A Probabilistic Framework of Transfer Learning –

Theory and Application

by

Na Zou

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2015 by the
Graduate Supervisory Committee:

Jing Li, Chair
Mustafa Baydogan
Connie Borror
Douglas Montgomery
Teresa Wu

ARIZONA STATE UNIVERSITY

December 2015

ABSTRACT

Transfer learning refers to statistical machine learning methods that integrate the knowledge of one domain (source domain) and the data of another domain (target domain) in an appropriate way, in order to develop a model for the target domain that is better than a model using the data of the target domain alone. Transfer learning emerged because classic machine learning, when used to model different domains, has to take on one of two mechanical approaches. That is, it will either assume the data distributions of the different domains to be the same and thereby developing one model that fits all, or develop one model for each domain independently. Transfer learning, on the other hand, aims to mitigate the limitations of the two approaches by accounting for both the similarity and specificity of related domains. The objective of my dissertation research is to develop new transfer learning methods and demonstrate the utility of the methods in real-world applications. Specifically, in my methodological development, I focus on two different transfer learning scenarios: spatial transfer learning across different domains and temporal transfer learning along time in the same domain. Furthermore, I apply the proposed spatial transfer learning approach to modeling of degenerate biological systems. Degeneracy is a well-known characteristic, widely-existing in many biological systems, and contributes to the heterogeneity, complexity, and robustness of biological systems. In particular, I study the application of one degenerate biological system which is to use transcription factor (TF) binding sites to predict gene expression across multiple cell lines. Also, I apply the proposed temporal transfer learning approach to change detection of dynamic network data. Change detection is a classic research area in Statistical Process Control (SPC), but change detection in network data has been limited studied. I integrate

the temporal transfer learning method called the Network State Space Model (NSSM) and SPC and formulate the problem of change detection from dynamic networks into a covariance monitoring problem. I demonstrate the performance of the NSSM in change detection of dynamic social networks.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Jing Li, for her guidance, encouragement, and support during my dissertation research. She is an outstanding mentor and the most dedicated and diligent researcher I have ever known. Her responsibility and positive life attitude influence me profoundly and become my lifelong assets. I would like to thank my dissertation committee members, Dr. Mustafa Baydogan, Dr. Connie Borror, Dr. Douglas Montgomery and Dr. Teresa Wu, for their valuable interactions and feedback. They provide helpful suggestions and insightful comments for my dissertation.

Members of AMIIL lab inspired me a lot through discussions, seminars, and project collaborations, and I would like to thank the following people for their valuable suggestions: Fei Gao, Min Zhang, Nathan Gaw, Yinlin Fu, Kung Wang, Shuluo Ning, Hyunsoo Yoon, Can Cui, Bing Si, Congzhe Su, Xiaonan Liu. I have been working closely with many colleagues from ASU and outside ASU during my Ph.D. I want to thank Yun Zhu, Wei Wang, Ji Zhu, Igor Yakushev, Dmitry Titov.

Last but not the least, I would like to thank my family: my husband for supporting me spiritually, my parents and parents-in-law for unconditionally and attentively taking care of my son.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1: Introduction

1.1 Background

Transfer learning is a basic learning ability of human beings. It refers to the ability that people can intelligently apply knowledge learned in one domain to solve the problem in another domain faster or with better solutions. For example, people with prior experience on learning music instruments may be found to be a quicker learner of a new instrument compared with people who have no experience at all. Transfer learning, in statistical machine learning, has a similar nature. It refers to methods that integrate the knowledge of one domain (source domain) and the data of another domain (target domain) in an appropriate way, in order to develop a model for the target domain that is better than a model using the data of the target domain alone. Transfer learning emerged because classic machine learning, when used to model different domains, has to take on one of two mechanical approaches. That is, it will either assume the data distributions of the different domains to be the same and thereby developing one model that fits all, or develop one model for each domain independently. Transfer learning, on the other hand, aims to mitigate the obvious limitations of the two approaches by accounting for both the similarity and specificity of related domains. Transfer learning is especially advantageous when the sample size of data in the target domain is too limited to produce a reliable model, due to timing, availability, or/and cost. Next, I give a few examples in various areas in which transfer learning is desirable:

Manufacturing: Rapid updating of product generations is a common characteristic of various manufacturing industries. When a new generation of a product is invented, it needs to be quickly introduced to the market. The production data on the new

generation (target domain) could be very limited to allow adequate process modeling, control, and optimization. On the other hand, abundant data and knowledge may have been accumulated from past generations of the same product (source domain). Transfer learning can make use of these past data and knowledge to model the new generation better and faster.

Health care: An important problem in health care especially cancer medicine is to use various sources of clinical data, such as imaging, genetics, and demographics, for cancer prognostics. In longitudinal studies, a particular interest is to follow along a cohort of patients with a specific cancer to model the association between clinical data and disease outcomes (e.g., mortality, recurrence) at different stages of the disease advancement. Patient drop-off is common, leaving less data for use in modeling later stages of the disease (target domain). Transfer learning can play an important role here by integrating the limited data with knowledge from the earlier stages (source domain). A similar setting is in the study of different subtypes of a cancer. When a new subtype is discovered (target domain), the patient number is usually limited. Transfer learning can help establish a model for the new subtype timely and reliably by transferring knowledge of the known subtypes (source domain) to the modeling of the new subtype.

1.2 State of the Art

The existing transfer learning methods primarily fall into three categories: instance transfer, feature transfer, and parameter transfer.

Instance transfer is a major type of transfer learning methods (Liao, Xue, and Carin 2005; Dai et al. 2007; Wu and Dietterich 2004). The basic idea is to reuse some samples/instances in the source domain as auxiliary data for the target domain. For

example, Dai et al. (2007) proposed a boosting algorithm called TrAdaBoost to iteratively reweight samples in the source domains to identify samples that are helpful for modeling the target domain. Although intuitive, instance transfer may be questioned for its validity. For example, if the source and target domains are two subtypes of a cancer, using the data of some patients in one subtype to model another subtype implies that these patients are misdiagnosed, which is not a reasonable assumption.

Feature transfer and parameter transfer are two other types of relevant methods. Feature transfer aims to identify good feature representations shared by the source and target domains. In an earlier work by Caruana (1997), the features are shared hidden layers for the neural network models across the domains. More recently, Argyrious et al. (2008) and Evgeniou and Pontil (2007) proposed to map the original high-dimensional predictor space to a low-dimensional feature space and the mapping is shared across the domains. Nonlinear mapping was study by Jebara (2004) for Support Vector Machines (SVMs) and by Ruchert and Kramer (2008) who designed a kernel-based approach aiming at finding a suitable kernel for the target domain. Interpretability, e.g., physical meaning of the shared features, is an issue for feature transfer especially nonlinear approaches.

Parameter transfer assumes that the old and target domains share some model parameters. For example, Liu et al. (2009) adopt L21-norm regularization for linear models to encourage the same predictors to be selected across the domains. Regularized approaches for nonlinear models like SVMs were also studied (T. Evgeniou and Pontil 2004). In addition to regularization, Bayesian statistics provide a nice framework by assuming the same prior distribution for the model parameters across the domains, which

has been adopted for Gaussian process models (Lawrence and Platt 2004; Schwaighofer, Tresp, and Yu 2005; Bonilla, Chai, and Williams 2008).

Limitations of the existing transfer learning methods are:

First, most existing methods require that the raw data of the source are available at the time of modeling the target domain. This can be computationally intensive when the source domain has massive amounts of data. Also, this requires keeping the data of the source domain in their complete form all the time, which consumes lots of storage. In this sense, the existing methods should be more appropriately called multitask learning methods. An alternative approach is to discard the data of the source domain but only keep the knowledge of a much smaller size to be transferred, which has been less studied in the literature.

Second, when there are more than one source domains, most existing methods assume that the relationship between each source domain and the target domain is similar. This assumption may not hold well in practice. For examples, there may be some source domains whose model parameters are positively correlated with those of the target domain, and some with negative correlations, and others with no correlations. A desirable approach should be able to automatically learn the different domain relationships from data and adaptively decide how much is transferred from each source domain to the target domain. This idea has been explored by a few existing methods (Bakker and Heskes 2003; Xue et al. 2007; Jacob et al. 2009), in which the domains are grouped into clusters and domains within each cluster are assumed to have similar relationships with one another. However, clustering is only able to reveal domain relationships qualitatively

(i.e., whether or not two domains belong to the same cluster), but not quantitatively (i.e., to how much extent two domains are related).

Third, while showing empirically good performance than single-domain learning (i.e., machine learning methods that model each domain independently), the existing literature lacks theoretical investigation on when and why transfer learning is better than single-domain learning.

1.3 Summary of Original Contributions

The objective of my dissertation research is to develop new transfer learning methods that overcome the aforementioned limitations of the existing methods and demonstrate the utility of the methods in real-data applications. In my methodological development, I focus on two different transfer learning scenarios: **spatial transfer learning** across different domains and **temporal transfer learning** along time in the same domain.

The original contributions of my dissertation research are summarized as follows:

- **Spatial transfer learning**: I develop a transfer learning method for predictive modeling, which can flexibly incorporate the data or knowledge of the source domains, whichever available, to the modeling of the target domain. The method can automatically estimate the relationship between domains and adaptively decide how much information to transfer from each source domain to the target domain. The method is also able to model high-dimensional data. I develop a computationally efficient algorithm in model estimation. In addition, I perform theoretical analysis to answer several important questions, such as: What difference it will make by transferring the knowledge/models of

the source domains instead of the data? Is transfer learning always better than learning using the data of the target domain alone? What knowledge from source domains or what type of source domains is most helpful for transfer learning?

- **Temporal transfer learning**: I develop a transfer learning method called Network State Space Model (NSSM) for characterizing the temporal evolution of dynamic network data. NSSM produces a model for the current time frame by integrating the current data with a model from the past time frame. For tractable parameter estimation of the NSSM, I develop an Expectation Propagation (EP) algorithm to produce a multivariate Gaussian approximation for the observation equation of the NSSM, and further use an Expectation-Maximization (EM) framework integrated with a Bayesian optimal smoothing (BOS) algorithm to estimate the parameters.
- **Applications**: (1) I apply the proposed spatial transfer learning approach to modeling of degenerate biological systems. Degeneracy is a well-known characteristic, widely-existing in many biological systems, and contributes to the heterogeneity, complexity, and robustness of biological systems. In particular, I study the application of one degenerate biological system which is to use transcription factor (TF) binding sites to predict gene expression across multiple cell lines. The proposed transfer learning method shows better prediction accuracy compared with competing methods. The biological findings revealed by the proposed method are also consistent with the literature. (2) I apply the temporal transfer learning approach, i.e., the NSSM,

to change detection in social networks. Change detection is a classic research area in Statistical Process Control (SPC), but change detection in network data has been little studied. I integrate the NSSM and SPC and formulate the problem of change detection from dynamic networks into a covariance monitoring problem. I demonstrate the performance of the NSSM by extensive simulation experiments and a real-world application on Enron's email communication networks.

The organization of my dissertation is the following: Chapter 2 presents my research development in spatial transfer learning; Chapter 3 presents the application of the spatial transfer learning method in modeling of degenerate biological systems. Chapter 4 presents my research development in temporal transfer learning, i.e., the NSSM, and integration of the NSSM with SPC for change detection in social networks.

Chapter 2: Spatial Transfer Learning by a Sparse Matrix-normal Penalized Approach

2.1 Introduction

We adopt a Bayesian formulation and develop a unique prior for the model coefficients of the source domains and target domain. This prior has two hyper-parameters respectively encoding the covariance structure of predictor coefficients and characterizing the correlation structure of the domains. We propose an efficient algorithm that allows estimation of the hyper-parameters together with the model coefficients, so that the correlation structure between domains does not need to be specified a priori but can be learned from data. We perform theoretical analysis to reveal several important questions, such as: what difference it will make by transferring the knowledge/models of the source domains instead of the data? Is transfer learning always better than learning using the data of the target domain alone? What knowledge from target domains or what type of target domains is most helpful for transfer learning? Lastly, we perform simulation studies to compare the performance of the transfer learning method with single-domain learning.

