrm{IBR}}(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad \mathrm{E}_\pi[c] f_\Theta(\mathbf{x}|0) \geq \\ & \quad (1 - \mathrm{E}_\pi[c]) f_\Theta(\mathbf{x}|1) \\ 1 & \text{otherwise} \end{cases}. \tag{2.133}$$

2.3.3.4 Trajectory-based IBR Classifier

Let $\Theta_s = [p_{2:m}, \theta_{1:n}(1:m), \lambda_{1:n}(1:m)]$ denote the parameters of the class $S = s$, for $s = 0, 1$, where $p_{2:m}$ means the parameters $p_2, p_3, \cdots, p_m$, and similarly for $\theta_{1:n}(1:m)$ and $\lambda_{1:n}(1:m)$. Furthermore, let $\mathbf{f}_s$ denote the Boolean network function of the class $S = s$ and $\mathcal{X} = [\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_m]$ denote the Boolean state trajectory at m consecutive times points at which $\mathcal{Y}$ has been observed. The $n \times 1$ Boolean vector $\mathbf{X}_k = [x_1(k), x_2(k), \cdots, x_n(k)]^T$ has the states of the n genes at time k, which are hidden and not observed.

Suppose that we obtain the expressions of n genes at m consecutive time points. Let $\mathbf{Y}_k = [y_1(k), \cdots, y_n(k)]^T$ denote the $n \times 1$ expression vector of n genes at time k, and $\mathcal{Y} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_m]$ denote a time trajectory of length m, containing the expression vectors at the m consecutive times. The problem is to optimally classify this observed trajectory $\mathcal{Y}$ to the class 0 (wild-type) or class 1 (mutated). Let c and $1-c$ be the prior probabilities of the class $S = 0$ and class $S = 1$, respectively.

Since we are uncertain about c, we use a beta prior

$$g(c) \sim \text{Beta}(a_c, b_c) = \frac{\Gamma(a_c + b_c)}{\Gamma(a_c)\Gamma(b_c)} c^{a_c - 1}(1 - c)^{b_c - 1}, \tag{2.134}$$

with mean

$$\text{E}[c] = \frac{a_c}{a_c + b_c}, \tag{2.135}$$

where a_c and b_c are known parameters.

According to (2.133), the IBR classifier for the trajectories is

$$\psi_{\text{IBR}}(\mathcal{Y}) = \begin{cases} 0 & \text{if } \; \text{E}[c] p(\mathcal{Y}|S = 0) \geq \\ & \qquad (1 - \text{E}[c]) p(\mathcal{Y}|S = 1) \\ 1 & \text{otherwise} \end{cases}, \tag{2.136}$$

where $p(\mathcal{Y}|S = s)$ is the effective class-conditional density of the trajectory $\mathcal{Y}$ in the class $S = s$ for $s = 0, 1$.

• Effective Class-Conditional Densities of Trajectories: The joint distribution of $\mathcal{Y}$, $\mathcal{X}$, and Θ_s given the class $S = s$ can be factorized as

$$\begin{aligned} p(\mathcal{Y}, \mathcal{X}, \Theta_s|S = s) = {} & g(p_{2:m}) P\left(\mathcal{X}|p_{2:m}, S = s\right) \\ & \times p\left(\mathcal{Y}|\theta_{1:n}(1:m), \lambda_{1:n}(1:m)\right) \\ & \times p\left(\theta_{1:n}(1:m), \lambda_{1:n}(1:m)|\mathcal{X}\right). \end{aligned} \tag{2.137}$$

In deriving (2.137), it is assumed that the state parameters $\{p_{2:m}\}$ are independent of the observation parameters $\{\theta_{1:n}(1:m), \lambda_{1:n}(1:m)\}$. Due to the independence assumption in the priors,

$$\begin{aligned} g(p_{2:m}) &= \prod_{k=1}^{m-1} g(p_{k+1}) \qquad (2.138) \\ &= \prod_{k=1}^{m-1} \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} p_{k+1}^{a-1}(1 - p_{k+1})^{b-1}, \end{aligned}$$

$$
\begin{aligned}
& p\left(\theta_{1:n}(1:m), \lambda_{1:n}(1:m)|\mathcal{X}\right) \\
&= \prod_{k=1}^{m}\prod_{j=1}^{n} p(\theta_j(k)|\lambda_j(k), x_j(k))p(\lambda_j(k)) \\
&= \prod_{k=1}^{m}\prod_{j=1}^{n} \frac{1}{\sqrt{2\pi(\kappa_0\lambda_j(k))^{-1}}}\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)} \\
&\times \exp\left(-\frac{(\theta_j(k)-\mu_j(k))^2}{2(\kappa_0\lambda_j(k))^{-1}}\right)\lambda_j(k)^{\alpha_0-1}\exp(-\beta_0\lambda_j(k)).
\end{aligned}
\tag{2.139}
$$

If we assume that the conditional expressions, given the parameters, of the n genes at m time points are independent, the likelihood of $\mathcal{Y}$ given the parameters in (2.137) can be written as

$$
\begin{aligned}
& p\left(\mathcal{Y}|\theta_{1:n}(1:m), \lambda_{1:n}(1:m)\right) \\
&= \prod_{k=1}^{m}\prod_{j=1}^{n} p\left(y_j(k)|\theta_j(k), \lambda_j(k)\right) \\
&= \prod_{k=1}^{m}\prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\lambda_j(k)^{-1}}}\exp\left(-\frac{(y_j(k)-\theta_j(k))^2}{2\lambda_j(k)^{-1}}\right).
\end{aligned}
\tag{2.140}
$$

We should note that the independency assumption in (2.140) is only for the observations, whereas the genes have interactions at the state level, following the underlying Boolean network, so that they cannot be considered independent. Due to the Markov property in (2.128), $p\left(\mathcal{X}|p_{2:m}, S=s\right)$ in (2.137) can be factored as

$$
\begin{aligned}
& P\left(\mathcal{X}|p_{2:m}, S=s\right) = \\
& P(\mathbf{X}_1|S=s)\prod_{k=1}^{m-1} P(\mathbf{X}_{k+1}|\mathbf{X}_k, p_{k+1}, S=s),
\end{aligned}
\tag{2.141}
$$

where $P(\mathbf{X}_{k+1}|\mathbf{X}_k, p_{k+1}, S=s)$ is the probability of transitioning from state $\mathbf{X}_k$ at time k to state $\mathbf{X}_{k+1}$ at time $k+1$, given the perturbation probability p_{k+1}, in the class $S=s$, and $P(\mathbf{X}_1|S=s)$ is the probability of the first state $\mathbf{X}_1$ in the class $S=s$. Let $\mathbf{x}^i$ denote the $n\times 1$ Boolean vector of the state i, for $i=1,2,\cdots,2^n$. Given the perturbation probability p_{k+1}, the conditional transition probability matrix (TPM) at time $k+1$, which is a $2^n\times 2^n$ matrix, in the class $S=s$ can be derived from (2.128) as

$$\begin{aligned}\mathbf{A}^{(s)}_{i,j}(k+1) &= P(\mathbf{X}_{k+1}=\mathbf{x}^j|\mathbf{X}_k=\mathbf{x}^i, p_{k+1}, S=s)\\ &= p_{k+1}^{\mathbf{d}(\mathbf{x}^j,\mathbf{f}_s(\mathbf{x}^i))}(1-p_{k+1})^{n-\mathbf{d}(\mathbf{x}^j,\mathbf{f}_s(\mathbf{x}^i))},\end{aligned} \tag{2.142}$$

where $\mathbf{d}(\mathbf{x}^j, \mathbf{f}_s(\mathbf{x}^i))$ is the Hamming distance between the two Boolean vectors $\mathbf{x}^j$ and $\mathbf{f}_s(\mathbf{x}^i)$.

For obtaining $p(\mathcal{Y}|S=s)$, we need to integrate out the joint distribution $p(\mathcal{Y}, \mathcal{X}, \Theta_s|S=s)$ in (2.137) with respect to Θ_s and $\mathcal{X}$:

$$\begin{aligned}p(\mathcal{Y}|S=s) &= \sum_{\mathcal{X}}\int_{\Theta_s} p(\mathcal{Y},\mathcal{X},\Theta_s|S=s) \qquad (2.143)\\ &= \sum_{\mathcal{X}}\left\{P(\mathbf{X}_1|S=s)\prod_{k=1}^{m-1}\int_{p_{k+1}} g(p_{k+1})p_{k+1}^{\mathbf{d}(\mathbf{X}_{k+1},\mathbf{f}_s(\mathbf{X}_k))}\right.\\ &\quad \times(1-p_{k+1})^{n-\mathbf{d}(\mathbf{X}_{k+1},\mathbf{f}_s(\mathbf{X}_k))}dp_{k+1}\\ &\quad \times\prod_{k=1}^{m}\prod_{j=1}^{n}\int_{\theta_j(k)}\int_{\lambda_j(k)} p(y_j(k)|\theta_j(k),\lambda_j(k))\\ &\quad \left.\times p(\theta_j(k)|\lambda_j(k),x_j(k))\; p(\lambda_j(k))\; d\theta_j(k)d\lambda_j(k)\right\}.\end{aligned}$$

Fortunately, as the priors are conjugate, we can analytically solve the integrals in (2.143). We will use the following lemmas to do so.

Lemma 1. *Let* $\mathbf{P}=(0\ 1)$ *be the domain of* p_{k+1}*. The following equation holds:*

$$\begin{aligned}K_1 &\triangleq \int_{\mathbf{P}} g(p_{k+1})p_{k+1}^{\mathbf{d}(\mathbf{X}_{k+1},\mathbf{f}_s(\mathbf{X}_k))}\\ &\quad\times(1-p_{k+1})^{n-\mathbf{d}(\mathbf{X}_{k+1},\mathbf{f}_s(\mathbf{X}_k))}dp_{k+1}\\ &= \frac{\Gamma(\mathbf{d}(\mathbf{X}_{k+1},\mathbf{f}_s(\mathbf{X}_k))+a)\Gamma(n-\mathbf{d}(\mathbf{X}_{k+1},\mathbf{f}_s(\mathbf{X}_k))+b)}{\Gamma(a)\Gamma(b)\Gamma(a+b+n)\Gamma(a+b)^{-1}}.\end{aligned} \tag{2.144}$$

Proof. See Appendix A.1. □

We assume that the observations $\mathcal{Y}$ occur in the steady state. As a result, in (2.143), $P(\mathbf{X}_1|S=s)$ is the steady-state probability of the state $\mathbf{X}_1$ in the class $S=s$, for $s=0,1$. The following lemma gives the steady-state distribution.

Lemma 2. *Let $\pi_i^{(s)} = P(\mathbf{X}_1 = \mathbf{x}^i | S = s)$ denote the steady-state probability of the i-th state in the class $S = s$, and $\pi^{(s)} = [\pi_1^{(s)}, \cdots, \pi_{2^n}^{(s)}]$ be the 1×2^n vector of the steady-state distribution. Then $\pi^{(s)}$ can be calculated from*

$$\pi^{(s)} = \pi^{(s)} \mathbf{M}^{(s)}, \qquad \sum_{i=1}^{2^n} \pi_i^{(s)} = 1, \tag{2.145}$$

where $\mathbf{M}^{(s)}$ is the transition probability matrix of the class $S = s$ with the entries

$$\mathbf{M}_{i,j}^{(s)} = \frac{\Gamma(\mathbf{d}(\mathbf{x}^j, \mathbf{f}_s(\mathbf{x}^i)) + a)\Gamma(n - \mathbf{d}(\mathbf{x}^j, \mathbf{f}_s(\mathbf{x}^i)) + b)}{\Gamma(a)\Gamma(b)\Gamma(a+b+n)\Gamma(a+b)^{-1}}. \tag{2.146}$$

Proof. See Appendix A.2. □

Lemma 3. *Let $\mathbf{\Omega} = (-\infty\ \infty)$ and $\mathbf{\Lambda} = (0\ \infty)$ be the domains of $\theta_j(k)$ and $\lambda_j(k)$, respectively, for $j = 1, \cdots, n$, and $k = 1, \cdots, m$. The following equation holds:*

$$\begin{aligned} K_2 &\triangleq \int_{\mathbf{\Omega}} \int_{\mathbf{\Lambda}} p(y_j(k)|\theta_j(k), \lambda_j(k)) \\ &\times p(\theta_j(k)|\lambda_j(k), x_j(k))\ p(\lambda_j(k))\ d\theta_j(k) d\lambda_j(k) \\ &= \frac{1}{(2\pi)^{\frac{1}{2}}} \left(\frac{\kappa_0}{\kappa_1}\right)^{\frac{1}{2}} \frac{\Gamma(\alpha_1)}{\Gamma(\alpha_0)} \frac{\beta_0^{\alpha_0}}{\beta_1^{\alpha_1}}, \end{aligned} \tag{2.147}$$

where

$$\begin{aligned} \kappa_1 &= \kappa_0 + 1, \\ \alpha_1 &= \alpha_0 + \frac{1}{2}, \\ \beta_1 &= \beta_0 + \frac{\kappa_0 (y_j(k) - \mu_0 - \delta_0 x_j(k))^2}{2(\kappa_0 + 1)}. \end{aligned} \tag{2.148}$$

Proof. See Appendix A.3. □

- Summing out $\mathcal{X}$:

Using (2.143), (1), (2.146), and (2.147), we have

$$p(\mathcal{Y}|S=s) = \sum_{\mathbf{X}_1} \cdots \sum_{\mathbf{X}_m} \Bigg\{ \pi_{\mathbf{X}_1}^{(s)}$$
$$\prod_{k=1}^{m-1} \frac{\Gamma(\mathbf{d}(\mathbf{X}_{k+1}, \mathbf{f}_s(\mathbf{X}_k)) + a)\Gamma(n - \mathbf{d}(\mathbf{X}_{k+1}, \mathbf{f}_s(\mathbf{X}_k)) + b)}{\Gamma(a)\Gamma(b)\Gamma(a+b+n)\Gamma(a+b)^{-1}}$$
$$\times \prod_{k=1}^{m} \prod_{j=1}^{n} \frac{1}{(2\pi)^{\frac{1}{2}}} \left(\frac{\kappa_0}{\kappa_1}\right)^{\frac{1}{2}} \frac{\Gamma(\alpha_1)}{\Gamma(\alpha_0)} \frac{\beta_0^{\alpha_0}}{\beta_1^{\alpha_1}} \Bigg\}. \tag{2.149}$$

We use the sum-product algorithm [53] to efficiently compute the summation in (2.149). Define the $2^n \times 1$ vector $\Phi(k)$ by

$$\Phi_i(k) = \left(\frac{\beta_0^{\alpha_0}}{(2\pi)^{\frac{1}{2}}} \left(\frac{\kappa_0}{\kappa_0+1}\right)^{\frac{1}{2}} \frac{\Gamma(\alpha_0 + \frac{1}{2})}{\Gamma(\alpha_0)} \right)^n$$
$$\times \prod_{j=1}^{n} \left[\beta_0 + \frac{\kappa_0 (y_j(k) - \mu_0 - \delta_0 \mathbf{x}_j^i)^2}{2(\kappa_0+1)} \right]^{-(\alpha_0 + \frac{1}{2})}, \tag{2.150}$$

for $i = 1, \cdots, 2^n$, where $\mathbf{x}_j^i$ is the j-th entry of the Boolean state $\mathbf{x}^i$. We define an auxiliary $2^n \times 1$ vector $\Pi^{(s)}(k)$ at the time k, for $k = 1, \cdots, m$, which is initialized and updated as follows:

$$\begin{aligned} \Pi^{(s)}(1) &= (\pi^{(s)})^T \circ \Phi(1), \\ \Pi^{(s)}(k+1) &= [\mathbf{M}^{(s)^T} \Pi^{(s)}(k)] \circ \Phi(k+1), \end{aligned} \tag{2.151}$$

for $k = 1, \cdots, m-1$, where T denotes the transpose, and $\circ$ is the Hadamard product. Once we have calculated $\Pi^{(s)}(m)$, the summation of (2.149) is equal to the l_1 norm $\parallel \Pi^{(s)}(m) \parallel_1$, which is the summation of the all 2^n entries of $\Pi^{(s)}(m)$. Therefore, (2.149) can be written as

$$p(\mathcal{Y}|S=s) = \parallel \Pi^{(s)}(m) \parallel_1 . \tag{2.152}$$

2.3.4 Results and Discussion

In this section, we consider a mammalian cell-cycle gene regulatory network [56], depicted in Figure 2.15, for evaluating our proposed trajectory-based IBR classifier. This GRN consists of $n = 10$ genes, whose interactions are shown in Figure 2.15. The Boolean functions of the corresponding Boolean network for this GRN are given in Table 2.4 [56]. We define class $S = 0$ as the wild-type class, whose network function $\mathbf{f}_0$ is in Table 2.4. According to [56], one mutated case which leads to cancer is when the gene p27 in the network is shut down and cannot be activated by its regulating genes, that is, $f_3 = 0$. Therefore, we define class $S = 1$ as the mutated class with the network function $\mathbf{f}_1$, which is the same as Table 2.4 with $f_3 = 0$. We analyze the effects of the hyperparameters and m on the classification error in Figures 2.17-2.24. In the simulations, we have set $a_c = b_c = 10$ and $\mu_0 = 10$. Therefore, the prior probability c of the class $S = 0$ will have the mean value $\mathrm{E}[c] = 0.5$.

Figure 2.17 shows the classification error versus m for $a = 1, 3, 5$, when the values of the other hyperparameters are $b = 100$, $\alpha_0 = 100$, $\beta_0 = 10^4$, $\delta_0 = 40$, and $\kappa_0 = 100$. We see that the classification error decreases by increasing m and converges to zero. This means that for long enough trajectories we can have perfect classification. The value of the hyperparameter a determines the amount of uncertainty for the gene perturbation probability p_k at time k. From a biological perspective, we know that p_k should be small. As a result, we have chosen $b = 100$, and $a = 1, 3, 5$, leading to the mean values of p_k, respectively, as $\mathrm{E}[p_k] = \frac{a}{a+b} \approx 0.01, 0.03, 0.05$. For a given b, the bigger value of a allows a wider range of p_k, which results in higher classification error.

Figure 2.18 represents the classification error versus m for different values of α_0 and β_0, when $a = 1$, $b = 100$, $\delta_0 = 40$, and $\kappa = 100$. As the precision (inverse of variance) has a Gamma(α_0, β_0) distribution, its mean and variance are equal to $\mathrm{E}[\lambda] = \frac{\alpha_0}{\beta_0}$ and $\mathrm{Var}[\lambda] = \frac{\alpha_0}{\beta_0^2} = \frac{\mathrm{E}[\lambda]}{\beta_0}$. Figure 2.18 shows the error curves for the two cases $\frac{\alpha_0}{\beta_0} = 10^{-3}$ and 2×10^{-3}, each having a different value of β_0, leading to a different variance of λ. As such, we can see the effects of both the mean and variance of the precision λ. Whenever $\frac{\alpha_0}{\beta_0}$ increases, there is lower variance in the outputs, which

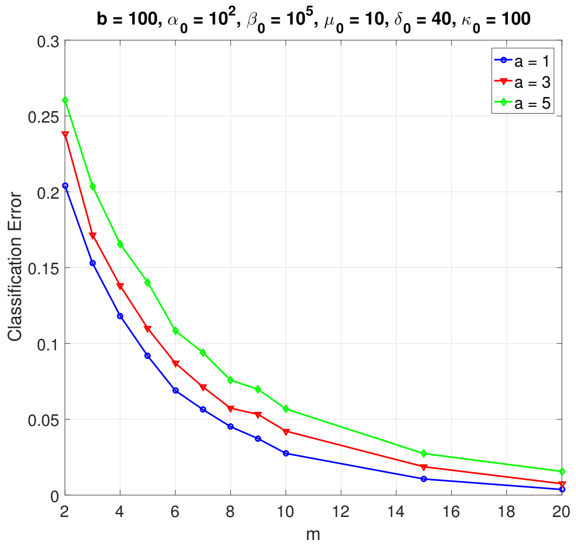


Figure 2.17: Classifier error versus m in cell-cycle network. Reprinted with permission from [3], ©2018 BMC.

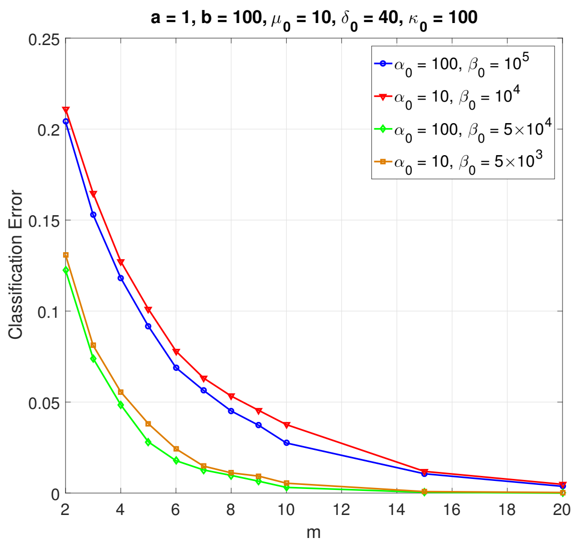


Figure 2.18: Classifier error versus m in cell-cycle network. Reprinted with permission from [3], ©2018 BMC.

results in lower errors. We also notice from Figure 2.18 that for a given value of $\frac{\alpha_0}{\beta_0}$, increasing β_0 decreases the error, the reason being that increased β_0 yields lower variance for the precision.

Figure 2.19 plots the error versus α_0 for $a = 1, 3, 5$, when $m = 5$, $b = 100$, $\delta_0 = 40$, $\kappa_0 = 100$, and $\beta_0 = 10^5$. For all values of a, the classification error is a decreasing function of α_0. Similarly, Figure 2.20 gives the error versus β_0 for $a = 1, 3, 5$, when $\alpha_0 = 10$ and the other hyperparameters

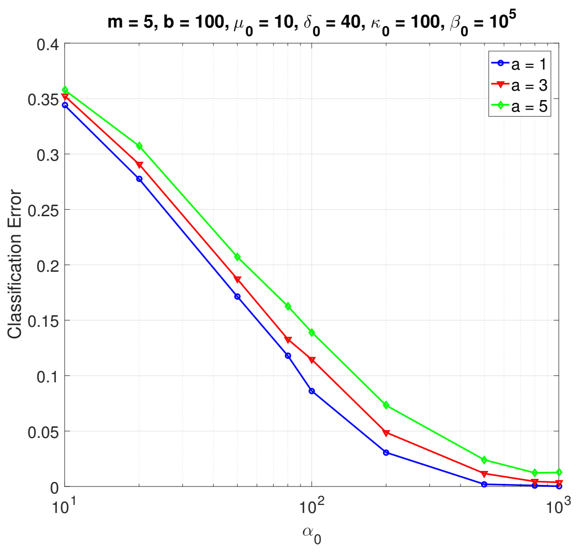


Figure 2.19: Classifier error versus α_0 in cell-cycle network. Reprinted with permission from [3], ©2018 BMC.

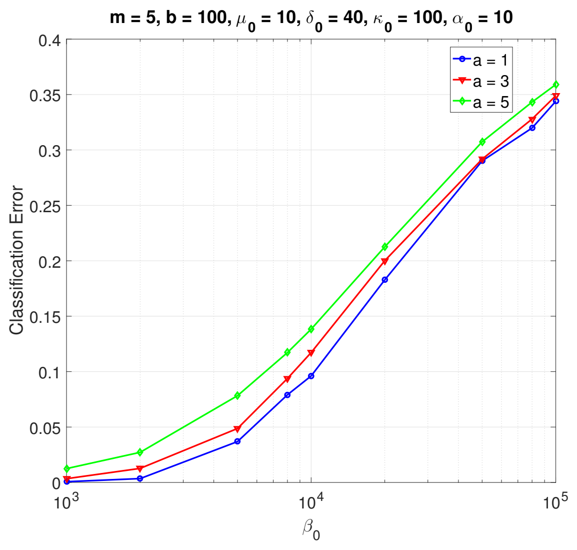


Figure 2.20: Classifier error versus β_0 in cell-cycle network. Reprinted with permission from [3], ©2018 BMC.

are the same as in Figure 2.19. Figure 2.20 shows an increasing trend of classification error as β_0 grows.

Figure 2.21 shows the error as a function of a, when $m = 5$, $b = 100$, $\delta_0 = 40$, $\kappa_0 = 100$, $\alpha_0 = 100$, and $\beta_0 = 10^5$. When a grows, the uncertainty of the perturbation probability grows as well. The error is an increasing function of a. Similarly, Figure 2.22 is for error versus b, when

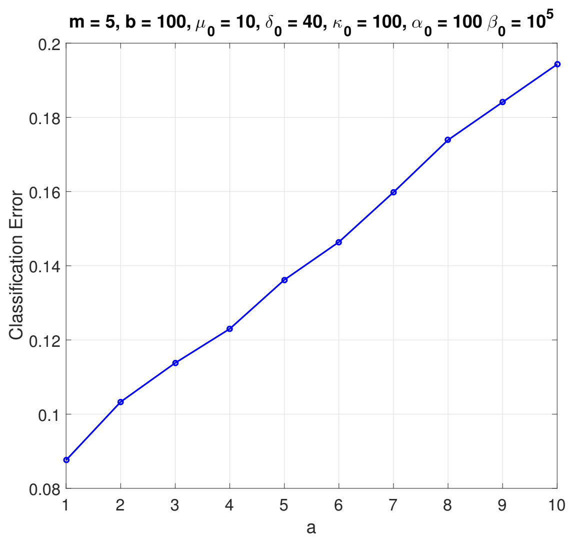


Figure 2.21: Classifier error versus a in cell-cycle network. Reprinted with permission from [3], ©2018 BMC.

$a = 1$, and the others are the same as in Figure 2.21. In Figure 2.22 the classification error is decreasing as b increases.

Figure 2.23 illustrates the error versus κ_0, for $m = 5$, $a = 1$, $b = 100$, $\delta_0 = 40$, $\alpha_0 = 100$, and $\beta_0 = 10^5$. From (2.132), the hyperparameter κ_0 controls the variance of the mean parameters $\theta_j(k)$. When κ_0 increases, the uncertainty of $\theta_j(k)$ is reduced and its density peaks at μ_0 and $\mu_0+\delta_0$ for the unexpressed and expressed states, respectively. Consequently, we expect better error rates for higher κ_0. Accordingly, in Figure 2.23 the classification error is a decreasing function of κ_0.

Figure 2.24 illustrates the error versus δ_0, the expression coefficient, for $m = 5$, $a = 1$, $b = 100$, $\kappa_0 = 100$, $\alpha_0 = 100$, and $\beta_0 = 10^5$. Having larger δ_0 means that the mean values of data for each gene in the unexpressed and expressed cases are well separated, which leads to lower classification error. As expected, in Figure 2.24 the error is decreasing as δ_0 gets larger.

2.3.5 Conclusion

In this section, we proposed a trajectory-based intrinsically Bayesian robust classifier for classification of single-cell gene-expression trajectories. We assumed that the expressions of the n genes, whose interactions are known in terms of a Boolean network, are observed in m consecu-

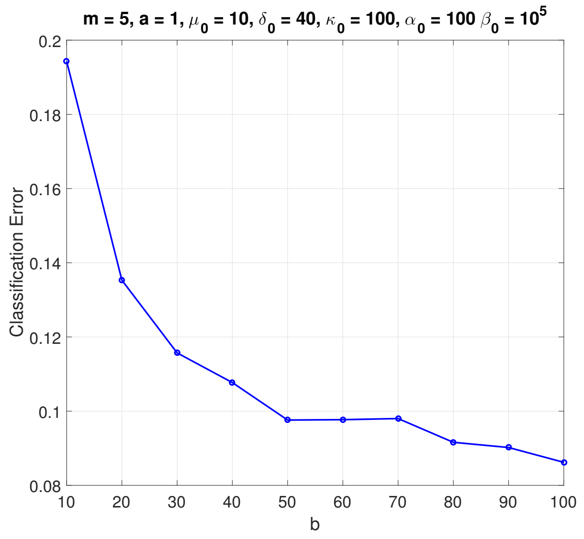


Figure 2.22: Classifier error versus b in cell-cycle network. Reprinted with permission from [3], ©2018 BMC.

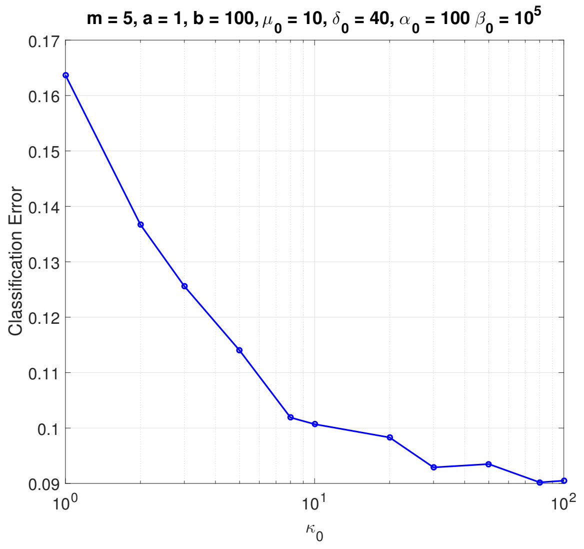


Figure 2.23: Classifier error versus κ_0 in cell-cycle network. Reprinted with permission from [3], ©2018 BMC.

tive time points, for both the wild-type class ($S = 0$) and mutated class ($S = 1$). As the parameters have uncertainty, we assigned priors for them. We assumed a beta distribution as a prior for both the probability of the class $S = 0$ and the gene perturbation probabilities at each time. We assumed a normal-gamma distribution for the mean and precision of the expressions at each time and for each gene, given the underlying states. As such, we derived closed-form solutions for the effective

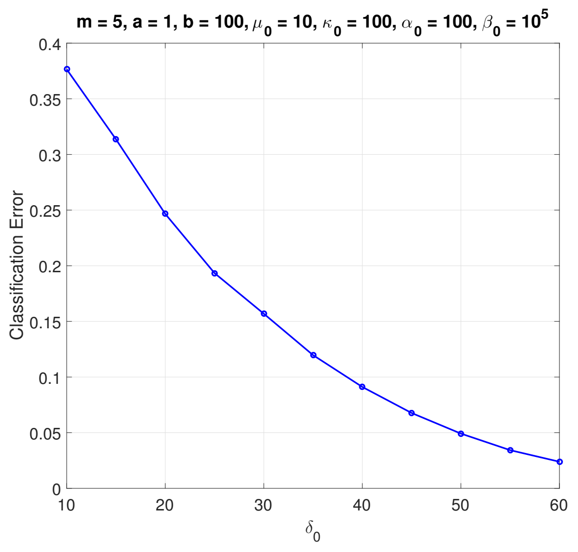


Figure 2.24: Classifier error versus δ_0 in cell-cycle network. Reprinted with permission from [3], ©2018 BMC.

class-conditional densities of the trajectories in each class, by which we defined the IBR classifier. The performance of the IBR classifier was evaluated in a cell-cycle gene regulatory network with 10 genes, for which we know the Boolean networks of the two classes. We also analyzed the effects of m and all the hyperparameters on the classification error.

3. BAYESIAN TRANSFER LEARNING AND REGRESSION*

3.1 Optimal Bayesian Transfer Learning

3.1.1 Overview

Transfer learning has recently attracted significant research attention, as it simultaneously learns from different source domains, which have plenty of labeled data, and transfers the relevant knowledge to the target domain with limited labeled data to improve the prediction performance. We propose a Bayesian transfer learning framework, in the homogeneous transfer learning scenario, where the source and target domains are related through the joint prior density of the model parameters. The modeling of joint prior densities enables better understanding of the transferability between domains. We define a joint Wishart distribution for the precision matrices of the Gaussian feature-label distributions in the source and target domains to act like a bridge that transfers the useful information of the source domain to help classification in the target domain by improving the target posteriors. Using several theorems from multivariate statistics, the posteriors and posterior predictive densities are derived in closed forms with hypergeometric functions of matrix argument, leading to our novel closed-form and fast Optimal Bayesian Transfer Learning (OBTL) classifier. Experimental results on both synthetic and real-world benchmark data confirm the superb performance of the OBTL compared to the other state-of-the-art transfer learning and domain adaptation methods.

3.1.2 Introduction

A basic assumption of traditional machine learning is that data in the training and test sets are independently sampled in one domain with the identical underlying distribution. However, with

*Reprinted with permission from "Optimal Bayesian Transfer Regression" by A. Karbalayghareh, X. Qian, and E. Dougherty 2018. IEEE Signal Processing Letters 25, no. 11, 1655-1659, Copyright 2018 IEEE.

*Reprinted with permission from "Optimal Bayesian Transfer Learning for Count Data" by A. Karbalayghareh, X. Qian, and E. Dougherty 2019. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Copyright 2019 IEEE.

*Reprinted with permission from "Optimal Bayesian Transfer Learning" by A. Karbalayghareh, X. Qian, and E. Dougherty 2018. IEEE Transactions on Signal Processing 66, no. 14, 3724-3739, Copyright 2018 IEEE.

the growing amount of heterogeneity in modern data, the assumption of having only one domain may not be reasonable. Transfer learning (TL) is a learning strategy that enables us to learn from a source domain with plenty of labeled data as well as a target domain with no or very few labeled data in order to design a better classifier in the target domain than the ones trained by target-only data for its generalization performance. This can reduce the effort of collecting labeled data for the target domain, which might be very costly, if not impossible. Due to its importance, there has been ongoing research on the topic of transfer learning and many surveys in the recent years covering transfer learning and domain adaptation methods from different perspectives [65-69].

If we train a model in one domain and directly apply it in another, the trained model may not generalize well, but if the domains are related, appropriate transfer learning and domain adaptation methods can borrow information from all the data across the domains to develop better generalizable models in the target domain. Transfer learning in medical genomics is desirable, since the number of labeled data samples is often very limited due to the difficulty of having disease samples and the prohibitive costs of human clinical trials. However, it is relatively easier to obtain gene-expression data for cell lines or other model species like mice or dogs. If these different life systems share the same underlying disease cellular mechanisms, we may utilize data in cell lines or model species as our source domain to develop transfer learning methods for more accurate human disease prognosis in the target domain [70, 71].

3.1.2.1 Related Works

Domain adaptation (DA) is a specific case of transfer learning where the source and target domains have the same classes or categories [66, 67, 69]. DA methods either adapt the model learned in the source domain to be applied in the target domain or adapt the source data so that the distribution can be close to the one of the target data. Depending on the availability of labeled target data, the DA methods are categorized as unsupervised and semi-supervised algorithms. Unsupervised DA problems applies to the cases where there are no labeled target data and the algorithm uses only unlabeled data in the target domain along with source labeled data [72]. Semi-supervised DA methods use both the unlabeled and a few labeled target data to learn a classifier in the target domain with the help of source labeled data [9,73-75].

Depending on whether the source and target domains have the same feature space with the same feature dimension, there are homogeneous and heterogeneous DA methods. The first direction in homogeneous DA is instance re-weighting, for which the most popular measure to re-weight the data is Maximum Mean Discrepancy (MMD) [76] between the two domains. Transfer Adaptive Boosting (TrAdaBoost) [77] is another method that adaptively sets the weights for the source and target samples during each iteration based on the relevance of source and target data to help train the target classifier. Another direction is model or parameter adaptation. There are several efforts to adapt the SVM classifier designed in the source domain for the target domain, for example, based on residual error [78,79]. Feature augmentation methods, such as Geodesic Flow Sampling (GFS) and Geodesic Flow Kernel (GFK) [72], derive intermediate subspaces using Geodesic flows, which interpolate between the source and target domains. Finding an invariant latent domain in which the distance between the empirical distributions of the source and target data is minimized is another direction to tackle the problem of domain adaptation, such as Invariant Latent Space (ILS) in [8]. Authors in [8] proposed to learn an invariant latent Hilbert space to address both the unsupervised and semi-supervised DA problems, where a notion of domain variance is simultaneously minimized while maximizing a measure of discriminatory power using Riemannian optimization techniques. Max-Margin Domain Transform (MMDT) [9] is a semi-supervised feature transformation DA method which uses a cost function based on the misclassification loss and jointly optimizes both the transformation and classifier parameters. Another domain-invariant representation method [80] matches the distributions in the source and target domains via a regularized optimal transportation model. Heterogeneous Feature Augmentation (HFA) [73] is a heterogeneous DA method which typically embeds the source and target data into a common latent space prior to data augmentation.

Domain adaption has been recently studied in deep learning frameworks like deep adaptation network (DAN) [81], residual transfer networks (RTN) [82], and models based on generative adversarial networks (GAN) such as domain adversarial neural network (DaNN) [83] and coupled

GAN (CoGAN) [84]. Although deep DA methods have shown promising results, they require a fairly large amount of labeled data.

3.1.2.2 Main Contributions

This section treats homogeneous transfer learning and domain adaptation from Bayesian perspectives, a key aim being better theoretical understanding when data in the source domain are transferrable to help learning in the target domain. When learning complex systems with limited data, Bayesian learning can integrate prior knowledge to compensate for the generalization performance loss due to the lack of data. Rooted in Optimal Bayesian Classifiers (OBC) [57, 58], which gives the classifiers having Bayesian minimum mean squared error (MMSE) over uncertainty classes of feature-label distributions, we propose a Bayesian transfer learning framework and the corresponding Optimal Bayesian Transfer Learning (OBTL) classifier to formulate the OBC in the target domain by taking advantage of both the available data and the joint prior knowledge in source and target domains. In this Bayesian learning framework, transfer learning from the source to target domain is through a joint prior probability density function for the model parameters of the feature-label distributions of the two domains. By explicitly modeling the dependency of the model parameters of the feature-label distribution, the posterior of the target model parameters can be updated via the joint prior probability distribution function in conjunction with the source and target data. Based on that, we derive the *effective* class-conditional densities of the target domain, by which the OBTL classifier is constructed.

Our problem definition is the same as the aforementioned domain adaptation methods, where there are plenty of labeled source data and few labeled target data. The source and target data follow different multivariate Gaussian distributions with arbitrary mean vectors and precision (inverse of covariance) matrices. For the OBTL, we define a joint Gaussian-Wishart prior distribution, where the two precision matrices in the two domains are jointly connected. This joint prior distribution for the two precision matrices of the two domains acts like a bridge through which the useful knowledge of the source domain can be transferred to the target domain, making the posterior of the target parameters tighter with less uncertainty.

With such a Bayesian transfer learning framework and several theorems from multivariate statistics, we define an appropriate joint prior for the precision matrices using hypergeometric functions of matrix argument, whose marginal distributions are Wishart as well. The corresponding closed-form posterior distributions for the target model parameters are derived by integrating out all the source model parameters. Having closed-form posteriors facilitates closed-form effective class-conditional densities. Hence, the OBTL classifier can be derived based on the corresponding hypergeometric functions and does not need iterative and costly techniques like MCMC sampling. Although the OBTL classifier has a closed form, computing these hypergeometric functions involves the computation of series of zonal polynomials, which is time-consuming and not scalable to high dimension. To resolve this issue, we use the Laplace approximations of these functions, which preserves the good prediction performance of the OBTL while making it efficient and scalable. The performance of the OBTL is tested on both synthetic data and real-world benchmark image datasets to show its superior performance over state-of-the-art domain adaption methods.

3.1.3 Bayesian Transfer Learning Framework

We consider a supervised transfer learning problem in which there are L common classes (labels) in each domain. Let $\mathcal{D}_s$ and $\mathcal{D}_t$ denote the labeled datasets of the source and target domains with the sizes of N_s and N_t, respectively, where $N_t \ll N_s$. Let $\mathcal{D}_s^l = \left\{\mathbf{x}_{s,1}^l, \mathbf{x}_{s,2}^l, \cdots, \mathbf{x}_{s,n_s^l}^l\right\}$, $l \in \{1, \cdots, L\}$, where n_s^l denotes the size of data in the source domain for the label l. Similarly, let $\mathcal{D}_t^l = \left\{\mathbf{x}_{t,1}^l, \mathbf{x}_{t,2}^l, \cdots, \mathbf{x}_{t,n_t^l}^l\right\}$, $l \in \{1, \cdots, L\}$, where n_t^l denotes the size of data in the target domain for the label l. There is no intersection between $\mathcal{D}_t^i$ and $\mathcal{D}_t^j$ and also between $\mathcal{D}_s^i$ and $\mathcal{D}_s^j$ for any $i, j \in \{1, \cdots, L\}$. Obviously, we have $\mathcal{D}_s = \cup_{l=1}^L \mathcal{D}_s^l$, $\mathcal{D}_t = \cup_{l=1}^L \mathcal{D}_t^l$, $N_s = \sum_{l=1}^L n_s^l$, and $N_t = \sum_{l=1}^L n_t^l$. Since we consider the homogeneous transfer learning scenario, where the feature spaces are the same in both the source and target domains, $\mathbf{x}_s^l$ and $\mathbf{x}_t^l$ are $d \times 1$ vectors for d features of the source and target domains, respectively.

Letting $\mathbf{x}^l = \left[\mathbf{x}_t^{l'}, \mathbf{x}_s^{l'}\right]'$ be a $2d \times 1$ augmented feature vector, $\mathbf{A}'$ denoting the transpose of matrix $\mathbf{A}$, a general joint sampling model would take the Gaussian form

$$\mathbf{x}^l \sim \mathcal{N}\left(\mu^l, \left(\mathbf{\Lambda}^l\right)^{-1}\right), \quad l \in \{1, \cdots, L\}, \tag{3.1}$$

with

$$\mu^l = \begin{bmatrix} \mu_t^l \\ \mu_s^l \end{bmatrix}, \quad \mathbf{\Lambda}^l = \begin{bmatrix} \mathbf{\Lambda}_t^l & \mathbf{\Lambda}_{ts}^l \\ \mathbf{\Lambda}_{ts}^{l\ \prime} & \mathbf{\Lambda}_s^l \end{bmatrix}, \tag{3.2}$$

where μ^l is the $2d \times 1$ mean vector, and $\mathbf{\Lambda}^l$ is the $2d \times 2d$ precision matrix. In this model, $\mathbf{\Lambda}_t^l$ and $\mathbf{\Lambda}_s^l$ account for the interactions of features within the source and target domains, respectively, and $\mathbf{\Lambda}_{ts}^l$ accounts for the interactions of the features across the source and target domains, for any class $l \in \{1, \cdots, L\}$. In this Gaussian setting, it is common to use a Wishart distribution as a prior for the precision matrix $\mathbf{\Lambda}^l$, since it is a conjugate prior.

In transfer learning, it is not realistic to assume joint sampling of the source and target domains. Therefore we cannot use the general joint sampling model. Instead, we assume that there are two datasets separately sampled from the source and target domains. Thus, we define a joint prior distribution for $\mathbf{\Lambda}_s^l$ and $\mathbf{\Lambda}_t^l$ by marginalizing out the term $\mathbf{\Lambda}_{ts}^l$. This joint prior distribution of the parameters of the source and target domains accounts for the dependency (or “relatedness") between the domains.

Given this adjustment to account for transfer learning, we utilize a Gaussian model for the feature-label distribution in each domain:

$$\mathbf{x}_z^l \sim \mathcal{N}\left(\mu_z^l, \left(\mathbf{\Lambda}_z^l\right)^{-1}\right), \quad l \in \{1, \cdots, L\}, \tag{3.3}$$

where subscript $z \in \{s, t\}$ denotes the source s or target t domain, μ_s^l and μ_t^l are $d \times 1$ mean vectors in the source and target domains for label l, respectively, $\mathbf{\Lambda}_s^l$ and $\mathbf{\Lambda}_t^l$ are the $d \times d$ precision matrices in the source and target domains for label l, respectively, and a joint Gaussian-Wishart distribution is employed as a prior for mean and precision matrices of the Gaussian models. Under these assumptions, the joint prior distribution for μ_s^l, μ_t^l, $\mathbf{\Lambda}_s^l$, and $\mathbf{\Lambda}_s^l$ takes the form

$$p\left(\mu_s^l, \mu_t^l, \mathbf{\Lambda}_s^l, \mathbf{\Lambda}_t^l\right) = p\left(\mu_s^l, \mu_t^l | \mathbf{\Lambda}_s^l, \mathbf{\Lambda}_t^l\right) p\left(\mathbf{\Lambda}_s^l, \mathbf{\Lambda}_t^l\right). \tag{3.4}$$

To facilitate conjugate priors, we assume that, for any class $l \in \{1, \cdots, L\}$, μ_s^l and μ_t^l are conditionally independent given $\mathbf{\Lambda}_s^l$ and $\mathbf{\Lambda}_t^l$, so that

$$p\left(\mu_s^l, \mu_t^l, \mathbf{\Lambda}_s^l, \mathbf{\Lambda}_t^l\right) = p\left(\mu_s^l | \mathbf{\Lambda}_s^l\right) p\left(\mu_t^l | \mathbf{\Lambda}_t^l\right) p\left(\mathbf{\Lambda}_s^l, \mathbf{\Lambda}_t^l\right), \tag{3.5}$$

and that both $p\left(\mu_s^l | \mathbf{\Lambda}_s^l\right)$ and $p\left(\mu_t^l | \mathbf{\Lambda}_t^l\right)$ are Gaussian,

$$\mu_z^l | \mathbf{\Lambda}_z^l \sim \mathcal{N}\left(\mathbf{m}_z^l, \left(\kappa_z^l \mathbf{\Lambda}_z^l\right)^{-1}\right), \tag{3.6}$$

where $\mathbf{m}_z^l$ is the $d \times 1$ mean vector of μ_z^l, and κ_z^l is a positive scalar hyperparameter. We need to define a joint distribution for $\mathbf{\Lambda}_s^l$ and $\mathbf{\Lambda}_t^l$. In the case of a prior for either $\mathbf{\Lambda}_s^l$ or $\mathbf{\Lambda}_t^l$, we use a Wishart distribution as the conjugate prior. Here we desire a joint distribution for $\mathbf{\Lambda}_s^l$ and $\mathbf{\Lambda}_t^l$, whose marginal distributions for both $\mathbf{\Lambda}_s^l$ and $\mathbf{\Lambda}_t^l$ are Wishart.

We present some definitions and theorems that will be used in deriving the OBTL classifier.

Definition 1. A random $d \times d$ symmetric positive-definite matrix $\mathbf{\Lambda}$ has a nonsingular Wishart distribution with ν degrees of freedom, $W_d(\mathbf{M}, \nu)$, if $\nu \geq d$ and $\mathbf{M}$ is a $d \times d$ positive-definite matrix ($\mathbf{M} > 0$) and the density is

$$p(\mathbf{\Lambda}) = \left[2^{\frac{\nu d}{2}} \Gamma_d\left(\frac{\nu}{2}\right) |\mathbf{M}|^{\frac{\nu}{2}}\right]^{-1} |\mathbf{\Lambda}|^{\frac{\nu-d-1}{2}} \operatorname{etr}\left(-\frac{1}{2}\mathbf{M}^{-1}\mathbf{\Lambda}\right), \tag{3.7}$$

where $|\mathbf{A}|$ is the determinant of $\mathbf{A}$, $\operatorname{etr}(\mathbf{A}) = \exp\left(\operatorname{tr}(\mathbf{A})\right)$ and $\Gamma_d(\alpha)$ is the multivariate gamma function given by

$$\Gamma_d(\alpha) = \pi^{\frac{d(d-1)}{4}} \prod_{i=1}^{d} \Gamma\left(\alpha - \frac{i-1}{2}\right). \tag{3.8}$$

Proposition 1. *[85]: If $\mathbf{\Lambda} \sim W_d(\mathbf{M}, \nu)$, and $\mathbf{A}$ is an $r \times d$ matrix of rank r, where $r \leq d$, then*

$\mathbf{A}\boldsymbol{\Lambda}\mathbf{A}' \sim W_r(\mathbf{AMA}', \nu)$.

Corollary 1. *If* $\boldsymbol{\Lambda} \sim W_d(\mathbf{M}, \nu)$ *and* $\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{12}' & \boldsymbol{\Lambda}_{22} \end{pmatrix}$*, where* $\boldsymbol{\Lambda}_{11}$ *and* $\boldsymbol{\Lambda}_{22}$ *are* $d_1 \times d_1$ *and* $d_2 \times d_2$ *submatrices, respectively, and if* $\mathbf{M} = \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{12}' & \mathbf{M}_{22} \end{pmatrix}$ *is the corresponding partition of* $\mathbf{M}$ *with* $\mathbf{M}_{11}$ *and* $\mathbf{M}_{22}$ *being two* $d_1 \times d_1$ *and* $d_2 \times d_2$ *submatrices, respectively, then* $\boldsymbol{\Lambda}_{11} \sim W_{d_1}(\mathbf{M}_{11}, \nu)$ *and* $\boldsymbol{\Lambda}_{22} \sim W_{d_2}(\mathbf{M}_{22}, \nu)$.

Using Corollary 1, we can ensure that using the Wishart distribution for the precision matrix $\boldsymbol{\Lambda}^l$ (3.2) of the joint model in (3.1) will lead to the Wishart marginal distributions for $\boldsymbol{\Lambda}_s^l$ and $\boldsymbol{\Lambda}_t^l$ in the source and target domains separately, which is a desired property. Now we introduce a theorem, proposed in [86], which gives the form of the joint distribution of the two submatrices of a partitioned Wishart matrix.

Theorem 1. *[86]: Let* $\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{12}' & \boldsymbol{\Lambda}_{22} \end{pmatrix}$ *be a* $(d_1 + d_2) \times (d_1 + d_2)$ *partitioned Wishart random matrix, where the diagonal partitions are of sizes* $d_1 \times d_1$ *and* $d_2 \times d_2$*, respectively. The Wishart distribution of* $\boldsymbol{\Lambda}$ *has* $\nu \geq d_1 + d_2$ *degrees of freedom and positive-definite scale matrix* $\mathbf{M} = \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{12}' & \mathbf{M}_{22} \end{pmatrix}$ *partitioned in the same way as* $\boldsymbol{\Lambda}$*. The joint distribution of the two diagonal partitions* $\boldsymbol{\Lambda}_{11}$ *and* $\boldsymbol{\Lambda}_{22}$ *have the density function given by*

$$\begin{aligned} & p(\boldsymbol{\Lambda}_{11}, \boldsymbol{\Lambda}_{22}) = \\ & K \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{M}_{11}^{-1} + \mathbf{F}'\mathbf{C}_2\mathbf{F}\right)\boldsymbol{\Lambda}_{11}\right) \operatorname{etr}\left(-\frac{1}{2}\mathbf{C}_2^{-1}\boldsymbol{\Lambda}_{22}\right) \\ & \times |\boldsymbol{\Lambda}_{11}|^{\frac{\nu-d_2-1}{2}} |\boldsymbol{\Lambda}_{22}|^{\frac{\nu-d_1-1}{2}} \; {}_0F_1\left(\frac{\nu}{2}; \frac{1}{4}\mathbf{G}\right), \end{aligned} \tag{3.9}$$

where $\mathbf{C}_2 = \mathbf{M}_{22} - \mathbf{M}_{12}'\mathbf{M}_{11}^{-1}\mathbf{M}_{12}$, $\mathbf{F} = \mathbf{C}_2^{-1}\mathbf{M}_{12}'\mathbf{M}_{11}^{-1}$, $\mathbf{G} = \boldsymbol{\Lambda}_{22}^{\frac{1}{2}}\mathbf{F}\boldsymbol{\Lambda}_{11}\mathbf{F}'\boldsymbol{\Lambda}_{22}^{\frac{1}{2}}$*,* $K^{-1} = 2^{\frac{(d_1+d_2)\nu}{2}}\Gamma_{d_1}\left(\frac{\nu}{2}\right)\Gamma_{d_2}\left(\frac{\nu}{2}\right)|\mathbf{M}|^{\frac{\nu}{2}}$*, and* ${}_0F_1$ *is the generalized matrix-variate hypergeometric function.*

Definition 2. [87]: The generalized hypergeometric function of one matrix argument is defined by

$$
\begin{aligned}
&{}_pF_q(a_1,\cdots,a_p;b_1,\cdots,b_q;\mathbf{X}) \\
&\qquad = \sum_{k=0}^{\infty}\sum_{\kappa\vdash k}\frac{(a_1)_\kappa\cdots(a_p)_\kappa}{(b_1)_\kappa\cdots(b_q)_\kappa}\frac{C_\kappa(\mathbf{X})}{k!},
\end{aligned}
\tag{3.10}
$$

where a_i, $i=1,\cdots,p$, and b_j, $j=1,\cdots,q$, are arbitrary complex (real in our case) numbers, $C_\kappa(\mathbf{X})$ is the zonal polynomial of $d\times d$ symmetric matrix $\mathbf{X}$ corresponding to the ordered partition $\kappa=(k_1,\cdots,k_d)$, $k_1\geq\cdots\geq k_d\geq 0$, $k_1+\cdots k_d=k$ and $\sum_{\kappa\vdash k}$ denotes summation over all partitions κ of k. The generalized hypergeometric coefficient $(a)_\kappa$ is defined by

$$
(a)_\kappa=\prod_{i=1}^{d}\left(a-\frac{i-1}{2}\right)_{k_i},
\tag{3.11}
$$

where $(a)_r=a(a+1)\cdots(a+r-1)$, $r=1,2,\cdots$, with $(a)_0=1$.

Conditions for convergence of the series in (3.10) are available in the literature [88]. From (3.10) it follows

$$
\begin{aligned}
{}_0F_0(\mathbf{X}) &= \sum_{k=0}^{\infty}\sum_{\kappa\vdash k}\frac{C_\kappa(\mathbf{X})}{k!}=\sum_{k=0}^{\infty}\frac{(\mathrm{tr}(\mathbf{X}))^k}{k!}=\mathrm{etr}(\mathbf{X}),\\
{}_1F_0(a;\mathbf{X}) &= \sum_{k=0}^{\infty}\sum_{\kappa\vdash k}\frac{(a)_\kappa C_\kappa(\mathbf{X})}{k!}=|\mathbf{I}_m-\mathbf{X}|^{-a},\ \ ||\mathbf{X}||<1,\\
{}_0F_1(b;\mathbf{X}) &= \sum_{k=0}^{\infty}\sum_{\kappa\vdash k}\frac{C_\kappa(\mathbf{X})}{(b)_\kappa k!},\\
{}_1F_1(a;b;\mathbf{X}) &= \sum_{k=0}^{\infty}\sum_{\kappa\vdash k}\frac{(a)_\kappa}{(b)_\kappa}\frac{C_\kappa(\mathbf{X})}{k!},\\
{}_2F_1(a,b;c;\mathbf{X}) &= \sum_{k=0}^{\infty}\sum_{\kappa\vdash k}\frac{(a)_\kappa(b)_\kappa}{(c)_\kappa}\frac{C_\kappa(\mathbf{X})}{k!},\quad ||\mathbf{X}||<1,
\end{aligned}
\tag{3.12}
$$

where $||\mathbf{X}||<1$ means that the maximum of the absolute values of the eigenvalues of $\mathbf{X}$ is less than 1. ${}_1F_1(a;b;\mathbf{X})$ and ${}_2F_1(a,b;c;\mathbf{X})$ are respectively called Confluent and Gauss hypergeometric functions of matrix argument. See Appendix B.1 for some useful theorems on zonal polynomials and generalized hypergeometric functions of matrix arguments. We use those theorems to derive the posterior densities and posterior predictive densities of the target parameters in closed forms in terms of Confluent and Gauss hypergeometric functions of matrix argument in Sections 3.1.4 and 3.1.5, respectively.

Now, using Theorem 1, we define the joint prior distribution, $p(\mathbf{\Lambda}_s^l, \mathbf{\Lambda}_t^l)$ in (3.5), of the precision matrices of the source and target domains for class $l \in \{1, \cdots, L\}$ as follows:

$$\begin{aligned} p(\mathbf{\Lambda}_t^l, \mathbf{\Lambda}_s^l) = & K^l \mathrm{etr}\left(-\frac{1}{2}\left(\left(\mathbf{M}_t^l\right)^{-1} + \mathbf{F}^{l'}\mathbf{C}^l\mathbf{F}^l\right)\mathbf{\Lambda}_t^l\right) \\ & \times \mathrm{etr}\left(-\frac{1}{2}\left(\mathbf{C}^l\right)^{-1}\mathbf{\Lambda}_s^l\right) \\ & \times \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l-d-1}{2}} \left|\mathbf{\Lambda}_s^l\right|^{\frac{\nu^l-d-1}{2}} {}_0F_1\left(\frac{\nu^l}{2}; \frac{1}{4}\mathbf{G}^l\right), \end{aligned} \tag{3.13}$$

where $\mathbf{M} = \begin{pmatrix} \mathbf{M}_t^l & \mathbf{M}_{ts}^l \\ \mathbf{M}_{ts}^{l\,'} & \mathbf{M}_s^l \end{pmatrix}$ is a $2d \times 2d$ positive definite scale matrix, $\nu^l \geq 2d$ denotes degrees of freedom, and

$$\begin{aligned} \mathbf{C}^l = & \ \mathbf{M}_s^l - \mathbf{M}_{ts}^{l\,'}\left(\mathbf{M}_t^l\right)^{-1}\mathbf{M}_{ts}^l, \\ \mathbf{F}^l = & \ \left(\mathbf{C}^l\right)^{-1}\mathbf{M}_{ts}^{l\,'}\left(\mathbf{M}_t^l\right)^{-1}, \\ \mathbf{G}^l = & \ \mathbf{\Lambda}_s^{l\,\frac{1}{2}}\mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l'}\mathbf{\Lambda}_s^{l\,\frac{1}{2}}, \\ \left(K^l\right)^{-1} = & \ 2^{d\nu^l}\Gamma_d^2\left(\frac{\nu^l}{2}\right)\left|\mathbf{M}^l\right|^{\frac{\nu^l}{2}}. \end{aligned} \tag{3.14}$$

Using Corollary 1, $\mathbf{\Lambda}_t^l$ and $\mathbf{\Lambda}_s^l$ have the following Wishart marginal distributions:

$$\mathbf{\Lambda}_z^l \sim W_d(\mathbf{M}_z^l, \nu^l), \quad l \in \{1, \cdots, L\}, \quad z \in \{s, t\}. \tag{3.15}$$

3.1.4 Posteriors of Target Parameters

Having defined the prior distributions in the previous section, we aim to derive the posterior distribution of the parameters of the target domain upon observing the training source $\mathcal{D}_s$ and target $\mathcal{D}_t$ datasets. The likelihood of the datasets $\mathcal{D}_t$ and $\mathcal{D}_s$ is conditionally independent given the parameters of the target and source domains. The dependence between the two domains is due to the dependence of the prior distributions of the precision matrices, as shown in Fig 3.1.

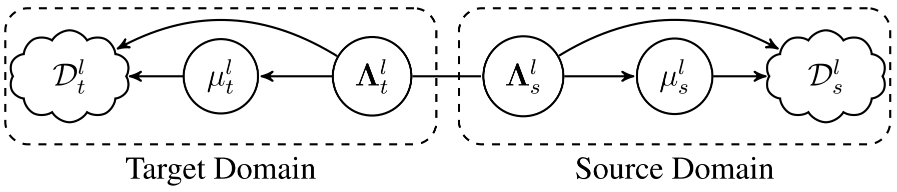


Figure 3.1: Dependency of the source and target domains through their precision matrices for any class $l \in \{1, \cdots, L\}$. Reprinted with permission from [4], ©2018 IEEE.

Within each domain, source or target, the likelihoods of the different classes are also conditionally independent given the parameters of the classes. As such, the joint likelihood of the datasets $\mathcal{D}_t$ and $\mathcal{D}_s$ can be written as

$$\begin{aligned} p(\mathcal{D}_t,\mathcal{D}_s|\mu_t,\mu_s,\mathbf{\Lambda}_t,\mathbf{\Lambda}_s) &= p(\mathcal{D}_t|\mu_t,\mathbf{\Lambda}_t)p(\mathcal{D}_s|\mu_s,\mathbf{\Lambda}_s) \\ &= p(\mathcal{D}_t^1,\cdots,\mathcal{D}_t^L|\mu_t^1,\cdots,\mu_t^L,\mathbf{\Lambda}_t^1,\cdots,\mathbf{\Lambda}_t^L) \\ &\quad \times p(\mathcal{D}_s^1,\cdots,\mathcal{D}_s^L|\mu_s^1,\cdots,\mu_s^L,\mathbf{\Lambda}_s^1,\cdots,\mathbf{\Lambda}_s^L) \\ &= \prod_{l=1}^{L} p(\mathcal{D}_t^l|\mu_t^l,\mathbf{\Lambda}_t^l)\prod_{l=1}^{L} p(\mathcal{D}_s^l|\mu_s^l,\mathbf{\Lambda}_s^l). \end{aligned} \tag{3.16}$$

The posterior of the parameters given $\mathcal{D}_t$ and $\mathcal{D}_s$ satisfies

$$\begin{aligned} &p(\mu_t,\mu_s,\mathbf{\Lambda}_t,\mathbf{\Lambda}_s|\mathcal{D}_t,\mathcal{D}_s) \\ &\propto p(\mathcal{D}_t,\mathcal{D}_s|\mu_t,\mu_s,\mathbf{\Lambda}_t,\mathbf{\Lambda}_s)p(\mu_t,\mu_s,\mathbf{\Lambda}_t,\mathbf{\Lambda}_s) \\ &\propto \prod_{l=1}^{L} p(\mathcal{D}_t^l|\mu_t^l,\mathbf{\Lambda}_t^l)\prod_{l=1}^{L} p(\mathcal{D}_s^l|\mu_s^l,\mathbf{\Lambda}_s^l)\prod_{l=1}^{L} p(\mu_t^l,\mu_s^l,\mathbf{\Lambda}_t^l,\mathbf{\Lambda}_s^l), \end{aligned} \tag{3.17}$$

where we assume that the priors of the parameters in different classes are independent, $p(\mu_t,\mu_s,\mathbf{\Lambda}_t,\mathbf{\Lambda}_s) = \prod_{l=1}^{L} p(\mu_t^l,\mu_s^l,\mathbf{\Lambda}_t^l,\mathbf{\Lambda}_s^l)$. From (3.5) and (3.17),

$$\begin{aligned} p(\mu_t,\mu_s,\mathbf{\Lambda}_t,\mathbf{\Lambda}_s|\mathcal{D}_t,\mathcal{D}_s) \propto \prod_{l=1}^{L} & p(\mathcal{D}_t^l|\mu_t^l,\mathbf{\Lambda}_t^l)p(\mathcal{D}_s^l|\mu_s^l,\mathbf{\Lambda}_s^l) \\ & \times p\left(\mu_s^l|\mathbf{\Lambda}_s^l\right)p\left(\mu_t^l|\mathbf{\Lambda}_t^l\right)p\left(\mathbf{\Lambda}_s^l,\mathbf{\Lambda}_t^l\right). \end{aligned} \tag{3.18}$$

We can see that the posterior of the parameters is equal to the product of the posteriors of the parameters of each class:

$$p(\mu_t, \mu_s, \mathbf{\Lambda}_t, \mathbf{\Lambda}_s|\mathcal{D}_t, \mathcal{D}_s) = \prod_{l=1}^{L} p(\mu_t^l, \mu_s^l, \mathbf{\Lambda}_t^l, \mathbf{\Lambda}_s^l|\mathcal{D}_t^l, \mathcal{D}_s^l), \tag{3.19}$$

where

$$\begin{aligned} p(\mu_t^l, \mu_s^l, \mathbf{\Lambda}_t^l, \mathbf{\Lambda}_s^l|\mathcal{D}_t^l, \mathcal{D}_s^l) &\propto p(\mathcal{D}_t^l|\mu_t^l, \mathbf{\Lambda}_t^l)p(\mathcal{D}_s^l|\mu_s^l, \mathbf{\Lambda}_s^l) \\ &\times p\left(\mu_s^l|\mathbf{\Lambda}_s^l\right) p\left(\mu_t^l|\mathbf{\Lambda}_t^l\right) p\left(\mathbf{\Lambda}_s^l, \mathbf{\Lambda}_t^l\right). \end{aligned} \tag{3.20}$$

Since we are interested in the posterior of the parameters of the target domain, we integrate out the parameters of the source domain in (3.19):

$$\begin{aligned} p(\mu_t, \mathbf{\Lambda}_t|\mathcal{D}_t, \mathcal{D}_s) &= \int_{\mu_s, \mathbf{\Lambda}_s} p(\mu_t, \mu_s, \mathbf{\Lambda}_t, \mathbf{\Lambda}_s|\mathcal{D}_t, \mathcal{D}_s) d\mu_s d\mathbf{\Lambda}_s \\ &= \prod_{l=1}^{L} \int_{\mu_s^l, \mathbf{\Lambda}_s^l} p(\mu_t^l, \mu_s^l, \mathbf{\Lambda}_t^l, \mathbf{\Lambda}_s^l|\mathcal{D}_t^l, \mathcal{D}_s^l) d\mu_s^l d\mathbf{\Lambda}_s^l \\ &= \prod_{l=1}^{L} p(\mu_t^l, \mathbf{\Lambda}_t^l|\mathcal{D}_t^l, \mathcal{D}_s^l), \end{aligned}$$

where

$$\begin{aligned} &p(\mu_t^l, \mathbf{\Lambda}_t^l|\mathcal{D}_t^l, \mathcal{D}_s^l) \\ &= \int_{\mu_s^l, \mathbf{\Lambda}_s^l} p(\mu_t^l, \mu_s^l, \mathbf{\Lambda}_t^l, \mathbf{\Lambda}_s^l|\mathcal{D}_t^l, \mathcal{D}_s^l) d\mu_s^l d\mathbf{\Lambda}_s^l \\ &\propto p(\mathcal{D}_t^l|\mu_t^l, \mathbf{\Lambda}_t^l)p\left(\mu_t^l|\mathbf{\Lambda}_t^l\right) \\ &\times \int_{\mu_s^l, \mathbf{\Lambda}_s^l} p(\mathcal{D}_s^l|\mu_s^l, \mathbf{\Lambda}_s^l)p\left(\mu_s^l|\mathbf{\Lambda}_s^l\right) p\left(\mathbf{\Lambda}_s^l, \mathbf{\Lambda}_t^l\right) d\mu_s^l d\mathbf{\Lambda}_s^l. \end{aligned} \tag{3.21}$$

Theorem 2. *Given the target $\mathcal{D}_t$ and source $\mathcal{D}_s$ data, the posterior distribution of target mean μ_t^l and target precision matrix $\mathbf{\Lambda}_t^l$ for the class $l \in \{1, \cdots, L\}$ has Gaussian-hypergeometric-function* *distribution*

$$\begin{aligned}
&p(\mu_t^l, \mathbf{\Lambda}_t^l | \mathcal{D}_t^l, \mathcal{D}_s^l) = \\
&A^l \left|\mathbf{\Lambda}_t^l\right|^{\frac{1}{2}} \exp\left(-\frac{\kappa_{t,n}^l}{2}\left(\mu_t^l - \mathbf{m}_{t,n}^l\right)' \mathbf{\Lambda}_t^l \left(\mu_t^l - \mathbf{m}_{t,n}^l\right)\right) \\
&\times \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l + n_t^l - d - 1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_t^l\right)^{-1} \mathbf{\Lambda}_t^l\right) \\
&\times \ {}_1F_1\left(\frac{\nu^l + n_s^l}{2}; \frac{\nu^l}{2}; \frac{1}{2}\mathbf{F}^l \mathbf{\Lambda}_t^l \mathbf{F}^{l'} \mathbf{T}_s^l\right),
\end{aligned} \tag{3.22}$$

where A^l is the constant of proportionality

$$\begin{aligned}
\left(A^l\right)^{-1} = \left(\frac{2\pi}{\kappa_{t,n}^l}\right)^{\frac{d}{2}} 2^{\frac{d\left(\nu^l + n_t^l\right)}{2}} \Gamma_d\left(\frac{\nu^l + n_t^l}{2}\right) \left|\mathbf{T}_t^l\right|^{\frac{\nu^l + n_t^l}{2}} \\
\times \ {}_2F_1\left(\frac{\nu^l + n_s^l}{2}, \frac{\nu^l + n_t^l}{2}; \frac{\nu^l}{2}; \mathbf{T}_s^l \mathbf{F}^l \mathbf{T}_t^l \mathbf{F}^{l'}\right),
\end{aligned} \tag{3.23}$$

and

$$\begin{aligned}
\kappa_{t,n}^l &= \kappa_t^l + n_t^l, \\
\mathbf{m}_{t,n}^l &= \frac{\kappa_t^l \mathbf{m}_t^l + n_t^l \bar{\mathbf{x}}_t^l}{\kappa_t^l + n_t^l}, \\
\left(\mathbf{T}_t^l\right)^{-1} &= \left(\mathbf{M}_t^l\right)^{-1} + \mathbf{F}^{l'} \mathbf{C}^l \mathbf{F}^l + \mathbf{S}_t^l \\
&\qquad + \frac{\kappa_t^l n_t^l}{\kappa_t^l + n_t^l}(\mathbf{m}_t^l - \bar{\mathbf{x}}_t^l)(\mathbf{m}_t^l - \bar{\mathbf{x}}_t^l)', \\
\left(\mathbf{T}_s^l\right)^{-1} &= \left(\mathbf{C}^l\right)^{-1} + \mathbf{S}_s^l + \frac{\kappa_s^l n_s^l}{\kappa_s^l + n_s^l}(\mathbf{m}_s^l - \bar{\mathbf{x}}_s^l)(\mathbf{m}_s^l - \bar{\mathbf{x}}_s^l)',
\end{aligned} \tag{3.24}$$

with sample means and covariances for $z \in \{s, t\}$ as

$$\bar{\mathbf{x}}_z^l = \frac{1}{n_z^l} \sum_{i=1}^{n_z^l} \mathbf{x}_{z,i}^l, \quad \mathbf{S}_z^l = \sum_{i=1}^{n_z^l} \left(\mathbf{x}_{z,i}^l - \bar{\mathbf{x}}_z^l\right)\left(\mathbf{x}_{z,i}^l - \bar{\mathbf{x}}_z^l\right)'.$$

Proof. See Appendix B.2. □

3.1.5 Effective Class-Conditional Densities

In classification, the feature-label distributions are written in terms of class-conditional densities and prior class probabilities, and the posterior probabilities of the classes upon observation of data are proportional to the product of class-conditional densities and prior class probabilities, according to the Bayes rule. This also holds in the Bayesian setting except we use effective class-conditional densities, as shown in [57, 58]. For optimal Bayesian classifier [57, 58], using the posterior predictive densities of the classes, called "effective class-conditional densities", leads to the optimal choices for classifiers in order to minimize the Bayesian error estimates of the classifiers. Similarly, we can derive the effective class-conditional densities for defining the OBTL classifier in the target domain, albeit with the posterior of the target parameters derived from both the target and source datasets.

Suppose that $\mathbf{x}$ denotes a $d \times 1$ new observed data point in the target domain that we aim to optimally classify into one of the classes $l \in \{1, \cdots, L\}$. In the context of the optimal Bayesian classifier, we need the effective class-conditional densities for the L classes, defined as

$$p(\mathbf{x}|l) = \int_{\mu_t^l, \mathbf{\Lambda}_t^l} p(\mathbf{x}|\mu_t^l, \mathbf{\Lambda}_t^l)\pi^\star(\mu_t^l, \mathbf{\Lambda}_t^l) d\mu_t^l d\mathbf{\Lambda}_t^l, \tag{3.25}$$

for $l \in \{1, \cdots, L\}$, where $\pi^\star(\mu_t^l, \mathbf{\Lambda}_t^l) = p(\mu_t^l, \mathbf{\Lambda}_t^l|\mathcal{D}_t^l, \mathcal{D}_s^l)$ is the posterior of $(\mu_t^l, \mathbf{\Lambda}_t^l)$ upon observation of $\mathcal{D}_t^l$ and $\mathcal{D}_s^l$.