2.2 Formulation

Let $X = (X_1, \dots, X_Q)$ denote Q predictors and Y denote the response. Assume that there are K related domains, where domains 1 to K-1 are source domains and domain K is a target domain. For each domain k, there is a model that links X to Y by coefficients w_k, $k = 1, \dots, K$. If $Y \in \mathbb{R}$, a common model is a linear regression, $Y = Xw_k + \varepsilon_k$. If $Y \in \{-1,1\}$, a classification model such as a logistic regression applies, e.g., $\log \frac{P(Y=1)}{P(Y=-1)} = Xw_k$. We propose the following prior for $W = (w_1, \dots, w_K)$:

$$\mathrm{p}(\mathbf{W}|\mathbf{\Omega},\mathbf{\Phi},\mathrm{b}) \propto \prod_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{Laplace}(\mathbf{w}_{\mathrm{k}};\mathrm{b}) \times \mathrm{MN}(\mathbf{W};0,\mathbf{\Omega},\mathbf{\Phi}). \tag{2.1}$$

This prior is formed based on the following considerations:

- $\mathrm{Laplace}(\mathrm{w_k};\mathrm{b})$ is a Laplace distribution for $\mathrm{w_k}$, i.e., $p(\mathbf{w}_k|b) = \left(\frac{1}{2b}\right)^Q exp\left(-\frac{1}{b}\sum_{q=1}^{Q}\left|w_{qk}\right|\right)$. w_{qk} is the q-th element of $\mathbf{w}_k$. Using a Laplace distribution in the prior is to facilitate "sparsity" in model estimation. The most well-known sparse model is probably the lasso model (Tibshirani 1994), which impose an L1 penalty on regression coefficients to shrink the estimates for many small be exactly zero, thus producing a sparse model. Tibshirani (1994) showed that the lasso estimate is equivalent to a Bayesian Maximum-A-Posteriori (MAP) estimate with a Laplace prior. Sparsity is an advantageous property for high-dimensional problems, which is the target setting of this dissertation.
- $MN(\mathbf{W};\mathbf{0},\mathbf{\Omega},\mathbf{\Phi})$ is a zero-mean matrix-variate normal distribution, whose probability density function is $p(\mathbf{W}|\mathbf{\Omega},\mathbf{\Phi}) = \frac{\exp\left(-\frac{1}{2}tr\left(\mathbf{\Phi}^{-1}\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T\right)\right)}{(2\pi)^{QK/2}|\mathbf{\Omega}|^{Q/2}|\mathbf{\Phi}|^{K/2}}$. $|\cdot|$and $tr(\cdot)$ denote the determinant and trace of a matrix, respectively. $\mathbf{\Omega} \in \mathbb{R}^{K\times K}$ and $\mathbf{\Phi} \in \mathbb{R}^{Q\times Q}$ are called column and row covariance matrices, respectively. It can be shown that $cov(\mathbf{w}^q) = \mathbf{\Phi}_{qq}\mathbf{\Omega}$. $\mathbf{w}^q$ is the q–th row of $\mathbf{W}$, which consists of regression coefficients for all the K domains corresponding to the q–th predictor. $\mathbf{\Phi}_{qq}$ is the q–th diagonal element of $\mathbf{\Phi}$. $cov(\cdot)$ denotes the covariance matrix of a vector. Therefore, $\mathbf{\Omega}$ encodes the prior knowledge about the correlation structure of the domains. Furthermore, it can be shown that $cov(\mathbf{w}_k) = \mathbf{\Omega}_{kk}\mathbf{\Phi}$. Therefore, $\mathbf{\Phi}$ encodes the prior knowledge about the correlation structure of the regression coefficients.

Next, we propose two modeling strategies depending on the availability of data. In Case I, data of the source domains 1 to K-1 is available. In Case II, data of the source domains is not available but only the knowledge/models. The latter case is more common especially in biology and medicine. At the time a new cell line or a new subtype of a disease is being studied, the researcher may only have access to the data of the target domain. Although he/she may gather abundant knowledge about existing cell lines or disease subtypes from the published works of other researchers, he/she can hardly access the data due to ownership or confidentiality.

Case I: Model the target domain using the data of all the domains

Let $\mathbf{y}_k$ and $\mathbf{X}_k$ denote the data for the response and predictors of the k-th domain $k = 1, \dots, K$. The likelihood for $\mathbf{y}_k$ given $\mathbf{X}_k$ and $\mathbf{w}_k$ is

$$p(\mathbf{y}_k|\mathbf{X}_k, \mathbf{w}_k) \sim N\left(\mathbf{y}_k; \mathbf{X}_k\mathbf{w}_k, \sigma^2\,\mathbf{I}_{n_k}\right) . \tag{2.2}$$

The posterior distribution of $\mathbf{W}$ based on the prior in (2.1) and likelihood in (2.2) is:

$$p(\mathbf{W}|\{\mathbf{y}_k, \mathbf{X}_k\}_{k=1}^K, \mathbf{\Omega}, \mathbf{\Phi}, b) \propto p(\mathbf{W}|\mathbf{\Omega}, \mathbf{\Phi}, b) \prod_{k=1}^K p(\mathbf{y}_k|\mathbf{X}_k, \mathbf{w}_k). \tag{2.3}$$

One way for estimating the regression coefficients of the target domain, $\mathbf{w}_K$, is to find a $\widehat{\mathbf{W}}$ that maximizes the posterior distribution of $\mathbf{W}$ in (2.3), i.e., $\widehat{\mathbf{W}}$ is a Bayesian MAP estimate for $\mathbf{W}$. This will naturally produce an estimate for $\mathbf{w}_K$, $\widehat{\mathbf{w}}_K$, as in $\widehat{\mathbf{W}} = (\widehat{\mathbf{w}}_1, \dots, \widehat{\mathbf{w}}_K)$, and estimates for the source domains, $\widehat{\mathbf{w}}_1, \dots, \widehat{\mathbf{w}}_{K-1}$, as a side product. Through some algebra, it can be derived that $\widehat{\mathbf{W}}$ can be obtained by solving the following optimization:

$$\widehat{\mathbf{W}}^{\mathrm{I}} = \operatorname{argmin}_{\mathbf{W}} \left\{\frac{1}{2\sigma^2}\sum_{k=1}^K \|\mathbf{y}_k - \mathbf{X}_k\mathbf{w}_k\|_2^2 + \frac{1}{b}\|\mathbf{W}\|_1 + \frac{1}{2}\left(Q log|\mathbf{\Omega}| + K log|\mathbf{\Phi}| + tr(\mathbf{\Phi}^{-1}\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T)\right)\right\}. \tag{2.4}$$

where $\|\cdot\|_2$ and $\|\cdot\|_1$ denote the L2 and L1 norms, respectively. The superscript "I" is used to differentiate this estimate from the one that will be presented in Case II.

(2.4) assumes that $\mathbf{W}$ is the only parameter to be estimated whereas σ^2, b, $\mathbf{\Omega}$, and $\mathbf{\Phi}$ are known. This assumption may be too strict. To relax this assumption, we propose the following approach: Let $\lambda_1 = 2\sigma^2/b$ and $\lambda_2 = \sigma^2$. Then, (2.4) is equivalent to (2.5):

$$\widehat{\mathbf{W}}^{\mathrm{I}} = \underset{\mathrm{W}}{\operatorname{argmin}} \left\{ \sum_{\mathrm{k}=1}^{\mathrm{K}} \|\mathbf{y}_{\mathrm{k}} - \mathbf{X}_{\mathrm{k}}\mathbf{w}_{\mathrm{k}}\|_2^2 + \lambda_1 \|\mathbf{W}\|_1 + \lambda_2 \left(\mathrm{Q}\log|\mathbf{\Omega}| + \mathrm{K}\log|\mathbf{\Phi}| + \mathrm{tr}(\mathbf{\Phi}^{-1}\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^{\mathrm{T}}) \right) \right\}. \quad (2.5)$$

Here, $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ serve as regularization parameters to control the sparsity of $\widehat{\mathbf{W}}^{\mathrm{I}}$ and the amount of prior knowledge used for estimating $\mathbf{W}$, respectively. λ_1 and λ_2 can be selected by a grid search according to some model selection criterion such as BIC and cross validation. This strategy for "estimating" σ^2 and b enjoys computational simplicity and was also adopted by other papers (Tibshirani 1994; Liu, Ji, and Ye 2009; Genkin, Lewis, and Madigan 2007). Furthermore, hyper-parameters $\mathbf{\Phi}$ and $\mathbf{\Omega}$ are matrices of potentially high dimensionality, the specification of which is more involved and will be discussed in detail in Section 2.4. For now, we assume that $\mathbf{\Phi}$ and $\mathbf{\Omega}$ are known.

Note that the MAP estimate for $\mathbf{W}$ is a point estimate, which would not allow statistical inference for characterizing the uncertainty in the estimation. A full Bayesian approach would be more preferable in this regard. However, the difficulty is that the posterior distribution for $\mathbf{W}$ in (2.3) does not have a parametric form. Although computational algorithms like Markov Chain Monte Carlo (MCMC) may be used, they are known to be computationally intensive. Therefore, we choose to use MAP in this chapter to gain efficiency (Smith 1998) and leave the full Bayesian approach for future investigation.

Case II: Model the target domain using data of the target domain and knowledge/models of the source domains

To develop a model for this case, we first re-organize the terms in (2.5) to separate the terms involving source domains from those involving only the target domain. Denote the objective function in (2.5) by $f(\mathbf{W})$. Let $\widetilde{\mathbf{W}} = (\mathbf{w}_1, \dots, \mathbf{w}_{K-1})$, so $\mathbf{W} = (\widetilde{\mathbf{W}}, \mathbf{w}_K)$. Also let $\boldsymbol{\Omega} = \begin{bmatrix} \widetilde{\boldsymbol{\Omega}} & \boldsymbol{\varpi}_K \\ \boldsymbol{\varpi}_K^T & \varsigma_K \end{bmatrix}$. Then, it can be shown that (please see details in Appendix 2.I):

$$f(\mathbf{W}) = f(\widetilde{\mathbf{W}}) + g(\mathbf{w}_K|\widetilde{\mathbf{W}}). \tag{2.6}$$

$\mathrm{f}(\widetilde{\mathrm{W}})$ takes the same form as $\mathrm{f(W)}$ but for the $\mathrm{K}-1$ source domains, i.e.,

$$f(\widetilde{\mathbf{W}}) = \sum_{k=1}^{K-1} \|\mathbf{y}_k - \mathbf{X}_k \mathbf{w}_k\|_2^2 + \lambda_1 \|\widetilde{\mathrm{W}}\|_1 + \lambda_2 \left(Q log|\widetilde{\boldsymbol{\Omega}}| + (K-1) log|\boldsymbol{\Phi}| + tr\left(\boldsymbol{\Phi}^{-1} \widetilde{\mathbf{W}} \widetilde{\boldsymbol{\Omega}}^{-1} \widetilde{\mathbf{W}}^T\right)\right). \tag{2.7}$$

and

$$g(\mathbf{w}_K|\widetilde{\mathbf{W}}) = \|\mathbf{y}_K - \mathbf{X}_K \mathbf{w}_K\|_2^2 + \lambda_1 \|\mathbf{w}_K\|_1 + \lambda_2 \left(log|\boldsymbol{\Sigma}_K| + (\mathbf{w}_K - \boldsymbol{\mu}_K)^T \boldsymbol{\Sigma}_K^{-1} (\mathbf{w}_K - \boldsymbol{\mu}_K)\right). \tag{2.8}$$

where

$$\boldsymbol{\mu}_K = \widetilde{\mathbf{W}} \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K. \tag{2.9}$$

and

$$\boldsymbol{\Sigma}_K = \left(\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K\right) \boldsymbol{\Phi}. \tag{2.10}$$

When data from the source domains is not available but only the knowledge/model in the form of $\widetilde{\mathbf{W}} = \widetilde{\mathbf{W}}^*$, the $f(\widetilde{\mathbf{W}}^*)$ in (2.6) becomes a constant. Therefore, minimizing $f(\mathbf{W})$ becomes minimizing $g(\mathbf{w}_K|\widetilde{\mathbf{W}}^*)$, i.e.,

$$\widehat{\mathbf{w}}_{\mathrm{K}}^{\mathrm{II}} = \underset{\mathbf{w}_{\mathrm{K}}}{\operatorname{argmin}} \quad \mathrm{g}(\mathbf{w}_{\mathrm{K}}|\widetilde{\mathbf{W}}^*)$$

$$= \text{argmin}_{\mathbf{w}_K} \ \|\mathbf{y}_K - \mathbf{X}_K\mathbf{w}_K\|_2^2 + \lambda_1\|\mathbf{w}_K\|_1 + \lambda_2\big(log|\mathbf{\Sigma}_K| + (\mathbf{w}_K - \boldsymbol{\mu}_K)^T\mathbf{\Sigma}_K^{-1}(\mathbf{w}_K - \boldsymbol{\mu}_K)\big). \quad (2.11)$$

with $\boldsymbol{\mu}_K = \widetilde{\mathbf{W}}^*\widetilde{\mathbf{\Omega}}^{-1}\boldsymbol{\varpi}_K$ and $\mathbf{\Sigma}_K$ given in (2.10).

Finally in this section, we would like to assess the difference between the estimates in Case I and Case II, i.e., $\widehat{\mathbf{w}}_K^{\mathrm{I}}$ as in $\widehat{\mathbf{W}}^{\mathrm{I}} = (\widehat{\mathbf{w}}_1^{\mathrm{I}}, \dots, \widehat{\mathbf{w}}_K^{\mathrm{I}})$ and $\widehat{\mathbf{w}}_K^{\mathrm{II}}$. Theorem 2.1 shows that the estimate in Case II is no better than Case I in terms of minimizing the objective function in the estimation. Case II is only as good as Case I when the knowledge/model of the source domains can be provided in its optimal form. The intuitive explanation about this finding is that since Case II utilizes the knowledge of the source domains, which may contain uncertainty or noise, it is only sub-optimal compared with using the data of the source domains directly (i.e., Case I). Please see Appendix A.II for a proof of Theorem 2.1.

Theorem 2.1: $f\left(\left(\widetilde{\mathbf{W}}^*, \widehat{\mathbf{w}}_K^{\mathrm{II}}\right)\right) \geq f\left(\widehat{\mathbf{W}}^{\mathrm{I}}\right)$. When $\widetilde{\mathbf{W}}^* = (\widehat{\mathbf{w}}_1^{\mathrm{I}}, \dots, \widehat{\mathbf{w}}_{K-1}^{\mathrm{I}})$, $\widehat{\mathbf{w}}_K^{\mathrm{II}} = \widehat{\mathbf{w}}_K^{\mathrm{I}}$ and $f\left(\left(\widetilde{\mathbf{W}}^*, \widehat{\mathbf{w}}_K^{\mathrm{II}}\right)\right) = f\left(\widehat{\mathbf{W}}^{\mathrm{I}}\right)$.

2.3 Theoretical Properties

This section aims to perform theoretical analysis to address the following questions: Is transfer learning always better than single-domain learning, i.e., learning using only the data of the target domain but neither the data nor the knowledge of the source domains (Theorem 2.2)? What knowledge from source domains or what type of source domains is most helpful for learning of the target domain (Theorems 2.3)?

Let (2.12) and (2.13) be the transfer learning and single-domain learning formulations targeted in this section, respectively. $\lambda \geq 0$. When $\lambda = 0$, (2.12) becomes (2.13).

$$\widehat{\mathbf{w}}_K = \text{argmin}_{\mathbf{w}_K} \quad \|\mathbf{y}_K - \mathbf{X}_K\mathbf{w}_K\|_2^2 + \lambda(\mathbf{w}_K - \boldsymbol{\mu}_K)^T(\mathbf{w}_K - \boldsymbol{\mu}_K) \tag{2.12}$$

$$\breve{\mathbf{w}}_K = \text{argmin}_{\mathbf{w}_K} \quad \|\mathbf{y}_K - \mathbf{X}_K\mathbf{w}_K\|_2^2 \tag{2.13}$$

Comparing (2.12) with (2.11) in the previous section, it can be seen that (2.12) is obtained from (2.11) by dropping the L1 norm, $\|\mathrm{w_K}\|_1$, and making $\Phi = \mathrm{I}$ and $\lambda = \frac{\lambda_2}{\varsigma_K - \varpi_K^T \widetilde{\Omega}^{-1} \varpi_K}$. This is to single out transfer learning from the sparsity and the covariance structure of regression coefficients in (2.11), so that the discussion in this section will be focused on transfer learning. Let $\mathrm{MSE}(\cdot)$ denote the Mean Square Error (MSE) of an estimator. It is known that the MSE is the sum of the variance and squared bias of an estimator, and is a commonly used criterion for comparing/choosing estimators.

Theorem 2.2: There always exists a $\boldsymbol{\lambda} > \mathbf{0}$ such that $\boldsymbol{MSE}(\widehat{\mathbf{w}}_K) < \boldsymbol{MSE}(\breve{\mathbf{w}}_K)$.

Please see a proof of this Theorem in Appendix 2.III. Theorem 2.2 provides theoretical assurance that the model coefficients of the target domain, $\mathbf{w}_K$, can be better estimated by transfer learning than single-domain learning in the sense of a smaller MSE. Next, we would like to investigate what type of knowledge from source domains or what type of source domains helps learning of the target domain better. Because knowledge from source domains is represented by $\boldsymbol{\mu}_K$ in (2.9), the question becomes what property of $\boldsymbol{\mu}_K$ leads to a better transfer learning. Definition 1 defines a distance measure between the knowledge from source domains, $\boldsymbol{\mu}_K$, and the target domain, $\mathbf{w}_K$, called the transfer learning distance. Theorem 2.3 further proves that the knowledge for source domains

that has a smaller transfer learning distance to the target domain will help achieve a smaller MSE in modeling the target domain.

Definition 2.1 (*transfer learning distance*): Define a transfer learning distance to be $\boldsymbol{d}(\boldsymbol{\mu}_K;\boldsymbol{\lambda}) \triangleq (\mathbf{w}_K - \boldsymbol{\mu}_K)^T\mathbf{B}^T\mathbf{B}(\mathbf{w}_K - \boldsymbol{\mu}_K)$, where $\mathbf{B} = \left(\mathbf{X}_K^T\mathbf{X}_K + \boldsymbol{\lambda}\mathbf{I}\right)^{-1}$.

The geometric interpretation of this distance measure is the following: Let Λ be a diagonal matrix of eigenvalues $\gamma_1, \dots, \gamma_Q$ for $X_K^T X_K$ and P be a matrix consisting of corresponding eigenvectors, i.e., ., $X_K^T X_K = P^T \Lambda P$. Furthermore, let $\alpha \triangleq P(\mu_K - w_K)$. The elements of α, $\alpha_1, \dots, \alpha_Q$, are indeed projections of $\mu_K - w_K$ onto the principal component axes of the data. Furthermore, it can be derived that the transfer learning distance is $d(\mu_K;\lambda) = \sum_{i=1}^{Q} \frac{\alpha_i^2}{(\gamma_i+\lambda)^2}$. Provided that the predictors are mean-centered, this expression of $d(\mu_K;\lambda)$ implies that the transfer learning distance is a scaled Euclidean distance in the principal component space.

Furthermore, suppose that there are two sets of knowledge from source domains to be compared, i.e., $\boldsymbol{\mu}_K^{(1)}$ and $\boldsymbol{\mu}_K^{(2)}$. Let $MSE\left(\widehat{\mathbf{w}}_K^{(i)};\lambda\right)$ be the MSE of the estimator for $\widehat{\mathbf{w}}_K$ using (2.9) with $\boldsymbol{\mu}_K = \boldsymbol{\mu}_K^{(i)}$. Let $\min_\lambda MSE\left(\widehat{\mathbf{w}}_K^{(i)}\right)$ denote the smallest MSE over all possible values of λ. $i = 1,2$.

Theorem 2.3: If $\boldsymbol{d}\left(\boldsymbol{\mu}_K^{(1)};\boldsymbol{\lambda}\right) \le \boldsymbol{d}\left(\boldsymbol{\mu}_K^{(2)};\boldsymbol{\lambda}\right)$ for $\forall\boldsymbol{\lambda} > \mathbf{0}$, then $\mathbf{min}_{\boldsymbol{\lambda}}\, \boldsymbol{MSE}\left(\widehat{\mathbf{w}}_K^{(1)}\right) \le \mathbf{min}_{\boldsymbol{\lambda}}\, \boldsymbol{MSE}\left(\widehat{\mathbf{w}}_K^{(2)}\right)$.

Please see a proof of this Theorem in Appendix 2.IV. For better illustration, we show the comparison of MSEs between five sets of knowledge from source domains in Figure 1. This is a simple example which consists of only one predictor. Therefore, $\mathbf{w}_K$

and $\boldsymbol{\mu}_K$ are scalars, w_K and μ_K. Assume that $w_K = 3$. $\mu_K^{(1)}$ through $\mu_K^{(5)}$ are 1.3, 1.6, 1.9, 2.2, 2.5, respectively, i.e., they are more and more close to the target domain in transfer learning distance. Figure 1 plots the MSEs of transfer learning using each of the five sets of knowledge. The observations are: (i) For each curve, there exists a $\lambda > 0$ whose corresponding MSE is smaller than the MSE of single-domain learning (i.e., the intercept on the vertical axis). This demonstrates Theorem 2.2. (ii) The smaller the transfer learning distance, the smaller the minimum MSE. This demonstrates Theorem 2.3.

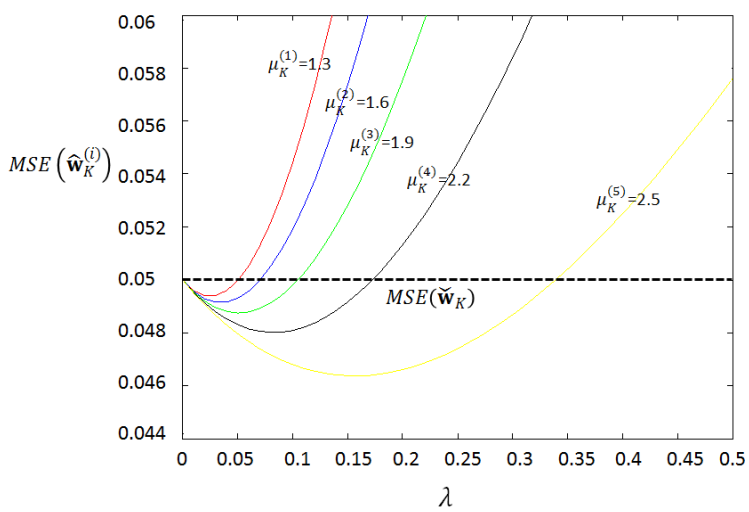


Figure 1. Transfer learning MSEs of five sets of knowledge from source domains, $\mathrm{MSE}\left(\hat{\mathrm{w}}_{\mathrm{K}}^{(\mathrm{i})}\right)$, $\mathrm{i} = 1, \ldots, 5$, and single-domain MSE, $\mathrm{MSE}\left(\check{\mathrm{w}}_{\mathrm{K}}\right)$ with true $\mathrm{w}_{\mathrm{K}} = 3$

Finally, we would like to discuss some practical implication of the Theorems. Theorem 2.2 needs little assumption to hold. However, this does not imply that transfer learning always gives better results than single-domain learning in practice. This is because in practice, λ is selected by a grid search according to some model selection criterion such as BIC and cross-validation. The λ that makes the MSE of the transfer learning estimator smaller than single-domain learning may be missed in this practical

search. Furthermore, as indicated by Theorem 2.3 and Figure 1, this risk is higher when the knowledge from source domain is farther from the target domain in transfer learning distance. For example, in Figure 1, when the knowledge is far away from the target domain, e.g., the top red curve, the range of λ within which the curve falls below the MSE of single-domain learning, $MSE\left(\breve{\mathbf{w}}_K\right)$, is very small. This small range of λ may be easily missed in a practical grid search, thus resulting in a transfer learning approach that has worse performance than single-domain learning.

2.4 Joint Estimation of Hyper-parameters and Parameters

$\mathbf{\Phi}$ is a hyper-parameter that encodes the correlation structure of the regression coefficients. For a specific domain, prior knowledge usually exists to specify $\mathbf{\Phi}$. This will be illustrated in Chapter 3. $\mathbf{\Omega}$ is another hyper-parameter that encodes the prior knowledge about the correlation structure between domains, which is difficult to specify precisely. Therefore, we propose an algorithm in this section to estimate both hyper-parameter $\mathbf{\Omega}$ and parameter $\mathbf{W}$ together. This will change (2.4) to (2.14):

Case III:

$$\left(\widehat{\mathbf{W}}^{\mathrm{III}}, \widehat{\mathbf{\Omega}}^{\mathrm{III}}\right) = \underset{\mathrm{W},\Omega}{\operatorname{argmin}} \left\{\sum_{\mathrm{k}=1}^{\mathrm{K}} \|\mathbf{y}_{\mathrm{k}} - \mathbf{X}_{\mathrm{k}}\mathbf{w}_{\mathrm{k}}\|_2^2 + \lambda_1\|\mathbf{W}\|_1 + \lambda_2\left(\mathrm{Q}\log|\mathbf{\Omega}| + \mathrm{tr}(\mathbf{\Phi}^{-1}\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^{\mathrm{T}})\right)\right\}. \quad (2.14)$$

(2.14) is the same as (2.4) except for treating $\mathbf{\Omega}$ as unknown.

Next, we will discuss an algorithm for solving (2.14). (2.14) is not a convex optimization with respect to all unknown parameters. However, given $\mathbf{\Omega}$, it becomes a convex optimization with respect to $\mathbf{W}$, which can be solved efficiently. Furthermore, given $\mathbf{W}$, the optimization problem with respect to Ω can be solved analytically, i.e.,

$$\widehat{\mathbf{\Omega}} = \frac{\mathbf{W}^{\mathrm{T}}\mathbf{\Phi}^{-1}\mathbf{W}}{\mathrm{Q}}. \quad (2.15)$$

Therefore, we propose an iterative algorithm that alternates between two sub-optimizations: solving $\mathbf{W}$ with $\mathbf{\Omega}$ fixed at their estimates in the previous iteration, and solving $\mathbf{\Omega}$ with $\mathbf{W}$ fixed at its estimate just obtained. Because each sub-optimization decreases the objective function, this iterative algorithm is guaranteed to converge to a local optimal solution. Note that joint estimation of parameters and hyper-parameters has also been adopted by other researchers (Idier 2013; Zhang and Yeung 2010).

A similar case to Case II (2.11) takes the form of (2.16):

<u>*Case IV:*</u>

$$(\widehat{\mathbf{w}}_K^{\mathrm{IV}}, \hat{\varsigma}_K^{\mathrm{IV}}, \widehat{\boldsymbol{\varpi}}_K^{\mathrm{IV}}) = \underset{\mathbf{w}_K, \varsigma_K, \boldsymbol{\varpi}_K}{\operatorname{argmin}} \left\{ \begin{array}{c} \|\mathbf{y}_K - \mathbf{X}_K \mathbf{w}_K\|_2^2 + \lambda_1 \|\mathbf{w}_K\|_1 \ + \\ \lambda_2 \left(log\left(\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\mathbf{\Omega}}^{-1} \boldsymbol{\varpi}_K\right) + \frac{1}{\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\mathbf{\Omega}}^{-1} \boldsymbol{\varpi}_K} (\mathbf{w}_K - \boldsymbol{\mu}_K)^T \mathbf{\Phi}^{-1} (\mathbf{w}_K - \boldsymbol{\mu}_K) \right) \end{array} \right\}. \quad (2.16)$$

(2.16) can be solved by an iterative algorithm that alternates between solving $\mathbf{w}_K$ with ς_K and $\boldsymbol{\varpi}_K$ fixed – a convex optimization, and solving ς_K and $\boldsymbol{\varpi}_K$ with $\mathbf{w}_K$ fixed analytically using (2.17):

$$\widehat{\boldsymbol{\varpi}}_K = \widetilde{\mathbf{W}}^T \mathbf{\Phi}^{-1} \mathbf{w}_K / Q, \hat{\varsigma}_K = \mathbf{w}_K{}^T \mathbf{\Phi}^{-1} \mathbf{w}_K / Q. \quad (2.17)$$

The derivation for (2.17) is in Appendix 2.V.