Theorem 3. *The effective class-conditional density, denoted by* $p(\mathbf{x}|l) = O_{\mathrm{OBTL}}(\mathbf{x}|l)$*, in the target domain is given by*

$$\begin{aligned} O_{\mathrm{OBTL}}(\mathbf{x}|l) &= \pi^{-\frac{d}{2}} \left(\frac{\kappa_{t,n}^l}{\kappa_{\mathbf{x}}^l}\right)^{\frac{d}{2}} \Gamma_d\left(\frac{\nu^l + n_t^l + 1}{2}\right) \\ &\times \Gamma_d^{-1}\left(\frac{\nu^l + n_t^l}{2}\right) \left|\mathbf{T}_{\mathbf{x}}^l\right|^{\frac{\nu^l + n_t^l + 1}{2}} \left|\mathbf{T}_t^l\right|^{-\frac{\nu^l + n_t^l}{2}} \\ &\times {}_2F_1\left(\frac{\nu^l + n_s^l}{2}, \frac{\nu^l + n_t^l + 1}{2}; \frac{\nu^l}{2}; \mathbf{T}_s^l \mathbf{F}^l \mathbf{T}_{\mathbf{x}}^l \mathbf{F}^{l'}\right) \\ &\times {}_2F_1^{-1}\left(\frac{\nu^l + n_s^l}{2}, \frac{\nu^l + n_t^l}{2}; \frac{\nu^l}{2}; \mathbf{T}_s^l \mathbf{F}^l \mathbf{T}_t^l \mathbf{F}^{l'}\right), \end{aligned} \tag{3.26}$$

where

$$\begin{aligned} \kappa_{\mathbf{x}}^l &= \kappa_{t,n}^l + 1 = \kappa_t^l + n_t^l + 1, \\ \left(\mathbf{T}_{\mathbf{x}}^l\right)^{-1} &= \left(\mathbf{T}_t^l\right)^{-1} + \frac{\kappa_{t,n}^l}{\kappa_{t,n}^l + 1} \left(\mathbf{m}_{t,n}^l - \mathbf{x}\right) \left(\mathbf{m}_{t,n}^l - \mathbf{x}\right)'. \end{aligned} \tag{3.27}$$

Proof. See Appendix B.3. □

3.1.6 Optimal Bayesian Transfer Learning Classifier

Let c_t^l be the prior probability that the target sample $\mathbf{x}$ belongs to the class $l \in \{1, \cdots, L\}$. Since $0 < c_t^l < 1$ and $\sum_{l=1}^{L} c_t^l = 1$, a Dirichlet prior is assumed:

$$(c_t^1, \cdots, c_t^L) \sim \text{Dir}(L, \xi_t), \tag{3.28}$$

where $\xi_t = (\xi_t^1, \cdots, \xi_t^L)$ are the concentration parameters, and $\xi_t^l > 0$ for $l \in \{1, \cdots, L\}$. As the Dirichlet distribution is a conjugate prior for the categorical distribution, upon observing $\mathbf{n} = (n_t^1, \cdots, n_t^L)$ data for class l in the target domain, the posterior has a Dirichlet distribution:

$$\begin{aligned}\pi^\star = (c_t^1, \cdots, c_t^L | \mathbf{n}) &\sim \text{Dir}(L, \xi_t + \mathbf{n}) \\ &= \text{Dir}(L, \xi_t^1 + n_t^1, \cdots, \xi_t^L + n_t^L),\end{aligned} \tag{3.29}$$

with the posterior mean of c_t^l as

$$\text{E}_{\pi^\star}(c_t^l) = \frac{\xi_t^l + n_t^l}{N_t + \xi_t^0}, \tag{3.30}$$

where $N_t = \sum_{l=1}^{L} n_t^l$ and $\xi_t^0 = \sum_{l=1}^{L} \xi_t^l$. As such, the optimal Bayesian transfer learning (OBTL) classifier for any new unlabeled sample $\mathbf{x}$ in the target domain is defined as

$$\Psi_{\text{OBTL}}(\mathbf{x}) = \arg \max_{l \in \{1, \cdots, L\}} \text{E}_{\pi^\star}(c_t^l) O_{\text{OBTL}}(\mathbf{x}|l), \tag{3.31}$$

which minimizes the expected error of the classifier in the target domain, that is, $\text{E}_{\pi^\star}[\varepsilon(\Theta_t, \Psi_{\text{OBTL}})] \leq \text{E}_{\pi^\star}[\varepsilon(\Theta_t, \Psi)]$, where $\varepsilon(\Theta_t, \Psi)$ is the error of any arbitrary classifier Ψ assuming the parameters $\Theta_t = \{c_t^l, \mu_t^l, \mathbf{\Lambda}_t^l\}_{l=1}^{L}$ of the feature-label distribution in the target domain, and the expectation is over the posterior $\pi^\star$ of Θ_t upon observation of data. If we do not have any prior knowledge for the selection of classes, we use the same concentration parameter for all the classes: $\xi_t = (\xi, \cdots, \xi)$. Hence, if the number of samples in each class is the same, $n_t^1 = \cdots = n_t^L$, the first term $\text{E}_{\pi^\star}(c_t^l)$ is the same for all the classes and (3.31) is reduced to:

$$\Psi_{\text{OBTL}}(\mathbf{x}) = \arg\max_{l\in\{1,\cdots,L\}} O_{\text{OBTL}}(\mathbf{x}|l). \tag{3.32}$$

We have derived the effective class-conditional densities in closed forms (3.26). However, deriving the OBTL classifier (3.31) requires computing the Gauss hypergeometric function of matrix argument. Computing the exact values of hypergeometirc functions of matrix argument using the series of zonal polynomials, as in (3.12), is time-consuming and is not scalable to high dimension. To facilitate computation, we propose to use the Laplace approximation of this function, as in [89], which is computationally efficient and scalable. See Appendix B.4 for the detailed description of the Laplace approximation of Gauss hypergeometric functions of matrix argument.

3.1.7 OBC in Target Domain

To see how the source data can help improve the performance, we compare the OBTL classifier with the OBC based on the training data only from the target domain. Using exactly the same modeling and parameters as the previous sections, the priors for μ_t^l and $\mathbf{\Lambda}_t^l$, from (3.6) and (3.15), are given by

$$\begin{aligned} \mu_t^l|\mathbf{\Lambda}_t^l &\sim \mathcal{N}\left(\mathbf{m}_t^l, \left(\kappa_t^l\mathbf{\Lambda}_t^l\right)^{-1}\right), \\ \mathbf{\Lambda}_t^l &\sim W_d(\mathbf{M}_t^l, \nu^l). \end{aligned} \tag{3.33}$$

Using Lemma 6 in Appendix B.2, upon observing the dataset $\mathcal{D}_t^l$, the posteriors of μ_t^l and $\mathbf{\Lambda}_t^l$ will be

$$\begin{aligned} \mu_t^l|\mathbf{\Lambda}_t^l, \mathcal{D}_t^l &\sim \mathcal{N}\left(\mathbf{m}_{t,n}^l, \left(\kappa_{t,n}^l\mathbf{\Lambda}_t^l\right)^{-1}\right), \\ \mathbf{\Lambda}_t^l|\mathcal{D}_t^l &\sim W_d(\mathbf{M}_{t,n}^l, \nu_{t,n}^l), \end{aligned} \tag{3.34}$$

where

$$\begin{aligned} \kappa_{t,n}^l = \kappa_t^l + n_t^l, \quad \nu_{t,n}^l = \nu^l + n_t^l, \quad \mathbf{m}_{t,n}^l &= \frac{\kappa_t^l\mathbf{m}_t^l + n_t^l\bar{\mathbf{x}}_t^l}{\kappa_t^l + n_t^l}, \\ \left(\mathbf{M}_{t,n}^l\right)^{-1} = \left(\mathbf{M}_t^l\right)^{-1} + \mathbf{S}_t^l + \frac{\kappa_t^l n_t^l}{\kappa_t^l + n_t^l}&(\mathbf{m}_t^l - \bar{\mathbf{x}}_t^l)(\mathbf{m}_t^l - \bar{\mathbf{x}}_t^l)', \end{aligned} \tag{3.35}$$

with the corresponding sample mean and covariance:

$$\bar{\mathbf{x}}_t^l = \frac{1}{n_t^l}\sum_{i=1}^{n_t^l}\mathbf{x}_{t,i}^l, \quad \mathbf{S}_t^l = \sum_{i=1}^{n_t^l}\left(\mathbf{x}_{t,i}^l - \bar{\mathbf{x}}_t^l\right)\left(\mathbf{x}_{t,i}^l - \bar{\mathbf{x}}_t^l\right)'. \tag{3.36}$$

By (3.25) and similar integral steps, the effective class-conditional densities $p(\mathbf{x}|l) = O_{\mathrm{OBC}}(\mathbf{x}|l)$ for the OBC are derived as [57]

$$\begin{aligned} O_{\mathrm{OBC}}(\mathbf{x}|l) = \pi^{-\frac{d}{2}}\left(\frac{\kappa_{t,n}^l}{\kappa_{t,n}^l+1}\right)^{\frac{d}{2}}\Gamma_d\left(\frac{\nu^l+n_t^l+1}{2}\right) \\ \times\,\Gamma_d^{-1}\left(\frac{\nu^l+n_t^l}{2}\right)\left|\mathbf{M}_{\mathbf{x}}^l\right|^{\frac{\nu^l+n_t^l+1}{2}}\left|\mathbf{M}_{t,n}^l\right|^{-\frac{\nu^l+n_t^l}{2}}, \end{aligned} \tag{3.37}$$

where

$$\left(\mathbf{M}_{\mathbf{x}}^l\right)^{-1} = \left(\mathbf{M}_{t,n}^l\right)^{-1} + \frac{\kappa_{t,n}^l}{\kappa_{t,n}^l+1}(\mathbf{m}_{t,n}^l - \mathbf{x})(\mathbf{m}_{t,n}^l - \mathbf{x})'. \tag{3.38}$$

The multi-class OBC [90], under a zero-one loss function, can be defined as

$$\Psi_{\mathrm{OBC}}(\mathbf{x}) = \arg\max_{l\in\{1,\cdots,L\}} \mathrm{E}_{\pi^\star}(c_t^l)O_{\mathrm{OBC}}(\mathbf{x}|l). \tag{3.39}$$

Similar to the OBTL, in the case of equal prior probabilities for the classes,

$$\Psi_{\mathrm{OBC}}(\mathbf{x}) = \arg\max_{l\in\{1,\cdots,L\}} O_{\mathrm{OBC}}(\mathbf{x}|l). \tag{3.40}$$

For binary classification, the definition of the OBC in (3.39) is equivalent to the definition in [57], where it is defined to be the binary classifier possessing the minimum Bayesian mean square error estimate [64] relative to the posterior distribution.

Theorem 4. *If* $\mathrm{M}_{ts}^l = \mathbf{0}$ *for all* $l \in \{1,\cdots,L\}$*, then*

$$\Psi_{\mathrm{OBTL}}(\mathbf{x}) = \Psi_{\mathrm{OBC}}(\mathbf{x}), \tag{3.41}$$

meaning that if there is no interaction between the source and target domains in all the classes a *priori, then the OBTL classifier turns to the OBC classifier in the target domain.*

Proof. If $\mathbf{M}_{ts}^l = \mathbf{0}$ for all $l \in \{1, \cdots, L\}$, then $\mathbf{F}^l = \mathbf{0}$. Since ${}_2F_1(a, b; c; \mathbf{0}) = 1$ for any values of a, b, and c, the Gauss hypergeometric functions will disappear in (3.26). From (3.24) and (3.35), $\mathbf{T}_t^l = \mathbf{M}_{t,n}^l$. From (3.27) and (3.38), $\mathbf{T}_{\mathbf{x}}^l = \mathbf{M}_{\mathbf{x}}^l$. As a result, $O_{\mathrm{OBTL}}(\mathbf{x}|l) = O_{\mathrm{OBC}}(\mathbf{x}|l)$, and consequently, $\Psi_{\mathrm{OBTL}}(\mathbf{x}) = \Psi_{\mathrm{OBC}}(\mathbf{x})$. □

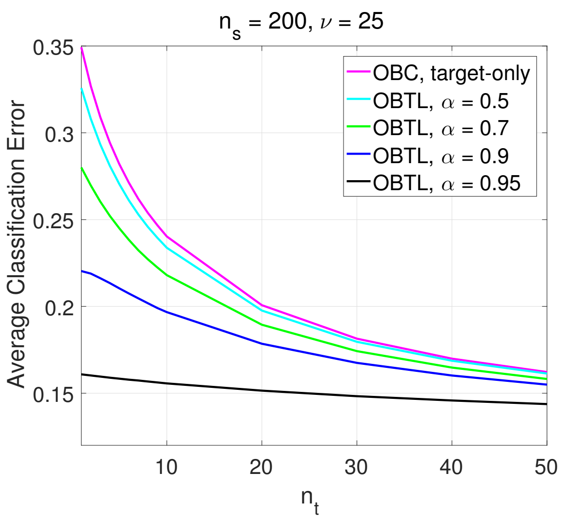
3.1.8 Experiments

3.1.8.1 Synthetic datasets

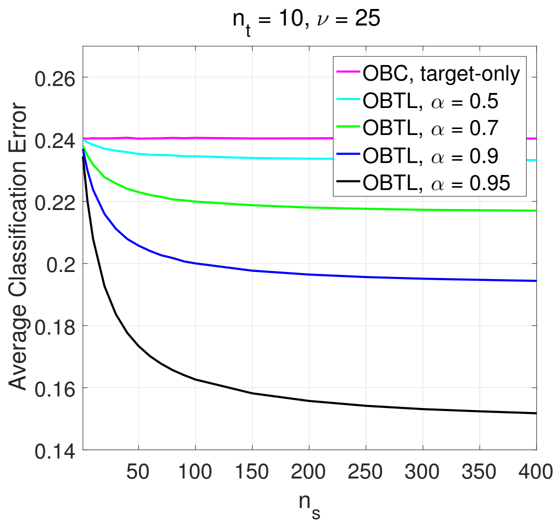
We have considered a simulation setup and evaluated the OBTL classifiers by the average classification error with different joint prior densities modeling the relatedness of the source and target domains. The setup is as follows. Unless mentioned, the feature dimension is $d = 10$, the number of classes in each domain is $L = 2$, the number of source training data per class is $n_s = n_s^l = 200$, the number of target training data per class is $n_t = n_t^l = 10$, $\nu = \nu^l = 25$, $\kappa_t = \kappa_t^l = 100$, $\kappa_s = \kappa_s^l = 100$, for both the classes $l = 1, 2$, $\mathbf{m}_t^1 = \mathbf{0}_d$, $\mathbf{m}_t^2 = 0.05 \times \mathbf{1}_d$, $\mathbf{m}_s^1 = \mathbf{m}_t^1 + \mathbf{1}_d$, and $\mathbf{m}_s^2 = \mathbf{m}_t^2 + \mathbf{1}_d$, where $\mathbf{0}_d$ and $\mathbf{1}_d$ are $d \times 1$ all-zero and all-one vectors, respectively. For the scale matrices, we choose $\mathbf{M}_t^l = k_t \mathbf{I}_d$, $\mathbf{M}_s^l = k_s \mathbf{I}_d$, and $\mathbf{M}_{ts}^l = k_{ts} \mathbf{I}_d$ for two classes $l = 1, 2$, where $\mathbf{I}_d$ is the $d \times d$ identity matrix. Note that choosing an identity matrix for $\mathbf{M}_{ts}^l$ makes sense when the order of the features in the two domains is the same. We have the constraint that the scale matrix $\mathbf{M}^l = \begin{pmatrix} \mathbf{M}_t^l & \mathbf{M}_{ts}^l \\ \mathbf{M}_{ts}^{l\,\prime} & \mathbf{M}_s^l \end{pmatrix}$ should be positive definite for any class l. It is easy to check the following corresponding constraints on k_t, k_s, and k_{ts}: $k_t > 0$, $k_s > 0$, and $|k_{ts}| < \sqrt{k_t k_s}$. We define $k_{ts} = \alpha \sqrt{k_t k_s}$, where $|\alpha| < 1$. In this particular example, the value of $|\alpha|$ shows the amount of relatedness between the source and target domains. If $|\alpha| = 0$, the two domains are not related and if $|\alpha|$ is close to one, we have greater relatedness. We set $k_t = k_s = 1$ and plot the average classification error curves for different values of $|\alpha|$. All the simulations assume equal prior probabilities for the classes, so we use (3.32) and (3.40) for the OBTL classifier and OBC, respectively.

We evaluate the prediction performance according to the common evaluation procedure of Bayesian learning by average classification errors. To sample from the prior (3.5) we first sample from a Wishart distribution $W_{2d}(\mathbf{M}^l, \nu^l)$ to get a sample for $\mathbf{\Lambda}^l = \begin{pmatrix} \mathbf{\Lambda}_t^l & \mathbf{\Lambda}_{ts}^l \\ \mathbf{\Lambda}_{ts}^{l\,\prime} & \mathbf{\Lambda}_s^l \end{pmatrix}$, for each class $l = 1, 2$, and then pick $(\mathbf{\Lambda}_t^l, \mathbf{\Lambda}_s^l)$, which is a joint sample from $p(\mathbf{\Lambda}_t^l, \mathbf{\Lambda}_s^l)$ in (3.13). Then given $\mathbf{\Lambda}_t^l$ and $\mathbf{\Lambda}_s^l$, we sample from (3.6) to get samples of μ_t^l and μ_s^l for $l = 1, 2$. Once we have μ_t^l, μ_s^l, $\mathbf{\Lambda}_t^l$, and $\mathbf{\Lambda}_s^l$, we generate 100 different training and test sets from (3.3). Training sets contain samples from both the target and source domains, but the test set contains only samples from the target domain. As the numbers of source and target training data per class are n_s and n_t, there are Ln_s and Ln_t source and target training data in total, respectively. We assume the size of the test set per class is 1000 in the simulations, so 2000 in total. For each training and test set, we use the OBTL classifier and its target-only version, OBC, and calculate the error. Then we average all the errors for 100 different training and test sets. We further repeat this whole process 1000 times for different realizations of $\mathbf{\Lambda}_t^l$ and $\mathbf{\Lambda}_s^l$, μ_t^l, and μ_s^l for $l = 1, 2$, and finally average all the errors and return the average classification error. Note that in all figures, the hyperparameters used in the OBTL classifier are the same as the ones used for simulating data, except for the figures showing the sensitivity of the performance with respect to different hyperparameters, in which case we assume that true values of the hyperparameters used for simulating data are unknown.

To examine how the source data improves the classifier in target domain, we compare the performance of the OBTL classifier with the OBC designed in the target domain alone. The average classification error versus n_t is depicted in Fig. 3.2a for the OBC and OBTL with different values of α. When α is close to one, the performance of the OBTL classifier is much better than that of the OBC, this due to the greater relatedness between the two domains and appropriate use of the source data. This performance improvement is especially noticeable when n_t is small, which reflects the real-world scenario. In Fig. 3.2a, we also observe that the errors of the OBTL classifier and OBC are converging to a similar value when n_t gets very large, meaning that the source data are redundant when there is a large amount of target data. When α is larger, the error curves converge faster to the optimal error, which is the average Bayes error of the target classifier. The corresponding Bayes error averaged over 1000 randomly generated distributions is equal to 0.122 in this simulation setup. Recall that when $\alpha = 0$, the OBTL classifier reduces to the OBC. In this



(a)



(b)

Figure 3.2: (a) Average classification error versus the number of target training data per class, n_t, (b) Average classification error versus the number of source training data per class, n_s. Reprinted with permission from [4], ©2018 IEEE.

particular example, the sign of α does not matter in the performance of the OBTL, which can be verified by (3.26). Hence, we can use $|\alpha|$ in all the cases.

Figure 3.2b depicts average classification error versus n_s for the OBC and OBTL with different values of α. The error of the OBC is constant for all n_s as it does not employ the source data. The error of the OBTL classifier equals that of the OBC when $n_s = 0$ and starts to decrease as n_s increases. In Fig. 3.2b when α is larger, the amount of improvement is greater since the two

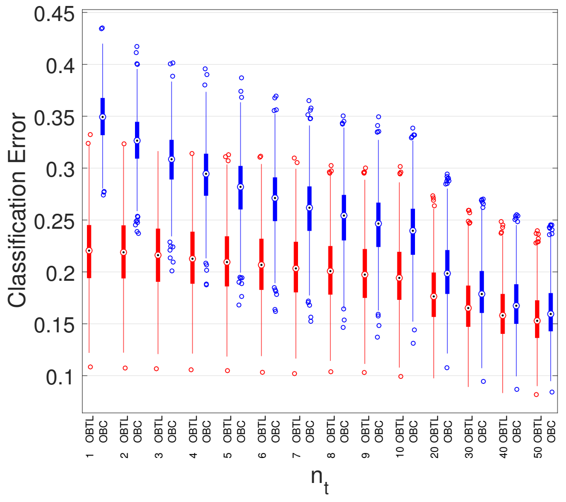


Figure 3.3: Box plots of 1000 simulated classification errors for different n_t. Blue denotes the OBC and red denotes the OBTL with $\alpha = 0.9$. Reprinted with permission from [4], ©2018 IEEE.

domains are more related. Another important point in Fig. 3.2b is that having very large source data when the two domains are highly related can compensate the lack of target data and lead to a target classification error as small as the Bayes error in the target domain.

Figure 3.3 illustrates the box plots of the simulated classification errors corresponding to the 1000 distributions randomly drawn from the prior distributions for both the OBC and OBTL with $\alpha = 0.9$, which show the variability for different numbers n_t of target data per class.

We investigate the sensitivity of the OBTL with respect to the hyperparameters. Fig. 3.4 represents the average classification error of the OBTL with respect to $|\alpha|$, where we assume that we do not know the true value α_{true} of the amount of relatedness between source and target domains. In Figs. 3.4a-3.4d we plot the error curves when $\alpha_{true} = 0.3, 0.5, 0.7, 0.9$, respectively. We observe several important trends in these figures. First of all, the performance gain of the OBTL towards the OBC depends heavily on the relatedness (value of α_{true}) of source and target and the value of α used in the classifier. Generally speaking, there exists an α_{max} in $(0, 1)$ such that for $|\alpha| < \alpha_{max}$, the OBTL has a performance gain towards the OBC, where the maximum gain is achieved at $|\alpha| = \alpha_{true}$ (it might not be exactly at α_{true} due to the Laplace approximation of the Gauss hypergeometric function). Second, the performance gain is higher when the two

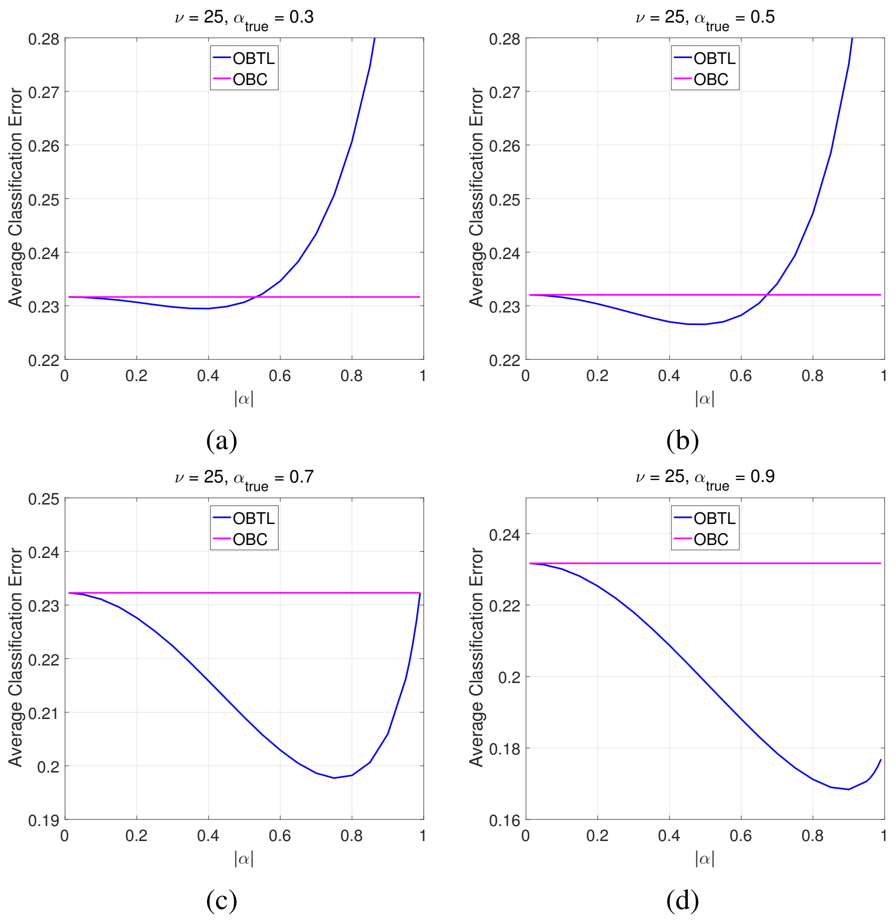
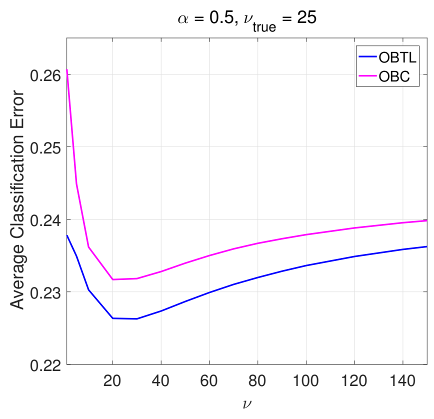


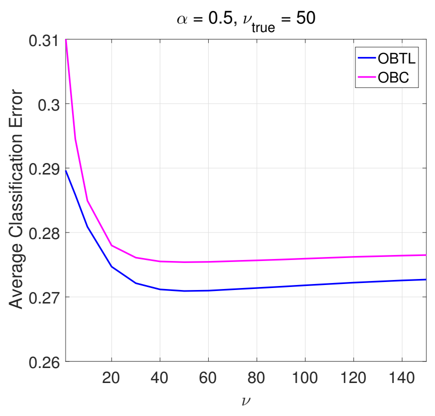
Figure 3.4: Average classification error vs $|\alpha|$. Reprinted with permission from [4], ©2018 IEEE.

domains are highly related (Fig. 3.4d). Third, when the two domains are very related, for example, $\alpha_{true} = 0.9$ in Fig. 3.4d, $\alpha_{max} = 1$, meaning that for any $|\alpha|$, the OBTL has performance gain towards the target-only OBC. However, when the source and target domains are not related much, like Figs. 3.4a and 3.4b, $\alpha_{max} < 1$, and choosing $|\alpha|$ greater than α_{max} leads to performance loss compared to the OBC. This means that exaggeration in the amount of relatedness between the two domains can hurt the transfer learning classifier when the two domains are not actually related, which refers to the concept of negative transfer.

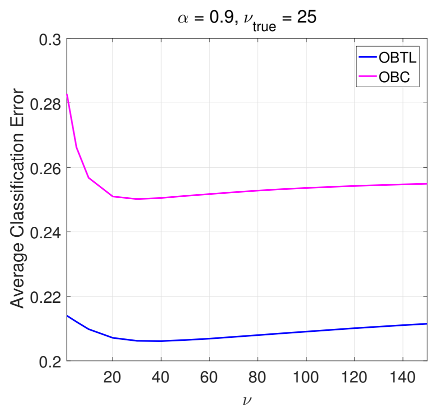
Figure 3.5 shows the errors versus ν, assuming unknown true value ν_{true}, for different values of α (0.5 and 0.9) and ν_{true} (25 and 50). The salient point here is that the performance of the OBTL classifier is not so sensitive to ν if it is chosen in its allowable range, that is, $\nu \geq 2d$. In Fig. 3.5, the error of the OBTL does not change much for $\nu \geq 2d = 20$. As a result, we can choose any



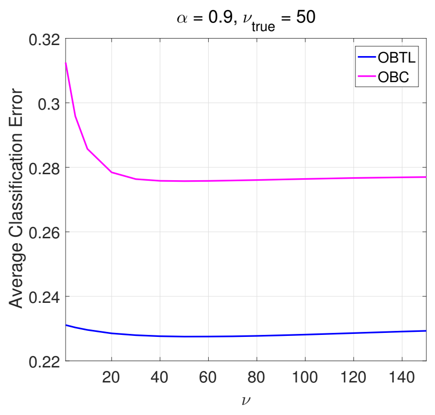
(a)



(b)



(c)



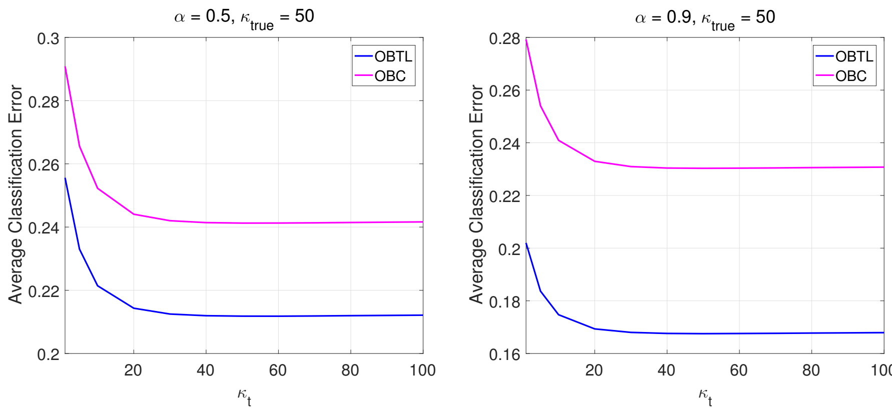
(d)

Figure 3.5: Average classification error vs ν. Reprinted with permission from [4], ©2018 IEEE.

arbitrary $\nu \geq 2d$ in real datasets without worrying about critical performance deterioration.

Figure 3.6 depicts average classification error versus κ_t for two different values of α (0.5 and 0.9), where the true value of κ_t is $\kappa_{true} = 50$. Similar to ν, if κ_t is greater than a value (20 in Fig. 3.6), the performance does not change much. According to (3.24), it is better to choose κ_t^l and κ_s^l to be proportional to n_t and n_s, respectively, since the values of updated means $\mathbf{m}_{t,n}^l$ and $\mathbf{m}_{s,n}^l$ are weighted averages of our prior knowledge about means, $\mathbf{m}_t^l$ and $\mathbf{m}_s^l$, and the sample means $\bar{\mathbf{x}}_t^l$ and $\bar{\mathbf{x}}_s^l$. Assuming that $\kappa_t = \beta_t n_t$ and $\kappa_s = \beta_s n_s$, for some $\beta_t, \beta_s > 0$, if we have higher confidence on our priors on means, we pick higher β_t and β_s (as in Fig. 3.6); but for the untrustworthy priors, we choose lower values for β_t and β_s.

Sensitivity results in Figs. 3.4, 3.5, and 3.6 reveal that in our simulation setup the performance improvement of the OBTL depends on the value of α and true relatedness (α_{true} in this example)



(a) (b)

Figure 3.6: Average classification error vs κ_t. Reprinted with permission from [4], ©2018 IEEE.

between the two domains and is not affected that much by the choices of other hyperparameters like ν, κ_t, and κ_s. We could have a reasonable range of α to get improved performance but the correct estimates of *relatedness* or *transferability* are critical, which is an important future research direction.

3.1.8.2 Real-world benchmark datasets

We test the OBTL classifier on *Office* [91] and *Caltech256* [92] image datasets, which have been adopted to help benchmark different transfer learning algorithms in the literature. We have used exactly the same evaluation setup and data splits of MMDT (Max-Margin Domain Transform) [9].

• **Office dataset:** This dataset has images in three different domains: *amazon*, *webcam*, and *dslr*. The dataset contains 31 classes including the office stuff like backpack, chair, keyboard, etc. The three domains *amazon*, *webcam*, and *dslr* contain images from Amazon's website, a webcam, and a digital single-lens reflex (dslr) camera, respectively, with different lighting and backgrounds. SURF [93] image features are used in all the domains, which are of dimension 800.

• **Office + Caltech256 dataset:** This dataset has $L = 10$ common classes of both *Office* and *Caltech256* datasets with the same feature dimension $d = 800$. According to the data splits of [9], the numbers of training data per class in the source domain are $n_s = 20$ for *amazon* and $n_s = 8$ for the other three domains, and in the target domain $n_t = 3$ for all the four domains. For this four-domain dataset, 20 random train-test splits have been created by [9]. We run the OBTL classifier on that 20 provided train-test splits and report the average accuracy. Note that the test data are solely from the target domains. Authors of MMDT [9] reduce the dimension to $d = 20$ using PCA. We follow the same procedure for the OBTL classifier.

Following the comparison framework of [8], which used the same evaluation setup of [9], we compare the OBTL's performance in terms of accuracy (10-class) in Table 3.1 with two target-only classifiers and four state-of-the-art semi-supervised transfer learning algorithms (including [8] itself). The evaluation setup is exactly the same for the OBTL and all the other six methods. As a result, we use the results of [8] for the first six methods in Table 3.1 and compare them with the OBTL classifier. The six methods are as follows.

• **1-NN-t and SVM-t:** The Nearest Neighbor (1-NN) and linear SVM classifiers designed using only the target data.

• **HFA [73]:** This Heterogeneous Feature Augmentation (HFA) method learns a common latent space between source and target domains using the max-margin approach and designs a classifier in that common space.

• **MMDT [9]:** This Max-Margin Domain Transform (MMDT) method learns a transformation between the source and target domains and employs the weighted SVM for classification.

• **CDLS [75]:** This Cross-Domain Landmark Selection (CDLS) is a semi-supervised heterogeneous domain adaptation method, which derives a domain-invariant feature space for improved classification performance.

• **ILS (1-NN) [8]:** This is a recent method that learns an Invariant Latent Space (ILS) to reduce the discrepancy between the source and target domains and uses Riemannian optimization techniques to match statistical properties between samples projected into the latent space from different domains.

In Table 3.1, we have calculated the accuracy of the OBTL classifier in 12 distinct experiments, where the source-target pairs are different (source $\rightarrow$ target) in each experiment. We have marked

Table 3.1: Semi-supervised accuracy for different source and target domains in the *Office+Caltech256* dataset using SURF features. Domain names are denoted as a: *amazon*, w: *webcam*, d: *dslr*, c: *Caltech256*. The numbers in red show the best accuracy and the numbers in blue show the second best accuracy in each column. The results of the first six methods have been adopted from [8]. Similar to [8], we have also used the evaluation setup of [9] for the OBTL. Reprinted with permission from [4], ©2018 IEEE.

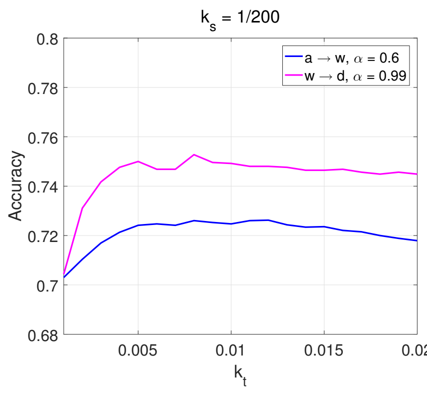
	a → w	a → d	a → c	w → a	w → d	w → c	d → a	d → w	d → c	c → a	c → w	c → d	Mean
1-NN-t	34.5	33.6	19.7	29.5	35.9	18.9	27.1	33.4	18.6	29.2	33.5	34.1	29.0
SVM-t	63.7	57.2	32.2	46.0	56.5	29.7	45.3	62.1	32.0	45.1	60.2	56.3	48.9
HFA [73]	57.4	55.1	31.0	**56.5**	56.5	29.0	42.9	60.5	30.9	43.8	58.1	55.6	48.1
MMDT [9]	64.6	56. 7	36.4	47.7	67.0	32.2	46.9	74.1	34.1	49.4	63.8	56.5	52.5
CDLS [75]	**68.7**	**60.4**	35.3	51.8	60.7	33.5	50.7	68.5	34.9	50.9	**66.3**	**59.8**	53.5
ILS (1-NN) [8]	59.7	49.8	**43.6**	54.3	**70.8**	**38.6**	**55.0**	**80.1**	**41.0**	**55.1**	62.9	56.2	**55.6**
OBTL	**72.4**	**60.2**	**41.5**	**55.0**	**75.0**	**37.4**	**54.4**	**83.2**	**40.3**	**54.8**	**71.1**	**61.5**	**58.9**

Table 3.2: The values of hyperparameter α of the OBTL used in each experiment. n_t and n_s are based on the data splits provided by [9]. Reprinted with permission from [4], ©2018 IEEE.

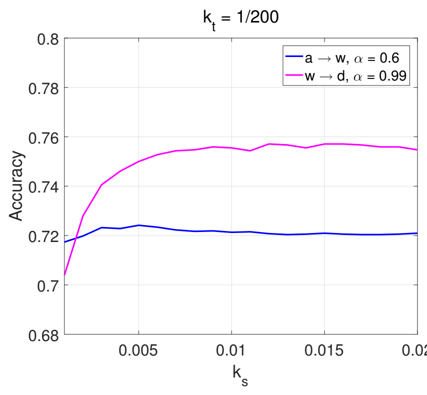
	a → w	a → d	a → c	w → a	w → d	w → c	d → a	d → w	d → c	c → a	c → w	c → d
n_t	3	3	3	3	3	3	3	3	3	3	3	3
n_s	20	20	20	8	8	8	8	8	8	8	8	8
α	0.6	0.75	0.99	0.9	0.99	0.99	0.9	0.99	0.99	0.85	0.5	0.75

the best accuracy in each column with red and the second best accuracy with blue. We see that the OBTL classifier has either the best or second best accuracy in all the 12 experiments. We have written the mean accuracy of each method in the last column, which has been averaged over all the 12 different experiments. The OBTL classifier has the best mean accuracy and the ILS [8] has the second best accuracy among all the methods. We have assumed equal prior probabilities for all the classes and used (3.32) for the OBTL classifier.