Finally in Section 2.3, we present the algorithms for solving the proposed transfer learning formulations in Case III and Case IV in Figures 2 and 3, respectively. Note that the algorithms also work for classification problems which replace the square-error loss in Case III and Case IV, $\|\mathbf{y}_k - \mathbf{X}_k \mathbf{w}_k\|_2^2$, with a logistic loss, $\sum_{i=1}^{n_K} log(1 + exp(-y_{iK} \mathbf{x}_{iK} \mathbf{w}_K))$. Because this loss function is also convex with respect to $\mathbf{w}_K$, the convex optimization solver in step 2.2 of Figures 2 and 3 naturally applies. Step 2.1 does not involve the loss function so it needs no change.

Input: data of the source domains and target domain, $\{\mathbf{y}_k, \mathbf{X}_k\}_{k=1}^{K}$; regularization parameters, λ_1 and λ_2.

Step 1: Obtain the covariance matrix $\mathbf{\Phi}$ of predictor coefficients.

Step 2: Alternate between 2.1 and 2.2 till convergence. Initialize $\mathbf{W}$ by fitting a lasso to each domain separately.

2.1: Solve $\mathbf{\Omega}$ by (2.15).

2.2: Solve $\mathbf{W}$ in (2.14) using a convex optimization solver like the accelerated gradient method (Liu, Ji, and Ye 2009).

Output: models of the source domains and target domain, $\widehat{\mathbf{W}}^{\mathrm{III}}$; relationship between the target domain and the source domains, $\widehat{\mathbf{\Omega}}^{\mathrm{III}}$

Figure 2. An algorithm for solving the transfer learning formulation in Case III

Input: knowledge about source domains $1, \ldots, K-1$, $\widetilde{\mathbf{W}}^{*}$; data for a target domain K, $\mathbf{X}_K$ and $\mathbf{y}_K$; regularization parameters, λ_1 and λ_2.

Step 1: Obtain the predictor covariance matrix $\mathbf{\Phi}$ of predictor coefficients.

Step 2: Alternate between 2.1 and 2.2 till convergence. Initialize $\mathbf{w}_K$ by fitting a lasso to data $\mathbf{X}_K$ and $\mathbf{y}_K$.

2.1: Solve ς_K and $\boldsymbol{\varpi}_K$ by (2.17).

2.2: Let $\boldsymbol{\mu}_K = \widetilde{\mathbf{W}}^{*}\widetilde{\mathbf{\Omega}}^{-1}\boldsymbol{\varpi}_K$ and solve $\mathbf{w}_K$ in (2.16) by a convex optimization solver like the accelerated gradient method (Liu, Ji, and Ye 2009).

Output: model of the target domain, $\widehat{\mathbf{w}}_K^{\mathrm{IV}}$; covariances between the target domain and the source domains, $\widehat{\boldsymbol{\varpi}}_K^{\mathrm{IV}}$; variance of the target domain, $\hat{\varsigma}_K^{\mathrm{IV}}$.

Figure 3. An algorithm for solving the transfer learning formulation in Case IV

2.5 Prediction

Given a new observation in the target domain, $\mathbf{x}_K^{*}$, we can predict its response variable by $\hat{y}_K^{*} = \mathbf{x}_K^{*T}\widehat{\mathbf{w}}_K$. $\widehat{\mathbf{w}}_K$ can be the $\widehat{\mathbf{w}}_K^{\mathrm{III}}$ in Case III or the $\widehat{\mathbf{w}}_K^{\mathrm{IV}}$ in Case IV, obtained from training data. Because the proposed transfer learning method only produces a point estimator for $\mathbf{w}_K$, statistical inference on $\mathbf{w}_K$ and the prediction has to be performed using resampling approaches such as bootstrap. This is a similar situation to lasso, for

which bootstrap-based statistical inference on the model coefficients has been studied by a number of papers (Knight and Fu 2000; Chatterjee and Lahiri 2010). Following the similar idea, we propose a residual bootstrap procedure to compute the prediction interval, which includes nine steps shown in Figure 4.

1) Under Case III or IV model, select $\hat{\lambda}_1$ and $\hat{\lambda}_2$ by cross-validation. Estimate $\widehat{w}_K$ from training data under $\hat{\lambda}_1$ and $\hat{\lambda}_2$.
2) Compute the residual for each of the n training data points, i.e., $r_{K,i} = y_{K,i} - x_{K,i}^T \widehat{w}_K$, $i = 1, \dots, n$
3) Center the residuals, i.e., $e_{K,i} = r_{K,i} - \bar{r}_K$, where $\bar{r}_K = \frac{1}{n}\sum_{i=1}^{n} r_{K,i}$. Pool the centered residuals together as $\{e_{K,1}, \dots, e_{K,n}\}$.
4) Draw a sample of size n from $\{e_{K,1}, \dots, e_{K,n}\}$ with replacement, i.e., $\{\tilde{e}_{K,1}, \dots, \tilde{e}_{K,n}\}$. Calculate the bootstrapped version of the response variable in the training data as $\tilde{y}_{K,i} = x_{K,i}^T \widehat{w}_K + \tilde{e}_{K,i}$, $i = 1, \dots, n$.
5) Using the bootstrap dataset $\{(\tilde{y}_{K,i}, x_{K,i}^T): i = 1, \dots, n\}$ and the $\hat{\lambda}_1$ and $\hat{\lambda}_2$ in 1), estimate the bootstrap version of w_K using Case III or IV model, $\widetilde{w}_K$.
6) Use the bootstrap estimator to predict the new observation, i.e., $\hat{\tilde{y}}_K^* = x_K^{*T} \widetilde{w}_K$.
7) Draw a single sample from $\{e_{K,1}, \dots, e_{K,n}\}$, $\tilde{e}_K$, and let $\tilde{y}_K^* = x_K^{*T} \widehat{w}_K + \tilde{e}_K$.
8) Compute the bootstrap prediction error as $\hat{\tilde{y}}_K^* - \tilde{y}_K^*$.
9) Repeat 4)-8) B times and obtain the empirical distribution of the bootstrap prediction error, G. Let $G^{-1}\left(1 - \frac{\alpha}{2}\right)$ and $G^{-1}\left(\frac{\alpha}{2}\right)$ be the $\left(1 - \frac{\alpha}{2}\right) \times 100\%$ and $\frac{\alpha}{2} \times 100\%$ percentiles for G, respectively. Then, the bootstrap prediction interval is $\left[x_K^{*T} \widehat{w}_K - G^{-1}\left(1 - \frac{\alpha}{2}\right), x_K^{*T} \widehat{w}_K - G^{-1}\left(\frac{\alpha}{2}\right)\right]$.

Figure 4. A residual bootstrap procedure to compute prediction interval

2.6 Simulation Study

A real-data application will be presented in the next chapter. In this section, we aim to assess the proposed method in aspects that cannot be assessed using real data. Prediction accuracy can be assessed using real data by cross validation. However, because the ground truth is unknown in real data analysis, variable selection accuracy cannot be assessed, including False Positive Rate (FPR) and False Negative Rate (FNR).

Here, FPR is the proportion of truly zero regression coefficients that are misidentified to be non-zero by the model. FNR is the proportion of truly non-zero regression coefficients that are misidentified to be zero by the model. Therefore, the simulation studies in this section focus on evaluating these error rates of the proposed method against competing methods. In particular, we use Area Under the Curve (AUC), which is an integrated measure for FPR and FNR. Another advantage of AUC is that it does not require a selection of the regularization parameters, but reflects the overall performance of the model over all possible values of the regularization parameters. The theoretically best AUC is one; the larger the AUC for a model, the better the performance.

In this section, we compare the proposed method in Case III with $\mathbf{\Phi} = \mathbf{I}$ with single domain learning. Note that although we only present the results for Case III due to space limit, similar results have been obtained for Case IV.

Consider three domains and the model, $Y_k = \sum_{q=1}^{Q} w_{qk} X_{qk} + \varepsilon_k$, $k = 1,2,3$. In each domain, there are 50 predictors, i.e., $Q = 50$. Domains 1 and 2 are highly correlated with each other but little correlated with domain 3. To achieve this, we set the coefficients of the first five predictors in domains 1 and 2 to be non-zero, i.e., $w_{qk} \neq 0, q = 1, \ldots, 5; k = 1,2$. To make the two domains non-identical, we randomly select one different predictor from X_6 to X_{50} in each domain to have a non-zero coefficient. For domain 3, we set the coefficients $w_{q3} \neq 0, q = 5, \ldots, 10$. Therefore, in each domain, there are six predictors with non-zero coefficients and all 44 others with zero coefficients. The value of each non-zero coefficient is randomly generated from a normal distribution $N(5,1)$. After generating the coefficients, we check the correlation between the three domains using their respective coefficients. The correlations are 0.81 between domains 1

and 2, and 0.05(0.06) between domain 1(2) and 3, which are good to serve our purpose. Next, we generate samples for the 50 predictors from a multivariate normal distribution with zero mean and covariance matrix $\Sigma_{ij} = 0.5^{|i-j|}, i, j = 1, \ldots, 50$. To focus on small-sample-size scenarios, 50 samples of the predictors are generated for each domain. The response variable of each sample is generated by the model $Y_k = \sum_{q=1}^{Q} w_{qk} X_{qk} + \varepsilon_k$, where ε_k is generated from $N(0,15)$.

The proposed method of transfer learning is compared with single-domain learning, i.e., a lasso model applied to each domain separately, on the simulation dataset. The process is repeated for 50 times; the average and standard derivation of the 50 AUCs for each method are reported in Table 1. It can be seen that transfer learning has a better average AUC performance than single-domain learning. It is also more stable by having a smaller standard deviation. Furthermore, having a little-correlated domain, i.e., domain 3, does not hurt the performance of transfer learning in domains 1 and 2. This is because the proposed transfer learning method can estimate the correlation structure of the domains from data, and therefor can adaptively decide how much information to transfer from one domain to another.

Table 1. AUC performances of transfer learning and single-domain learning

		Domain 1	Domain 2	Domain 3
Proposed transfer Learning	Average	0.943447	0.955568	0.949735
	Standard deviation	0.042415	0.037728	0.041266
Single-domain learning	Average	0.871061	0.868485	0.889432
	Standard deviation	0.099003	0.101911	0.084644

Chapter 3: Application of Spatial Transfer Learning in Predictive Modeling of Degenerate Biological Systems

3.1 Introduction

We apply the proposed method in Chapter 2 to an application on predictive modeling of degenerate biological systems. Degeneracy exists in many biological systems and processes, and is referred to as the phenomenon that structurally different elements perform the same/similar function or yield the same/similar output. Degeneracy contributes to the heterogeneity, complexity, and robustness of biological systems. Specifically, we propose to use a graph to represent the qualitative knowledge about degeneracy with uncertainty, and set the corresponding hyper-parameter to be the Laplacian matrix of the graph. We theoretically prove that this has an effect of pushing the coefficients of degenerate elements to be similar, thus nicely reflecting the nature of degenerate elements that they perform the similar function. Then, we apply the proposed method integrating degeneracy and transfer learning to a real-world application of using TF binding sites to predict gene expression across multiple cell lines. The proposed method shows better prediction accuracy compared with competing methods. The biological findings revealed by the proposed model are also consistent with the literature.

3.2 Need of Transfer Learning in Modeling Biological Systems

An essential problem in biological system informatics is to build a predictive model with high-dimensional predictors. This can be a challenging problem for a target domain in which the data is scarce due to resource limitation or timing of the modeling. Often times, there may be some source domains related to but not exactly the same as the target domain, in which abundant knowledge have existed. This makes transfer learning highly desirable. Next, we give three examples:

(i) Modeling the predictive relationship between transcription factors (TFs) and gene expression is of persistent interest in system biology. TFs are proteins that bind to the upstream region of a gene and regulate the expression level of the gene. Knowledge of TFs-expression relationship may have existed for a number of known cell lines. To model a new cell line, it is advantageous to adopt transfer learning to make good use of the existing knowledge of the known cell lines, because the experimental data for the new cell line may be limited.

(ii) In cancer genomics, a prominent interest is to use gene expression to predict disease prognosis. Knowledge may have existed for several known subtypes of a cancer. When a new subtype is discovered, the patient number is usually limited. Transfer learning can help establish a model for the new subtype timely and reliably by transferring knowledge of the known subtypes to the modeling of the new subtype.

(iii) Biomedical imaging, such as Positron Emission Tomography (PET) and Magnetic Resonance Imaging (MRI), has been used to predict cognitive performance. In longitudinal studies, a particular interest is to follow along a cohort of patients with a brain disease such as the Alzheimer's disease to identify the imaging-cognition associations at different stages of the disease advancement. Patient drop-off is common, leaving less data for use in modeling later stages of the disease. Transfer learning can play an important role here by integrating the limited data with knowledge from the earlier stages.

3.3 Degenerate Biological Systems

This chapter focuses on transfer learning in *degenerate* biological systems. Degeneracy is a well-known characteristic of biological systems. In the seminal paper by

Edelman and Gally (2001), degeneracy was referred to as the phenomenon that structurally different elements perform the same/similar function or yield the same/similar output. The paper also provided ample evidence to show that degeneracy exists in many biological systems and processes. Degeneracy contributes to the heterogeneity, complexity, and robustness of biological systems.

A closely related concept to degeneracy is redundancy, which may be more familiar to the engineering society. Degeneracy is different from redundancy in three major aspects:

(a) Degeneracy is a characteristic for structurally different elements, whereas redundancy is one for structurally identical elements. In fact, although prevalent in engineering systems, true redundancy hardly exists in biological systems due to the rare presence of identical elements.

(b) Degenerate elements work in a stochastic fashion, whereas redundant elements work according to deterministic design logic, e.g., A will work if B fails.

(c) Degenerate elements deliver the same/similar function under some condition. When the condition changes, these degenerate elements may deliver different functions. This property leads to strong selection under environmental changes. In essence, degeneracy is a prerequisite for natural selection and evolution. Redundancy, on the other hand, does not have such a strong tie to environment.

Degeneracy exists all the three example presented earlier. In (i), due to the difficulty of measuring TFs directly and precisely, the association between TFs and gene expression is usually studied by modeling the association between TF binding sites and gene expression. The binding site of a TF is a short DNA sequence where the TF binds. It

is known that the same TF can have alternative binding sites (F. Li and Zhang 2010), and as a result, these alternative binding sites should have the similar association with gene expression. The alternative binding sites of the same TF are degenerate elements. In (ii), genes in the same pathway may be degenerate elements in the sense that different genes in the pathway may have similar association with disease prognosis. This explains the growing interest in cancer genomics that aims at identifying how gene pathway as a whole affects prognosis rather than the effect of individual genes (Vogelstein and Kinzler 2004). In (iii), brain regions that are strongly connected in a brain connectivity network may be degenerate elements because their functions may have similar association with cognition (Huang et al. 2012).

Although degeneracy has been extensively discussed in the biological literature, its implication to statistical modeling has not been rigorously defined. Consider a biological system with Q elements, $X_1, \dots, X_Q$, jointly performing a function or yielding an output Y. For example, $X_1, \dots, X_Q$ may be Q potential binding sites of some TFs of interest which bind to the upstream region of a gene to regulate the gene's expression. Y is expression level of the gene. In the context of a predictive model, $X_1, \dots, X_Q$ are predictors and Y is the response variable. If a subset $\{X_{(1)}, \dots, X_{(q)}\} \subset \{X_1, \dots, X_Q\}$ consists of degenerate elements, e.g., they are potential binding sites of a TF, then according to the definition of degeneracy, $\{X_{(1)}, \dots, X_{(q)}\}$ should satisfy two conditions: (1) they are structurally different; (2) they perform the similar function, which means that their respective coefficients, $\{w_{(1)}, \dots, w_{(q)}\}$, that link them to Y should satisfy $\|w_{(i)} - w_{(j)}\| < \epsilon$, $\forall i, j \in \{1, \dots, q\}, i \neq j$. $\|\cdot\|$ is an appropriate norm and ϵ is a biologically

defined threshold. A degenerate system may contain more than one subset of degenerate elements such as the subsets corresponding to different TFs. The challenge in modeling a degenerate system is how to build the biological knowledge about the degeneracy into statistical modeling, especially considering that the knowledge is often qualitative and with uncertainty.

3.4 Modeling of Degeneracy

Recall that the proposed transfer learning method in Chapter 2 includes a hyper-parameter, $\mathbf{\Phi}$, that encodes the prior knowledge about the correlation structure of the regression coefficients. This is indeed the prior knowledge about degeneracy. In real-world applications, it is common that some qualitative knowledge about the degeneracy exists, which can be represented by a graph $G = \{\mathbf{X}, \mathbf{E}\}$. The nodes in the graph are elements of the system, i.e., predictors $\mathbf{X}$ in the predictive model. $\mathbf{E} = \{X_i \sim X_j\}$ is a set of edges. a_{ij} is the edge weight. No edge between two nodes implies that the nodes are not degenerate elements to each other. If there is an edge between two nodes, the edge weight reflects the level of certainty that the nodes are degenerate elements. Next, we will discuss how to construct such a graph for the three examples presented previously:

In (i), nodes/predictors are potential TF binding sites. A potential binding site is a short DNA sequence in the upstream promoter region of a gene, e.g., ACGCGT, ATGCGC. The letters in each word (i.e., each binding site) can only be from the DNA alphabet $\{A, C, G, T\}$. If focusing on all κ-letter-long words, called κ-mers, there will be 4^{κ} nodes in the graph. It is known that the binding sites with similar word composition are more likely to be alternative binding sites of the same TF (X. Li et al. 2010). The similarity between two binding sites can be measured by the number of letters they have

in common in their respective words. For example, the similarity between ACGCGT and ATGCGC is 4, because they share four letters in the same position. A formal definition of this similarity between two binding sites X_i and X_j is $\kappa - H\{X_i, X_j\}$. $H\{X_i, X_j\}$ is the so-called Hemming distance defined as $H\{X_i, X_j\} = \sum_{l=1}^{L} I(c_{il} \neq c_{jl})$ (Li and Zhang 2010). $I(\cdot)$ is an indicator function. c_{il} is the l-th letter in the word of binding site X_i. Using this similarity measure, two nodes X_i and X_j do not have an edge if they do not have any common letter in the same position; they have an edge otherwise and the edge weight is their similarity. Likewise, in example (ii), nodes of the graph are genes and edges can be put between genes according to known pathway databases such as KEGG (http://www.genome.jp/kegg/) and BioCarta (http://www.biocarta.com/). In example (iii), nodes of the graph are brain regions and edges can be put between brain regions with known functional or anatomical connectivity (Huang et al. 2010).To incorporate the graph into our model, the graph is first converted to a Laplacian matrix, **L**, i.e.,

$$\mathbf{L}_{ij} = \begin{cases} d_i & if\ i = j \\ -a_{ij} & if\ i \neq j\ and\ X_i \sim X_j \\ 0 & otherwise \end{cases}, \tag{3.1}$$

where $d_i = \sum_{X_i \sim X_j} a_{ij}$ is called the degree of node X_i. It is known that **L** is always non-negative definite and it encodes many properties of the graph (Chung 1999). If the graph encodes the degeneracy of the system, **L** can be reasonably used to replace the $\mathbf{\Phi}^{-1}$ in the optimization problems in Case III and Case IV of chapter 2. Then, we obtain the following optimization problems for Case V and Case VI, respectively.

Case V:

$$\widehat{\mathbf{W}}^{\mathrm{V}} = \underset{\mathbf{W}}{\operatorname{argmin}} \left\{ \sum_{k=1}^{K} \|\mathbf{y}_k - \mathbf{X}_k \mathbf{w}_k\|_2^2 + \lambda_1 \|\mathbf{W}\|_1 + \lambda_2 \left(Qlog|\mathbf{\Omega}| + tr(\mathbf{L}\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T) \right) \right\},$$

(3.2)

Case VI:

$$\widehat{\mathbf{w}}_K^{\mathrm{VI}} = \operatorname{argmin}_{\mathbf{w}_K} \left\{ \begin{array}{c} \|\mathbf{y}_K - \mathbf{X}_K\mathbf{w}_K\|_2^2 + \lambda_1\|\mathbf{w}_K\|_1 \ + \\ \lambda_2\left(Qlog\left(\varsigma_K - \boldsymbol{\varpi}_K^T\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\right) + \frac{1}{\varsigma_K - \boldsymbol{\varpi}_K^T\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K}(\mathbf{w}_K - \boldsymbol{\mu}_K)^T\mathbf{L}(\mathbf{w}_K - \boldsymbol{\mu}_K)\right) \end{array} \right\}. \quad (3.3)$$

Case V and Case VI can be solved by the algorithm in Figures 2 and 3.

Next, we would like to provide some theoretical analysis to reveal the role of the graph in the optimizations/estimations. We will focus on Case VI; a similar result can be obtained for Case V. For notation simplicity, we further simply (3.3) into:

$$\widehat{\mathbf{w}}_K^{\mathrm{VI}} = \operatorname{argmin}_{\mathbf{w}_K} \ \left\{\|\mathbf{y}_K - \mathbf{X}_K\mathbf{w}_K\|_2^2 + \lambda_1\|\mathbf{w}_K\|_1 \ + \lambda_2(\mathbf{w}_K - \boldsymbol{\mu}_K)^T\mathbf{L}(\mathbf{w}_K - \boldsymbol{\mu}_K)\right\}. \quad (3.4)$$

by dropping the constant $Qlog\left(\varsigma_K - \boldsymbol{\varpi}_K^T\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\right)$ and re-using λ_2 to represent $\frac{\lambda_2}{\varsigma_K - \boldsymbol{\varpi}_K^T\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K}$ in (3.3).

Lemma 3.1: The optimization in (3.4) is equivalent to:

$$\widehat{\mathbf{w}}_K^{\mathrm{VI}} = \operatorname{argmin}_{\mathbf{w}_K} \ \left\{\|\mathbf{y}_K - \mathbf{X}_K\mathbf{w}_K\|_2^2 + \lambda_1\|\mathbf{w}_K\|_1 \ + \lambda_2 \sum_{X_i \sim X_j} a_{ij}\left((w_{iK} - \mu_{iK}) - (w_{jK} - \mu_{jK})\right)^2\right\}. \quad (3.5)$$

(3.5) enables a clear interpretation for the role of the graph. That is, the edge weight between two nodes/predictors, a_{ij}, plays an role of regulating the closeness between the estimated coefficients of the two predictors, w_{iK} and w_{jK}, after adjusting for the knowledge transferred from source domains that is embraced in μ_{iK} and μ_{jK}. Consider a special case of no transfer learning, i.e., $\mu_{iK} = 0$ and $\mu_{jK} = 0$. Then, the larger the edge weight a_{ij}, the closer the estimated w_{iK} and w_{jK} should be in order to achieve the minimization in (3.5). Theorem 3.1 below shows the existence of an upper

bound for the difference between the estimated w_{iK} and w_{jK}, and this upper bound is inversely related to the edge weight a_{ij}.

Theorem 3.1: Let $\widehat{w}_{ik}^{VI}, \widehat{w}_{jk}^{VI} \in \widehat{\mathbf{w}}_K^{\mathrm{VI}}$ be the estimated coefficients for predictors X_i and X_j. Let $\mathbf{x}_{iK}$ and $\mathbf{x}_{jK}$ be the data vectors for X_i and X_j, respectively. Suppose that $\widehat{w}_{ik}^{VI}\widehat{w}_{jk}^{VI} > 0$ and $a_{ij} \gg a_{uv}$. a_{uv} is the weight of any edge other than $X_i \sim X_j$. Then, for fixed λ_1 and λ_2 and a square-error loss,

$$\left|(\widehat{w}_{ik}^{VI} - \mu_{iK}) - \left(\widehat{w}_{jk}^{VI} - \mu_{jK}\right)\right| \leq \frac{\|\mathbf{x}_{iK} - \mathbf{x}_{jK}\|_2}{2\lambda_2} \times \sqrt{\frac{\|\mathbf{y}_K\|_2^2}{a_{ij}^2} + \frac{2\lambda_2(\mu_{iK} - \mu_{jK})^2}{a_{ij}}}. \quad (3.6)$$

The proof for Lemma 3.1 and Theorem 3.1 can be found in Appendix 3.I. In the upper bound in (3.6), the data for the target domain, $\mathbf{x}_{iK}$, $\mathbf{x}_{jK}$, and $\mathbf{y}_K$, knowledge transferred from the source domains, μ_{iK} and μ_{jK}, and λ_2 can be considered as given. Then, the upper bound is inversely related to the edge weight a_{ij}.

3.5 Simulation Study

In this section, we use AUC previously mentioned in chapter 2 to compare models with degeneracy modeling (the proposed method) and without degeneracy modeling (lasso).

Consider a single domain and the model $\mathrm{Y} = \sum_{\mathrm{q}=1}^{\mathrm{Q}} \mathrm{w_q X_q} + \varepsilon$ with 50 predictors, i.e., $\mathrm{Q} = 50$. Suppose that the 50 predictors fall into 10 non-overlapping subsets; each subset consists of five predictors as its degenerate elements. Coefficients of the first two subsets, $\{\mathrm{w}_1, \mathrm{w}_2, \mathrm{w}_3, \mathrm{w}_4, \mathrm{w}_5\}$ and $\{\mathrm{w}_6, \mathrm{w}_7, \mathrm{w}_8, \mathrm{w}_9, \mathrm{w}_{10}\}$ are non-zero and generated from $\mathrm{N}(5,1)$ and $\mathrm{N}(1,1)$, respectively. Coefficients of the rest three subsets are zero. This is to reflect the reality that some degenerate elements of the system may not relate to the

particular response of interest. Next, we want to generate samples for the 50 predictors. The way these samples are generated must follow the biology of how the degenerate elements are formed, so it is different from chapter 2. Specifically, assuming that the 10 subsets correspond to 10 TFs, we first generate 10 TFs, $\mathrm{TF}_1, \ldots, \mathrm{TF}_{10}$, from $\mathrm{N}(0,1)$. Next, to reflect the stochastic nature of the degenerate elements corresponding to each TF_i, we generate TF_i's corresponding five predictors/degenerate elements from $\mathrm{N}(\rho \times \mathrm{TF}_i, 1-\rho^2)$. ρ corresponds to the correlation between TF_i and its corresponding degenerate elements. We try different correlation levels for generality. 50 samples are generated for each correlation level.