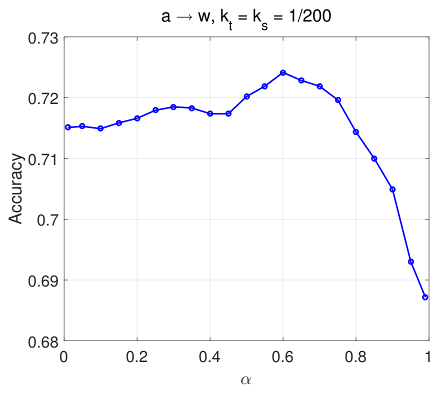
• **Hyperparameters of the OBTL:** We assume the same values of hyperparameters for all the 10 classes in each domain, so we can drop the superscript l denoting the class label. We set $\nu = 10d = 200$ for all the experiments. We choose α separately in each experiment since the relatedness between distinct pairs of domains are different. For $\mathbf{m}_t$ and $\mathbf{m}_s$, we pool all the target and source data in all the 10 classes, respectively, and use the sample means of the datasets. We fix $\beta_t = \beta_s = 1$ (meaning that $\kappa_t = n_t$ and $\kappa_s = n_s$) and $k_t = k_s = 1/\nu = 1/200 = 0.005$. The mean



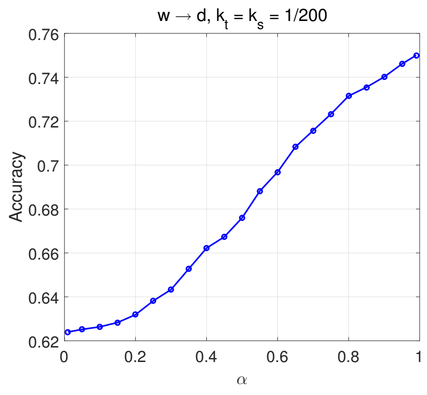
(a)



(b)



(c)



(d)

Figure 3.7: Accuracy in the *Office+Caltech256* dataset versus: (a) k_t when $k_s = 1/200$ and for two experiments $a \to w, \alpha = 0.6$ and $w \to d, \alpha = 0.99$, (b) k_s when $k_t = 1/200$ and for two experiments $a \to w, \alpha = 0.6$ and $w \to d, \alpha = 0.99$, (c) α when $k_t = k_s = 1/200$ and for the experiment $a \to w$, (d) α when $k_t = k_s = 1/200$ and for the experiment $w \to d$. Reprinted with permission from [4], ©2018 IEEE.

of the Wishart precision matrix $\mathbf{\Lambda}_z$, for $z \in \{s, t\}$, with scale matrix $\mathbf{M}_z$ and ν degrees of freedom is $\nu\mathbf{M}_z$. Consequently, $E(\mathbf{\Lambda}_t) = E(\mathbf{\Lambda}_s) = I_d$, which is a reasonable choice, since the provided datasets of [9] have been normally standardized. Therefore, the only hyperparameter and the most important one is α ($\in (0,1)$), which shows the relatedness between the two domains. Figs. 3.7a and 3.7b demonstrate that the accuracy is robust for $k_t \in (0.005, 0.02)$ and $k_s \in (0.005, 0.02)$, respectively. Figs. 3.7a and 3.7b are corresponding to two experiments: $a \to w, \alpha = 0.6$ and $w \to d, \alpha = 0.99$. Figs. 3.7c and 3.7d show interesting results. We have already seen similar behavior in the synthetic data as well. In the case of $a \to w$, accuracy grows smoothly by increasing α, reaches the maximum at $\alpha = 0.6$, and decreases afterwards. This verifies the fact that the source domain a cannot help the target domain w that much. On the contrary, accuracy increases monotonically in

Fig. 3.7d, in the case of $w \to d$, and the difference between accuracy for $\alpha = 0.01$ and $\alpha = 0.99$ is huge. This confirms that the source domain w is very related to the target domain d and helps it a lot. Interestingly, this coincides with the findings from the literature that the two domains w and d are highly related. We choose the values of α in each experiment which give the best accuracy. They are shown in Table 3.2. The values of α in Table 3.2 also reveal the amount of relatedness between any pairs of source and target domains. For example, both $w \to d$ and $d \to w$ have high relatedness with $\alpha = 0.99$, which has already been verified by other papers as well [72].

3.1.9 Conclusion

We constructed a Bayesian transfer learning framework to tackle the supervised transfer learning problem. The proposed Optimal Bayesian Transfer Learning (OBTL) classifier can deal with the lack of labeled data in the target domain and is optimal in this new Bayesian framework since it minimizes the expected classification error. We obtained the closed-form posterior distribution of the target parameters and accordingly the closed-form effective class-conditional densities in the target domain to define the OBTL classifier. As the OBTL's objective function consists of hypergeometric functions of matrix argument, we used the Laplace approximations of those functions to derive a computationally efficient and scalable OBTL classifier, while preserving its superior performance. We compared the performance of the OBTL with its target-only version, OBC, to see how transferring from source to target domain can help. We tested the OBTL classifier with real-world benchmark image datasets and demonstrated its excellent performance compared to other state-of-the-art domain adaption methods.

3.2 Optimal Bayesian Transfer Regression

3.2.1 Overview

Transfer learning studies effective ways to derive better predictors for a system of interest in a *target* domain, where there is lack of data, by utilizing data from other related systems as *source* domain(s). In this section we define a Bayesian transfer learning framework for regression to integrate data between the domains through a joint prior distribution for the source and target parameters. We derive closed-form posteriors of the target parameters integrating both the source and target data, from which closed-form effective joint distributions in the target domain can be derived in terms of generalized hypergeometric functions of matrix argument to define the Optimal Bayesian Transfer Regression (OBTR) operator. We show that the OBTR improves the mean squared error (MSE) when the source and target domains are related on both synthetic and real-world data.

3.2.2 Introduction

Traditional machine learning methods assume that the training and test data follow the same probability distribution and work poorly when that assumption does not hold. Transfer learning and domain adaptation techniques attempt to address this issue and have been studied in recent years [65-68]. Suppose we have two domains with different distributions, *target* domain and *source* domain. The goal is to design an operator (for classification or regression) in the target domain, assuming that the target domain has a very small number of training data. On the other side, there are plenty of training data in the source domain. If the source data are somehow related to the target data, leveraging those data can benefit the operator design in the target domain. Transfer learning strives to transfer related data and knowledge from the source to the target domain.

The Optimal Bayesian Classifier (OBC) minimizes the expected classification error over an uncertainty class of feature-label distributions [57, 58]. We followed the derivation of the OBC (which is for one domain) and developed a Bayesian framework for transfer learning in previous section, in which the relatedness between domains is defined via a joint prior distribution between the model parameters of the two domains. The Optimal Bayesian Transfer Learning (OBTL) [4] classifier optimally transfers the data and knowledge from source to target domain and yields the minimum expected classification error. The Optimal Bayesian Regression (OBR) proposed in [60] minimizes the expected Mean Squared Error (MSE) over an uncertainty class of joint distributions and outperforms traditional Bayesian Linear Regression (BLR) [94, 95]. In this section we adopt the Bayesian transfer learning framework [4] and propose the Optimal Bayesian Transfer Regression (OBTR) for the target domain utilizing data across domains. We show that it outperforms the OBR in terms of MSE when the data across domains are related. Unlike distribution-free transfer learning methods [10, 70], here we are interested in finding optimal regression operators for an assumed class of distributions. We particularly consider Gaussian distributions, with the benefit of having closed-form posterior distributions.

3.2.3 Preliminaries

Suppose $\mathbf{x}$ is a $d \times 1$ feature vector and y is the output. A joint process $F(y, \mathbf{x})$ governs the interactions of the output and the features. The goal is to find the optimal operator $\psi(\mathbf{x})$ for the observed point $\mathbf{x}$ which minimizes the MSE $\mathrm{E}[(y - \psi(\mathbf{x}))^2]$:

$$\psi(\mathbf{x}) = \underset{\psi \in \mathcal{F}}{\operatorname{argmin}}\, \mathrm{E}_F[(y - \psi(\mathbf{x}))^2], \tag{3.42}$$

where $\mathcal{F}$ denotes the class of all operators. It is well known that the optimal operator is the conditional expectation of y given $\mathbf{x}$, $\psi(\mathbf{x}) = \mathrm{E}_F[y|\mathbf{x}]$. If $\mathbf{x}$ and y follow a joint Gaussian distribution, the optimal operator becomes linear [60, 94, 95]. Let $\mathbf{z} = [y, \mathbf{x}^{'}]'$ be the joint $(d+1) \times 1$ vector which follows the multivariate Gaussian distribution $\mathbf{z} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$, where $\mu = [\mu_y, \mu_{\mathbf{x}}^{'}]'$ and $\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{yy} & \mathbf{\Sigma}_{y\mathbf{x}} \\ \mathbf{\Sigma}_{y\mathbf{x}}^{'} & \mathbf{\Sigma}_{\mathbf{xx}} \end{pmatrix}$. In this case, the distribution of y given $\mathbf{x}$ is another Gaussian: $y|\mathbf{x} \sim \mathcal{N}(\mu_{y|\mathbf{x}}, \Sigma_{y|\mathbf{x}})$, where $\mu_{y|\mathbf{x}} = \mu_y + \mathbf{\Sigma}_{y\mathbf{x}}\mathbf{\Sigma}_{\mathbf{xx}}^{-1}(\mathbf{x} - \mu_{\mathbf{x}})$ and $\mathrm{Var}(y|\mathbf{x}) = \mathbf{\Sigma}_{yy} - \mathbf{\Sigma}_{y\mathbf{x}}\mathbf{\Sigma}_{\mathbf{xx}}^{-1}\mathbf{\Sigma}_{y\mathbf{x}}^{'}$. As a result, the optimal operator (which is linear) and the minimum mean squared error (MMSE) are given by

$$\begin{aligned} \psi(\mathbf{x}) &= \mu_y + \mathbf{\Sigma}_{y\mathbf{x}}\mathbf{\Sigma}_{\mathbf{xx}}^{-1}(\mathbf{x} - \mu_{\mathbf{x}}), \\ \mathrm{E}_F[(y - \psi(\mathbf{x}))^2] &= \mathbf{\Sigma}_{yy} - \mathbf{\Sigma}_{y\mathbf{x}}\mathbf{\Sigma}_{\mathbf{xx}}^{-1}\mathbf{\Sigma}_{y\mathbf{x}}^{'}. \end{aligned} \tag{3.43}$$

Although we know the optimal operator as in (3.43), the true parameters μ_y, $\mu_{\mathbf{x}}$, $\mathbf{\Sigma}_{yy}$, $\mathbf{\Sigma}_{y\mathbf{x}}$, and $\mathbf{\Sigma}_{\mathbf{xx}}$ are unknown and need to be estimated from the training data. Let $\hat{\theta}$ denote the estimate of θ and $\psi_n(.)$ be the optimal operator with the estimated parameters from n training data. In this case,

$$\psi_n(\mathbf{x}) = \hat{\mu}_y + \hat{\mathbf{\Sigma}}_{y\mathbf{x}}\hat{\mathbf{\Sigma}}_{\mathbf{xx}}^{-1}(\mathbf{x} - \hat{\mu}_{\mathbf{x}}), \tag{3.44}$$

and its MSE $\mathrm{E}_F[(y - \psi_n(\mathbf{x}))^2]$ can be numerically found using test data $(y, \mathbf{x})$ from the process $F(y, \mathbf{x})$. When there is little training data, and n is small for the feature dimension d, the estimated parameters, especially the covariance matrices, will not be accurate, leading to a weak operator and higher MSE. Bayesian methods are employed to incorporate prior knowledge in order to compensate for the lack of data.

3.2.4 Optimal Bayesian Regression

The Optimal Bayesian Filter (OBF) and Optimal Bayesian Regression (OBR) assume that the joint process $(y, \mathbf{x})$ belongs to an uncertainly class of processes defined by the parameter set Θ, each $\theta \in \Theta$ corresponding to a distribution $F_\theta(y, \mathbf{x})$, and the aim is to minimize the MSE relative to Θ [60]. Let $\mathcal{D} = \{(y_1, \mathbf{x}_1^{'})^{'}, \cdots, (y_n, \mathbf{x}_n^{'})^{'}\}$ contain n random training samples and $\pi(\theta, \mathcal{D})$ denote the joint distribution over Θ and the sampling process. The dependence of $(y, \mathbf{x})$ on θ is denoted by writing $(y_\theta, \mathbf{x}_\theta)$. The OBR is defined as

$$\psi_{\mathrm{OBR}}(\mathbf{x}, \mathcal{D}) = \underset{\psi \in \mathcal{F}}{\mathrm{argmin}}\, \mathrm{E}_{\pi^*}\mathrm{E}_F[(y_\theta - \psi(\mathbf{x}_\theta))^2], \tag{3.45}$$

where the first expectation is relative to the posterior distribution of the parameters given the training data, $\pi^*(\theta) = \pi(\theta|\mathcal{D})$. The solution to (3.45) is the conditional expectation

$$\psi_{\mathrm{OBR}}(\mathbf{x}, \mathcal{D}) = \mathrm{E}_{\pi^*}\mathrm{E}_F[y|\mathbf{x}, \theta] = \mathrm{E}_{\pi^*}[\psi(\mathbf{x})|\theta], \tag{3.46}$$

where $\psi(\mathbf{x})$ is the optimal operator given in (3.42). In the case of the Gaussian distribution, $\psi(\mathbf{x})$ as in (3.43), the OBR is still linear and can be written as

$$\psi_{\text{OBR}}(\mathbf{x}, \mathcal{D}) = \mathrm{E}_{\pi^*}[\mu_y] - \mathrm{E}_{\pi^*}[\mathbf{\Sigma}_{y\mathbf{x}}\mathbf{\Sigma}_{\mathbf{xx}}^{-1}\mu_{\mathbf{x}}] + \mathrm{E}_{\pi^*}[\mathbf{\Sigma}_{y\mathbf{x}}\mathbf{\Sigma}_{\mathbf{xx}}^{-1}]\mathbf{x}. \tag{3.47}$$

It is shown in [60] that the OBR in (3.46) can also be written as

$$\psi_{\text{OBR}}(\mathbf{x}, \mathcal{D}) = \mathrm{E}_{F_{\text{eff}}^{\mathcal{D}}}[y|\mathbf{x}], \tag{3.48}$$

where $F_{\text{eff}}^{\mathcal{D}}(y, \mathbf{x})$ is the effective joint distribution of $(y, \mathbf{x})$,

$$F_{\text{eff}}^{\mathcal{D}}(y, \mathbf{x}) = \int_{\Theta} F_\theta(y, \mathbf{x})\pi^*(\theta)d\theta. \tag{3.49}$$

Suppose $\mathbf{z} = [y, \mathbf{x}']'$ follows the Gaussian distribution $\mathbf{z} \sim \mathcal{N}(\mu, \mathbf{\Lambda}^{-1})$, where $\mu = [\mu_y, \mu_{\mathbf{x}}']'$ and $\mathbf{\Lambda} = \mathbf{\Sigma}^{-1} = \begin{pmatrix} \mathbf{\Lambda}_{yy} & \mathbf{\Lambda}_{y\mathbf{x}} \\ \mathbf{\Lambda}_{y\mathbf{x}}' & \mathbf{\Lambda}_{\mathbf{xx}} \end{pmatrix}$ is the precision matrix. The uncertain parameters are $\theta = (\mu, \mathbf{\Lambda})$. We use the conjugate Gaussian-Wishart prior distribution for μ and $\mathbf{\Lambda}$: $\mathbf{\Lambda} \sim W_{d+1}(\mathbf{M}, \nu)$ and $\mu|\Lambda \sim \mathcal{N}(\mathbf{m}, (\kappa\mathbf{\Lambda})^{-1})$, where $\mathbf{M} = \begin{pmatrix} \mathbf{M}_{yy} & \mathbf{M}_{y\mathbf{x}} \\ \mathbf{M}_{y\mathbf{x}}' & \mathbf{M}_{\mathbf{xx}} \end{pmatrix} > 0$ is the positive-definite scale matrix, $\nu \geq d + 1$ is the degrees of freedom, $\mathbf{m} = [\mathbf{m}_y, \mathbf{m}_{\mathbf{x}}']'$ is the prior mean, and $\kappa > 0$ is a positive scalar. Due to the conjugacy, the posterior of θ is another Gaussian-Wishart distribution: $\mathbf{\Lambda}|\mathcal{D} \sim W_{d+1}(\mathbf{M}_n, \nu_n)$ and $\mu|\mathbf{\Lambda}, \mathcal{D} \sim \mathcal{N}(\mathbf{m}_n, (\kappa_n\mathbf{\Lambda})^{-1})$, where $\kappa_n = \kappa + n$, $\nu_n = \nu + n$, $\mathbf{m}_n = \frac{\kappa\mathbf{m}+n\bar{\mathbf{z}}}{\kappa+n}$, and $\mathbf{M}_n^{-1} = \mathbf{M}^{-1} + S + \frac{\kappa n}{\kappa+n}(\mathbf{m} - \bar{\mathbf{z}})(\mathbf{m} - \bar{\mathbf{z}})'$. Here $\bar{\mathbf{z}} = \frac{1}{n}\sum_{i=1}^n \mathbf{z}_i$ is the sample mean and $S = \sum_{i=1}^n (\mathbf{z}_i - \bar{\mathbf{z}})(\mathbf{z}_i - \bar{\mathbf{z}})'$ is the sample covariance matrix. Similar to the derivations in [4,57], the effective joint distribution can be derived as a multivariate student's t-distribution

$$F_{\text{eff}}^{\mathcal{D}}(\mathbf{z}) = \mathcal{T}\left(\nu_n - d, \mathbf{m}_n, \left(\frac{\kappa_n(\nu_n - d)}{\kappa_n + 1}\mathbf{M}_n\right)^{-1}\right). \tag{3.50}$$

Lemma 4. *[96] Suppose $\mathbf{x} \in \mathbb{R}^d$ has a multivariate t-distribution $\mathcal{T}(\mathbf{x}; \nu, \mu, \Sigma)$ with $\nu > 0$*

degrees of freedom, the location $\mu \in \mathbb{R}^d$, and the scale matrix $\mathbf{\Sigma} \in \mathbb{R}^{d\times d}$,

$$p(\mathbf{x}|\nu, \mu, \mathbf{\Sigma}) = \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\frac{\nu}{2})(\nu\pi)^{\frac{d}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} \times \left(1 + \frac{1}{\nu}(\mathbf{x}-\mu)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right)^{-\frac{\nu+d}{2}}. \tag{3.51}$$

If $\mathbf{x} = [\mathbf{x}_1', \mathbf{x}_2']'$ with the dimensions of $\mathbf{x}_1$ and $\mathbf{x}_2$ being d_1 and d_2, respectively, and with the corresponding partitions of the location $\mu = [\mu_1', \mu_2']'$ and scale matrix $\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} \\ \mathbf{\Sigma}_{12}' & \mathbf{\Sigma}_{22} \end{pmatrix}$, the conditional distribution of $\mathbf{x}_1$ given $\mathbf{x}_2$ is a t-distribution $p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{T}(\mathbf{x}_1; \nu_{1|2}, \mu_{1|2}, \mathbf{\Sigma}_{1|2})$, where

$$\nu_{1|2} = \nu + d_2, \quad \mu_{1|2} = \mu_1 + \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \mu_2),$$
$$\Sigma_{1|2} = \frac{\nu + (\mathbf{x}_2-\mu_2)'\mathbf{\Sigma}_{22}^{-1}(\mathbf{x}_2-\mu_2)}{\nu + d_2}(\mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}\mathbf{\Sigma}_{12}').$$

Using (3.48), (3.50), and Lemma 4, the OBR is derived as

$$\psi_{\text{OBR}}(\mathbf{x}) = \mathbf{m}_{n,y} + \Phi_{y\mathbf{x}}\Phi_{\mathbf{x}\mathbf{x}}^{-1}(\mathbf{x} - \mathbf{m}_{n,\mathbf{x}}), \tag{3.52}$$

where $\mathbf{m}_n = [\mathbf{m}_{n,y}, \mathbf{m}_{n,\mathbf{x}}']'$ and $\Phi = \begin{pmatrix} \Phi_{yy} & \Phi_{y\mathbf{x}} \\ \Phi_{y\mathbf{x}}' & \Phi_{\mathbf{x}\mathbf{x}} \end{pmatrix} = \frac{\kappa_n+1}{\kappa_n(\nu_n-d)}\mathbf{M}_n^{-1}$.

3.2.5 Optimal Bayesian Transfer Regression

Suppose there are a target t and a source s domain, where $\mathbf{z}_z = [y_z, \mathbf{x}_z']'$ and $\mathbf{z}_z \sim \mathcal{N}(\mu_z, \mathbf{\Lambda}_z^{-1})$ for $z \in \{t, s\}$. We constructed in [4] a Bayesian transfer learning framework and defined the relatedness between the domains by a joint prior distribution for the parameters of the models. We adopt some results of [4] here for the regression problem and refer the readers to [4] for the detailed definitions and proofs, due to the lack of space. Suppose the joint prior distribution

$$p(\mu_t, \mu_s, \mathbf{\Lambda}_t, \mathbf{\Lambda}_s) = p(\mu_t|\mathbf{\Lambda}_t)p(\mu_s|\mathbf{\Lambda}_s)p(\mathbf{\Lambda}_t, \mathbf{\Lambda}_s), \tag{3.53}$$

where it is assumed μ_t and μ_s are conditionally independent given $\mathbf{\Lambda}_t$ and $\mathbf{\Lambda}_s$ in order to have conjugacy and closed-form posteriors. Similar to the one-domain scenario,

$$\mu_z | \mathbf{\Lambda}_z \sim \mathcal{N}(\mathbf{m}_z, (\kappa_z \mathbf{\Lambda}_z)^{-1}). \tag{3.54}$$

The key issue in [4] is to define a joint prior for the two precision matrices $\mathbf{\Lambda}_t$ and $\mathbf{\Lambda}_s$ that bridges the two domains in a fully Bayesian framework. The joint distribution for $\mathbf{\Lambda}_t$ and $\mathbf{\Lambda}_s$ is defined as

$$\begin{aligned} p(\mathbf{\Lambda}_t, \mathbf{\Lambda}_s) = K \text{etr}\left(-\frac{1}{2}\left(\mathbf{M}_t{}^{-1} + \mathbf{F}'\mathbf{C}\mathbf{F}\right)\mathbf{\Lambda}_t\right) \\ \times \text{ etr}\left(-\frac{1}{2}\mathbf{C}^{-1}\mathbf{\Lambda}_s\right) |\mathbf{\Lambda}_t|^{\frac{\nu-d-2}{2}} |\mathbf{\Lambda}_s|^{\frac{\nu-d-2}{2}} \ {}_0F_1\left(\frac{\nu}{2}; \frac{1}{4}\mathbf{G}\right), \end{aligned} \tag{3.55}$$

where $\mathbf{M} = \begin{pmatrix} \mathbf{M}_t & \mathbf{M}_{ts} \\ \mathbf{M}'_{ts} & \mathbf{M}_s \end{pmatrix}$ is a $2(d+1) \times 2(d+1)$ positive definite scale matrix, $\nu \geq 2(d+1)$ denotes degrees of freedom, $\mathbf{C} = \mathbf{M}_s - \mathbf{M}'_{ts}\mathbf{M}_t{}^{-1}\mathbf{M}_{ts}$, $\mathbf{F} = \mathbf{C}^{-1}\mathbf{M}'_{ts}\mathbf{M}_t^{-1}$, $\mathbf{G} = \mathbf{\Lambda}_s{}^{\frac{1}{2}}\mathbf{F}\mathbf{\Lambda}_t\mathbf{F}'\mathbf{\Lambda}_s{}^{\frac{1}{2}}$, and $K^{-1} = 2^{(d+1)\nu}\Gamma^2_{d+1}\left(\frac{\nu}{2}\right)|\mathbf{M}|^{\frac{\nu}{2}}$. The function ${}_0F_1$ is called the matrix-variate generalized hypergeometric function [87]. Furthermore, the marginal distributions of the target and source precision matrices are Wishart: $\mathbf{\Lambda}_z \sim W_{d+1}(\mathbf{M}_z, \nu)$ for $z \in \{t, s\}$.

Let $\mathcal{D}_z = \{\mathbf{z}_{z,1}, \cdots, \mathbf{z}_{z,n_z}\}$ denote n_z training data in the domain $z \in \{t, s\}$. Since we have defined a joint prior, we will also have a joint posterior distribution for μ_t, μ_s, $\mathbf{\Lambda}_t$, and $\mathbf{\Lambda}_s$ given the two datasets $\mathcal{D}_t$ and $\mathcal{D}_s$. However, as our aim is to design an operator in the target domain, the source parameters are integrated out in the joint posterior, yielding the posterior distribution of the target parameters μ_t and $\mathbf{\Lambda}_t$. Having the posterior in the target domain, we use (3.49) to derive the effective joint distribution of $\mathbf{z} = [y, \mathbf{x}']'$ in the target domain. The following theorems are used in the OBTR, with the detailed proofs in [4].

Theorem 5. *Given the target $\mathcal{D}_t$ and source $\mathcal{D}_s$ data, the posterior distribution of target mean μ_t and target precision matrix $\mathbf{\Lambda}_t$ has a Gaussian-Hypergeometric-function distribution*

$$\begin{aligned} & p(\mu_t, \mathbf{\Lambda}_t | \mathcal{D}_t, \mathcal{D}_s) = \\ & A |\mathbf{\Lambda}_t|^{\frac{1}{2}} \exp\left(-\frac{\kappa_{t,n}}{2}(\mu_t - \mathbf{m}_{t,n})' \mathbf{\Lambda}_t (\mu_t - \mathbf{m}_{t,n})\right) \\ & \times |\mathbf{\Lambda}_t|^{\frac{\nu+n_t-d-1}{2}} \text{etr}\left(-\frac{1}{2}\mathbf{T}_t^{-1}\mathbf{\Lambda}_t\right) \\ & \times \ {}_1F_1\left(\frac{\nu+n_s}{2}; \frac{\nu}{2}; \frac{1}{2}\mathbf{F}\mathbf{\Lambda}_t\mathbf{F}'\mathbf{T}_s\right), \end{aligned} \tag{3.56}$$

where A is the constant of proportionality

$$A^{-1} = \left(\frac{2\pi}{\kappa_{t,n}}\right)^{\frac{d+1}{2}} 2^{\frac{(d+1)(\nu+n_t)}{2}} \Gamma_{d+1}\left(\frac{\nu+n_t}{2}\right) |\mathbf{T}_t|^{\frac{\nu+n_t}{2}} \\ \times \ {}_2F_1\left(\frac{\nu+n_s}{2}, \frac{\nu+n_t}{2}; \frac{\nu}{2}; \mathbf{T}_s\mathbf{F}\mathbf{T}_t\mathbf{F}^{'}\right), \tag{3.57}$$

and

$$\kappa_{t,n} = \kappa_t + n_t, \qquad \mathbf{m}_{t,n} = \frac{\kappa_t\mathbf{m}_t + n_t\bar{\mathbf{z}}_t}{\kappa_t + n_t}, \\ \mathbf{T}_t^{-1} = \mathbf{M}_t^{-1} + \mathbf{F}^{'}\mathbf{C}\mathbf{F} + \mathbf{S}_t \\ + \frac{\kappa_t n_t}{\kappa_t + n_t}(\mathbf{m}_t - \bar{\mathbf{z}}_t)(\mathbf{m}_t - \bar{\mathbf{z}}_t)^{'}, \\ \mathbf{T}_s^{-1} = \mathbf{C}^{-1} + \mathbf{S}_s + \frac{\kappa_s n_s}{\kappa_s + n_s}(\mathbf{m}_s - \bar{\mathbf{z}}_s)(\mathbf{m}_s - \bar{\mathbf{z}}_s)^{'}, \tag{3.58}$$

with sample means and covariances for $z \in \{s, t\}$ as

$$\bar{\mathbf{z}}_z = \frac{1}{n_z}\sum_{i=1}^{n_z}\mathbf{z}_{z,i}, \quad \mathbf{S}_z = \sum_{i=1}^{n_z}\left(\mathbf{z}_{z,i} - \bar{\mathbf{z}}_z\right)\left(\mathbf{z}_{z,i} - \bar{\mathbf{z}}_z\right)^{'}.$$

Theorem 6. *The effective joint distribution of $\mathbf{z} = [y, \mathbf{x}^{'}]^{'}$ in the target domain is given by*

$$F_{\text{eff}}^{\mathcal{D}_t,\mathcal{D}_s}(\mathbf{z}) = \pi^{-\frac{d+1}{2}} \left(\frac{\kappa_{t,n}}{\kappa_{\mathbf{z}}}\right)^{\frac{d+1}{2}} \Gamma_{d+1}\left(\frac{\nu+n_t+1}{2}\right) \\ \times \Gamma_{d+1}^{-1}\left(\frac{\nu+n_t}{2}\right) |\mathbf{T}_{\mathbf{z}}|^{\frac{\nu+n_t+1}{2}} |\mathbf{T}_t|^{-\frac{\nu+n_t}{2}} \\ \times \ {}_2F_1\left(\frac{\nu+n_s}{2}, \frac{\nu+n_t+1}{2}; \frac{\nu}{2}; \mathbf{T}_s\mathbf{F}\mathbf{T}_{\mathbf{z}}\mathbf{F}^{'}\right) \\ \times \ {}_2F_1^{-1}\left(\frac{\nu+n_s}{2}, \frac{\nu+n_t}{2}; \frac{\nu}{2}; \mathbf{T}_s\mathbf{F}\mathbf{T}_t\mathbf{F}^{'}\right), \tag{3.59}$$

where

$$\kappa_{\mathbf{z}} = \kappa_{t,n} + 1 = \kappa_t + n_t + 1, \\ \mathbf{T}_{\mathbf{z}}^{-1} = \mathbf{T}_t^{-1} + \frac{\kappa_{t,n}}{\kappa_{t,n}+1}\left(\mathbf{m}_{t,n} - \mathbf{z}\right)\left(\mathbf{m}_{t,n} - \mathbf{z}\right)^{'}. \tag{3.60}$$

The functions ${}_1F_1$ and ${}_2F_1$ utilized in Theorems 5 and 6 are called Confluent and Gauss hypergeometric functions of matrix argument, respectively [87]. Having derived the effective joint distribution in our transfer learning framework, we can define the Optimal Bayesian Transfer Regression (OBTR) operator, similar to the OBR in (3.48), as

$$\psi_{\mathrm{OBTR}}(\mathbf{x}, \mathcal{D}_t, \mathcal{D}_s) = \mathrm{E}_{F_{\mathrm{eff}}^{\mathcal{D}_t,\mathcal{D}_s}}[y|\mathbf{x}]. \tag{3.61}$$

Although we have derived a closed-form joint distribution for $\mathbf{z} = [y, \mathbf{x}']'$ in Theorem 6, deriving the closed-form conditional expectation $\mathrm{E}_{F_{\mathrm{eff}}^{\mathcal{D}_t,\mathcal{D}_s}}[y|\mathbf{x}]$ in (3.61) is not straightforward if not impossible, since $F_{\mathrm{eff}}^{\mathcal{D}_t,\mathcal{D}_s}(\mathbf{z})$ includes the Gauss hypergeometric functions. Based on the Bayes rule, the conditional distribution is proportional to the joint distribution. Given any test point $\mathbf{x}$, we can sample from $F_{\mathrm{eff}}^{\mathcal{D}_t,\mathcal{D}_s}(y|\mathbf{x}) \propto F_{\mathrm{eff}}^{\mathcal{D}_t,\mathcal{D}_s}(y, \mathbf{x}) = F_{\mathrm{eff}}^{\mathcal{D}_t,\mathcal{D}_s}(\mathbf{z})$ using a Markov Chain Monte Carlo (MCMC) method and consequently can compute $\mathrm{E}_{F_{\mathrm{eff}}^{\mathcal{D}_t,\mathcal{D}_s}}[y|\mathbf{x}]$ by averaging the generated samples; however, it is not efficient since the costly MCMC process have to be done for every test data point.

We propose a method to use only $d + 1$ MCMC processes, which can considerably reduce the computations. Although we do not know the exact formula for the OBTR in (3.61), we know from (3.47) that it is linear in $\mathbf{x}$, since only the posterior distributions of the target parameters are different from the OBR. Therefore, the OBTR has the general linear form

$$\psi_{\mathrm{OBTR}}(\mathbf{x}, \mathcal{D}_t, \mathcal{D}_s) = \alpha + \beta'\mathbf{x}, \tag{3.62}$$

where $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}^d$. We use $d + 1$ test points to determine α and β. Let $\mathbf{0}_d$ denote the all-zero vector. From (3.61) and (3.62), α is derived as $\alpha = \mathrm{E}_{F_{\mathrm{eff}}^{\mathcal{D}_t,\mathcal{D}_s}}[y|\mathbf{0}_d]$. Let $\mathbf{e}_i = [0, \cdots, 1, \cdots, 0]'$ denote the basis vector whose ith entry is one and the rest are zero. The ith entry of the vector β can be found from (3.61) and (3.62) as $\beta_i = \mathrm{E}_{F_{\mathrm{eff}}^{\mathcal{D}_t,\mathcal{D}_s}}[y|\mathbf{e}_i] - \alpha$ for each $i \in \{1, \cdots, d\}$. We use Hamiltonian Monte Carlo (MHC) [97], which is more accurate than the Metropolis-Hasting random walk samplers for complex functions since it takes into account the derivative information of the functions as well. Since direct evaluation of the Gauss hypergeometric function is time-consuming, we use its Laplace approximation, proposed in [4], to speed up the HMC computations. Once α and β are derived, there is no need for HMC, and (3.62) can be used to predict any new test point.

3.2.6 Experiments

3.2.6.1 Synthetic data

We first compare the MSE results by OBTR and OBR based on simulated data to see how transfer learning can help improve the MSE in the target domain, especially when the number of target training data is small. Let $\mathbf{S}_t$ and $\mathbf{S}_s$ be our prior beliefs for the covariance matrices of the target and source domains, respectively, which marginally follow Inverse-Wishart distributions. As a result, we choose $\mathbf{M}_t^{-1} = (\nu - 2(d+1) - 1)\mathbf{S}_t = (\nu - 2d - 3)\mathbf{S}_t$ and similarly $\mathbf{M}_s^{-1} = (\nu - 2d - 3)\mathbf{S}_s$. In the OBTR framework, $\mathbf{M}_{ts}$ has the key transferring role but there is a constraint that the scale matrix $\mathbf{M} = \begin{pmatrix} \mathbf{M}_t & \mathbf{M}_{ts} \\ \mathbf{M}_{ts}' & \mathbf{M}_s \end{pmatrix}$ should be positive definite, or equivalently, $\mathbf{M}_t - \mathbf{M}_{ts}\mathbf{M}_s^{-1}\mathbf{M}_{ts}' > 0$. For satisfying the positive definiteness of $\mathbf{M}$, we consider a special structure which is easier to follow. Suppose $\mathbf{S}_t = \mathbf{A}$ and $\mathbf{S}_s = k_s\mathbf{A}$, where $\mathbf{A} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a symmetric positive definite matrix and $k_s > 0$. For this structure it can easily be shown that $\mathbf{M} > 0$ if $\mathbf{M}_{ts} = \tau[\sqrt{k_s}(\nu - 2d - 3)\mathbf{A}]^{-1}$ and $|\tau| < 1$. We see τ as a representative for the relatedness between the source and target domains. When $\tau = 0$, we have $\mathbf{M}_{ts} = \mathbf{0}$, and similar to [4] where we proved that the OBTL reduces to the OBC, here the OBTR becomes identical with the OBR. As a result, there is no transfer learning when $\tau = 0$. However, when $|\tau|$ is closer to one, we have a stronger bridge and higher relatedness between the domains, and the abundance of data in the source domain can lead to a much better MSE in the target domain. We consider two simulated examples. In both cases, we set $d = 10$, $\nu = 25$, $\kappa_t = \kappa_s = 100$, $\mathbf{m}_t = \mathbf{0}_d$ and $\mathbf{m}_s = 2 \times \mathbf{1}_d$. In case 1, we use $\mathbf{A} = 5 \times I_d$, $k_s = 1$, and $\tau = 0.9$. In case 2, we use $k_s = 2$, $\tau = 0.95$, $\mathbf{A}_{ii} = 2$, and $\mathbf{A}_{ij} = \rho\sqrt{\mathbf{A}_{ii}\mathbf{A}_{jj}}$, for any $i, j = 1, \cdots, d+1$ with $\rho = 0.7$.