Table 2. Best AUC performances of proposed model considering degeneracy and lasso

ρ	Proposed model considering degeneracy	Lasso (no consideration of degeneracy)
0.6	0.8138	0.6963
0.7	0.8475	0.6875
0.8	0.8388	0.6888

To apply the proposed method, we first build a graph that puts an edge between each pair of predictors in each of the five subsets (no edge between the subsets) to represent the qualitative prior knowledge about the degeneracy. The edge weight is set to be one. The graph is then converted to a Laplacian matrix L and used in the proposed method. A lasso model is also applied to the simulation datasets as a model not taking the degeneracy into account. The process is repeated for 50 times. The average AUC performances of the two methods are comparable. However, when the best AUC

performances of the two methods are compared, the proposed method is significantly better, as can be seen in Table 2.

3.6 Application of Gene Expression Prediction by Transcription Factors

We present an application of modeling the predictive relationship between TFs and gene expression. Potential TF binding sites are predictors and gene expression is the response. Eight human cell lines (H1, K562, GM12878, HUVEC, HSMM, NHLF, NHEF, and HMEC) are considered as eight domains. Since the simulation studies presented the results for Case V, here we present the results for Case VI. To apply the model in Case VI, one cell line is treated as the target domain and all the others are treated as the source domains. The data for the predictors are obtained as follows: We download the RefSeq Gene annotation track for human genome sequence (hg19) from the University of California Santa Cruz Genome Browser (USCS, http://genome.ucsc.edu). Then, we scan the promoter region of each gene (i.e., 1000bp upstream of the transcription state site) and count the occurrence of each κ-mer. Recall that a κ-mer is a κ-letter-long word describing a potential binding site. We do this for $\kappa = 6$ and obtain data for 4^6 predictors, and for $\kappa = 7$ and obtain data for 4^7 predictors. $\kappa = 6,7$ are common choices for binding site studies (X. Li et al. 2010). A minor technical detail is that in human cell lines, a word and its reverse complement should be considered the same predictor. This reduces the 6-mer predictors to 2080 and 7-mer predictors to 8192. Furthermore, we obtain data for the response variable, i.e., gene expression, for the eight cell lines from the Gene Expression Omnibus (GEO) database under the accession number GSE26386 (Ernst et al. 2011). A total of 16324 genes on all chromosomes are included. This is the sample size.

Recall in Section 3.4, we mentioned that a graph can be constructed to represent the prior knowledge about the degeneracy. Nodes are predictors, i.e., κ-mers. The similarity between two κ-mers is $\kappa - H\{X_i, X_j\}$. $H\{X_i, X_j\}$ is the Hamming distance. We consider an unweighted graph here. Specifically, there is an edge between X_i and X_j, if $\kappa - H\{X_i, X_j\} \geq s$, i.e., X_i and X_j share at least s letters in the same position of their respective words. There is no edge between X_i and X_j otherwise. s is a tuning parameter in our method.

3.6.1 Comparison to Methods without Transfer Learning or without Degeneracy Modeling

The method without degeneracy modeling is the model in Case VI but with $\mathbf{L} = \mathbf{I}$. The method without transfer learning is a lasso model applied to data of the target domain alone. Each method has some tuning parameters to select. For example, the tuning parameters for the proposed method include λ_1, λ_2, and s. We find that $s = 5$ is a consistently good choice across different choices for λ_1 and λ_2. λ_1 and λ_2 can be selected based on model selection criteria such as BIC and AIC. However, each criterion has some known weakness and there is no such a criterion that works universally well under all situations. To avoid drawing biased conclusion, we do not stick to any single model selection criterion. Instead, we run the model on a wide range of values for λ_1 and λ_2, i.e., λ_1, $\lambda_2 \in [10^{-5}, 10^3]$, and report the average performance. Similar strategies are adopted for the two competing methods. This is indeed a common practice for comparison of different methods each of which has parameters to be tuned (Wang et al. 2012).

All 2080 6-mers are used as predictors. To compare the three methods in challenging predictive problems, i.e., problems with small sample sizes, only the 1717 genes on chromosome 1 are included. Furthermore, one cell line is treated as the target domain and all the other cell lines are treated as the source domains. The knowledge of the source domains, i.e., $\widetilde{\mathbf{W}}^*$, is obtained using the model in Case VI applied to the data of the source domains. The data of the target domain is divided into 10 folds. Nine folds of data are used, together with $\widetilde{\mathbf{W}}^*$, to train a model, and the model is applied to the remaining one fold to compute a performance metric such as the MSE. The average MSE, $\overline{\mathrm{MSE}}$, over the 10 folds is computed. This entire procedure is repeated for each of the eight cell lines as the target domain and the eight $\overline{\mathrm{MSE}}s$ are averaged to get $\overline{\overline{\mathrm{MSE}}}$. This $\overline{\overline{\mathrm{MSE}}}$ can be obtained for each pair of λ_1 and λ_2 in their range $[10^{-5}, 10^3]$. Averaging the $\overline{\overline{\mathrm{MSE}}}s$ over the range gives $\overline{\overline{\overline{\mathrm{MSE}}}}$. Table 3 shows the results of comparison. It is clear that both transfer learning and degeneracy modeling in the proposed method help prediction in the target domain. Transfer learning is crucially important, without which the prediction is significantly impaired.

Table 3. Comparison of three methods by $\overline{\overline{\overline{\mathrm{MSE}}}}$

	Proposed method vs. transfer learning without degeneracy	Proposed method vs. lasso (no transfer learning)
$\dfrac{\overline{\overline{\overline{\mathrm{MSE}}}}(\text{competing}) - \overline{\overline{\overline{\mathrm{MSE}}}}(\text{proposed})}{\overline{\overline{\overline{\mathrm{MSE}}}}(\text{proposed})} \times 100\%$	16.25%	920.44%

3.6.2 Robustness of the Proposed Method to Noisy Source Domains

One distinguished feature of the proposed method is the ability to learn the relationship between each source domain and the target domain from data, and adaptively

decide how much knowledge to transfer from each source domain. To test this, we can include some “noisy” source domains. If the proposed method has the ability it claims to have, it should transfer little knowledge from the noisy domains and its performance should not be affected much. Specifically, we create the noisy source domains by destroying the correspondence between the response and predictors of each gene in these domains. For example, if we want to create a noisy source domain out of a cell line, we shuffle the gene expression data of that cell line. Next, we apply the proposed method to the data of the target domain with transfer learning from the source domains (note that some of the source domains have been shuffled to become noisy source domains). We compare the estimated model coefficients of the target domain and those obtained by keeping all the source domains as they are (i.e., no shuffling) by calculating their correlation coefficient. Table 4 shows this correlation coefficient with four, five, and six source domains shuffled. Cell line GM12878 is the target domain. When applying the proposed method, λ_1 and λ_2 are selected by 10-fold cross validation. It can be seen that the proposed method is almost not affected when less than five out of seven source domains are noisy domains. Furthermore, we also compute the correlation between the model coefficients of the target domain with and without transfer learning (no shuffling) and this correlation is 0.793765, which is at the similar level to that when there are more than five noisy domains. Finally, we would like to know if transfer learning can still outperform single-domain learning (i.e., lasso for the target domain) even with knowledge transferred from noisy domains. This result is summarized in Table 5, which further demonstrates the robustness of the proposed method.

Table 4. Correlation between model coefficients of the target domain with and without shuffled noisy source domains

Four out of seven source domains are shuffled	Five out of seven source domains are shuffled	Six out of seven source domains are shuffled
0.998065	0.816133	0.763273

Table 5. Comparison between transfer learning with shuffled noisy source domains and single-domain learning

	Four out of seven source domains are shuffled	Five out of seven source domains are shuffled	Six out of seven source domains are shuffled
$\frac{\overline{\overline{\mathrm{MSE}}}(\text{lasso}) - \overline{\overline{\mathrm{MSE}}}(\text{transfer learning})}{\overline{\overline{\mathrm{MSE}}}(\text{transfer learning})} \times 100\%$	22.42%	20.26%	19.95%

3.6.3 Understanding the Degenerate System

The purpose of predictive modeling is not only to predict a response but also to facilitate understanding of the problem domain. To achieve this, we apply the proposed method to one cell line, GM12878, treating this cell line as the target domain and all other cell lines as the source domains. Predictors are all 8192 7-mers. 7-mers contain richer binding site information than 6-mers, but analysis of 7-mers has been limited because of the dimension. Focusing on 7-mers can also test the capability of our method in handling very large dimensional predictors. The response is a binary indicator variable that indicates if a gene is expressed or unexpressed, so a logistic loss function is used in our method. This has a purpose of testing the capability of our method in classification problems. Also, it is more reasonable to assume that binding site counts like 7-mers can explain a majority of the variability in expressed/unexpressed genes than the variability in the numerical gene expression levels. The latter is more involved, as the expression level

is affected by a variety of other factors than binding site counts. 16324 genes on all chromosomes are included in the analysis.

Unlike Section 3.6.1 in which comparison of prediction accuracy between methods is the primary goal, here we want to obtain a model for the 7-mer-gene-expression relationship, and based on the identified relationship, to understand the system better. For this purpose, model selection is unavoidable. We use 10-fold cross validation to choose the optimal λ_1 and λ_2, which are ones giving the smallest average classification error over the 10 folds. Table 6 shows the classification performance of our method in terms of True Positive Rate (TPR), True Negative Rate (TNR), and accuracy. The definition of TPR is: among all the genes classified as expressed, the proportion that is truly expressed. TNR is: among all the genes classified as unexpressed, the proportion that is truly unexpressed. Accuracy is the proportion of correctly classified genes. An observation is that TPR is higher than TNR, which is expected, because classification of unexpressed genes is supposed to be harder than expressed genes. The accuracy is 0.70, which is satisfactory in this application, considering the complexity of the biological system. Given satisfactory accuracy, we can now proceed and use the model for knowledge discovery. To do this, we use all the data of GM12878 to fit a model under the optimal λ_1 and λ_2, which is called "the model" in the subsequent discussion.

Table 6. Classification performance for GM12878 treated as the target domain

TPR	TNR	Accuracy
0.84	0.60	0.70

In knowledge discovery, our goal is to characterize the degeneracy of the target domain, i.e., GM12878. Note that although we have used a graph to encode the

degeneracy, it is before seeing any data and is only qualitative. It can now be better characterized by the model that incorporates both the graph and the data of the target domain as well as knowledge transferred from the source domains. Specifically, the following steps are performed:

First, we examine the estimated coefficients of the 7-mers and eliminate those 7-mers with zero coefficients from the graph. These 7-mers are considered not significantly affecting gene expression. Then, we rank the remaining 7-mers according to the magnitudes of their coefficients and choose the top 50 7-mers for the subsequent analysis. This helps us focus on identifying the degeneracy most relevant to gene expression. Some of the 50 7-mers are connected in the graph and some are not; in fact, they fall into different clusters. We define a cluster to be a group of 7-mers, each of which is connected with at least one other 7-mer in the group. The clusters are shown in Table 7. Each cluster is suspected to correspond to a TF and the 7-mers in the cluster are believed to be alternative binding sites of the TF. To verify this, we compute a Position Specific Scoring Matrix (PSSM) for each cluster. PSSM has been commonly used to characterize binding site uncertainty (X. Li et al. 2010). A PSSM is a $\kappa \times 4$ matrix. κ is the number of positions in a κ-mer. $\kappa = 7$ in our case. Each row of a PSSM is a probability distribution over $\{A, C, G, T\}$. Let $p_i(s)$ denote the probability of s, $s = \{A, C, G, T\}$, for row/position i, $i = 1, \ldots, \kappa$. $\sum_{s=\{A,C,G,T\}} p_i(s) = 1$. $p_i(s)$ can be calculated by $p_i(s) = \frac{n_i(s)}{C}$, where C is the cluster size and $n_i(s)$ is the number of occurrences of s at position i among all the 7-mers in the cluster. Because our model outputs an estimated coefficient for each 7-mer, we modify this conventional formula by $p_i(s) = \frac{\sum_{c=1}^{C} \widehat{w}_c \, I(\mathbf{r}_{ci}=s)}{\sum_{c=1}^{C} \widehat{w}_c}$. $\widehat{w}_c$ is the estimated

coefficient for the c-th 7-mer in the cluster. $I(\cdot)$ is an indicator function. $\mathbf{r}_{ci}$ is the letter at the i-th position of the c-th 7-mer. This modified formula works better in our case because it takes the response variable into consideration by incorporating the model coefficients. Taking cluster 1 of GM12878 in Table 7 as an example, the PSSM is:

$$\begin{array}{c} \begin{array}{cccc} \text{A} & \text{C} & \text{G} & \text{T} \end{array} \\ \begin{bmatrix} 0.88 & 0.12 & 0 & 0 \\ 0.12 & 0.88 & 0 & 0 \\ 0 & 0 & 0.47 & 0.53 \\ 0 & 0 & 0 & 1 \\ 0 & 0.65 & 0.23 & 0.12 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.34 & 0.66 \end{bmatrix} \end{array}$$

A PSSM can be represented in a compact form by a motif logo, which stacks up the four letters $\{A, C, G, T\}$ at each position i and the letter height is proportional to its probability $p_i(s)$. Please see Table 7 for the PSSM motif logos for all the clusters.