To sample from the prior (3.53) we first sample from a Wishart distribution $W_{2(d+1)}(\mathbf{M}, \nu)$ to get a sample for $\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_t & \mathbf{\Lambda}_{ts} \\ \mathbf{\Lambda}_{ts}' & \mathbf{\Lambda}_s \end{pmatrix}$ and then pick $(\mathbf{\Lambda}_t, \mathbf{\Lambda}_s)$, which is a joint sample from $p(\mathbf{\Lambda}_t, \mathbf{\Lambda}_s)$ in (3.55). Then given $\mathbf{\Lambda}_t$ and $\mathbf{\Lambda}_s$, we sample from (3.54) to get samples of μ_t and μ_s. Once we have μ_t, μ_s, $\mathbf{\Lambda}_t$, and $\mathbf{\Lambda}_s$, we generate 20 different training and test sets. Training sets contain samples from both the target and source domains, but the test set contains only samples from the target

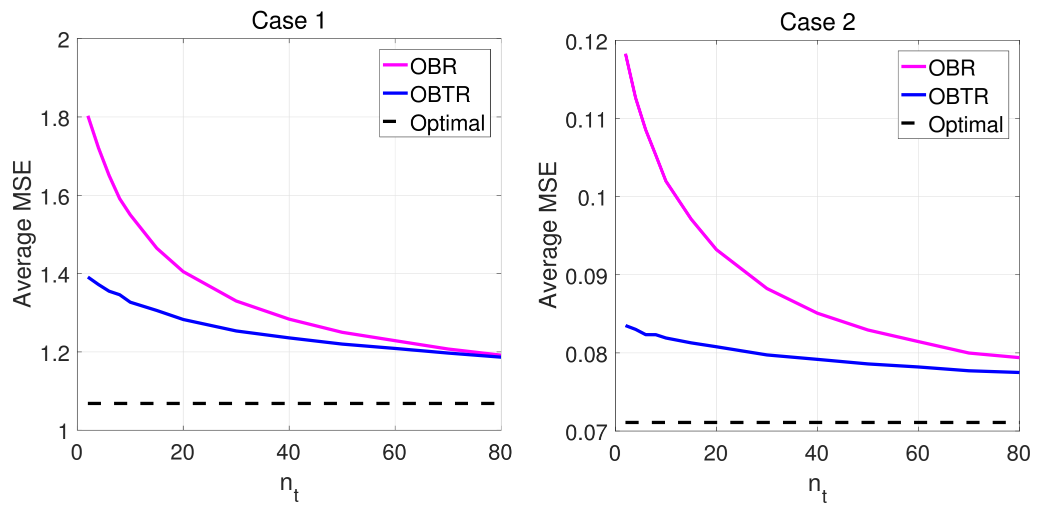


Figure 3.8: Average MSE versus n_t for the two cases, assuming $n_s = 500$. Reprinted with permission from [5], ©2018 IEEE.

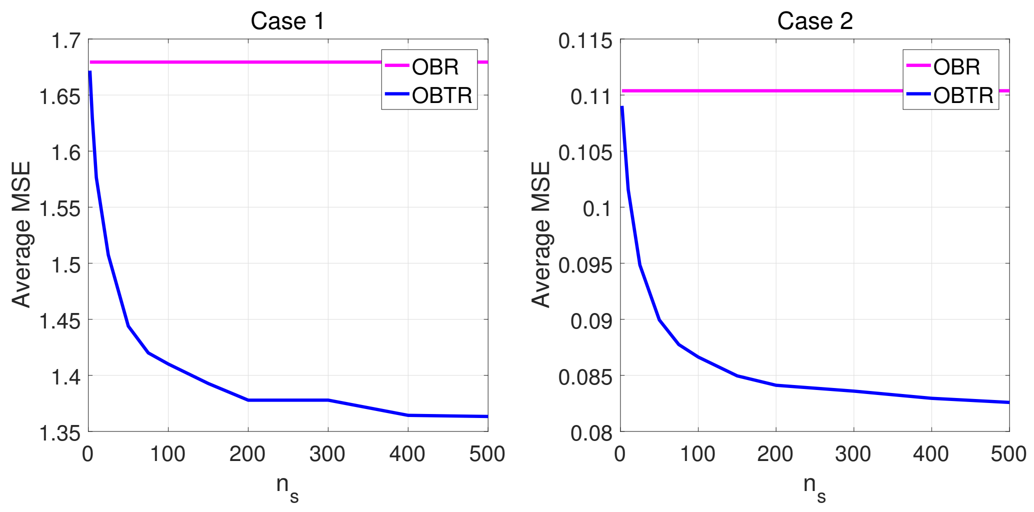


Figure 3.9: Average MSE versus n_s for the two cases, assuming $n_t = 5$. Reprinted with permission from [5], ©2018 IEEE.

domain. We set the size of the test set at 10^4 in the simulations. For each training and test set, we use the OBTR and its target-only version, OBR, and calculate the MSE. Then we average all the errors for 20 different training and test sets. We further repeat this whole process 100 times for different realizations of $\mathbf{\Lambda}_t$, $\mathbf{\Lambda}_s$, μ_t, and μ_s, and finally average all the MSE's and return the average MSE.

Figure 3.8 shows the average MSE versus n_t for the OBTR and OBR, assuming $n_s = 500$. We also include the average MSE of the optimal operator with known true parameters for the sake of comparison. We see that for both cases, the OBTR has better performance than the OBR in the range of small n_t, which is the reasonable region in transfer learning scenarios with lack of data in the target domain. The OBTR and OBR converge to each other and both to the optimal MSE when $n_t \to \infty$, meaning that we have enough data in the target domain to learn the model well without the need of source data and transfer learning. Figure 3.9 demonstrates the average MSE versus n_s for the OBTR and OBR in both cases, assuming $n_t = 5$. The average MSE of the OBR is constant, as it does not use the source data. For the OBTR, we see that its average MSE decreases monotonically with respect to n_s and then converges to a value which pertains to the amount of relatedness between the domains.

3.2.6.2 Real-world data

We further test the OBTR and OBR on five benchmark regression datasets: Concrete, Housing, Automobile MPG, Protein, and Yacht, available in the UCI repository. Authors in [10] have generated three different sets of source and target data for each of these datasets. Thus, there are three cases for each dataset, and for each case one set is considered as the target data and the other two as the source data. Table 3.3 shows the Root Mean Squared (RMS) error of the OBR and OBTR with $\tau = 0.3, 0.5, 0.7, 0.9, 0.99$ averaged over 100 runs. In each run, we randomly choose $n_t = 10$ training target data, $n_s = 200$ training source data, and 100 test target data. We first normalize all the features and output in each domain to have zero mean and standard deviation one. For all the cases, we see that the OBTR outperforms the OBR thanks to the relatedness between the domains and transfer learning. Since the data are normalized, we choose the hyperparameters as: $\mathbf{m}_t = \mathbf{m}_s = \mathbf{0}_d$, $\mathbf{A} = I_d$ and $k_s = 1$. We also choose $\kappa_t = n_t$, $\kappa_s = n_s$, and $\nu = 3(d+1)$.

3.2.7 Conclusion

In this section, we studied Bayesian transfer learning for regression. We constructed a fully Bayesian transfer learning framework by defining a joint prior distribution over the model parameters of the target and source domains. We proposed the OBTR which optimally transfers the useful

Table 3.3: Average RMS errors on five UCI datasets, each divided into three cases as in [10]. In each case, target and source data are different. Bold font marks the lowest error for the OBR and OBTR with different values of τ. In each case, $n_t = 10$ and $n_s = 200$. C 1 means Case 1. Reprinted with permission from [5], ©2018 IEEE.

Method	Concrete ($d = 7$)			Housing ($d = 12$)			MPG ($d = 6$)			Protein ($d = 8$)			Yacht ($d = 5$)		
	C 1	C 2	C 3	C 1	C 2	C 3	C 1	C 2	C 3	C 1	C 2	C 3	C 1	C 2	C 3
OBR	11.35	12.48	14.87	5.10	5.04	8.57	2.78	3.53	4.65	5.37	5.93	6.34	0.42	1.74	16.28
OBTR, $\tau = 0.3$	11.24	12.28	14.73	4.99	4.89	8.37	2.75	3.50	4.61	5.35	5.92	6.30	0.42	1.73	16.21
OBTR, $\tau = 0.5$	11.06	11.99	14.39	4.73	4.69	8.08	2.68	3.46	4.48	5.31	5.85	6.24	0.41	1.72	16.08
OBTR, $\tau = 0.7$	10.77	11.48	13.93	4.45	4.41	**7.97**	2.61	3.36	4.35	5.23	5.74	6.08	0.41	1.70	15.87
OBTR, $\tau = 0.9$	**10.51**	10.93	13.29	4.22	4.16	8.25	2.52	3.23	4.21	5.10	5.56	5.92	0.39	1.65	15.29
OBTR, $\tau = 0.99$	10.65	**10.92**	**13.15**	**4.21**	**4.07**	8.59	**2.40**	**3.09**	**4.16**	**5.06**	**5.48**	**5.87**	**0.38**	**1.62**	**14.86**

knowledge from the source to target domain. We showed by synthetic and real-world data that the OBTR outperforms OBR, its target-only counterpart.

3.3 Optimal Bayesian Transfer learning for Count Data

3.3.1 Overview

There is often limited amount of omics data to design predictive models in biomedicine. Knowing that these omics data come from underlying processes that may share common pathways and disease mechanisms, it may be beneficial for designing a more accurate and reliable predictor in a *target* domain of interest, where there is a lack of labeled data, to leverage available data in relevant *source* domains. In this section, we focus on developing *Bayesian transfer learning* methods for analyzing next-generation sequencing (NGS) data to help improve predictions in the target domain. We formulate transfer learning in a fully Bayesian framework and define the relatedness by a joint prior distribution of the model parameters of the source and target domains. Defining joint priors acts as a bridge across domains, through which the related knowledge of source data is transferred to the target domain. We focus on RNA-seq discrete count data, which are often overdispersed. To appropriately model them, we consider the Negative Binomial model and propose an Optimal Bayesian Transfer Learning (OBTL) classifier that minimizes the expected classification error in the target domain. We evaluate the performance of the OBTL classifier via both synthetic and cancer data from The Cancer Genome Atlas (TCGA).

3.3.2 Introduction

Due to the high dimensionality of omics data, effective predictive models demand a great number of relevant samples, which are difficult and costly to collect. This puts machine learning in bioinformatics in small-sample scenarios [15, 98], imposing critical concerns on the reproducibility of results derived from omics data. It is desired to leverage the shared knowledge of various sources of omics data to design a better predictor in a target domain of interest which suffers from lack of samples. Since the distributions of the source and target data may differ significantly, training on source data and testing on target data will give higher error. Transfer learning and domain adaptation have been introduced to address this issue [65, 67, 68]. Suppose we have two different domains with different distributions, target and source domains. The goal is to design a classifier in the target domain and evaluate it with the target test data, assuming that the target domain has a very small number of labeled data for training. There may be available labeled training data in other relevant source domains, for example, from relevant disease models or similar studies. Leveraging those data can benefit the training of the classifier in the target domain by transferring useful knowledge from the source to the target domain.

Recently, transfer learning has been used in genomics in order to bring knowledge from mice to the human domain [99]. The authors in [99] have exploited gene expression data for diseases in mice to improve the predictions for similar human diseases and to derive more confident differential gene expression analysis. In [100] the authors have studied how multi-task learning could better predict phenotype-gene associations in Human Phenotype Ontology (HPO) by effectively summarizing the ontology structure, and how transfer learning across HPO and Gene Ontology (GO) could bring useful training information. The study in [101] presents two novel approaches for incorporating information from a secondary domain for improving cancer drug sensitivity prediction in a target domain. The first approach in [101] is based on latent variable cost optimization and the second approach considers polynomial mapping between the two databases. The study in [71] has proposed a transfer learning method for classification and biomarker discovery which transfers the classifications rules between domains and shows better performance in genomics and proteomics than the target-only dataset and also the union of the datasets. The study in [70] considers transfer learning for degenerate biological systems. Degeneracy refers to the phenomenon that structurally different elements of the system perform the same/similar function or yield the same/similar output; the authors of [70] integrate transfer learning and degeneracy under a Bayesian framework and present good predictive accuracy for the relationship between transcription factors and gene expression across multiple cell lines.

Although there are many transfer learning methods in the literature, there is lack of a clear definition of the relatedness between two domains and a rigorous theoretical approach in the sense of minimizing the transfer-based classification error. Most endeavors in this field are based on domain adaptation techniques, which derive another common space in which the target and source data follow a similar distribution after a space transformation, regardless of considering the classification errors. On previous sections, we followed the framework of the Optimal Bayesian Classifier (OBC) [57, 58], which minimizes the expected classification error over an uncertainty class of feature-label distributions in one domain, and developed a Bayesian framework for transfer learning in [4,5], where the relatedness between domains is defined in terms of a joint prior distribution between the model parameters of the corresponding feature-label distributions in the two domains. We then proposed the Optimal Bayesian Transfer Learning (OBTL) classifier, which optimally transfers the knowledge from the source to the target domain and yields the minimum expected classification error. In [4], we addressed the transfer learning for continuous data with the assumption of Gaussian feature-label distributions and derived the OBTL classifier in a closed form in terms of hypergeometric functions of matrix argument.

In this section, we develop an OBTL classifier for count data from next-generation sequencing (NGS), which has been ubiquitous in modern biology and medicine research. NGS provides access to genome-scale expression profiles, such as RNA-seq count data in different samples and conditions, to help derive systematic understanding of cellular mechanisms. As noted, transfer learning has been considered in different biological problems [70,71,102], for instance, to transfer knowledge from prevalent diseases with much more labeled data to design accurate classifiers for those for which there is little data.

Owing to both technical and biological variations in NGS count data, the Negative Binomial (NB) distribution is well suited for modeling those data and accounts for large dispersion in the data [50, 103-105]. Therefore, we use NB to model feature-label distributions. Our goal is to design an optimal classifier in the target domain using a few labeled target training data as well as available labeled source training data. We define joint prior distributions between the NB mean parameters as well as between the NB shape (inverse of dispersion) parameters in the two domains. Having defined such a joint prior, the posterior distributions of the parameters in the two domains can be obtained simultaneously. As a result, when the source data are related to the target data, they can improve the inference of the target posteriors and lead to a more accurate classifier in the target domain. Unlike [4], in this section we have no closed-form OBTL classifier. We employ Hamiltonian Monte Carlo (HMC) [97], which is a leading Markov Chain Monte Carlo (MCMC) method, to derive the posterior samples and effective class-conditional densities, on which the OBTL classifier is defined. We compare the OBTL to its corresponding target domain OBC to see when and how transfer learning and source data can help reach a lower error. We also evaluate the OBTL classifier on lung cancer data from The Cancer Genome Atlas (TCGA) [106] to classify two subtypes of non-small cell lung cancer (NSCLC), lung adenocarcinoma (LUAD) versus lung squamous cell carcinoma (LUSC), and observe its performance improvement. We further assess the performance of the OBTL classifier using kidney cancer data from TCGA for the classification of two main subtypes of kidney cancer called kidney renal clear cell carcinoma (KIRC) and kidney renal papillary cell carcinoma (KIRP).

3.3.3 Method

3.3.3.1 Bayesian Transfer Learning Framework for Count Data

We consider a supervised transfer learning problem in which there are L common classes (labels) in each domain. Let $\mathcal{D}_s$ and $\mathcal{D}_t$ denote the labeled datasets of the source and target domains with sizes of N_s and N_t, respectively, where $N_t \ll N_s$. Let $\mathcal{D}_s^l = \left\{\mathbf{x}_{s,i,j}^l\right\}$, where $l \in \{1, \cdots, L\}$, $i \in \{1, \cdots, d\}$, and $j \in \{1, \cdots, n_s^l\}$, and $\mathcal{D}_s^l$ contains the n_s^l data points with the counts of d genes in the source domain for the class l. Similarly, let $\mathcal{D}_t^l = \left\{\mathbf{x}_{t,i,j}^l\right\}$, where $l \in \{1, \cdots, L\}$, $i \in \{1, \cdots, d\}$, and $j \in \{1, \cdots, n_t^l\}$, and $\mathcal{D}_t^l$ contains the n_t^l data points with the counts of d genes in the target domain for the class l. Therefore, $\mathcal{D}_s = \bigcup_{l=1}^L \mathcal{D}_s^l$, $\mathcal{D}_t = \bigcup_{l=1}^L \mathcal{D}_t^l$, $N_s = \sum_{l=1}^L n_s^l$, and $N_t = \sum_{l=1}^L n_t^l$. We use the Negative Binomial distribution to model the gene expression count data, which are often overdispersed, in each domain:

$$\mathbf{x}_{z,i,j}^l \sim \mathrm{NB}(\mu_{z,i}^l, r_{z,i}^l), \tag{3.63}$$

with the probability mass function (PMF)

$$P(\mathbf{x}_{z,i,j}^l = k|\mu_{z,i}^l, r_{z,i}^l) = \frac{\Gamma(k + r_{z,i}^l)}{\Gamma(r_{z,i}^l)\Gamma(k+1)} \left(\frac{\mu_{z,i}^l}{\mu_{z,i}^l + r_{z,i}^l}\right)^k \left(\frac{r_{z,i}^l}{\mu_{z,i}^l + r_{z,i}^l}\right)^{r_{z,i}^l}, \tag{3.64}$$

where $z \in \{s, t\}$ denotes the source, s, or target, t, domains; $\Gamma(\cdot)$ is the gamma function which is defined as $\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx$ and for the count value k is equal to $\Gamma(k) = (k-1)!$; $\mu_{z,i}^l$ and $r_{z,i}^l$ are respectively the mean and shape parameters of the gene i in domain z and class l. The shape parameter $r_{z,i}^l$ is the inverse of the dispersion parameter $\phi_{z,i}^l$ ($r_{z,i}^l = 1/\phi_{z,i}^l$) of the Negative Binomial model, which accounts for overdispersion in data, meaning that the variance can be much greater than the mean. The mean and variance of $\mathbf{x}_{z,i,j}^l$ are

$$\mathrm{E}(\mathbf{x}_{z,i,j}^l) = \mu_{z,i}^l, \quad \mathrm{Var}(\mathbf{x}_{z,i,j}^l) = \mu_{z,i}^l + \frac{\left(\mu_{z,i}^l\right)^2}{r_{z,i}^l}. \tag{3.65}$$

• Prior distributions:

In a Bayesian framework, we need to define prior distributions for the model parameters. For transfer learning, we should define the priors in such a way that the relevant model parameters in different domains relate to each other. We address this issue by defining joint prior distributions between $\mu_{s,i}^l$ and $\mu_{t,i}^l$ and between $r_{s,i}^l$ and $r_{t,i}^l$ for any $i \in \{1, \cdots, d\}$ and $l \in \{1, \cdots, L\}$. For computationally feasible inference, we assume that the priors for different genes and classes are independent in each domain. Let $\mu = \left\{\mu_{\{s,t\},\{1:d\}}^{\{1:L\}}\right\}$ and $r = \left\{r_{\{s,t\},\{1:d\}}^{\{1:L\}}\right\}$ denote respectively all the mean and shape parameters of the d genes in L classes and two domains s and t. The prior is factorized as

$$p(\mu, r) = \prod_{l=1}^{L}\prod_{i=1}^{d} p\left(\mu_{s,i}^l, \mu_{t,i}^l\right) p\left(r_{s,i}^l, r_{t,i}^l\right), \tag{3.66}$$

where it is assumed that the mean and shape parameters are independent; $p\left(\mu_{s,i}^l, \mu_{t,i}^l\right)$ is the joint probability distribution between the mean parameters of the source and target domains for the gene i and the class l; similarly, $p\left(r_{s,i}^l, r_{t,i}^l\right)$ is the joint probability distribution between the shape parameters of the source and target domains for gene i and class l.

We need to define joint priors for two sets of positive parameters: $\mu_{s,i}^l$ and $\mu_{t,i}^l$, $r_{s,i}^l$ and $r_{t,i}^l$, so that we can control the amount of correlation between them to simulate weaker to stronger dependency between the features of the two domains. One way of deriving such a joint prior is to integrate out the off-diagonal entry of a 2×2 Wishart matrix, which yields the joint distribution of its diagonal entries, which are both positive. Their correlation can be altered by defining an appropriate scale matrix.

Definition 3. A random $d \times d$ symmetric positive-definite matrix $\mathbf{\Lambda}$ has a nonsingular Wishart distribution with ν degrees of freedom, $W_d(\mathbf{M}, \nu)$, if $\nu \geq d$ and $\mathbf{M}$ is a $d \times d$ positive-definite matrix ($\mathbf{M} > 0$), and the density is

$$p(\mathbf{\Lambda}) = \left[2^{\frac{\nu d}{2}} \Gamma_d\left(\frac{\nu}{2}\right) |\mathbf{M}|^{\frac{\nu}{2}}\right]^{-1} |\mathbf{\Lambda}|^{\frac{\nu-d-1}{2}} \mathrm{etr}\left(-\frac{1}{2}\mathbf{M}^{-1}\mathbf{\Lambda}\right), \tag{3.67}$$

where $|\mathbf{A}|$ is the determinant of $\mathbf{A}$, $\mathrm{etr}(\mathbf{A}) = \exp\left(\mathrm{tr}(\mathbf{A})\right)$ and $\Gamma_d(\alpha)$ is the multivariate gamma function given by $\Gamma_d(\alpha) = \pi^{\frac{d(d-1)}{4}} \prod_{i=1}^d \Gamma\left(\alpha - \frac{i-1}{2}\right)$.

If $\mathrm{Vec}(\mathbf{\Lambda})$ denotes the $d^2 \times 1$ vectorized representation of $\mathbf{\Lambda}$ by stacking the columns of the matrix $\mathbf{\Lambda}$ on top of one another, then

$$\mathrm{Cov}[\mathrm{Vec}(\mathbf{\Lambda})] = \nu[\mathbf{M} \otimes \mathbf{M}][I_{d^2} + T], \tag{3.68}$$

where $\otimes$ denotes the Kronecker product and I_{d^2} is the identity matrix of size d^2. Here T is the transpose operator, i.e., $T\mathrm{Vec}(\mathbf{A}) = \mathrm{Vec}(\mathbf{A}')$, which can be written in terms of basis vectors e_i, for $i \in \{1, \cdots, d\}$ (a column vector whose ith entry is one and the remaining entries are zeros), as

$$T = \sum_{i=1}^{d}\sum_{j=1}^{d}[e_j \otimes e_i][e_i \otimes e_j]'. \tag{3.69}$$

Lemma 5. *[85]: If $\mathbf{\Lambda} \sim W_d(\mathbf{M}, \nu)$, and $\mathbf{A}$ is an $r \times d$ matrix of rank r, where $r \leq d$, then $\mathbf{A}\mathbf{\Lambda}\mathbf{A}' \sim W_r(\mathbf{A}\mathbf{M}\mathbf{A}', \nu)$.*

Corollary 2. *If $\mathbf{\Lambda} \sim W_2(\mathbf{M}, \nu)$, $\mathbf{\Lambda} = \left(\begin{smallmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{12} & \lambda_{22} \end{smallmatrix}\right)$, and $\mathbf{M} = \left(\begin{smallmatrix} m_{11} & m_{12} \\ m_{12} & m_{22} \end{smallmatrix}\right)$, then $\lambda_{ii} \sim m_{ii}\chi_\nu^2$ for $i = 1, 2$, where χ_ν^2 denotes the Chi-squared distribution with ν degrees of freedom. As a result, their mean and variance are $\mathrm{E}(\lambda_{ii}) = \nu m_{ii}$ and $\mathrm{Var}(\lambda_{ii}) = 2\nu m_{ii}^2$ for $i = 1, 2$, which can also be verified by (3.68). Furthermore, based on (3.68), the covariance and correlation between λ_{11} and λ_{22} are respectively*

$$\mathrm{Cov}(\lambda_{11}, \lambda_{22}) = 2\nu m_{12}^2, \quad \rho_\lambda = \frac{m_{12}^2}{m_{11}m_{22}}. \tag{3.70}$$

Theorem 7. *[86]: Let $\mathbf{\Lambda} = \left(\begin{smallmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{12} & \lambda_{22} \end{smallmatrix}\right)$ be a 2×2 Wishart random matrix with $\nu \geq 2$ degrees of freedom and positive-definite scale matrix $\mathbf{M} = \left(\begin{smallmatrix} m_{11} & m_{12} \\ m_{12} & m_{22} \end{smallmatrix}\right)$. The joint distribution of the two diagonal entries λ_{11} and λ_{22} have density function given by*

$$\begin{aligned} &p(\lambda_{11}, \lambda_{22}) = \\ &K \, \exp\left(-\frac{1}{2}\left(m_{11}^{-1} + c_2 f^2\right)\lambda_{11}\right) \exp\left(-\frac{1}{2}c_2^{-1}\lambda_{22}\right) \\ &\times (\lambda_{11})^{\frac{\nu}{2}-1} (\lambda_{22})^{\frac{\nu}{2}-1} \; {}_0F_1\left(\frac{\nu}{2}; \frac{1}{4}g\right), \end{aligned} \tag{3.71}$$

where $c_2 = m_{22} - m_{12}^2 m_{11}^{-1}$, $f = c_2^{-1} m_{12} m_{11}^{-1}$, $g = f^2 \lambda_{11}\lambda_{22}$, $K^{-1} = 2^\nu \Gamma^2\left(\frac{\nu}{2}\right)|\mathbf{M}|^{\frac{\nu}{2}}$, and ${}_0F_1$ is the generalized hypergeometric function [87].

Definition 4. [87]: The generalized hypergeometric function is defined by

$$\begin{aligned} &{}_pF_q(a_1, \cdots, a_p; b_1, \cdots, b_q; x) \\ &\qquad = \sum_{k=0}^{\infty} \frac{(a_1)_k \cdots (a_p)_k}{(b_1)_k \cdots (b_q)_k} \frac{x^k}{k!}, \end{aligned} \tag{3.72}$$

where a_i, $i = 1, \cdots, p$, and b_j, $j = 1, \cdots, q$, are arbitrary numbers and $(a)_k = a(a+1)\cdots(a+k-1)$, $k = 1, 2, \cdots$ is the Pochhammer symbol with $(a)_0 = 1$.

Some special cases of generalized hypergeometric functions (3.72) are

$$
\begin{aligned}
{}_0F_0(x) &= \sum_{k=0}^{\infty} \frac{x^k}{k!} = \exp(x), \\
{}_1F_0(a;x) &= \sum_{k=0}^{\infty} \frac{(a)_k x^k}{k!} = (1-x)^{-a}, \ \ |x|<1, \\
{}_0F_1(b;x) &= \sum_{k=0}^{\infty} \frac{x^k}{(b)_k k!},
\end{aligned}
\tag{3.73}
$$

where ${}_0F_1(b;x)$ is called the confluent hypergeometric limit function, which is closely related to the Bessel functions:

$$
J_\alpha(x) = \frac{(\frac{x}{2})^\alpha}{\Gamma(\alpha+1)}\, {}_0F_1\left(\alpha+1; -\frac{1}{4}x^2\right). \tag{3.74}
$$

Now, using Corollary 2 and Theorem 7, for every gene $i \in \{1, \cdots, d\}$ and class $l \in \{1, \cdots, L\}$, we can define the following joint prior for $\mu^l_{s,i}$ and $\mu^l_{t,i}$:

$$
\begin{aligned}
p(\mu^l_{s,i}, \mu^l_{t,i}) &= K^l_{\mu,i} \exp\left(-\frac{\mu^l_{s,i}}{2m^l_{s,i}(1-\rho^l_{\mu,i})}\right) \\
&\times \exp\left(-\frac{\mu^l_{t,i}}{2m^l_{t,i}(1-\rho^l_{\mu,i})}\right) \left(\mu^l_{s,i}\right)^{\frac{\nu_\mu}{2}-1} \left(\mu^l_{t,i}\right)^{\frac{\nu_\mu}{2}-1} \\
&\times \ {}_0F_1\left(\frac{\nu_\mu}{2}; \frac{\rho^l_{\mu,i}}{4m^l_{s,i}m^l_{t,i}\left(1-\rho^l_{\mu,i}\right)^2}\mu^l_{s,i}\mu^l_{t,i}\right),
\end{aligned}
\tag{3.75}
$$

where $\mathrm{E}(\mu^l_{z,i}) = \nu_\mu m^l_{z,i}$, $\mathrm{Var}(\mu^l_{z,i}) = 2\nu_\mu (m^l_{z,i})^2$ for $z \in \{s,t\}$, and $\mathrm{Corr}(\mu^l_{s,i}, \mu^l_{t,i}) = \rho^l_{\mu,i}$. Similarly, for every gene $i \in \{1, \cdots, d\}$ and class $l \in \{1, \cdots, L\}$, we can define the following joint prior for $r^l_{s,i}$ and $r^l_{t,i}$:

$$
\begin{aligned}
p(r^l_{s,i}, r^l_{t,i}) &= K^l_{r,i} \exp\left(-\frac{r^l_{s,i}}{2s^l_{s,i}(1-\rho^l_{r,i})}\right) \\
&\times \exp\left(-\frac{r^l_{t,i}}{2s^l_{t,i}(1-\rho^l_{r,i})}\right) \left(r^l_{s,i}\right)^{\frac{\nu_r}{2}-1} \left(r^l_{t,i}\right)^{\frac{\nu_r}{2}-1} \\
&\times \ {}_0F_1\left(\frac{\nu_r}{2}; \frac{\rho^l_{r,i}}{4s^l_{s,i}s^l_{t,i}\left(1-\rho^l_{r,i}\right)^2} r^l_{s,i} r^l_{t,i}\right),
\end{aligned}
\tag{3.76}
$$

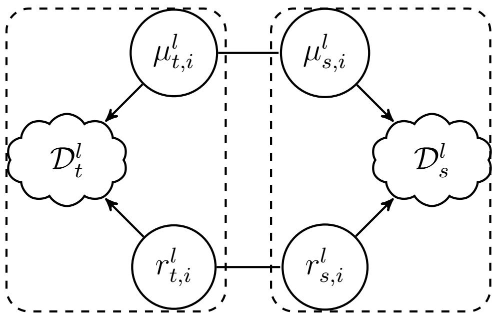


Figure 3.10: Dependency of the source and target domains through their mean and shape parameters for any gene $i \in \{1, \cdots, d\}$ and class $l \in \{1, \cdots, L\}$. Reprinted with permission from [6], ©2019 IEEE.

where $\mathrm{E}(r^l_{z,i}) = \nu_r s^l_{z,i}$, $\mathrm{Var}(r^l_{z,i}) = 2\nu_r (s^l_{z,i})^2$ for $z \in \{s,t\}$, and $\mathrm{Corr}(r^l_{s,i}, r^l_{t,i}) = \rho^l_{r,i}$.

Dependency of the source and target domains through their mean and shape parameters is shown in Fig. 3.10. Defining joint prior distributions for the mean and shape parameters connects the two domains through which the knowledge of rich source data can be transferred by improving the posterior distributions of the target parameters. The OBTL uses the improved posteriors of the target parameters to yield better prediction accuracy whenever the source domain is related to the target domain.

• Posteriors of Target Parameters:

Independent assumptions for all the genes $i \in \{1, \cdots, d\}$, classes $l \in \{1, \cdots, L\}$, domains $z \in \{s,t\}$, and samples $j \in \{1, \cdots, n^l_z\}$ is similarly adopted for the joint data likelihood function:

$$P(\mathcal{D}_s, \mathcal{D}_t | \mu, r) = \prod_{z \in \{s,t\}} \prod_{l=1}^{L} \prod_{i=1}^{d} \prod_{j=1}^{n^l_z} P(\mathbf{x}^l_{z,i,j} | \mu^l_{z,i}, r^l_{z,i}). \tag{3.77}$$

Therefore, the posteriors of different genes in different classes can be factorized as

$$\begin{aligned} &p\left(\mu^l_{s,i}, \mu^l_{t,i}, r^l_{s,i}, r^l_{t,i} \middle| \mathcal{D}^l_s, \mathcal{D}^l_t\right) \propto \\ &p\left(\mu^l_{s,i}, \mu^l_{t,i}\right) p\left(r^l_{s,i}, r^l_{t,i}\right) \prod_{z \in \{s,t\}} \prod_{j=1}^{n^l_z} P(\mathbf{x}^l_{z,i,j} | \mu^l_{z,i}, r^l_{z,i}), \end{aligned} \tag{3.78}$$

for all $i \in \{1, \cdots, d\}$ and $l \in \{1, \cdots, L\}$. Since we are interested in the posteriors of the target parameters for the sake of classification in the target domain, we can derive them by integrating out the source parameters in the joint posterior (3.78) as

$$p\left(\mu_{t,i}^l, r_{t,i}^l \middle| \mathcal{D}_s^l, \mathcal{D}_t^l\right) \propto \int_{\mu_{s,i}^l} \int_{r_{s,i}^l} p\left(\mu_{s,i}^l, \mu_{t,i}^l\right) p\left(r_{s,i}^l, r_{t,i}^l\right) \times \prod_{z \in \{s,t\}} \prod_{j=1}^{n_z^l} P(\mathbf{x}_{z,i,j}^l | \mu_{z,i}^l, r_{z,i}^l) d\mu_{s,i}^l dr_{s,i}^l. \tag{3.79}$$

Since the joint prior is not conjugate for the joint likelihood, the joint posterior, and consequently the target posteriors will not have closed forms. We utilize Markov Chain Monte Carlo (MCMC) methods to obtain posterior samples in the target domain. We use Hamiltonian Monte Carlo (HMC) [97], which has a superior performance to random walk MCMC samplers. To this end, we employ the STAN software [107], which has efficiently implemented the HMC.

3.3.3.2 Effective Class-Conditional Densities

For the optimal Bayesian classifier [57, 58], using the posterior predictive densities of the classes, called “effective class-conditional densities", leads to optimal classifiers to minimize the Bayesian expected errors of the classifiers. Similarly, we can derive the effective class-conditional densities for defining the OBTL classifier in the target domain, albeit, with the posterior of the target parameters derived from both the target and source samples.

Suppose that $\mathbf{x}$ denotes a $d \times 1$ new observed data point in the target domain, and we aim to optimally classify it to one of the classes $l \in \{1, \cdots, L\}$. In the context of the OBC, we need the effective class-conditional densities (or posterior predictive densities) of the L classes, defined as

$$p(\mathbf{x}|l) = \int_{\mu_t^l, r_t^l} p(\mathbf{x}|\mu_t^l, r_t^l) \pi^\star(\mu_t^l, r_t^l) d\mu_t^l dr_t^l, \tag{3.80}$$

for $l \in \{1, \cdots, L\}$, where $\pi^\star(\mu_t^l, r_t^l) = p(\mu_t^l, r_t^l | \mathcal{D}_t^l, \mathcal{D}_s^l)$ is the posterior of (μ_t^l, r_t^l) upon observation of $\mathcal{D}_t^l$ and $\mathcal{D}_s^l$. Since we do not have the closed form of the posterior $\pi^\star(\mu_t^l, r_t^l)$, we cannot do the integration of (3.80) in closed form. Instead, we use the posterior samples that were generated by

HMC sampling to approximate the integration of (3.80). Suppose we have N posterior samples from all of d genes in L classes. Then the approximation is given by:

$$p(\mathbf{x}|l) = \frac{1}{N}\sum_{j=1}^{N}\prod_{i=1}^{d} p(\mathbf{x}_i|\bar{\mu}^l_{t,i,j}, \bar{r}^l_{t,i,j}), \tag{3.81}$$

where $\bar{\mu}^l_{t,i,j}$ and $\bar{r}^l_{t,i,j}$ are the jth posterior sample of gene i in class l of the target domain for the mean and shape parameters, respectively. We denote the effective class-conditional density of the OBTL by $O_{\text{OBTL}}(\mathbf{x}|l) = p(\mathbf{x}|l)$.