Furthermore, the PSSM of each cluster can be compared with databases of known TFs to see if there is a match. We used the Motif-based Sequence Analysis Tools (http://meme.nbcr.net/meme) for the matching. Table 7 shows the top five matched TFs for each cluster, according to the significance level of each match. If less than five matched TFs are found, then all the matched TFs will be shown. If no match is found, there is a "N/A". Out of the eight clusters, six have at least one match with known TFs. Clusters 1, 2, and 6 are enriched with SPI1, Ets, Elk, , FLI1, FEV, GABP, and EHF, which are well-known TFs for important basic cell functions. Cluster 3 is enriched with AP-1 and NF-E2, which are related to Golgi membrane and nucleus that are also basic cell functions. Clusters 5 and 7 are enriched with Zfx and CNOT3. CNOT3 is a Leukocyte Receptor Cluster Member 2 and Zfx is required for the renewal process in

hematopoietic cells. As GM12878 is a lymphocyte cell, these blood transcription factors are specific to this cell line. Clusters 4 and 5 do not match with any known TFs. However, only 10-20% of total human TFs are known so far. The unmatched clusters indeed present an interesting opportunity for identifying new TFs.

This entire analysis for GM12878 is also performed for other cell lines. For each cell line, clusters of 7-mers exist and a large majority of the clusters can be matched to known TFs. Also, some clusters are common across the cell lines. These are the clusters whose matched TFs are related to basic cell functions. There are also some cell-line-specific clusters such as clusters 5 and 7 for GM12878. As other examples, there is a cluster enriched with CTF1 for HMEC. CTF1 is known to be in entracellar region. As HMEC is an epithelial cell, CTF1 is specific to this cell line. In addition, there is a cluster enriched with MyoD and another cluster enriched with MEF-2 for HSMM. MyoD is related to muscle cell differentiation and MEF-2 is a myocyte enhancer factor, both being specific to HSMM. The identified common and cell-line-specific cluster structures verifies transfer learning's ability of modeling related but not exactly the same domains.

Table 7. Clusters of 7-mers and matching with known TFs for GM12878

Clusters	7-mers	Est.	Motif logo	Matched known TFs
1	AAGTGCT	0.005808		SPI1, Ets, Elk-1, FLI1, FEV
	ACGTGCT	0.005497		
	ACGTTCT	0.005367		
	ACGTCCT	0.005963		
	ACTTCCT	0.009086		
	ACTTCCG	0.010709		
	CCTTCCG	0.005685		
2	GGCGGAA	0.007632		GABP, Elk-1, Ehf_primary, Eip74EF, ERF
	GCCGGAA	0.006837		
	ACCGGAA	0.006497		
	CCCGGAA	0.006982		
	TCCGGAA	0.005488		
3	ACTAAGT	0.005597		AP-1, NF-E2
	ACTCAGT	0.005881		
	ACCCAGT	0.006013		
4	ATGACAT	-0.00594		N/A
	ATCACAT	-0.00588		
5	CAGGCCG	0.006636		Zfx, CNOT3
	AAGGCCG	0.00586		
6	CCGGAAG	0.009778		ELK-1, GABPA, Eip74EF, SAP-1a, EHF
	CCGGAGG	0.005296		
7	AGGCCGC	0.005775		Zfx
	AGGCCGG	0.005715		
8	TAGACTA	0.006607		N/A
	TAAACTA	0.006447		

Chapter 4: Temporal Transfer Learning for Modelling and Change Detection of Dynamic Networks by a Network State Space Model

4.1 Introduction

In many data-rich domains, the data exist in the form of a network that consists of nodes and edges. Typical examples include social networks, gene networks, brain networks, and supply networks. A network naturally evolves over time: people's interaction in a social network may become more and more often as they become more acquainted with each other; functional connectivity of a human brain may become more and more sparse with aging; a supply network may have seasonality. This results in a time series of networks, also referred to as dynamic networks in this paper. Dynamic Network Modeling (DNM) has been a popular research area in Computer Science (CS) in recent years (McCallum, Corrada-Emmanuel, and Wang 2005). Most existing methods extend previously developed models on cross-sectional or static network data (i.e., data in the form of a network at a single time point or at an aggregate view) to temporal models. DNM is typically used for community detection, prediction, characterization of temporal trends, and visualization.

In addition to the variability of natural evolution, another type of variability in dynamic network data is associated with assignable causes. The latter variability is what we refer to as "changes" in this paper. Examples of assignable causes include preparation for a terrorist attack that leads to changes in the social network of members in the terrorist group, a brain disease that leads to changes in a person's brain connectivity network, and new government regulation that leads to changes in a supply network. Accurate and timely change detection from dynamic network data is of critical importance in many practical domains: it can generate alerts for a potential terrorist attack, preparing the

authorities and people to properly respond; it can detect the onset of a brain disease, making treatment and disease management more effective. However, change detection has been little addressed by the existing DNM research.

Change detection is a classic research area in Statistical Process Control (SPC). Numerous approaches have been developed for change detection from univariate or multivariate time series data (Alwan and Roberts 1988; Apley and Shi 1999; Berthouex, Hunter, and Pallesen 1978; Dooley and Kapoor 1990; Dooley et al. 1986), but not from time series of networks. Change detection from network data has been studied by a number of researchers in recent years. Most of the research shares a similar logical procedure that first extracts aggregated measures of network topology (e.g., density, degree, clustering coefficient, and scan statistics) and then treats these measures as univariate or multivariate data to which conventional SPC approaches become immediately applicable (Marchette 2012; McCulloh and Carley 2011; Neil et al. 2013; Park et al. 2013; Priebe et al. 2005). A different line of work monitors the formation mechanism of network topology. For example, Azarnoush et al. (2015) proposed a method that relates the edge probability to node attributes by a logistic regression and formulates a likelihood ratio test to detect changes in the regression coefficients. However, in all the aforementioned research, network data collected at different temporal snapshots are not treated as time series but independent observations, i.e., the natural evolution of the dynamic networks is not modeled in development of the change detection methods.

In this paper, we study change detection from dynamic (i.e., time series) network data. There are two essential steps in the methodological development: First, we need to

develop a model capable of characterizing the natural evolution of dynamic networks with high accuracy. This would provide a proper baseline model against which changes can be detected. Second, we need to develop a change detection method capable of combining the baseline model with new data in a principled way to detect changes with statistical rigor. Additionally, because the baseline model developed in the first step will be used in the change detection in the second step, it is important that the baseline model takes a form that not only ensures high accuracy in fitting the network data with natural evolution but also facilitates detection of various changes in new data.

To serve the purpose of the first step, we propose a State Space Model (SSM), called Network SSM (NSSM), to characterize the natural evolution of dynamic networks by attributing the observed network evolution to the evolution of latent state vectors that represent the "social propensities" of nodes. SSM is a classic modelling approach for multivariate time series data when the data are noisy and there is difficulty for directly characterizing the evolution of the observed data (Shi 2006). These conditions are clearly true for network data. In addition, defining the latent state vectors to be social propensities of nodes in the NSSM shares a similar idea to (Azarnoush et al. 2015) that elucidates the formation of edges in the networks, i.e., two nodes with more similar social propensities should be more likely to have an edge. The difference is that the method in Azarnoush et al. (2015) requires the social propensities to be observed node attributes, whereas the NSSM assumes them to be latent state vectors. Because the state vectors are not observed, we have the flexibility of assuming their probability distribution to make the subsequent modeling more convenient. Specifically, we assume that the state vectors are multivariate Gaussian and their temporal evolution is characterized by a linear

Markovian state equation. Also, we propose a novel observation equation of the NSSM to link the state vectors of nodes to the observed edges in the network at each time point. The form of the observation equation allows us to further develop an Expectation Propagation (EP) algorithm to adequately approximate it by a multivariate Gaussian distribution of the state vectors. Utilizing the Gaussian approximation, parameter estimation for the NSSM becomes tractable. For parameter estimation, we adopt the Expectation-Maximization (EM) framework but develop a Bayesian optimal smoothing (BOS) algorithm to compute the expectation in the E-step that suits the NSSM. We call this integrated algorithm an EM-BOS algorithm in this paper.

Furthermore, after an NSSM is estimated from dynamic network data with natural evolution, we propose to integrate the NSSM into the logical procedure of SPC to detect changes in new networks. Specifically, we propose to compare a predicted covariance matrix of the state vector at a future time t^* using the NSSM with an estimated covariance matrix using the network data at t^* alone. Covariance monitoring is a well-established research area in SPC. Many approaches have been developed to detect various changes in a covariance matrix of multivariate data with statistical rigor. The state space formulation of the NNSM allows these approaches to be adopted to detect various changes in network data.

Finally, we would like to stress that the NSSM is flexible in the sense that it can be easily extended to model a broad spectrum of network data and to integrate network and non-network data, such as networks with hyper-edges, multi-dimensional networks, and integration of node attributes and external/environmental factors with network data.

This would allow for monitoring and change detection in a variety of application domains.

4.2 Related Work in Network Modeling and SPC

Network modeling: This is a popular research area in CS with most motivation examples and applications in social networks. Earlier research focused on static networks. Because of the static nature, the research typically modelled the network at a single time point or at an aggregate view. Classic approaches include the Exponential Random Graph Model (ERGM) and extensions (Hanneke, Fu, and Xing 2010) which are descriptive in nature, and the Stochastic Block Model (SBM) and extensions (Airoldi et al. 2009; Nowicki and Snijders 2001) that aim for community detection. Another popular approach is the Latent Space Model (LSM) (Handcock, Raftery, and Tantrum 2007; Hoff, Raftery, and Handcock 2002; Globerson et al. 2007; Miller, Griffiths, and Jordan 2009). LSM works by embedding nodes of the network into a low-dimensional latent space in which the relative positions of the nodes reflect their relationship in the observed network. LSM is flexible in the sense that once the latent positions of the nodes are estimated, they can be used to achieve various goals such as visualization, community detection, and link prediction.

DNM has attracted much attention in recent years. Research has been done to extend the models previously developed for static networks to modelling of time series data in the form of networks. For example, a temporal ERGM (TERGM) (Hanneke, Fu, and Xing 2010) was developed as an extension to the original static ERGM. The static ERGM considers the probability distribution of the network to follow that from an exponential family, in which the sufficient statistics are pre-defined graph statistics such

as the number of edges, number of k-stars, and number of triangles. In the TERGM, the sufficient statistics are defined on two successive networks to characterize their temporal evolution, such as the stability, reciprocity, and transitivity statistics. Also, several methods have been developed to extend the static SBM to temporal models. Xing, Fu, and Song (2010) and Ho, Song, and Xing (2011) proposed temporal extensions of a mixed-membership version of the static SBM using linear state space models for the real-valued class memberships. In (Yang et al. 2011), Yang et al. proposed a temporal extension of the static SBM that explicitly models nodes changing between classes over time by using a transition matrix that specifies the probability that a node in class i at time t switches to class j at time $t + 1$ (Xu and Hero 2014). In addition, temporal LSMs for dynamic networks have also been developed. Hoff (2011) proposed a dynamic latent factor model analogous to an eigenvalue decomposition with time-invariant eigenvectors and time-varying eigenvalues. The model is applicable to many types of data in the form of multi-way arrays, including dynamic networks. Lee and Priebe (2011) proposed a latent process model for multi-relational dynamic networks using random dot product spaces. Sarkar and Moore (2005) proposed to embed nodes of the dynamic networks into a p-dimensional Euclidian latent space and use a temporal transition model to prohibit large movement of each node along successive time points. Goals of the aforementioned dynamic network models include characterization of the temporal trend, prediction, and visualization, but not change detection.

Monitoring and change detection in statistical process control (SPC): Two related research areas to this paper are SPC for time series data and SPC for network data. Numerous SPC control charts have been developed for univariate and multivariate time

series data (Alwan and Roberts 1988; Apley and Shi 1999; Berthouex, Hunter, and Pallesen 1978; Dooley and Kapoor 1990; Dooley et al. 1986). These methods are not applicable to time series in the form of networks. On the other hand, SPC methods for network data have been developed by a number of researchers in recent years. Most existing research applies SPC to aggregated measures of network topology, such as density, number of triangles, global clustering coefficient, and scan statistics (Marchette 2012; McCulloh and Carley 2011; Neil et al. 2013; Park et al. 2013; Priebe et al. 2005). Azarnoush et al. (2015) recently developed a method with a different perspective, which monitors the formation mechanism of network topology. By attributing the formation of network topology to node attributes, the method adopts a logistic regression to link edge probability with node attributes and uses a likelihood ratio test to detect changes. However, all the aforementioned SPC methods for network data do not consider the networks to be time series, but independent observations.