3.3.3.3 Optimal Bayesian Transfer Learning Classifier

Let c^l_t be the prior probability that the target sample $\mathbf{x}$ belongs to the class $l \in \{1, \cdots, L\}$. Since $0 < c^l_t < 1$ and $\sum_{l=1}^{L} c^l_t = 1$, a Dirichlet prior is assumed:

$$(c^1_t, \cdots, c^L_t) \sim \text{Dir}(L, \xi_t), \tag{3.82}$$

where $\xi_t = (\xi^1_t, \cdots, \xi^L_t)$ are the concentration parameters, and $\xi^l_t > 0$ for $l \in \{1, \cdots, L\}$. As the Dirichlet distribution is a conjugate prior for the categorical distribution, upon observing $\mathbf{n} = (n^1_t, \cdots, n^L_t)$ data for class l in the target domain, the posterior has a Dirichlet distribution:

$$\begin{aligned}\pi^\star = (c^1_t, \cdots, c^L_t|\mathbf{n}) &\sim \text{Dir}(L, \xi_t + \mathbf{n}) \\ &= \text{Dir}(L, \xi^1_t + n^1_t, \cdots, \xi^L_t + n^L_t),\end{aligned} \tag{3.83}$$

with the posterior mean of c^l_t as

$$\text{E}_{\pi^\star}(c^l_t) = \frac{\xi^l_t + n^l_t}{N_t + \xi^0_t}, \tag{3.84}$$

where $N_t = \sum_{l=1}^{L} n^l_t$ and $\xi^0_t = \sum_{l=1}^{L} \xi^l_t$. As such, the optimal Bayesian transfer learning (OBTL) classifier for any new unlabeled sample $\mathbf{x}$ in the target domain is defined as

$$\Psi_{\text{OBTL}}(\mathbf{x}) = \arg\max_{l\in\{1,\cdots,L\}} \text{E}_{\pi^\star}(c^l_t) O_{\text{OBTL}}(\mathbf{x}|l), \tag{3.85}$$

which minimizes the expected error of the classifier in the target domain, that is, $\mathrm{E}_{\pi^\star}[\varepsilon(\Theta_t, \Psi_{\mathrm{OBTL}})] \leq \mathrm{E}_{\pi^\star}[\varepsilon(\Theta_t, \Psi)]$, where $\varepsilon(\Theta_t, \Psi)$ is the error of any arbitrary classifier Ψ assuming the parameters $\Theta_t = \left\{c_t^{\{1:L\}}, \mu_{t,\{1:d\}}^{\{1:L\}}, r_{t,\{1:d\}}^{\{1:L\}}\right\}$ of the feature-label distribution in the target domain, and the expectation is over the posterior $\pi^\star$ of Θ_t.

If we do not have any prior knowledge for the selection of classes, we use the same concentration parameter for all the classes: $\xi_t = (\xi, \cdots, \xi)$. Hence, if the number of samples in each class is the same, $n_t^1 = \cdots = n_t^L$, the first term $\mathrm{E}_{\pi^\star}$ is the same for all the classes and (3.85) is reduced to:

$$\Psi_{\mathrm{OBTL}}(\mathbf{x}) = \arg\max_{l \in \{1, \cdots, L\}} O_{\mathrm{OBTL}}(\mathbf{x}|l). \tag{3.86}$$

3.3.3.4 OBC in Target Domain

To see how the source data can help improve the performance, we compare the OBTL classifier with the OBC when there is only the target domain. Using Corollary 1 and exactly the same modeling and parameters as the OBTL, the priors for $\mu_{t,i}^l$ and $r_{t,i}^l$ are scaled Chi-squared distributions:

$$\begin{aligned} \mu_{t,i}^l &\sim m_{t,i}^l \mathcal{X}_{\nu_\mu}^2, \\ r_{t,i}^l &\sim s_{t,i}^l \mathcal{X}_{\nu_r}^2, \end{aligned} \tag{3.87}$$

with the probability density functions

$$\begin{aligned} p(\mu_{t,i}^l) &= \left[(2m_{t,i}^l)^{\frac{\nu_\mu}{2}} \Gamma\left(\frac{\nu_\mu}{2}\right)\right]^{-1} {\mu_{t,i}^l}^{\frac{\nu_\mu}{2}-1} \exp\left(\frac{-\mu_{t,i}^l}{2m_{t,i}^l}\right), \\ p(r_{t,i}^l) &= \left[(2s_{t,i}^l)^{\frac{\nu_r}{2}} \Gamma\left(\frac{\nu_r}{2}\right)\right]^{-1} {r_{t,i}^l}^{\frac{\nu_r}{2}-1} \exp\left(\frac{-r_{t,i}^l}{2s_{t,i}^l}\right). \end{aligned} \tag{3.88}$$

Let $\mu = \left\{\mu_{t,\{1:d\}}^{\{1:L\}}\right\}$ and $r = \left\{r_{t,\{1:d\}}^{\{1:L\}}\right\}$ denote respectively all the mean and shape parameters of the d genes in L classes in the target domain. The prior is factorized as

$$p(\mu, r) = \prod_{l=1}^{L} \prod_{i=1}^{d} p\left(\mu_{t,i}^l\right) p\left(r_{t,i}^l\right). \tag{3.89}$$

The likelihood of the target data is

$$P(\mathcal{D}_t|\mu, r) = \prod_{l=1}^{L} \prod_{i=1}^{d} \prod_{j=1}^{n_t^l} P(\mathbf{x}_{t,i,j}^l | \mu_{t,i}^l, r_{t,i}^l). \tag{3.90}$$

Therefore, the posteriors of different genes in different classes can be factorized as

$$\begin{aligned} &p\left(\mu_{t,i}^l, r_{t,i}^l \middle| \mathcal{D}_t^l\right) \propto \\ &p\left(\mu_{t,i}^l\right) p\left(r_{t,i}^l\right) \prod_{j=1}^{n_t^l} P(\mathbf{x}_{t,i,j}^l | \mu_{t,i}^l, r_{t,i}^l), \end{aligned} \tag{3.91}$$

for all $i \in \{1, \cdots, d\}$ and $l \in \{1, \cdots, L\}$. Similar to the OBTL, here also we cannot derive the posteriors in closed forms and use HMC to get samples from the posteriors of the parameters $\mu_{t,i}^l$ and $r_{t,i}^l$, for all $i \in \{1, \cdots, d\}$ and $l \in \{1, \cdots, L\}$. Then using (3.81) yields us the effective class-conditional density of the OBC, which we denote by $O_{\text{OBC}}(\mathbf{x}|l)$.

The multi-class OBC [90], under a zero-one loss function, is defined as

$$\Psi_{\text{OBC}}(\mathbf{x}) = \arg \max_{l \in \{1, \cdots, L\}} \mathrm{E}_{\pi^*}(c_t^l) O_{\text{OBC}}(\mathbf{x}|l). \tag{3.92}$$

Similar to the OBTL, in the case of equal prior probabilities for the classes,

$$\Psi_{\text{OBC}}(\mathbf{x}) = \arg \max_{l \in \{1, \cdots, L\}} O_{\text{OBC}}(\mathbf{x}|l). \tag{3.93}$$

For binary classification, the definition of the OBC in (3.92) is equivalent to the definition in [57], where it is defined to be the binary classifier possessing the minimum Bayesian mean square error estimate [64] relative to the posterior distribution.

If the correlations between the mean and shape parameters of the target and source domains are zero for all the genes in all the classes, that is, $\rho_{\mu,i}^l = 0$ and $\rho_{r,i}^l = 0$, for $i \in \{1, \cdots, d\}$ and $l \in \{1, \cdots, L\}$, then the joint priors for the OBTL in (3.75) and (3.76) become $p(\mu_{s,i}^l, \mu_{t,i}^l) = p(\mu_{s,i}^l)p(\mu_{t,i}^l)$ and $p(r_{s,i}^l, r_{t,i}^l) = p(r_{s,i}^l)p(r_{t,i}^l)$, indicating that all the mean and shape parameters are independent between the two domains. Having independent priors and likelihoods leads to independent posteriors for the two domains. In other words, source data cannot help improve the target posteriors and the target posterior is identical to the OBC in the target domain. Therefore, the OBTL becomes identical to the OBC. On the contrary, if we have higher correlations between the parameters of the two domains, that is, $\rho^l_{\mu,i}$ and $\rho^l_{r,i}$ are closer to 1, then more knowledge from the source domain is transferred to the target and the OBTL yields much better performance than the OBC.

3.3.4 Experiments and Discussion

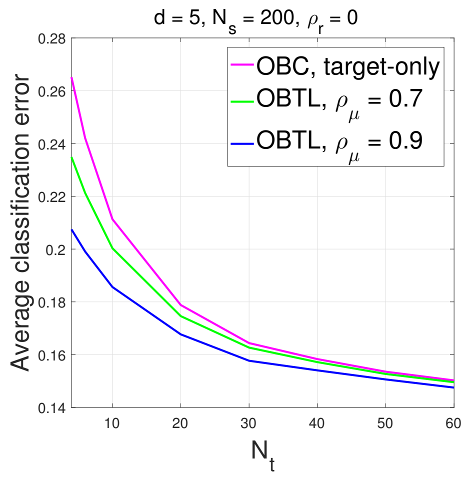
3.3.4.1 Synthetic datasets

We consider a simulation setup and evaluate the OBTL classifiers by average classification error with different joint prior densities modeling the relatedness of the source and target domains. Unless mentioned, the feature dimension is $d = 5$, the number of classes in each domain is $L = 2$, the number of source training data per class is $n_s = n^l_s = 100$, the number of target training data per class is $n_t = n^l_t = 5$, for $l \in \{1, 2\}$. We set $\nu_\mu = \nu_r = 4$, $m^1_{t,i} = 1000/\nu_\mu$, $m^2_{t,i} = 1500/\nu_\mu$, $m^1_{s,i} = 5000/\nu_\mu$, $m^2_{s,i} = 6000/\nu_\mu$, $s^1_{t,i} = s^2_{t,i} = 1/\nu_r$, and $s^1_{t,i} = s^2_{t,i} = 0.5/\nu_r$ for all the genes $i \in \{1, \cdots, d\}$. As such, the expected values of the parameters are $\mathrm{E}(\mu^1_{t,i}) = 1000$, $\mathrm{E}(\mu^2_{t,i}) = 1500$, $\mathrm{E}(\mu^1_{s,i}) = 5000$, $\mathrm{E}(\mu^2_{s,i}) = 6000$, $\mathrm{E}(r^1_{t,i}) = \mathrm{E}(r^2_{t,i}) = 1$, and $\mathrm{E}(r^1_{s,i}) = \mathrm{E}(r^2_{s,i}) = 0.5$ for all $i \in \{1, \cdots, d\}$. The variances of the parameters are $\mathrm{Var}(\mu^l_{z,i}) = 2[\mathrm{E}(\mu^l_{z,i})]^2/\nu_\mu$ and $\mathrm{Var}(r^l_{z,i}) = 2[\mathrm{E}(r^l_{z,i})]^2/\nu_r$ for $z \in \{s, t\}$, $l \in \{1, 2\}$, and $i \in \{1, \cdots, d\}$. All simulations assume equal prior probabilities for the classes, so we use (3.86) and (3.93) for the OBTL classifier and OBC, respectively.

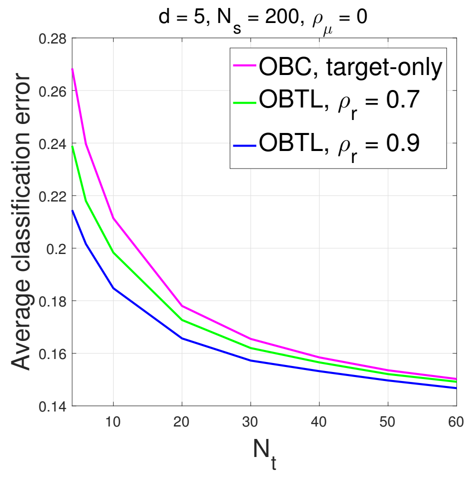
We evaluate the prediction performance according to the common evaluation procedure of Bayesian learning by average classification error. To sample from the prior (3.75) we first sample from a Wishart distribution $W_2(\mathbf{M}^l_i, \nu^l)$ with the scale matrix of $\mathbf{M}^l_i = \begin{pmatrix} m^l_{t,i} & m^l_{ts,i} \\ m^l_{ts,i} & m^l_{s,i} \end{pmatrix}$, where $m^l_{ts,i} = \sqrt{\rho^l_{\mu,i} m^l_{t,i} m^l_{s,i}}$, to get a sample for $\mathbf{\Lambda}^l_i = \begin{pmatrix} \mu^l_{t,i} & \mu^l_{ts,i} \\ \mu^l_{ts,i} & \mu^l_{s,i} \end{pmatrix}$, for each class $l = 1, 2$ and each gene $i \in \{1, \cdots, d\}$. We then pick $(\mu^l_{t,i}, \mu^l_{s,i})$, which is a joint sample from $p(\mu^l_{t,i}, \mu^l_{s,i})$ in (3.75). Similarly, to sample from the prior (3.76), we first sample from a Wishart distribution $W_2(\mathbf{S}_i^l, \nu^l)$ with the scale matrix of $\mathbf{S}_i^l = \begin{pmatrix} s_{t,i}^l & s_{ts,i}^l \\ s_{ts,i}^l & s_{s,i}^l \end{pmatrix}$, where $s_{ts,i}^l = \sqrt{\rho_{r,i}^l s_{t,i}^l s_{s,i}^l}$, to get a sample for $\mathbf{R}_i^l = \begin{pmatrix} r_{t,i}^l & r_{ts,i}^l \\ r_{ts,i}^l & r_{s,i}^l \end{pmatrix}$, for each class $l = 1, 2$ and each gene $i \in \{1, \cdots, d\}$. We then pick $(r_{t,i}^l, r_{s,i}^l)$, which is a joint sample from $p(r_{t,i}^l, r_{s,i}^l)$ in (3.76). Once we have $\mu_{t,i}^l$, $\mu_{s,i}^l$, $r_{t,i}^l$, and $r_{s,i}^l$, for all $i \in \{1, \cdots, d\}$, and $l \in \{1, \cdots, L\}$, we generate a training and test set from (3.63). The training set contains samples from both the target and source domains, but the test set contains only samples from the target domain. As the numbers of source and target training data per class is n_s and n_t, there are $N_t = Ln_s$ and $N_s = Ln_t$ source and target training data in total, respectively. We assume the size of the test set per class is 500 in the simulations, so 1000 in total. Given the training and test set, we use the OBTL and its target-only version, OBC, and calculate the error. We further repeat this whole process 1000 times for different realizations of $\mu_{t,i}^l$, $\mu_{s,i}^l$, $r_{t,i}^l$, and $r_{s,i}^l$, for all $i \in \{1, \cdots, d\}$, and $l \in \{1, \cdots, L\}$, and finally average all the errors and return the average classification error.

To examine how the source data improves the classifier in the target domain, we compare the performance of the OBTL classifier with the OBC designed in the target domain alone. The average classification error versus N_t is depicted in Figs. 3.11(a), 3.11(b), and 3.11(c) for the OBC and OBTL with different values of ρ_μ and ρ_r (we use the same value for all the classes and genes, $\rho_\mu = \rho_{\mu,i}^l$ and $\rho_r = \rho_{r,i}^l$ for $l \in \{1, 2\}$ and $i \in \{1, \cdots, d\}$). When ρ_μ and ρ_r are close to one, the performance of the OBTL classifier is much better than that of the OBC due to the greater relatedness between the two domains and appropriate use of the source data. This performance improvement is especially noticeable when N_t is small, which reflects the real-world scenario. In Fig. 3.11, we also observe that the errors of the OBTL classifier and OBC are converging to a similar value when N_t gets very large, meaning that the source data are redundant when there is a large number of target data. When ρ_μ and ρ_r are larger, the error curves converge faster to the optimal error, which is the average Bayes error of the target classifier. Recall that when $\rho_\mu = 0$ and $\rho_r = 0$, the OBTL classifier reduces to the OBC.

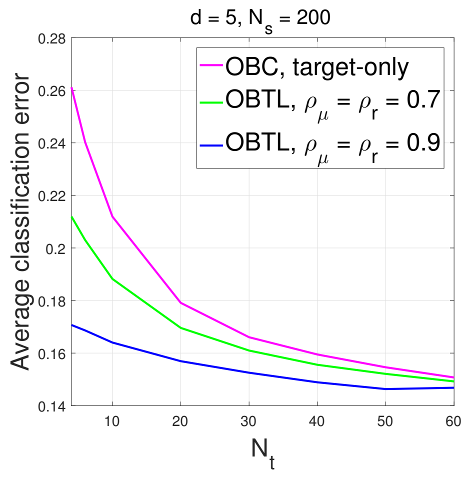
Figures 3.11(d), 3.11(e), and 3.11(f) depict average classification error versus N_s for the OBC and OBTL with different values of ρ_μ and ρ_r. The OBC error is constant for all N_s as it does not



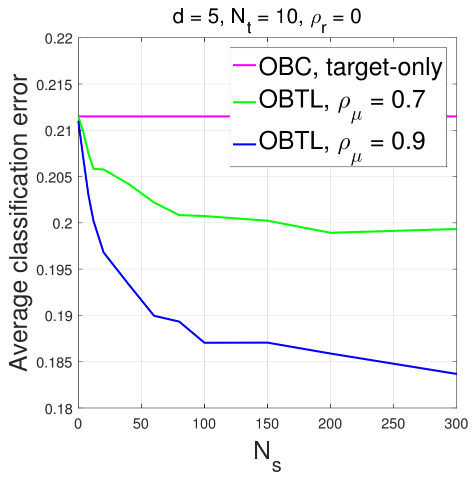
(a)



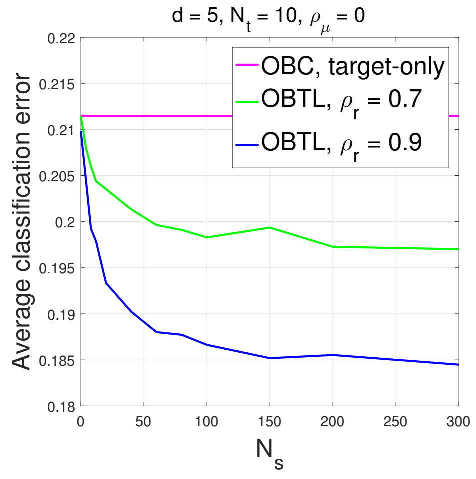
(b)



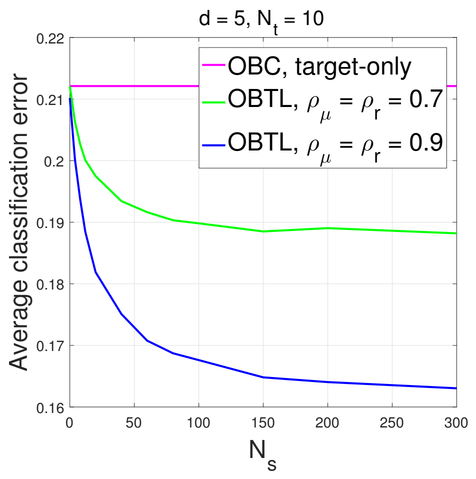
(c)



(d)



(e)



(f)

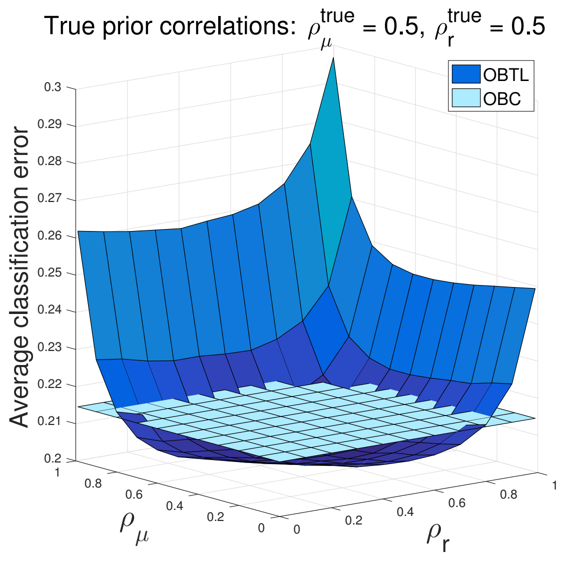
Figure 3.11: Average classification error versus the number of target training data N_t and the number of source training data N_s. Reprinted with permission from [6], ©2019 IEEE.

employ the source data. The error of the OBTL classifier equals that of the OBC when $N_s = 0$ and starts to decrease as N_s increases. In Figs. 3.11(d), 3.11(e), and 3.11(f), when ρ_μ and ρ_r are larger, the improvement is greater since the two domains are more related.

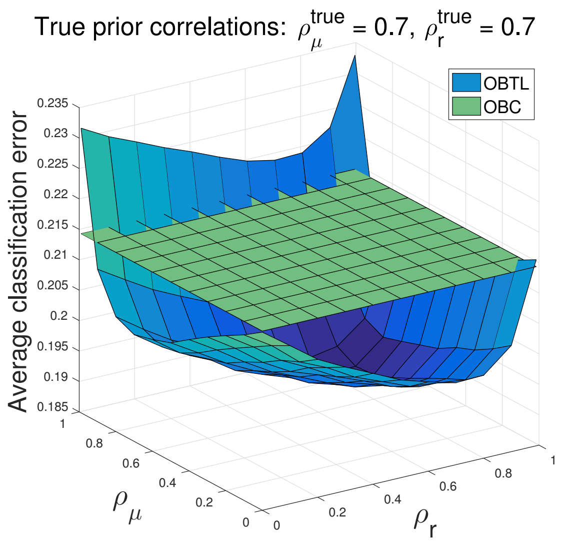
In the OBTL's framework, ρ_r and ρ_μ have a bridging task between the two domains and transfer the knowledge from the source to the target domain. We analyze in Fig. 3.12 the effects of not choosing the true prior correlations ρ_r and ρ_μ in the classifier. Figures 3.12(a), 3.12(b), and 3.12(c) demonstrate the average classification error versus ρ_r and ρ_μ ($0 \leq \rho_r, \rho_\mu < 1$), used in the OBTL classifier, when the true correlation priors from which data have been generated are $\rho_r = \rho_\mu = 0.5, 0.7, 0.9$, respectively. We also include the constant error plane of the OBC for the sake of comparison, which hits the OBTL's surface in $\rho_r = \rho_\mu = 0$, as discussed previously. The minimum error occurs in the true values of the prior correlations, and the distance between the OBTL's minimum error and the OBC's error is greater when the true correlations are larger. The most important observation in Fig. 3.12 is that the OBTL outperforms the OBC for any chosen values of ρ_r and ρ_μ when the two domains are highly related (Fig. 3.12c), and even in the case of less relatedness between the domains (Fig. 3.12a and Fig. 3.12b), the OBTL works better than the OBC for a wide range of chosen values for ρ_r and ρ_μ around the true values. This assures us that if we fail to use the true prior correlations, we will still get improvements using the OBTL.

3.3.4.2 Real TCGA datasets

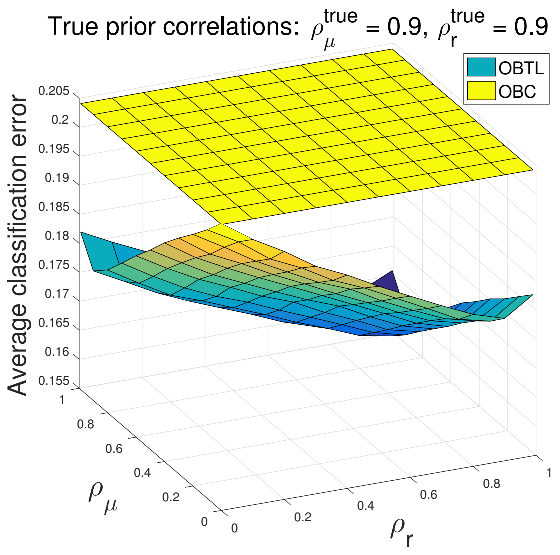
• Lung Cancer: We consider the classification of two subtypes of non-small cell lung cancer (NSCLC), lung adenocarcinoma (LUAD) versus lung squamous cell carcinoma (LUSC). According to the American Cancer Society, NSCLC accounts for 80% to 85% of lung cancers. Moreover, 40% of all lung cancers are LUAD and 25% to 30% of them are LUSC. We have used The Cancer Genome Atlas (TCGA) datasets for both LUAD and LUCS. We have downloaded those datasets using the R package "TCGA2STAT" [108] and extracted the RNA-Seq count data for the both classes: LUAD and LUSC. There are two types of RNA-Seq measurements for cancer types in TCGA: RNA-Seq and RNA-Seq-v2. Since the number of samples in RNA-Seq-v2 is larger than that in RNA-Seq, we consider the first one as the source domain and the second one as the tar-



(a)

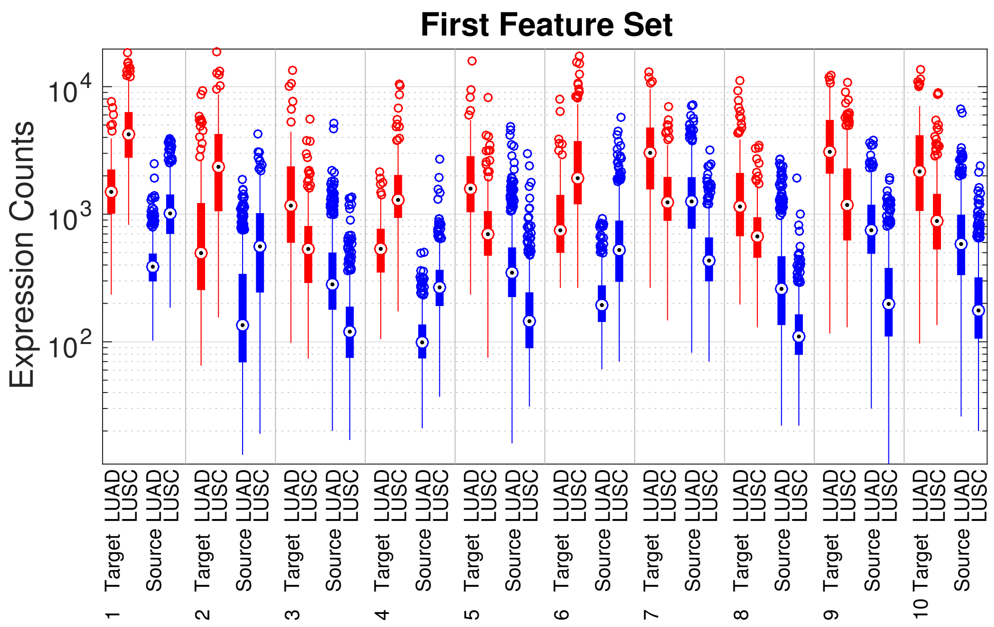


(b)

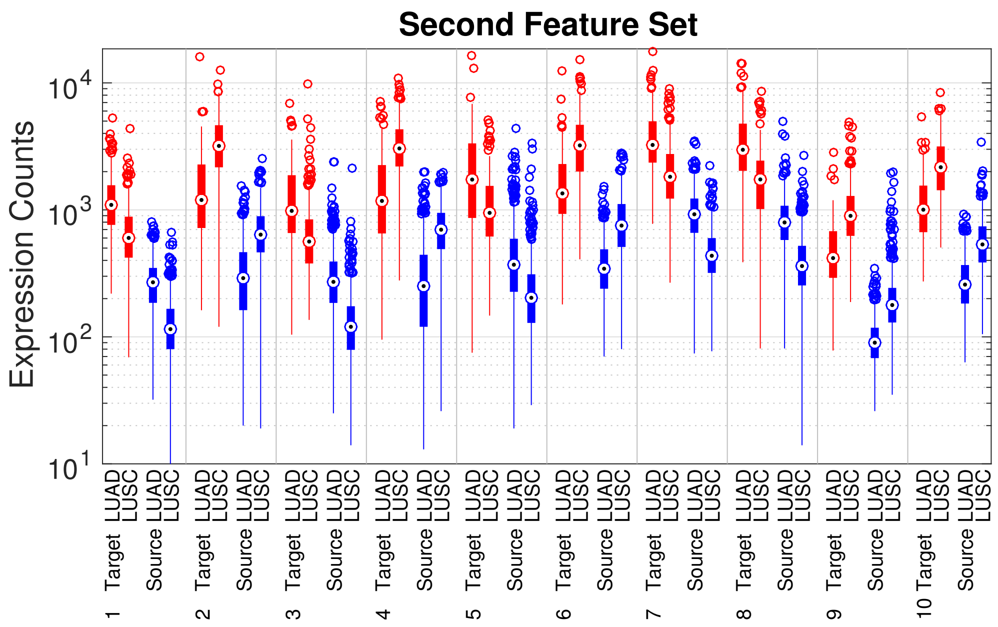


(c)

Figure 3.12: Average classification error versus ρ_r and ρ_μ used in the OBTL classifier. The true data generating values are written on top of each figure. Reprinted with permission from [6], ©2019 IEEE.

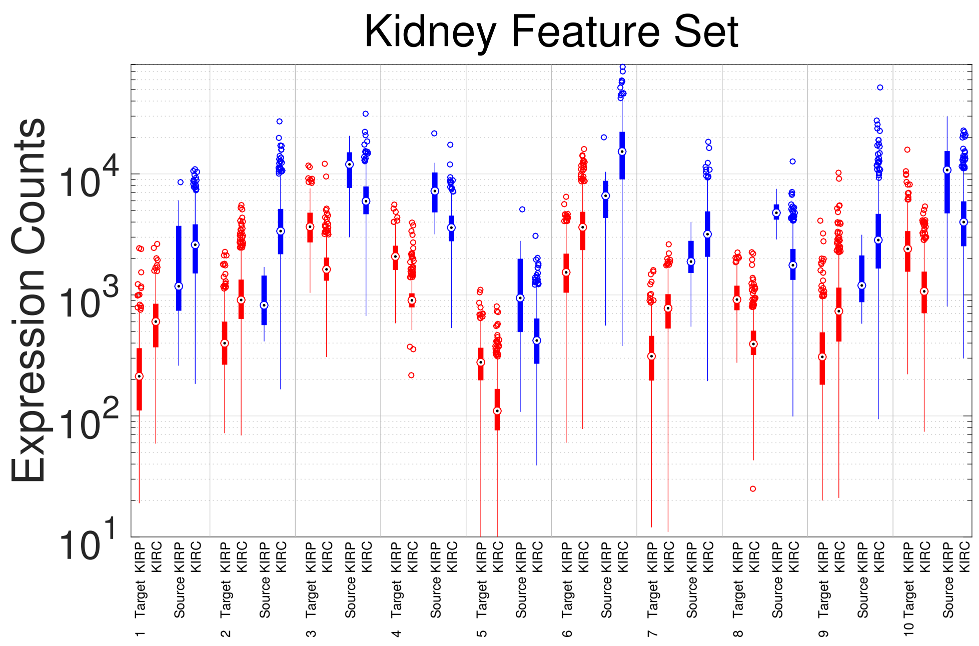


(a)



(b)

Figure 3.13: RNA-Seq counts of ten genes in two domains and for two classes LUAD and LUSC from TCGA. Red denotes the target domain, which is RNA-Seq data from TCGA. Blue denotes the source domain, which is RNA-Seq-v2 from TCGA. We see that the source domain has lower values than the target domain. (a) First feature set with the following ten genes (ordered from 1 to 10 successively): USP31, FGF11, CLCF1, C15orf41, KLF2, TMEM79, CD302, SDHAP3, TSPAN12, CABLES1, (b) Second feature set with the following ordered genes (ordered from 1 to 10 successively): ACBD4, DTL, DISP1, BUB1B, MTMR11, CHAF1A, C9orf7, SIGIRR, C1orf74, GEN1. Reprinted with permission from [6], ©2019 IEEE.



(a)

Figure 3.14: RNA-Seq counts of ten genes in two domains and for two classes KIRP and KIRC from TCGA. Red denotes the target domain, which is RNA-Seq-v2 data from TCGA. Blue denotes the source domain, which is RNA-Seq from TCGA. We see that the source domain has larger values than the target domain. (a) Kidney feature set with the following ten genes (ordered from 1 to 10 successively): MCC, PTP4A3, ABHD14B, VPS25, C9orf116, LEPREL1, NOSTRIN, GTF2IRD1, GEM, MMP24. Reprinted with permission from [6], ©2019 IEEE.

get domain. Tumor sample sizes in the target domain (RNA-Seq) are 125 and 223 for LUAD and LUSC, respectively, and in the source domain (RNA-Seq-v2) are 515 and 501 for LUAD and LUSC, respectively.

We evaluate the performance of the OBTL classifier using two feature sets of size $d = 10$. Our goal in this section is not feature selection. We want to show that for any feature set transfer learning can help improve the target classifier, and the amount of improvement depends on how much those features are related between the two domains. We choose two feature sets as follows. We drop all genes having at least one sample of less than 10 or more than 20,000 reads in the two domains. Then we use the edgeR package and sort all the genes by log-fold change values between the two classes LUAD and LUSC in the target domain. We choose two feature sets of size $d = 10$ from different regions of the ordered list to have different ranges of classification errors. Those

Table 3.4: The average error of the OBTL classifier using the first feature set in the TCGA data for the classification of LUAD and LUAC assuming different values of ρ_μ and ρ_r. The corresponding average error for the OBC is *0.1312*. The minimum error is written in bold. Reprinted with permission from [6], ©2019 IEEE.

OBTL	$\rho_r = 0$	$\rho_r = 0.3$	$\rho_r = 0.5$	$\rho_r = 0.7$	$\rho_r = 0.9$	$\rho_r = 0.99$
$\rho_\mu = 0$	0.1315	0.1278	0.1286	0.1262	0.1225	0.1201
$\rho_\mu = 0.3$	0.1290	0.1293	0.1255	0.1252	0.1220	0.1209
$\rho_\mu = 0.5$	0.1272	0.1279	0.1255	0.1241	0.1207	0.1189
$\rho_\mu = 0.7$	0.1267	0.1245	0.1228	0.1211	0.1189	0.1173
$\rho_\mu = 0.9$	0.1176	0.1181	0.1162	0.1151	0.1121	0.1124
$\rho_\mu = 0.99$	0.1024	0.1025	0.1023	0.1026	**0.1009**	0.1041

feature sets have corresponding RNA-Seq-v2 samples, which make the source dataset. The first feature set consists of the ordered genes USP31, FGF11, CLCF1, C15orf41, KLF2, TMEM79, CD302, SDHAP3, TSPAN12, CABLES1. The second feature set consists of the ordered genes ACBD4, DTL, DISP1, BUB1B, MTMR11, CHAF1A, C9orf7, SIGIRR, C1orf74, GEN1. Figure 3.13 shows the distributions of the genes in the first and second feature sets in both target and source domains and for each class LUAD and LUSC. We see in Fig. 3.13 that the source data have lower values than the target data in each class. We create 50 random training and test splits from both domains and compute the average classification error over those 50 splits in Tables (3.4) and (3.5). In each split, we randomly choose $n_t^l = 5$ training target data per class and $n_s^l = 100$ training source data per class. As a result, the total numbers of training target and source data are respectively $N_t = 10$ and $N_s = 200$. For each split, we choose 100 random test data per class (200 in total) from the target domain. Note that we use only the target data for the test, since the goal is to design a classifier in the target domain.

Regarding the hyperparameters, we set $m_{s,i}^l = 500/\nu_\mu$ and $m_{t,i}^l = 2000/\nu_\mu$ for $l \in \{1, 2\}$, and $i \in \{1, \cdots, d\}$. This makes the expected values of the mean parameters equal to $\mathrm{E}(\mu_{s,i}^l) = 500$ and $\mathrm{E}(\mu_{t,i}^l) = 2000$, and the reason for choosing them in this way is that the target data have larger values than the source data for any gene. We choose $s_{z,i}^l = 4/\nu_r$ for $z \in \{t, s\}$, $l \in \{1, 2\}$, and $i \in \{1, \cdots, d\}$. This makes the expected values of the shape parameters equal to $\mathrm{E}(r_{z,i}^l) = 4$ for $z \in \{t, s\}$, $l \in \{1, 2\}$, and $i \in \{1, \cdots, d\}$. We also choose $\nu_\mu = \nu_r = 2$ to have the least

Table 3.5: The average error of the OBTL classifier using the second feature set in the TCGA data for the classification of LUAD and LUAC assuming different values of ρ_μ and ρ_r. The corresponding average error for the OBC is *0.1776*. The minimum error is written in bold. Reprinted with permission from [6], ©2019 IEEE.

OBTL	$\rho_r = 0$	$\rho_r = 0.3$	$\rho_r = 0.5$	$\rho_r = 0.7$	$\rho_r = 0.9$	$\rho_r = 0.99$
$\rho_\mu = 0$	0.1787	0.1771	0.1764	0.1723	0.1689	0.1673
$\rho_\mu = 0.3$	0.1775	0.1752	0.1735	0.1701	0.1650	0.1668
$\rho_\mu = 0.5$	0.1751	0.1727	0.1714	0.1686	0.1652	0.1670
$\rho_\mu = 0.7$	0.1697	0.1700	0.1680	0.1651	0.1601	0.1612
$\rho_\mu = 0.9$	0.1614	0.1597	0.1581	0.1545	0.1498	0.1520
$\rho_\mu = 0.99$	0.1398	0.1386	0.1373	0.1340	0.1295	**0.1274**

informative priors. Tables (3.4) and (3.5) represent the average error of the OBTL classifier for different values of ρ_μ and ρ_r. For the first feature set in Table (3.4), the error of the OBC is 0.1312, while the best error of the OBTL is 0.1009 (for $\rho_\mu = 0.99$ and $\rho_r = 0.9$). For the second feature set in Table (3.5), the error of the OBC is 0.1776, while the best error of the OBTL is 0.1274 (for $\rho_\mu = 0.99$ and $\rho_r = 0.99$). These results show that the features in each feature set are highly related between the two domains, and therefore, the source data in the OBTL classifier make the error rate considerably less than that of the target-only OBC. We assume equal prior probabilities $(1/2)$ for both classes LUAD and LUSC, so we use (3.86) and (3.93) for the OBTL classifier and OBC, respectively.

• Kidney Cancer: Knowing which type of cell makes up a kidney tumor helps doctors plan treatment. According to the American Cancer Society, the two most common types of kidney cancers are kidney renal clear cell carcinoma (KIRC) and kidney renal papillary cell carcinoma (KIRP). We downloaded the datasets for two kinds of kidney cancers and obtained RNA-Seq and RNA-Seq-v2 data for both cancer types via “TCGA2STAT". Unlike the lung cancer case, in kidney cancer subtyping, we consider RNA-Seq-v2 as the target domain and RNA-Seq as the source domain, and we show again that transfer learning can help improve the accuracy of the target classifier and the OBTL works better than the target-only case. Tumor sample sizes in the target domain (RNA-Seq-v2) are 290 and 533 for KIRP and KIRC, respectively, and in the source domain (RNA-Seq) are 14 and 469 for KIRP and KIRC, respectively. Similar to the lung cancers, we pick

Table 3.6: The average error of the OBTL classifier using the kidney feature set in the TCGA data for the classification of KIRP and KIRC assuming different values of ρ_μ and ρ_r. The corresponding average error for the OBC is *0.0866*. The minimum error is written in bold. Reprinted with permission from [6], ©2019 IEEE.

OBTL	$\rho_r = 0$	$\rho_r = 0.3$	$\rho_r = 0.5$	$\rho_r = 0.7$	$\rho_r = 0.9$	$\rho_r = 0.99$
$\rho_\mu = 0$	0.0870	0.0837	0.0836	0.0831	0.0817	0.0843
$\rho_\mu = 0.3$	0.0862	0.0842	0.0833	0.0822	0.0822	0.0823
$\rho_\mu = 0.5$	0.0858	0.0854	0.0833	0.0805	0.0810	0.0823
$\rho_\mu = 0.7$	0.0856	0.0842	0.0821	0.0803	0.0797	0.0815
$\rho_\mu = 0.9$	0.0847	0.0834	0.0805	0.0797	0.0778	0.0789
$\rho_\mu = 0.99$	0.0798	0.0776	0.0775	0.0758	**0.0731**	0.0741

$N_t = 10$ target data, $N_s = 200$ source data, and 200 test data for any of 50 random training and test data splits.

Similar to the process we did for lung cancer, we choose $d = 10$ genes as our feature set: MCC, PTP4A3, ABHD14B, VPS25, C9orf116, LEPREL1, NOSTRIN, GTF2IRD1, GEM, MMP24. The expression values of these ten genes in both domains have been plotted in Fig. 3.14. Regarding the hyperparameters, we set $m^l_{s,i} = 5000/\nu_\mu$ and $m^l_{t,i} = 1000/\nu_\mu$ for $l \in \{1, 2\}$ and $i \in \{1, \cdots, d\}$. This makes the expected values of the mean parameters equal to $\mathrm{E}(\mu^l_{s,i}) = 5000$ and $\mathrm{E}(\mu^l_{t,i}) = 1000$. We choose $s^l_{z,i} = 4/\nu_r$ for $z \in \{t, s\}$, $l \in \{1, 2\}$, and $i \in \{1, \cdots, d\}$. This makes the expected values of the shape parameters equal to $\mathrm{E}(r^l_{z,i}) = 4$ for $z \in \{t, s\}$, $l \in \{1, 2\}$, and $i \in \{1, \cdots, d\}$. We also choose $\nu_\mu = \nu_r = 2$ to have the least informative priors. Table (3.6) shows the average error of the OBTL classifier for different values of ρ_μ and ρ_r. The error of the OBC is 0.0866, while the best error of the OBTL is 0.0731 (for $\rho_\mu = 0.99$ and $\rho_r = 0.9$). These results show that the features are highly related between the two domains, and therefore the OBTL outperforms the target-only OBC. Similar to the lung cancer case, we assume equal prior probabilities (1/2) for both classes KIRP and KIRC, so we use (24) and (31) for the OBTL classifier and OBC, respectively.

3.3.5 Conclusion

We constructed a Bayesian transfer learning framework to address supervised transfer learning for NGS count data. The Optimal Bayesian Transfer Learning (OBTL) classifier compensates for the lack of labeled data in the target domain by transferring the relevant knowledge from the source domain with available labeled data and is optimal in this novel Bayesian framework since it minimizes the expected classification error. We defined a joint prior model for the parameters of the target and source domains and learned the parameters by HMC for joint posterior sampling. Assuming the two domains are related, either through their mean or shape parameters, source data helped obtain more accurate target posterior samples, from which we derived the effective class-conditional densities as well as the OBTL classifier. We compared the performance of the OBTL with its target-only version, OBC, to see how transferring from source to target domain can help. We tested the OBTL classifier using RNA-Seq data from TCGA and demonstrated its performance improvement.

4. SUMMARY

In chapter 2, we studied classification of single-cell gene expression trajectories coming from two classes, healthy and mutated (cancerous) using Boolean networks with perturbation (BNps) to model the dynamics of each class at the state level, meaning that each class has its own BNp, which we partially know based on gene pathways. We employed a Gaussian model at the observation level to show the expression values of the genes given the hidden states at each time point. We used the expectation maximization (EM) methodology to learn the BNps and the unknown model parameters, derived closed-form updates for the parameters, and proposed a learning algorithm. After learning, a plug-in Bayes classifier was used to classify the unlabeled trajectories. The effect of missing data was also considered. We then proposed an intrinsically Bayesian robust classifier for classification of single-cell gene expression trajectories, which minimized the expected classification error over the uncertainty class of parameters.

In chapter 3 we constructed a Bayesian transfer learning framework to tackle the supervised transfer learning problem. The proposed Optimal Bayesian Transfer Learning (OBTL) classifier could deal with the lack of labeled data in the target domain and is optimal in this new Bayesian framework since it minimizes the expected classification error. We obtained the closed-form posterior distribution of the target parameters and accordingly the closed-form effective class-conditional densities in the target domain to define the OBTL classifier. As the OBTL's objective function consisted of hypergeometric functions of matrix argument, we used the Laplace approximations of those functions to derive a computationally efficient and scalable OBTL classifier, while preserving its superior performance. We compared the performance of the OBTL with its target-only version, OBC, to see how transferring from source to target domain can help. We tested the OBTL classifier with real-world benchmark image datasets and demonstrated its excellent performance compared to other state-of-the-art domain adaption methods. We further extended the OBTL to a regression problem and proposed Optimal Bayesian Transfer Regression (OBTR). We finally generalized the OBTL to count data and showed that it can effectively employ RNA-Seq count cancer data from TCGA and improve the cancer classification accuracy using heterogeneous sources of data.

REFERENCES

[1] A. Karbalayghareh, U. Braga-Neto, J. Hua, and E. R. Dougherty, "Classification of state trajectories in gene regulatory networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, pp. 68–82, Jan 2018.

[2] A. Karbalayghareh, U. Braga-Neto, and E. R. Dougherty, "Classification of single-cell gene expression trajectories from incomplete and noisy data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, pp. 193–207, Jan 2019.

[3] A. Karbalayghareh, U. Braga-Neto, and E. R. Dougherty, "Intrinsically bayesian robust classifier for single-cell gene expression trajectories in gene regulatory networks," *BMC Systems Biology*, vol. 12, p. 23, Mar 2018.

[4] A. Karbalayghareh, X. Qian, and E. R. Dougherty, "Optimal Bayesian transfer learning," *IEEE Transactions on Signal Processing*, vol. 66, pp. 3724–3739, July 2018.

[5] A. Karbalayghareh, X. Qian, and E. R. Dougherty, "Optimal Bayesian transfer regression," *IEEE Signal Processing Letters*, vol. 25, pp. 1655–1659, Nov 2018.

[6] A. Karbalayghareh, X. Qian, and E. R. Dougherty, "Optimal bayesian transfer learning for count data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2019.

[7] I. Shmulevich and E. R. Dougherty, *Probabilistic Boolean networks: the modeling and control of gene regulatory networks*. SIAM, 2010.

[8] S. Herath, M. Harandi, and F. Porikli, "Learning an invariant hilbert space for domain adaptation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3956–3965, July 2017.

[9] J. Hoffman, E. Rodner, T. Darrell, J. Donahue, and K. Saenko, "Efficient learning of domain-invariant image representations," in *International Conference on Learning Representations (ICLR)*, 2013.

[10] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proceedings of the 27th International Conference on Machine Learning*, pp. 863–870, 2010.

[11] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, *et al.*, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *science*, vol. 286, no. 5439, pp. 531–537, 1999.

[12] M. Bittner, P. Meltzer, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dor, *et al.*, "Molecular classification of cutaneous malignant melanoma by gene expression profiling," *Nature*, vol. 406, no. 6795, p. 536, 2000.

[13] J. Khan, J. S. Wei, M. Ringner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, *et al.*, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature medicine*, vol. 7, no. 6, p. 673, 2001.

[14] I. Hedenfalk, D. Duggan, Y. Chen, M. Radmacher, M. Bittner, R. Simon, P. Meltzer, B. Gusterson, M. Esteller, M. Raffeld, *et al.*, "Gene-expression profiles in hereditary breast cancer," *New England Journal of Medicine*, vol. 344, no. 8, pp. 539–548, 2001.

[15] U. M. Braga-Neto and E. R. Dougherty, "Is cross-validation valid for small-sample microarray classification?," *Bioinformatics*, vol. 20, no. 3, pp. 374–380, 2004.

[16] T. Mehta, M. Tanik, and D. B. Allison, "Towards sound epistemological foundations of statistical methods for high-dimensional biology," *Nature genetics*, vol. 36, no. 9, p. 943, 2004.

[17] C. Sima and E. R. Dougherty, "What should be expected from feature selection in small-sample settings," *Bioinformatics*, vol. 22, no. 19, pp. 2430–2436, 2006.

[18] L. Ein-Dor, O. Zuk, and E. Domany, "Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer," *Proceedings of the National Academy of Sciences*, vol. 103, no. 15, pp. 5923–5928, 2006.

[19] U. Braga-Neto, "Fads and fallacies in the name of small-sample microarray classification-a highlight of misunderstanding and erroneous usage in the applications of genomic signal processing," *IEEE Signal Processing Magazine*, vol. 1, no. 24, pp. 91–99, 2007.

[20] A. Dupuy and R. M. Simon, "Critical review of published microarray studies for cancer outcome and guidelines on statistical analysis and reporting," *Journal of the National Cancer Institute*, vol. 99, no. 2, pp. 147–157, 2007.

[21] E. R. Dougherty, M. Brun, J. M. Trent, and M. L. Bittner, "Conditioning-based modeling of contextual genomic regulation," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, no. 2, pp. 310–320, 2009.

[22] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, *et al.*, "Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles," *Proceedings of the National Academy of Sciences*, vol. 102, no. 43, pp. 15545–15550, 2005.

[23] L. Tian, S. A. Greenberg, S. W. Kong, J. Altschuler, I. S. Kohane, and P. J. Park, "Discovering statistically significant pathways in expression profiling studies," *Proceedings of the National Academy of Sciences*, vol. 102, no. 38, pp. 13544–13549, 2005.

[24] A. H. Bild, G. Yao, J. T. Chang, Q. Wang, A. Potti, D. Chasse, M.-B. Joshi, D. Harpole, J. M. Lancaster, A. Berchuck, *et al.*, "Oncogenic pathway signatures in human cancers as a guide to targeted therapies," *Nature*, vol. 439, no. 7074, p. 353, 2006.

[25] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, *et al.*, "Gene ontology: tool for the unification of biology," *Nature genetics*, vol. 25, no. 1, p. 25, 2000.

[26] Z. Guo, T. Zhang, X. Li, Q. Wang, J. Xu, H. Yu, J. Zhu, H. Wang, C. Wang, E. J. Topol, *et al.*, "Towards precise classification of cancers based on robust gene functional expression profiles," *BMC bioinformatics*, vol. 6, no. 1, p. 58, 2005.

[27] E. Lee, H.-Y. Chuang, J.-W. Kim, T. Ideker, and D. Lee, “Inferring pathway activity toward precise disease classification,” *PLoS computational biology*, vol. 4, no. 11, p. e1000217, 2008.

[28] J. Su, B.-J. Yoon, and E. R. Dougherty, “Accurate and reliable cancer classification based on probabilistic inference of pathway activity,” *PloS one*, vol. 4, no. 12, p. e8161, 2009.

[29] Z. Bar-Joseph, Z. Siegfried, M. Brandeis, B. Brors, Y. Lu, R. Eils, B. D. Dynlacht, and I. Simon, “Genome-wide transcriptional analysis of the human cell cycle identifies genes differentially regulated in normal and cancer cells,” *Proceedings of the National Academy of Sciences*, vol. 105, no. 3, pp. 955–960, 2008.

[30] A. R. Wu, N. F. Neff, T. Kalisky, P. Dalerba, B. Treutlein, M. E. Rothenberg, F. M. Mburu, G. L. Mantalas, S. Sim, M. F. Clarke, *et al.*, “Quantitative assessment of single-cell RNA-sequencing methods,” *Nature methods*, vol. 11, no. 1, pp. 41–46, 2014.

[31] A. Ståhlberg and M. Kubista, “The workflow of single-cell expression profiling using quantitative real-time pcr,” *Expert review of molecular diagnostics*, vol. 14, no. 3, pp. 323–331, 2014.

[32] C. Trapnell, D. Cacchiarelli, J. Grimsby, P. Pokharel, S. Li, M. Morse, N. J. Lennon, K. J. Livak, T. S. Mikkelsen, and J. L. Rinn, “The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells,” *Nature biotechnology*, vol. 32, no. 4, pp. 381–386, 2014.

[33] I. Shmulevich, E. R. Dougherty, and W. Zhang, “From boolean to probabilistic boolean networks as models of genetic regulatory networks,” *Proceedings of the IEEE*, vol. 90, no. 11, pp. 1778–1792, 2002.

[34] R. Pal, A. Datta, and E. R. Dougherty, “Optimal infinite-horizon control for probabilistic boolean networks,” *IEEE Transactions on Signal Processing*, vol. 54, no. 6, pp. 2375–2387, 2006.

[35] X. Qian and E. R. Dougherty, “Effect of function perturbation on the steady-state distribution of genetic regulatory networks: optimal structural intervention,” *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4966–4976, 2008.

[36] X. Qian and E. R. Dougherty, “On the long-run sensitivity of probabilistic boolean networks,” *Journal of theoretical biology*, vol. 257, no. 4, pp. 560–577, 2009.

[37] M. Brun, E. R. Dougherty, and I. Shmulevich, “Steady-state probabilities for attractors in probabilistic boolean networks,” *Signal Process.*, vol. 85, pp. 1993–2013, Oct. 2005.

[38] S. A. Kauffman, “Metabolic stability and epigenesis in randomly constructed genetic nets,” *Journal of theoretical biology*, vol. 22, no. 3, pp. 437–467, 1969.

[39] M. S. Esfahani, B.-J. Yoon, and E. R. Dougherty, “Probabilistic reconstruction of the tumor progression process in gene regulatory networks in the presence of uncertainty,” *BMC bioinformatics*, vol. 12, no. 10, p. S9, 2011.

[40] A. Karbalayghareh, U. Braga-Neto, and E. R. Dougherty, “Classification of gaussian trajectories with missing data in boolean gene regulatory networks,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1078–1082, March 2017.

[41] M. Imani and U. M. Braga-Neto, “Maximum-likelihood adaptive filter for partially observed boolean dynamical systems,” *IEEE Transactions on Signal Processing*, vol. 65, pp. 359–371, Jan 2017.

[42] C. Trapnell, “Defining cell types and states with single-cell genomics,” *Genome research*, vol. 25, no. 10, pp. 1491–1498, 2015.

[43] D. Usoskin, A. Furlan, S. Islam, H. Abdo, P. Lönnerberg, D. Lou, J. Hjerling-Leffler, J. Haeggström, O. Kharchenko, P. V. Kharchenko, *et al.*, “Unbiased classification of sensory neuron types by large-scale single-cell rna sequencing,” *Nature neuroscience*, vol. 18, no. 1, pp. 145–153, 2015.

[44] T. Ilicic, J. K. Kim, A. A. Kolodziejczyk, F. O. Bagger, D. J. McCarthy, J. C. Marioni, and S. A. Teichmann, “Classification of low quality cells from single-cell rna-seq data,” *Genome Biology*, vol. 17, p. 29, Feb 2016.

[45] K. Shekhar, S. W. Lapan, I. E. Whitney, N. M. Tran, E. Z. Macosko, M. Kowalczyk, X. Adiconis, J. Z. Levin, J. Nemesh, M. Goldman, *et al.*, “Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics,” *Cell*, vol. 166, no. 5, pp. 1308–1323, 2016.

[46] C. Shao and T. HÃűfer, “Robust classification of single-cell transcriptome data by nonnegative matrix factorization,” *Bioinformatics*, vol. 33, no. 2, pp. 235–242, 2017.

[47] M. Zamanighomi, Z. Lin, T. Daley, A. Schep, W. J. Greenleaf, and W. H. Wong, “Unsupervised clustering and epigenetic classification of single cells,” *bioRxiv*, 2017.

[48] P. V. Kharchenko, L. Silberstein, and D. T. Scadden, “Bayesian approach to single-cell differential expression analysis,” *Nature methods*, vol. 11, no. 7, pp. 740–742, 2014.

[49] Z. Wang, S. Jin, G. Liu, X. Zhang, N. Wang, D. Wu, Y. Hu, C. Zhang, Q. Jiang, L. Xu, and Y. Wang, “Dtwscore: differential expression and cell clustering analysis for time-series single-cell rna-seq data,” *BMC Bioinformatics*, vol. 18, p. 270, May 2017.

[50] S. Z. Dadaneh, X. Qian, and M. Zhou, “Bnp-seq: Bayesian nonparametric differential expression analysis of sequencing count data,” *Journal of the American Statistical Association*, vol. 113, no. 521, pp. 81–94, 2018.

[51] J. E. Reid and L. Wernisch, “Pseudotime estimation: deconfounding single cell time series,” *Bioinformatics*, vol. 32, no. 19, pp. 2973–2980, 2016.

[52] V. Moignard, S. Woodhouse, L. Haghverdi, A. J. Lilly, Y. Tanaka, A. C. Wilkinson, F. Buettner, I. C. Macaulay, W. Jawaid, E. Diamanti, *et al.*, “Decoding the regulatory network of early blood development from single-cell gene expression measurements,” *Nature biotechnology*, vol. 33, no. 3, pp. 269–276, 2015.

[53] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb 2001.

[54] U. Neto and E. Dougherty, *Error Estimation for Pattern Recognition*. IEEE Press Series on Biomedical Engineering, Wiley, 2015.

[55] E. Shapiro, T. Biezuner, and S. Linnarsson, “Single-cell sequencing-based technologies will revolutionize whole-organism science,” *Nature Reviews Genetics*, vol. 14, no. 9, p. 618, 2013.

[56] A. Fauré, A. Naldi, C. Chaouiya, and D. Thieffry, “Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle,” *Bioinformatics*, vol. 22, no. 14, pp. e124–e131, 2006.

[57] L. A. Dalton and E. R. Dougherty, “Optimal classifiers with minimum expected error within a Bayesian frameworkâĂŤPart I: Discrete and gaussian models,” *Pattern Recognition*, vol. 46, no. 5, pp. 1288 – 1300, 2013.

[58] L. A. Dalton and E. R. Dougherty, “Optimal classifiers with minimum expected error within a Bayesian framework âĂŤ Part II: Properties and performance analysis,” *Pattern Recognition*, vol. 46, no. 5, pp. 1301 – 1314, 2013.

[59] L. A. Dalton and E. R. Dougherty, “Intrinsically optimal Bayesian robust filtering,” *IEEE Transactions on Signal Processing*, vol. 62, pp. 657–670, Feb 2014.

[60] X. Qian and E. R. Dougherty, “Bayesian regression with network prior: Optimal Bayesian filtering perspective,” *IEEE Transactions on Signal Processing*, vol. 64, pp. 6243–6253, Dec 2016.

[61] R. Dehghannasiri, M. S. Esfahani, and E. R. Dougherty, “Intrinsically Bayesian robust Kalman filter: An innovation process approach,” *IEEE Transactions on Signal Processing*, vol. 65, pp. 2531–2546, May 2017.

[62] S. Boluki, M. S. Esfahani, X. Qian, and E. R. Dougherty, “Constructing pathway-based priors within a gaussian mixture model for Bayesian regression and classification,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2017.

[63] S. Boluki, M. S. Esfahani, X. Qian, and E. R. Dougherty, "Incorporating biological prior knowledge for bayesian learning via maximal knowledge-driven information priors," *BMC Bioinformatics*, vol. 18, p. 552, Dec 2017.

[64] L. A. Dalton and E. R. Dougherty, "Bayesian minimum mean-square error estimation for classification error-Part I: Definition and the Bayesian MMSE error estimator for discrete classification," *IEEE Transactions on Signal Processing*, vol. 59, pp. 115–129, Jan 2011.

[65] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[66] H. Venkateswara, S. Chakraborty, and S. Panchanathan, "Deep-learning systems for domain adaptation in computer vision: Learning transferable feature representations," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 117–129, 2017.

[67] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE signal processing magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[68] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, p. 9, 2016.

[69] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.

[70] N. Zou, Y. Zhu, J. Zhu, M. Baydogan, W. Wang, and J. Li, "A transfer learning approach for predictive modeling of degenerate biological systems," *Technometrics*, vol. 57, no. 3, pp. 362–373, 2015.

[71] P. Ganchev, D. Malehorn, W. L. Bigbee, and V. Gopalakrishnan, "Transfer learning of classification rules for biomarker discovery and verification from molecular profiling studies," *Journal of biomedical informatics*, vol. 44, pp. S17–S23, 2011.

[72] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2066–2073, IEEE, 2012.

[73] L. Duan, D. Xu, and I. Tsang, "Learning with augmented features for heterogeneous domain adaptation," *ICML*, 2012.

[74] J. Hoffman, E. Rodner, J. Donahue, B. Kulis, and K. Saenko, "Asymmetric and category invariant feature transformations for domain adaptation," *International journal of computer vision*, vol. 109, no. 1-2, pp. 28–41, 2014.

[75] Y.-H. Hubert Tsai, Y.-R. Yeh, and Y.-C. Frank Wang, "Learning cross-domain landmarks for heterogeneous domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5081–5090, 2016.

[76] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[77] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning*, pp. 193–200, ACM, 2007.

[78] L. Duan, I. W. Tsang, D. Xu, and S. J. Maybank, "Domain transfer svm for video concept detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1375–1381, IEEE, 2009.

[79] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 5, pp. 770–787, 2010.

[80] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1853–1865, Sept 2017.

[81] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International Conference on Machine Learning*, pp. 97–105, 2015.

[82] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Unsupervised domain adaptation with residual transfer networks,” in *Advances in Neural Information Processing Systems*, pp. 136–144, 2016.

[83] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[84] M.-Y. Liu and O. Tuzel, “Coupled generative adversarial networks,” in *Advances in neural information processing systems*, pp. 469–477, 2016.

[85] R. J. Muirhead, *Aspects of multivariate statistical theory*. John Wiley & Sons, 2009.

[86] K. Halvorsen, V. Ayala, and E. Fierro, “On the marginal distribution of the diagonal blocks in a blocked Wishart random matrix,” *International Journal of Analysis*, vol. 2016, pp. 1–5, 2016.

[87] D. K. Nagar and J. C. Mosquera-Benıtez, “Properties of matrix variate hypergeometric function distribution,” *Applied Mathematical Sciences*, vol. 11, no. 14, pp. 677–692, 2017.

[88] A. G. Constantine, “Some non-central distribution problems in multivariate analysis,” *Ann. Math. Statist.*, vol. 34, pp. 1270–1285, 12 1963.

[89] R. W. Butler and A. T. A. Wood, “Laplace approximations for hypergeometric functions with matrix argument,” *The Annals of Statistics*, vol. 30, no. 4, pp. 1155–1177, 2002.

[90] L. A. Dalton and M. R. Yousefi, “On optimal Bayesian classification and risk estimation under multiple classes,” *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2015, no. 1, p. 8, 2015.

[91] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains,” in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV’10, (Berlin, Heidelberg), pp. 213–226, Springer-Verlag, 2010.

[92] G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset,” *Technical Report 7694, California Institute of Technology*, 2007.

[93] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[94] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[95] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009.

[96] M. Roth, *On the multivariate t distribution*. Linköping University Electronic Press, 2012.

[97] R. M. Neal *et al.*, “MCMC using Hamiltonian dynamics,” *Handbook of Markov Chain Monte Carlo*, vol. 2, no. 11, 2011.

[98] B. Hanczar, J. Hua, C. Sima, J. Weinstein, M. Bittner, and E. R. Dougherty, “Small-sample precision of roc-related estimates,” *Bioinformatics*, vol. 26, no. 6, pp. 822–830, 2010.

[99] R. Normand, W. Du, M. Briller, R. Gaujoux, E. Starosvetsky, A. Ziv-Kenet, G. Shalev-Malul, R. J. Tibshirani, and S. S. Shen-Orr, “Found In Translation: a machine learning model for mouse-to-human inference,” *Nature methods*, vol. 15, no. 12, p. 1067, 2018.

[100] R. Petegrosso, R. Kuang, S. Park, and T. H. Hwang, “Transfer learning across ontologies for phenomeâĂŞgenome association prediction,” *Bioinformatics*, vol. 33, pp. 529–536, Feb 2017.

[101] S. R. Dhruba, R. Rahman, K. Matlock, S. Ghosh, and R. Pal, “Application of transfer learning for cancer drug sensitivity prediction,” *BMC Bioinformatics*, vol. 19, p. 497, Dec 2018.

[102] C. Y. Park, A. K. Wong, C. S. Greene, J. Rowland, Y. Guan, L. A. Bongo, R. D. Burdine, and O. G. Troyanskaya, “Functional knowledge transfer for high-accuracy prediction of under-studied biological processes,” *PLOS Computational Biology*, vol. 9, 03 2013.

[103] M. D. Robinson, D. J. McCarthy, and G. K. Smyth, “edger: a bioconductor package for differential expression analysis of digital gene expression data,” *Bioinformatics*, vol. 26, no. 1, pp. 139–140, 2010.

[104] S. Anders and W. Huber, “Differential expression analysis for sequence count data,” *Genome Biology*, vol. 11, p. R106, Oct 2010.

[105] S. Z. Dadaneh, M. Zhou, and X. Qian, “Bayesian negative binomial regression for differential expression with confounding factors,” *Bioinformatics*, vol. 34, no. 19, pp. 3349–3356, 2018.

[106] R. L. Grossman, A. P. Heath, V. Ferretti, H. E. Varmus, D. R. Lowy, W. A. Kibbe, and L. M. Staudt, “Toward a shared vision for cancer genomic data,” *New England Journal of Medicine*, vol. 375, no. 12, pp. 1109–1112, 2016.

[107] B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell, “Stan: A probabilistic programming language,” *Journal of Statistical Software, Articles*, vol. 76, no. 1, pp. 1–32, 2017.

[108] Y.-W. Wan, G. I. Allen, and Z. Liu, “TCGA2STAT: simple TCGA data access for integrated statistical analysis in R,” *Bioinformatics*, vol. 32, no. 6, pp. 952–954, 2016.

[109] D. K. Nagar and S. Nadarajah, “Appell’s hypergeometric functions of matrix arguments,” *Integral Transforms and Special Functions*, vol. 28, no. 2, pp. 91–112, 2017.

[110] A. K. Gupta, D. K. Nagar, and L. E. Sánchez, “Properties of matrix variate confluent hypergeometric function distribution,” *Journal of Probability and Statistics*, vol. 2016, 2016.

APPENDIX A

APPENDICES OF CHAPTER 2

A.1 Proof of lemma 1

Since $g(p_{k+1})$ is a valid beta probability density, as in (2.129), its integration with respect to p_{k+1} will be one:

$$\int_{\mathbf{P}} g(p_{k+1}) dp_{k+1} = \int_{\mathbf{P}} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p_{k+1}^{a-1}(1-p_{k+1})^{b-1} dp_{k+1} = 1. \tag{A.1}$$

Hence,

$$\int_{\mathbf{P}} p_{k+1}^{a-1}(1-p_{k+1})^{b-1} dp_{k+1} = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}. \tag{A.2}$$

After replacing $g(p_{k+1})$ in (1),

$$K_1 = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_{\mathbf{P}} p_{k+1}^{\mathbf{d}(\mathbf{X}_{k+1}, \mathbf{f}_s(\mathbf{X}_k))+a-1} \times (1-p_{k+1})^{n-\mathbf{d}(\mathbf{X}_{k+1}, \mathbf{f}_s(\mathbf{X}_k))+b-1} dp_{k+1}. \tag{A.3}$$

Using (A.2) and (A.3), K_1 is derived as in (1). □

A.2 Proof of lemma 2

It is well-known that the steady-state distribution of a time-homogeneous TPM is obtained from (2). The conditional TPM $\mathbf{A}^{(\mathbf{s})}(k+1)$ in (2.142) is time-inhomogeneous, since each time has its own perturbation probability p_{k+1}. Since the prior distribution of p_{k+1} in (2.129) is the same for every k, integrating the conditional TPM $\mathbf{A}^{(\mathbf{s})}(k+1)$, for every k, over the prior distribution of

p_{k+1} yields a time-homogeneous TPM with the (i,j)-th entry as

$$\mathbf{M}_{i,j}^{(s)} = \int_{\mathbf{P}} \mathbf{A}_{i,j}^{(s)}(k+1) g(p_{k+1}) dp_{k+1} = \int_{\mathbf{P}} g(p_{k+1}) p_{k+1}^{\mathbf{d}(\mathbf{x}^j, \mathbf{f}_s(\mathbf{x}^i))} (1-p_{k+1})^{n-\mathbf{d}(\mathbf{x}^j,\mathbf{f}_s(\mathbf{x}^i))} dp_{k+1}. \tag{A.4}$$

Lemma 1 and (A.4) result in (2.146). □

A.3 Proof of lemma 3

From (2.132), the normal-gamma prior for $\theta_j(k)$ and $\lambda_j(k)$ is

$$\begin{aligned} &p(\theta_j(k), \lambda_j(k)|x_j(k)) = p(\theta_j(k)|\lambda_j(k), x_j(k))\, p(\lambda_j(k)) \\ &= \frac{1}{Z_0}\lambda_j(k)^{\alpha_0 - \frac{1}{2}} \\ &\times \exp\left(-\frac{\lambda_j(k)}{2}\left[\kappa_0(\theta_j(k)-\mu_j(k))^2 + 2\beta_0\right]\right), \end{aligned} \tag{A.5}$$

where

$$Z_0 = \left(\frac{2\pi}{\kappa_0}\right)^{\frac{1}{2}} \frac{\Gamma(\alpha_0)}{\beta_0^{\alpha_0}}. \tag{A.6}$$

The likelihood from (2.131) is

$$\begin{aligned} &p(y_j(k)|\theta_j(k), \lambda_j(k)) \\ &= \frac{1}{(2\pi)^{\frac{1}{2}}}\lambda_j(k)^{\frac{1}{2}} \exp\left(-\frac{\lambda_j(k)}{2}(y_j(k)-\theta_j(k))^2\right). \end{aligned} \tag{A.7}$$

Therefore, for the posterior,

$$\begin{aligned} &p(\theta_j(k), \lambda_j(k)|y_j(k), x_j(k)) \propto p(y_j(k)|\theta_j(k), \lambda_j(k)) \\ &\quad \times p(\theta_j(k), \lambda_j(k)|x_j(k)) \quad \propto \quad \lambda_j(k)^{\alpha_0} \exp\left(-\frac{\lambda_j(k)}{2}\right. \\ &\left.\left[\kappa_0(\theta_j(k)-\mu_j(k))^2 + 2\beta_0 + (y_j(k)-\theta_j(k))^2\right]\right) \\ &\propto \lambda_j(k)^{\alpha_1-\frac{1}{2}} \exp\left(-\frac{\lambda_j(k)}{2}\left[\kappa_1(\theta_j(k)-\eta_j(k))^2 + 2\beta_1\right]\right), \end{aligned} \tag{A.8}$$

where κ_1, α_1, and β_1 are given in (2.148), and $\eta_j(k)$ is defined by

$$\eta_j(k) = \frac{\kappa_0 \mu_j(k) + y_j(k)}{\kappa_0 + 1}. \tag{A.9}$$

Comparing (A.8) with (A.5), we see that the posterior also has the following normal-gamma density:

$$\begin{aligned} &p(\theta_j(k), \lambda_j(k) | y_j(k), x_j(k)) = \\ &\frac{1}{Z_1} \lambda_j(k)^{\alpha_1 - \frac{1}{2}} \exp\left(-\frac{\lambda_j(k)}{2} \left[\kappa_1 (\theta_j(k) - \eta_j(k))^2 + 2\beta_1 \right] \right), \end{aligned} \tag{A.10}$$

where

$$Z_1 = \left(\frac{2\pi}{\kappa_1} \right)^{\frac{1}{2}} \frac{\Gamma(\alpha_1)}{\beta_1^{\alpha_1}}. \tag{A.11}$$

Since the posterior density in (A.10) integrates to 1,

$$\begin{aligned} &\int_{\mathbf{\Omega}} \int_{\mathbf{\Lambda}} \lambda_j(k)^{\alpha_1 - \frac{1}{2}} \exp\left(-\frac{\lambda_j(k)}{2} \right. \\ &\left. \left[\kappa_1 (\theta_j(k) - \eta_j(k))^2 + 2\beta_1 \right] \right) d\theta_j(k) d\lambda_j(k) = Z_1. \end{aligned} \tag{A.12}$$

Finally, K_2 in (2.147) can be written as

$$\begin{aligned} K_2 &= \frac{1}{(2\pi)^{\frac{1}{2}}} \frac{1}{Z_0} \int_{\mathbf{\Omega}} \int_{\mathbf{\Lambda}} \lambda_j(k)^{\alpha_1 - \frac{1}{2}} \\ &\times \exp\left(-\frac{\lambda_j(k)}{2} \left[\kappa_1 (\theta_j(k) - \eta_j(k))^2 + 2\beta_1 \right] \right) \\ &\quad d\theta_j(k) d\lambda_j(k) \\ &= \frac{1}{(2\pi)^{\frac{1}{2}}} \frac{Z_1}{Z_0} = \frac{1}{(2\pi)^{\frac{1}{2}}} \left(\frac{\kappa_0}{\kappa_1} \right)^{\frac{1}{2}} \frac{\Gamma(\alpha_1)}{\Gamma(\alpha_0)} \frac{\beta_0^{\alpha_0}}{\beta_1^{\alpha_1}}, \end{aligned}$$

which finishes the proof. □

APPENDIX B

APPENDICES OF CHAPTER 3

B.1 Theorems for Zonal Polynomials and Generalized Hypergeometric Functions of Matrix Argument

Theorem 8. *[85]: Let $\mathbf{Z}$ be a complex symmetric matrix whose real part is positive-definite, and let $\mathbf{X}$ be an arbitrary complex symmetric matrix. Then*

$$\begin{aligned}\int_{\mathbf{R}>0} & \operatorname{etr}(-\mathbf{ZR})|\mathbf{R}|^{\alpha-\frac{d+1}{2}} C_\kappa(\mathbf{RX}) d\mathbf{R} \\ &= \Gamma_d(\alpha)(\alpha)_\kappa |\mathbf{Z}|^{-\alpha} C_\kappa(\mathbf{XZ}^{-1}),\end{aligned} \tag{B.1}$$

the integration being over the space of positive-definite $d \times d$ matrices, and valid for all complex numbers α satisfying $\operatorname{Re}(\alpha) > \frac{d-1}{2}$. $\Gamma_d(\alpha)$ is the multivariate gamma function defined in (3.8).

Theorem 9. *[109]: The zonal polynomials are invariant under orthogonal transformation. That is, for a $d \times d$ symmetric matrix $\mathbf{X}$,*

$$C_\kappa(\mathbf{X}) = C_\kappa(\mathbf{HXH}'), \tag{B.2}$$

where $\mathbf{H}$ is an orthogonal matrix of order d. If $\mathbf{R}$ is a symmetric positive-definite matrix of order d, then

$$C_\kappa(\mathbf{RX}) = C_\kappa(\mathbf{R}^{1/2}\mathbf{XR}^{1/2}). \tag{B.3}$$

As a result, if $\mathbf{R}$ is a symmetric positive-definite matrix, the hypergeometric function has the following property:

$$\begin{aligned}& {}_pF_q(a_1, \cdots, a_p; b_1, \cdots, b_q; \mathbf{RX}) \\ &= {}_pF_q(a_1, \cdots, a_p; b_1, \cdots, b_q; \mathbf{R}^{1/2}\mathbf{XR}^{1/2}).\end{aligned} \tag{B.4}$$

Theorem 10. *[110]: If $\mathbf{Z} > 0$ and $\mathrm{Re}(\alpha) > \frac{d-1}{2}$, and $\mathbf{X}$ is a $d \times d$ symmetric matrix, we have*

$$\begin{aligned}
&\int_{\mathbf{R}>0} \mathrm{etr}(-\mathbf{Z}\mathbf{R})|\mathbf{R}|^{\alpha-\frac{d+1}{2}} \\
&\qquad \times {}_pF_q(a_1, \cdots, a_p; b_1, \cdots, b_q; \mathbf{R}\mathbf{X}) d\mathbf{R} \\
&= \int_{\mathbf{R}>0} \mathrm{etr}(-\mathbf{Z}\mathbf{R})|\mathbf{R}|^{\alpha-\frac{d+1}{2}} \\
&\qquad \times {}_pF_q(a_1, \cdots, a_p; b_1, \cdots, b_q; \mathbf{R}^{1/2}\mathbf{X}\mathbf{R}^{1/2}) d\mathbf{R} \\
&= \Gamma_d(\alpha)|\mathbf{Z}|^{-\alpha}\ {}_{p+1}F_q(a_1, \cdots, a_p, \alpha; b_1, \cdots, b_q; \mathbf{X}\mathbf{Z}^{-1}).
\end{aligned}$$

B.2 Proof of Theorem 2

We require the following lemma.

Lemma 6. *[85] If $\mathcal{D} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ where $\mathbf{x}_i$ is a $d \times 1$ vector and $\mathbf{x}_i \sim \mathcal{N}(\mu, (\mathbf{\Lambda})^{-1})$, for $i = 1, \cdots, n$, and $(\mu, \mathbf{\Lambda})$ has a Gaussian-Wishart prior, such that, $\mu|\mathbf{\Lambda} \sim \mathcal{N}(\mathbf{m}, (\kappa\mathbf{\Lambda})^{-1})$ and $\mathbf{\Lambda} \sim W_d(\mathbf{M}, \nu)$, then the posterior of $(\mu, \mathbf{\Lambda})$ upon observing $\mathcal{D}$ is also a Gaussian-Wishart distribution:*

$$\begin{aligned}
\mu|\mathbf{\Lambda}, \mathcal{D} &\sim \mathcal{N}(\mathbf{m}_n, (\kappa_n\mathbf{\Lambda})^{-1}), \\
\mathbf{\Lambda}|\mathcal{D} &\sim W_d(\mathbf{M}_n, \nu_n),
\end{aligned} \tag{B.5}$$

where

$$\begin{aligned}
\kappa_n &= \kappa + n, \\
\nu_n &= \nu + n, \\
\mathbf{m}_n &= \frac{\kappa\mathbf{m} + n\bar{\mathbf{x}}}{\kappa + n}, \\
\mathbf{M}_n^{-1} &= \mathbf{M}^{-1} + \mathbf{S} + \frac{\kappa n}{\kappa + n}(\mathbf{m} - \bar{\mathbf{x}})(\mathbf{m} - \bar{\mathbf{x}})',
\end{aligned} \tag{B.6}$$

depending on the sample mean and covariance matrix

$$\begin{aligned}
\bar{\mathbf{x}} &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i, \\
\mathbf{S} &= \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})'.
\end{aligned} \tag{B.7}$$

We now provide the proof. From (3.3), for each domain $z \in \{s, t\}$,

$$p(\mathcal{D}_z^l | \mu_z^l, \mathbf{\Lambda}_z^l) = (2\pi)^{-\frac{dn_z^l}{2}} \left|\mathbf{\Lambda}_z^l\right|^{\frac{n_z^l}{2}} \exp\left(-\frac{1}{2}\mathbf{Q}_z^l\right), \tag{B.8}$$

where $\mathbf{Q}_z^l = \sum_{i=1}^{n_z^l} \left(\mathbf{x}_{z,i}^l - \mu_z^l\right)' \mathbf{\Lambda}_z^l \left(\mathbf{x}_{z,i}^l - \mu_z^l\right)$. Moreover, from (3.6), for each domain $z \in \{s, t\}$,

$$\begin{aligned} p\left(\mu_z^l | \mathbf{\Lambda}_z^l\right) &= (2\pi)^{-\frac{d}{2}} \left(\kappa_z^l\right)^{\frac{d}{2}} \left|\mathbf{\Lambda}_z^l\right|^{\frac{1}{2}} \\ &\quad \times \exp\left(-\frac{\kappa_z^l}{2} \left(\mu_z^l - \mathbf{m}_z^l\right)' \mathbf{\Lambda}_z^l \left(\mu_z^l - \mathbf{m}_z^l\right)\right). \end{aligned} \tag{B.9}$$

From (3.13), (3.21), (B.8), and (B.9),

$$\begin{aligned} p(\mu_t^l, \mathbf{\Lambda}_t^l | \mathcal{D}_t^l, \mathcal{D}_s^l) &\propto \left|\mathbf{\Lambda}_t^l\right|^{\frac{n_t^l}{2}} \exp\left(-\frac{1}{2}\mathbf{Q}_t^l\right) \left|\mathbf{\Lambda}_t^l\right|^{\frac{1}{2}} \\ &\times \exp\left(-\frac{\kappa_t^l}{2} \left(\mu_t^l - \mathbf{m}_t^l\right)' \mathbf{\Lambda}_t^l \left(\mu_t^l - \mathbf{m}_t^l\right)\right) \\ &\times \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l - d - 1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\left(\mathbf{M}_t^l\right)^{-1} + \mathbf{F}^{l'} \mathbf{C}^l \mathbf{F}^l\right) \mathbf{\Lambda}_t^l\right) \\ &\times \int_{\mu_s^l, \mathbf{\Lambda}_s^l} \left\{ \left|\mathbf{\Lambda}_s^l\right|^{\frac{n_s^l}{2}} \exp\left(-\frac{1}{2}\mathbf{Q}_s^l\right) \left|\mathbf{\Lambda}_s^l\right|^{\frac{1}{2}} \right. \\ &\times \exp\left(-\frac{\kappa_s^l}{2} \left(\mu_s^l - \mathbf{m}_s^l\right)' \mathbf{\Lambda}_s^l \left(\mu_s^l - \mathbf{m}_s^l\right)\right) \\ &\times \left|\mathbf{\Lambda}_s^l\right|^{\frac{\nu^l - d - 1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{C}^l\right)^{-1} \mathbf{\Lambda}_s^l\right) \\ &\times \left. {}_0F_1\left(\frac{\nu^l}{2}; \frac{1}{4} \mathbf{\Lambda}_s^{l\,\frac{1}{2}} \mathbf{F}^l \mathbf{\Lambda}_t^l \mathbf{F}^{l'} \mathbf{\Lambda}_s^{l\,\frac{1}{2}}\right) \right\} d\mu_s^l d\mathbf{\Lambda}_s^l. \end{aligned} \tag{B.10}$$

Using Lemma 6 we can simplify (B.10) as

$$\begin{aligned}
&p(\mu_t^l, \mathbf{\Lambda}_t^l | \mathcal{D}_t^l, \mathcal{D}_s^l) \\
&\propto \left|\mathbf{\Lambda}_t^l\right|^{\frac{1}{2}} \exp\left(-\frac{\kappa_{t,n}^l}{2}\left(\mu_t^l - \mathbf{m}_{t,n}^l\right)' \mathbf{\Lambda}_t^l \left(\mu_t^l - \mathbf{m}_{t,n}^l\right)\right) \\
&\times \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l + n_t^l - d - 1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_t^l\right)^{-1}\mathbf{\Lambda}_t^l\right) \\
&\int_{\mu_s^l, \mathbf{\Lambda}_s^l} \left\{\left|\mathbf{\Lambda}_s^l\right|^{\frac{1}{2}} \exp\left(-\frac{\kappa_{s,n}^l}{2}\left(\mu_s^l - \mathbf{m}_{s,n}^l\right)' \mathbf{\Lambda}_s^l \left(\mu_s^l - \mathbf{m}_{s,n}^l\right)\right)\right. \\
&\times \left|\mathbf{\Lambda}_s^l\right|^{\frac{\nu^l + n_s^l - d - 1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_s^l\right)^{-1}\mathbf{\Lambda}_s^l\right) \\
&\left.\times {}_0F_1\left(\frac{\nu^l}{2}; \frac{1}{4}\mathbf{\Lambda}_s^{l\,\frac{1}{2}}\mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l'}\mathbf{\Lambda}_s^{l\,\frac{1}{2}}\right)\right\} d\mu_s^l d\mathbf{\Lambda}_s^l,
\end{aligned} \tag{B.11}$$

where

$$\begin{aligned}
&\kappa_{t,n}^l = \kappa_t^l + n_t^l, \qquad \kappa_{s,n}^l = \kappa_s^l + n_s^l, \\
&\mathbf{m}_{t,n}^l = \frac{\kappa_t^l\mathbf{m}_t^l + n_t^l\bar{\mathbf{x}}_t^l}{\kappa_t^l + n_t^l}, \quad \mathbf{m}_{s,n}^l = \frac{\kappa_s^l\mathbf{m}_s^l + n_s^l\bar{\mathbf{x}}_s^l}{\kappa_s^l + n_s^l}, \\
&\left(\mathbf{T}_t^l\right)^{-1} = \left(\mathbf{M}_t^l\right)^{-1} + \mathbf{F}^{l'}\mathbf{C}^l\mathbf{F}^l + \mathbf{S}_t^l \\
&\qquad + \frac{\kappa_t^l n_t^l}{\kappa_t^l + n_t^l}(\mathbf{m}_t^l - \bar{\mathbf{x}}_t^l)(\mathbf{m}_t^l - \bar{\mathbf{x}}_t^l)', \\
&\left(\mathbf{T}_s^l\right)^{-1} = \left(\mathbf{C}^l\right)^{-1} + \mathbf{S}_s^l + \frac{\kappa_s^l n_s^l}{\kappa_s^l + n_s^l}(\mathbf{m}_s^l - \bar{\mathbf{x}}_s^l)(\mathbf{m}_s^l - \bar{\mathbf{x}}_s^l)',
\end{aligned} \tag{B.12}$$

with sample means and covariances for $z \in \{s, t\}$ as

$$\bar{\mathbf{x}}_z^l = \frac{1}{n_z^l}\sum_{i=1}^{n_z^l}\mathbf{x}_{z,i}^l, \quad \mathbf{S}_z^l = \sum_{i=1}^{n_z^l}\left(\mathbf{x}_{z,i}^l - \bar{\mathbf{x}}_z^l\right)\left(\mathbf{x}_{z,i}^l - \bar{\mathbf{x}}_z^l\right)'.$$

Using the equation

$$\int_{\mathbf{x}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)'\mathbf{\Lambda}(\mathbf{x} - \mu)\right) d\mathbf{x} = (2\pi)^{\frac{d}{2}}|\mathbf{\Lambda}|^{-\frac{1}{2}}, \tag{B.13}$$

and integrating out μ_s^l in (B.11) yields

$$\begin{aligned} & p(\mu_t^l, \mathbf{\Lambda}_t^l | \mathcal{D}_t^l, \mathcal{D}_s^l) \\ & \propto \left|\mathbf{\Lambda}_t^l\right|^{\frac{1}{2}} \exp\left(-\frac{\kappa_{t,n}^l}{2}\left(\mu_t^l - \mathbf{m}_{t,n}^l\right)' \mathbf{\Lambda}_t^l \left(\mu_t^l - \mathbf{m}_{t,n}^l\right)\right) \\ & \times \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l + n_t^l - d - 1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_t^l\right)^{-1}\mathbf{\Lambda}_t^l\right) \\ & \times \int_{\mathbf{\Lambda}_s^l} \left\{\left|\mathbf{\Lambda}_s^l\right|^{\frac{\nu^l + n_s^l - d - 1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_s^l\right)^{-1}\mathbf{\Lambda}_s^l\right)\right. \\ & \qquad \left. \times \ {}_0F_1\left(\frac{\nu^l}{2}; \frac{1}{4}\mathbf{\Lambda}_s^{l\,\frac{1}{2}}\mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l'}\mathbf{\Lambda}_s^{l\,\frac{1}{2}}\right)\right\} d\mathbf{\Lambda}_s^l. \end{aligned} \tag{B.14}$$

The integral, I, in (B.14) can be done using Theorem 10 as

$$\begin{aligned} I = {} & \Gamma_d\left(\frac{\nu^l + n_s^l}{2}\right) \\ & \times \left|2\mathbf{T}_s^l\right|^{\frac{\nu^l + n_s^l}{2}} \ {}_1F_1\left(\frac{\nu^l + n_s^l}{2}; \frac{\nu^l}{2}; \frac{1}{2}\mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l'}\mathbf{T}_s^l\right), \end{aligned} \tag{B.15}$$

where ${}_1F_1(a; b; \mathbf{X})$ is the Confluent hypergeometric function with the matrix argument $\mathbf{X}$. As a result, (B.14) becomes

$$\begin{aligned} & p(\mu_t^l, \mathbf{\Lambda}_t^l | \mathcal{D}_t^l, \mathcal{D}_s^l) = \\ & A^l \left|\mathbf{\Lambda}_t^l\right|^{\frac{1}{2}} \exp\left(-\frac{\kappa_{t,n}^l}{2}\left(\mu_t^l - \mathbf{m}_{t,n}^l\right)' \mathbf{\Lambda}_t^l \left(\mu_t^l - \mathbf{m}_{t,n}^l\right)\right) \\ & \times \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l + n_t^l - d - 1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_t^l\right)^{-1}\mathbf{\Lambda}_t^l\right) \\ & \times \ {}_1F_1\left(\frac{\nu^l + n_s^l}{2}; \frac{\nu^l}{2}; \frac{1}{2}\mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l'}\mathbf{T}_s^l\right), \end{aligned} \tag{B.16}$$

where the constant of proportionality, A^l, makes the integration of the posterior $p(\mu_t^l, \mathbf{\Lambda}_t^l | \mathcal{D}_t^l, \mathcal{D}_s^l)$

with respect to μ_t^l and $\mathbf{\Lambda}_t^l$ equal to one. Hence,

$$\begin{aligned}\left(A^l\right)^{-1} = &\int_{\mathbf{\Lambda}_t^l} \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l+n_t^l-d-1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_t^l\right)^{-1}\mathbf{\Lambda}_t^l\right)\left|\mathbf{\Lambda}_t^l\right|^{\frac{1}{2}} \\ &\times \int_{\mu_t^l} \exp\left(-\frac{\kappa_{t,n}^l}{2}\left(\mu_t^l - \mathbf{m}_{t,n}^l\right)^{\prime}\mathbf{\Lambda}_t^l\left(\mu_t^l - \mathbf{m}_{t,n}^l\right)\right) d\mu_t^l \\ &\times {}_1F_1\left(\frac{\nu^l+n_s^l}{2};\frac{\nu^l}{2};\frac{1}{2}\mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l^{\prime}}\mathbf{T}_s^l\right) d\mathbf{\Lambda}_t^l.\end{aligned} \tag{B.17}$$

Using (B.13), the inner integral equals to $(2\pi)^{\frac{d}{2}}|\kappa_{t,n}^l\mathbf{\Lambda}_t^l|^{-\frac{1}{2}} = \left(\frac{2\pi}{\kappa_{t,n}^l}\right)^{\frac{d}{2}}|\mathbf{\Lambda}_t^l|^{-\frac{1}{2}}$. Hence,

$$\begin{aligned}\left(A^l\right)^{-1} = \left(\frac{2\pi}{\kappa_{t,n}^l}\right)^{\frac{d}{2}} &\int_{\mathbf{\Lambda}_t^l} \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l+n_t^l-d-1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_t^l\right)^{-1}\mathbf{\Lambda}_t^l\right) \\ &\times {}_1F_1\left(\frac{\nu^l+n_s^l}{2};\frac{\nu^l}{2};\frac{1}{2}\mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l^{\prime}}\mathbf{T}_s^l\right) d\mathbf{\Lambda}_t^l.\end{aligned} \tag{B.18}$$

With the variable change $\Omega = \mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l^{\prime}}$, we have $d\Omega = |\mathbf{F}^l|^{d+1}d\mathbf{\Lambda}_t^l$ and $\mathbf{\Lambda}_t^l = \left(\mathbf{F}^l\right)^{-1}\Omega\left(\mathbf{F}^{l^{\prime}}\right)^{-1}$. Since $\operatorname{tr}(\mathbf{ABCD}) = \operatorname{tr}(\mathbf{BCDA}) = \operatorname{tr}(\mathbf{CDAB}) = \operatorname{tr}(\mathbf{DABC})$ and $|\mathbf{ABC}| = |\mathbf{A}||\mathbf{B}||\mathbf{C}|$, A^l can be derived as

$$\begin{aligned}\left(A^l\right)^{-1} &= \left(\frac{2\pi}{\kappa_{t,n}^l}\right)^{\frac{d}{2}} |\mathbf{F}^l|^{-\left(\nu^l+n_t^l\right)} \int_{\Omega}\Bigg\{|\Omega|^{\frac{\nu^l+n_t^l-d-1}{2}} \\ &\quad\times \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{F}^{l^{\prime}}\right)^{-1}\left(\mathbf{T}_t^l\right)^{-1}\mathbf{F}^{l^{-1}}\Omega\right) \\ &\quad\times {}_1F_1\left(\frac{\nu^l+n_s^l}{2};\frac{\nu^l}{2};\frac{1}{2}\Omega\mathbf{T}_s^l\right)\Bigg\} d\Omega \\ &= \left(\frac{2\pi}{\kappa_{t,n}^l}\right)^{\frac{d}{2}} 2^{\frac{d\left(\nu^l+n_t^l\right)}{2}}\Gamma_d\left(\frac{\nu^l+n_t^l}{2}\right)\left|\mathbf{T}_t^l\right|^{\frac{\nu^l+n_t^l}{2}} \\ &\quad\times {}_2F_1\left(\frac{\nu^l+n_s^l}{2},\frac{\nu^l+n_t^l}{2};\frac{\nu^l}{2};\mathbf{T}_s^l\mathbf{F}^l\mathbf{T}_t^l\mathbf{F}^{l^{\prime}}\right),\end{aligned} \tag{B.19}$$

where the second equality follows from Theorem 10, and ${}_2F_1(a,b;c;\mathbf{X})$ is the Gauss hypergeometric function with the matrix argument $\mathbf{X}$. As such, we have derived the closed-form posterior distribution of the target parameters $(\mu_t^l, \mathbf{\Lambda}_t^l)$ in (3.22), where A^l is given by (3.23).

B.3 Proof of Theorem 3

The likelihood $p(\mathbf{x}|\mu_t^l, \mathbf{\Lambda}_t^l)$ and posterior $p(\mu_t^l, \mathbf{\Lambda}_t^l|\mathcal{D}_t^l, \mathcal{D}_s^l)$ are given in (3.3) and (3.22), respectively. Hence,

$$\begin{aligned} p(\mathbf{x}|l) = (2\pi)^{-\frac{d}{2}} A^l \int_{\mu_t^l, \mathbf{\Lambda}_t^l} \Big\{ & |\mathbf{\Lambda}_t^l|^{\frac{1}{2}} \\ & \times \exp\left(-\frac{1}{2}\left(\mathbf{x}-\mu_t^l\right)' \mathbf{\Lambda}_t^l \left(\mathbf{x}-\mu_t^l\right)\right) \\ & \times \left|\mathbf{\Lambda}_t^l\right|^{\frac{1}{2}} \exp\left(-\frac{\kappa_{t,n}^l}{2}\left(\mu_t^l-\mathbf{m}_{t,n}^l\right)' \mathbf{\Lambda}_t^l \left(\mu_t^l-\mathbf{m}_{t,n}^l\right)\right) \\ & \times \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l+n_t^l-d-1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_t^l\right)^{-1}\mathbf{\Lambda}_t^l\right) \\ & \times {}_1F_1\left(\frac{\nu^l+n_s^l}{2}; \frac{\nu^l}{2}; \frac{1}{2}\mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l'}\mathbf{T}_s^l\right)\Big\} d\mu_t^l d\mathbf{\Lambda}_t^l. \end{aligned} \tag{B.20}$$

Similarly, we can simplify (B.20) as

$$\begin{aligned} p(\mathbf{x}|l) = (2\pi)^{-\frac{d}{2}} A^l \int_{\mu_t^l, \mathbf{\Lambda}_t^l} \Big\{ & |\mathbf{\Lambda}_t^l|^{\frac{1}{2}} \\ & \times \exp\left(-\frac{\kappa_{\mathbf{x}}^l}{2}\left(\mu_t^l-\mathbf{m}_{\mathbf{x}}^l\right)' \mathbf{\Lambda}_t^l \left(\mu_t^l-\mathbf{m}_{\mathbf{x}}^l\right)\right) \\ & \times \left|\mathbf{\Lambda}_t^l\right|^{\frac{\nu^l+n_t^l+1-d-1}{2}} \operatorname{etr}\left(-\frac{1}{2}\left(\mathbf{T}_{\mathbf{x}}^l\right)^{-1}\mathbf{\Lambda}_t^l\right) \\ & \times {}_1F_1\left(\frac{\nu^l+n_s^l}{2}; \frac{\nu^l}{2}; \frac{1}{2}\mathbf{F}^l\mathbf{\Lambda}_t^l\mathbf{F}^{l'}\mathbf{T}_s^l\right)\Big\} d\mu_t^l d\mathbf{\Lambda}_t^l, \end{aligned} \tag{B.21}$$

where

$$\begin{aligned} \kappa_{\mathbf{x}}^l = \kappa_{t,n}^l + 1 = \kappa_t^l + n_t^l + 1, \quad \mathbf{m}_{\mathbf{x}}^l = \frac{\kappa_{t,n}^l \mathbf{m}_{t,n}^l + \mathbf{x}}{\kappa_{t,n}+1}, \\ \left(\mathbf{T}_{\mathbf{x}}^l\right)^{-1} = \left(\mathbf{T}_t^l\right)^{-1} + \frac{\kappa_{t,n}^l}{\kappa_{t,n}^l+1}\left(\mathbf{m}_{t,n}^l-\mathbf{x}\right)\left(\mathbf{m}_{t,n}^l-\mathbf{x}\right)'. \end{aligned} \tag{B.22}$$

The integration in (B.21) is similar to the one in (B.17). As a result, using (3.23),

$$\begin{aligned} p(\mathbf{x}|l) = (2\pi)^{-\frac{d}{2}} A^l \left(\frac{2\pi}{\kappa_{\mathbf{x}}^l}\right)^{\frac{d}{2}} 2^{\frac{d\left(\nu^l+n_t^l+1\right)}{2}} \Gamma_d\left(\frac{\nu^l+n_t^l+1}{2}\right) \\ \left|\mathbf{T}_{\mathbf{x}}^l\right|^{\frac{\nu^l+n_t^l+1}{2}} {}_2F_1\left(\frac{\nu^l+n_s^l}{2}, \frac{\nu^l+n_t^l+1}{2}; \frac{\nu^l}{2}; \mathbf{T}_s^l\mathbf{F}^l\mathbf{T}_{\mathbf{x}}^l\mathbf{F}^{l'}\right). \end{aligned} \tag{B.23}$$

By replacing the value of A^l, we have the effective class-conditional density. We denote $O_{\mathrm{OBTL}}(\mathbf{x}|l) = p(\mathbf{x}|l)$, since it is the objective function for the OBTL classifier. As such,

$$\begin{aligned} O_{\mathrm{OBTL}}(\mathbf{x}|l) &= \pi^{-\frac{d}{2}} \left(\frac{\kappa_{t,n}^l}{\kappa_{\mathbf{x}}^l}\right)^{\frac{d}{2}} \Gamma_d\left(\frac{\nu^l + n_t^l + 1}{2}\right) \\ &\times \Gamma_d^{-1}\left(\frac{\nu^l + n_t^l}{2}\right) \left|\mathbf{T}_{\mathbf{x}}^l\right|^{\frac{\nu^l+n_t^l+1}{2}} \left|\mathbf{T}_t^l\right|^{-\frac{\nu^l+n_t^l}{2}} \\ &\times \ {}_2F_1\left(\frac{\nu^l + n_s^l}{2}, \frac{\nu^l + n_t^l + 1}{2}; \frac{\nu^l}{2}; \mathbf{T}_s^l \mathbf{F}^l \mathbf{T}_{\mathbf{x}}^l \mathbf{F}^{l'}\right) \\ &\times \ {}_2F_1^{-1}\left(\frac{\nu^l + n_s^l}{2}, \frac{\nu^l + n_t^l}{2}; \frac{\nu^l}{2}; \mathbf{T}_s^l \mathbf{F}^l \mathbf{T}_t^l \mathbf{F}^{l'}\right). \end{aligned} \tag{B.24}$$

B.4 Laplace Approximation of the Gauss Hypergeometric Function of Matrix Argument

The Gauss hypergeomeric function has the following integral representation:

$$\begin{aligned} {}_2F_1(a, b; c; \mathbf{X}) &= B_d^{-1}(a, c-a) \\ &\times \int_{0_d < \mathbf{Y} < \mathbf{I}_d} |\mathbf{Y}|^{a-\frac{d+1}{2}} |\mathbf{I}_d - \mathbf{Y}|^{c-a-\frac{d+1}{2}} |\mathbf{I}_d - \mathbf{X}\mathbf{Y}|^{-b} d\mathbf{Y}, \end{aligned} \tag{B.25}$$

which is valid under the following conditions: $\mathbf{X} \in \mathbf{C}^{d\times d}$ is symmetric and satisfies $\mathrm{Re}(\mathbf{X}) < \mathbf{I}_d$, $\mathrm{Re}(a) > \frac{d-1}{2}$, and $\mathrm{Re}(c-a) > \frac{d-1}{2}$. $B_d(\alpha, \beta)$ is the multivariate beta function

$$B_d(\alpha, \beta) = \frac{\Gamma_d(\alpha)\Gamma_d(\beta)}{\Gamma_d(\alpha+\beta)}, \tag{B.26}$$

where $\Gamma_d(\alpha)$ is the multivariate gamma function defined in (3.8). The Laplace approximation is one common solution to approximate the integral

$$I = \int_{y \in D} h(y) \exp(-\lambda g(y)) dy, \tag{B.27}$$

where $D \subseteq \mathbf{R}^d$ is an open set and λ is a real parameter. If $g(\lambda)$ has a unique minimum over D at point $\hat{y} \in D$, then the Laplace approximation to I is given by

$$\tilde{I} = (2\pi)^{\frac{d}{2}} \lambda^{-\frac{d}{2}} |g^{''}(\hat{y})|^{-\frac{1}{2}} h(\hat{y}) \exp(-\lambda g(\hat{y})), \tag{B.28}$$

where $g^{''}(y) = \frac{\partial^2 g(y)}{\partial y \partial y^T}$ is the Hessian of $g(y)$. The hypergeometric function ${}_2F_1(a, b; c; \mathbf{X})$ depends only on the eigenvalues of the symmetric matrix $\mathbf{X}$. Hence, without loss of generality, it is assumed that $\mathbf{X} = \mathrm{diag}\{x_1, \cdots, x_d\}$. The following g and h functions are used for (B.25):

$$\begin{aligned} g(\mathbf{Y}) &= -a \log |\mathbf{Y}| - (c-a) \log |\mathbf{I}_d - \mathbf{Y}| + \log |\mathbf{I}_d - \mathbf{X}\mathbf{Y}|, \\ h(\mathbf{Y}) &= B_d^{-1}(a, c-a) |\mathbf{Y}|^{-\frac{d+1}{2}} |\mathbf{I}_d - \mathbf{Y}|^{-\frac{d+1}{2}}. \end{aligned} \tag{B.29}$$

Using (B.28) and (B.29), the Laplace approximation to ${}_2F_1(a, b; c; \mathbf{X})$ is given by [89]

$$\begin{aligned} {}_2\tilde{F}_1(a, b; c; \mathbf{X}) = & \frac{2^{\frac{d}{2}} \pi^{\frac{d(d+1)}{4}}}{B_d(a, c-a)} J_{2,1}^{-\frac{1}{2}} \\ & \times \prod_{i=1}^{d} \{\hat{y}_i^a (1-\hat{y}_i)^{c-a} (1-x_i\hat{y}_i)^{-b}\}, \end{aligned} \tag{B.30}$$

where $\hat{y}_i$ is defined as

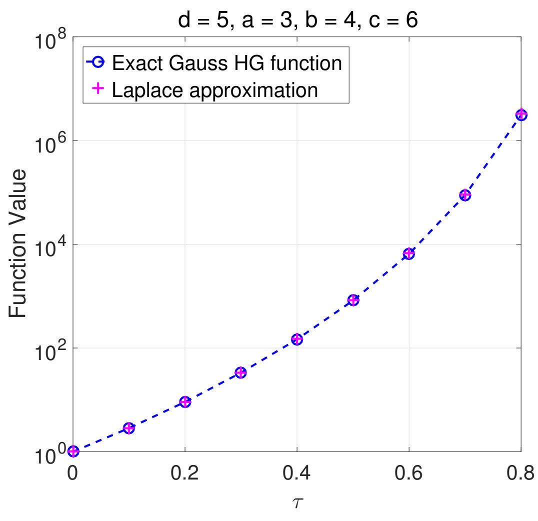
$$\hat{y}_i = \frac{2a}{\sqrt{\tau^2 - 4ax_i(c-b)} - \tau}, \tag{B.31}$$

with $\tau = x_i(b-a) - c$, and

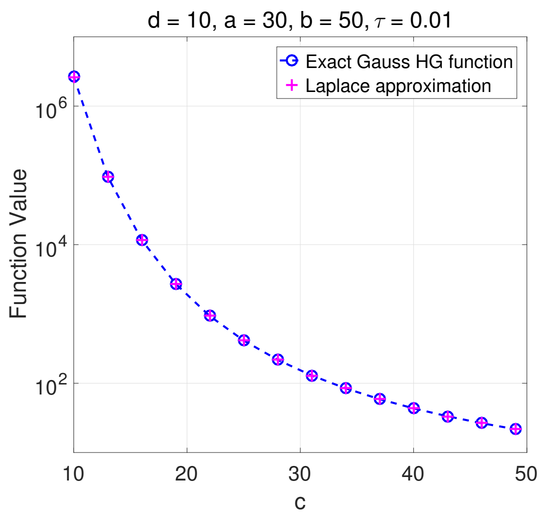
$$J_{2,1} = \prod_{i=1}^{d} \prod_{j=i}^{d} \{a(1-\hat{y}_i)(1-\hat{y}_j) + (c-a)\hat{y}_i\hat{y}_j - bL_iL_j\}, \tag{B.32}$$

with

$$L_i = \frac{x_i\hat{y}_i(1-\hat{y}_i)}{1-x_i\hat{y}_i}. \tag{B.33}$$



(a)



(b)

Figure B.1: Exact values of function ${}_2F_1(a,b;c;\tau I_d)$ and its Laplace approximation ${}_2\hat{F}_1(a,b;c;\tau I_d)$ versus: (a) τ, for $d=5$, $a=3$, $b=4$, and $c=6$, (b) c, for $d=10$, $a=30$, $b=50$, and $\tau=0.01$. Reprinted with permission from [4], ©2018 IEEE.

The value of ${}_2F_1(a,b;c;\mathbf{X})$ at $\mathbf{X}=\mathbf{0}$ is 1, that is, ${}_2F_1(a,b;c;\mathbf{0})=1$. As a result, the Laplace approximation in (B.30) is calibrated at $\mathbf{X}=\mathbf{0}$ to give the calibrated Laplace approximation [89]:

$$\begin{aligned}{}_2\hat{F}_1(a,b;c;\mathbf{X}) &= \frac{{}_2\tilde{F}_1(a,b;c;\mathbf{X})}{{}_2\tilde{F}_1(a,b;c;\mathbf{0})} = c^{cd-\frac{d(d+1)}{4}} R_{2,1}^{-\frac{1}{2}} \\ &\times \prod_{i=1}^{d}\left\{\left(\frac{\hat{y}_i}{a}\right)^{a}\left(\frac{1-\hat{y}_i}{c-a}\right)^{c-a}(1-x_i\hat{y}_i)^{-b}\right\},\end{aligned} \tag{B.34}$$

where

$$\begin{aligned}R_{2,1} = \prod_{i=1}^{d}\prod_{j=i}^{d}\Bigg\{&\frac{\hat{y}_i\hat{y}_j}{a}+\frac{(1-\hat{y}_i)(1-\hat{y}_j)}{c-a} \\ &-\frac{bx_ix_j\hat{y}_i\hat{y}_j(1-\hat{y}_i)(1-\hat{y}_j)}{(1-x_i\hat{y}_i)(1-x_j\hat{y}_j)a(c-a)}\Bigg\}.\end{aligned} \tag{B.35}$$

According to [89], the relative error of the approximation remains uniformly bounded:

$$\sup|\log\ {}_2\hat{F}_1(a,b;c;\mathbf{X}) - \log\ {}_2F_1(a,b;c;\mathbf{X})| < \infty, \tag{B.36}$$

supremum being over $c \geq c_0 > \frac{d-1}{2}$, $a,b \in R$, and $0_d \leq \mathbf{X} < (1-\epsilon)I_d$ for any $\epsilon \in (0,1)$. Authors provide in [89] some numerical examples to show how well this approximation works. We also follow the same way and show two plots in Fig. B.1, which demonstrate a very good numerical accuracy for several different setups. As mentioned, the hypergeometric function ${}_2F_1(a, b; c; \mathbf{X})$ of matrix argument is only a function of the eigenvalues of $\mathbf{X}$. So, we fix $\mathbf{X} = \tau I_d$ and draw the exact and approximate values of ${}_2F_1(a, b; c; \tau I_d)$ versus τ (note $0 < \tau < 1$ for convergence as mentioned in the definition of ${}_2F_1(a, b; c; \mathbf{X})$ in (3.12)) in Fig. B.1a for $d = 5$, $a = 3$, $b = 4$, and $c = 6$. Fig. B.1b shows the exact and approximate values of ${}_2F_1(a, b; c; \tau I_d)$ versus c for $d = 10$, $a = 30$, $b = 50$, and $\tau = 0.01$. The authors stated in [89] that when the integral representation is not valid, that is, when $c - a < \frac{d-1}{2}$, this Laplace approximation still gives good accuracy. We also see that approximation in Fig. B.1b is accurate for all range of c, even though the integral representation is not valid for $c < a + \frac{d-1}{2} = 34.5$. We also note that this approximation is more accurate in the smaller function values.