4.3 Overview of the Proposed Methodology

Figure 5 shows the proposed methodological framework. Two key components of the methodology include development of the NSSM, which is presented in Section 4.4, and development of the change detection method by integrating the NSSM and SPC, which is presented in Section 4.5. Furthermore, Section 4.6 presents simulation experiments and a real-data application. Section 7 is the conclusion.

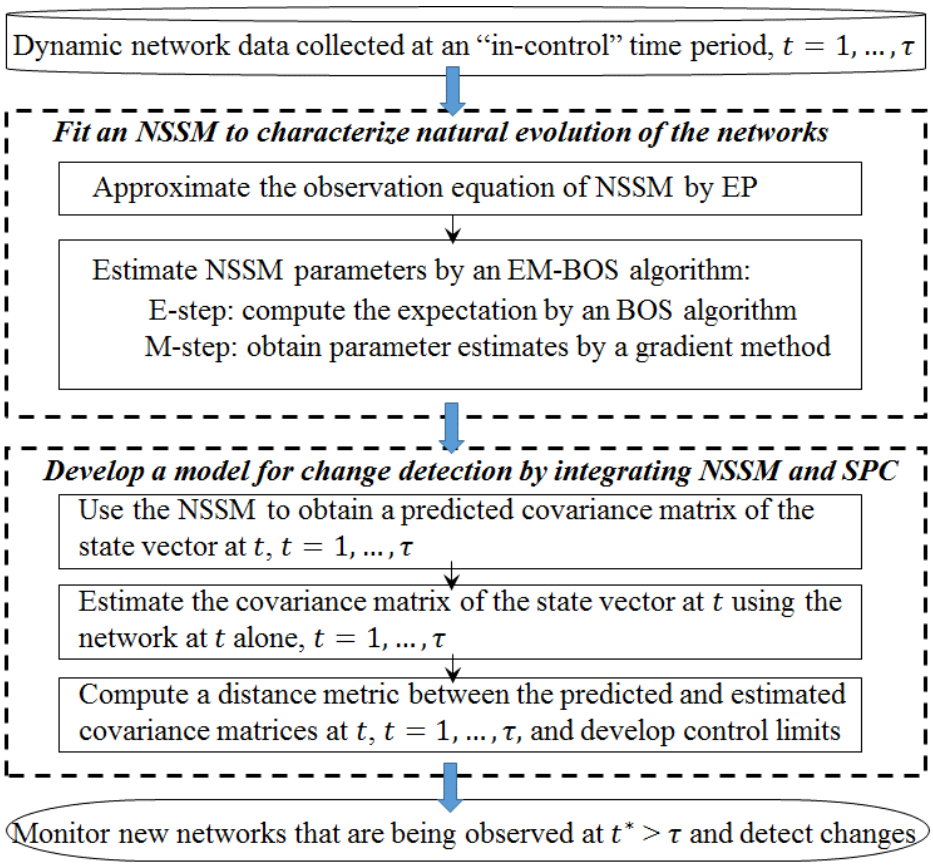


Figure 5. Proposed methodological framework

4.4 NSSM for Characterizing Natural Evolution of Dynamic Networks

In this section, we first present the mathematical formulation of the NSSM (Section 4.4.1). Then, we present the development of an EP algorithm for approximating the observation equation of the NSSM (Section 4.4.2). Next, we discuss several extensions of the NSSM to model various types of network data and incorporate non-network data with network data (Section 4.4.3). Finally, we present the development of an EM-BOS algorithm for parameter estimation of the NSSM (Section 4.4.4).

4.4.1 Model Formulation

A general SSM includes a state equation and an observation equation. The state equation models the dynamics of latent state variables. The observation equation links the

latent state variables with observational data. In the proposed NSSM, a network observed at time t is represented by $G_t = \{\mathbf{v}, \mathbf{y}_t\}$. $\mathbf{v}$ is a set of n nodes . $\mathbf{y}_t$ consists of the edges. In this paper, we focus on unweighted undirected networks. So, each edge in $\mathbf{y}_t$, i.e., $y_{ij,t}$, is treated as a Bernoulli random variable and $i < j$. Furthermore, based on the notion that the probability of having an edge between two nodes should be related to the states of the nodes, we propose one state variable for each node, x_i, $i = 1, .., n$. x_i reflects a node's propensity of interacting with others, called "social propensity" in this paper. Let $\mathbf{x}_t = (x_{1t}, \dots, x_{nt})^T$ be the state vector at time t. The temporal evolution of the state vector can be represented by a linear model in (4.1), which is the state equation of the NSSM:

$$\mathbf{x}_t = \mathbf{A}_{t-1}\mathbf{x}_{t-1} + \mathbf{q}_{t-1}. \tag{4.1}$$

$\mathbf{q}_{t-1} \sim N(\mathbf{0}, \mathbf{Q}_{t-1})$. $\mathbf{A}_{t-1}$ is a matrix of linear coefficients. The initial state vector is assumed to follow a zero-mean multivariate Gaussian distribution, i.e., $\mathbf{x}_1 \sim N(\mathbf{0}, \mathbf{\Sigma})$. To link the state vector with the network data at time t, an "ideal" observation equation can take the following form:

$$p_{ideal}(y_{ij,t} = 1 \,|x_{it}, x_{jt}) = \begin{cases} 1 & if\ x_{it}x_{jt} > 0 \\ 0 & if\ x_{it}x_{jt} < 0 \end{cases}. \tag{4.2}$$

That is, there is an edge between two nodes if their respective states have the same sign. (4.2) is called the "ideal" observation equation because it does not consider the measurement noise in network data. To account for the measurement noise, we propose the following observation equation:

$$p(y_{ij,t} = 1 \,|x_{it}, x_{jt}) = \iint p_{ideal}(y_{ij,t} = 1 \,|x_{it} + \delta_i, x_{jt} + \delta_j) N(0, \sigma_\delta^2) N(0, \sigma_\delta^2)\, d\delta_i d\delta_j, \tag{4.3}$$

where $\delta_i, \delta_j \sim N\left(0, \sigma_\delta^2\right)$ represent measurement noise. Through some algebra, we can simplify (4.3) into:

$$p\left(y_{ij,t} = 1 \,\middle|\, x_{it}, x_{jt}\right) = \phi\left(\frac{x_{it}}{\sigma_\delta}\right)\phi\left(\frac{x_{jt}}{\sigma_\delta}\right) + \left\{1 - \phi\left(\frac{x_{it}}{\sigma_\delta}\right)\right\}\left\{1 - \phi\left(\frac{x_{jt}}{\sigma_\delta}\right)\right\}. \tag{4.4}$$

$\phi(\cdot)$ is the cumulative density function (CDF) of the standard normal distribution. To understand the properties of (4.4), we plot the probability contours of (4.4) with two noise levels in Figure 6. The observations are: 1) Nodes whose states have the same sign have a higher probability of having an edge than nodes having opposite signs. 2) The magnitude of states affects the probability of having an edge in a positive way if the states of two variables have the same sign, i.e., the greater the magnitude, the higher the probability, but in a negative way if the states have opposite signs. 3) The probability of having an edge decreases as the measurement noise level increases.

Finally, based on the edge-specific observation equation in (4.4) and the assumption that the edges are independent of each other given the states of their respective nodes, we can write the observation equation of the network as follows:

$$p(\mathbf{y}_t | \mathbf{x}_t) = \prod_{ij} p\left(y_{ij,t} \,\middle|\, x_{it}, x_{jt}\right). \tag{4.4*}$$

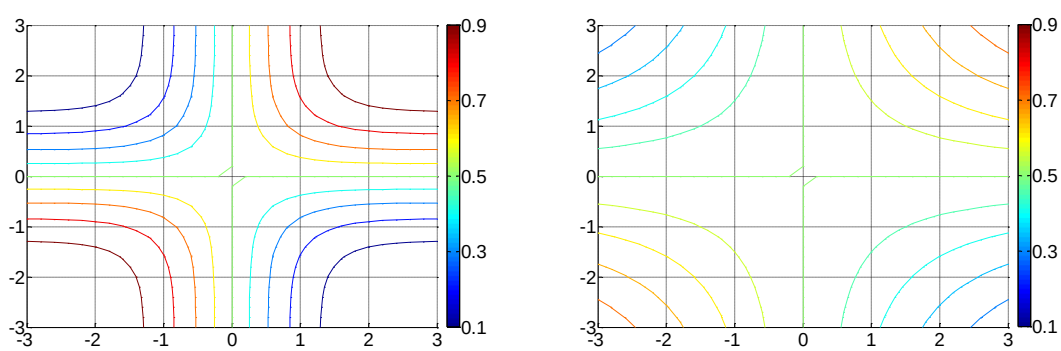


Figure 6. Probability contour plots of the proposed observation equation in (4.4) with two different measurement noise levels: (a) $\sigma_\delta = 1$; and (b) $\sigma_\delta = 3$

4.4.2 Gaussian Approximation of the Observation Equation by an EP Algorithm

To preserve computational tractability in the subsequent parameter estimation and change detection, we consider $p(\mathbf{y}_t|\mathbf{x}_t)$ as a function of $\mathbf{x}_t$ when the network $\mathbf{y}_t$ is observed, and approximate $p(\mathbf{y}_t|\mathbf{x}_t)$ by a multivariate Gaussian distribution of $\mathbf{x}_t$. The fact that the observation equation is composed by Gaussian CDFs also supports the validity of a Gaussian approximation. Among approximation algorithms such as Laplace (Shun and McCULLAGH 1995), variational Bayes (Cevher et al. 2008), and EP, EP is known to work well when the original/exact distribution takes the form of a factorization (T. P. Minka 1999) (please see a brief introduction of EP in Appendix C). This makes EP an ideal choice for our case because the joint distribution of the edges in a network can be written as a product of the edges' respective distributions according to the observation equation in (4.4*). We develop an EP algorithm to find a Gaussian approximation for the observation equation of the NSSM. The result is presented in Proposition 4.1 and the detailed derivation for finding the approximation is provided in Appendix C.

Proposition 4.1: A Gaussian approximation for the observation equation $p(\mathbf{y}_t|\mathbf{x}_t)$ in the NSSM is given by:

$$p(\mathbf{y}_t|\mathbf{x}_t) \approx N(\mathbf{x}_t|\mathbf{0}, \mathbf{\Pi}_t^{-1}),$$

where $\mathbf{\Pi}_t$ is a function of $\mathbf{y}_t$ and σ_δ , and is found by EP.

4.4.3 Model Extensions

The NSSM proposed in Section 4.4.1 can be extended in several ways to model a broad spectrum of network data and to integrate network and non-network data. Specifically, we propose four extended NSSM as follows:

1) Networks with hyperedges. An edge connects two nodes, while a hyperedge can connect/include any number of nodes. An edge characterizes only pair-wise interaction, but a hyperedge can characterize the complex interaction among the members in a group, e.g., people attending the same meeting. A network consisting of hyperedges is called a hypergraph (Kalman 1960). Many social networks and biological networks take the form of a hypergraph. To model a hypergraph, we can use the same state equation as (4.1) but modify the observation equation as follows: Let e_k be a hyperedge that connects/includes nodes $\{x_{k_1}, \dots, x_{k_n}\}$. Then, we adopt the same idea as (4.2) and assume that there is a hyperedge on the nodes if the nodes' respective states have the same sign, i.e.,

$$p_{ideal}(e_{k,t} = 1 \,|x_{k_1,t}, \dots, x_{k_n,t}) = \begin{cases} 1 & if\ sign(x_{k_1,t}) = \cdots sign(x_{k_n,t}) \\ 0 & otherwise \end{cases}.$$

Furthermore, considering measurement noise in network data, we can obtain the observation equation in a similar format to (4.4):

$$p(e_{k,t} = 1 \,|x_{k_1,t}, \dots, x_{k_n,t}) = \phi\left(\frac{x_{k_1,t}}{\sigma_\delta}\right) \times \dots \times \phi\left(\frac{x_{k_n,t}}{\sigma_\delta}\right) + \left\{1 - \phi\left(\frac{x_{k_1,t}}{\sigma_\delta}\right)\right\} \times \dots \times \left\{1 - \phi\left(\frac{x_{k_n,t}}{\sigma_\delta}\right)\right\}. \quad (4.5)$$

2) Multi-dimensional networks. At each time t, there may be more than one network for the same set of nodes. For example, a group of people may interact with each other through multi-media such as phone call, email, facebook, and twitter. We call each of these networks a "dimension" in this paper. To model multi-dimensional networks, we can use the same state equation as (4.1), but have one observation equation for each dimension of the networks. This is to assume the multi-dimensional networks to be

different realizations for the underlying social propensities (i.e., states) of the nodes. Specifically, the observation equation for the k-th dimension of the networks is:

$$p\left(y_{ij,t}^{(k)} = 1 \left| x_{it}, x_{jt} \right.\right) = \phi\left(\frac{x_{it}}{\sigma_\delta^{(k)}}\right)\phi\left(\frac{x_{jt}}{\sigma_\delta^{(k)}}\right) + \left\{1 - \phi\left(\frac{x_{it}}{\sigma_\delta^{(k)}}\right)\right\}\left\{1 - \phi\left(\frac{x_{jt}}{\sigma_\delta^{(k)}}\right)\right\}, k = 1, \dots, K. \tag{4.6}$$

If any of the dimensions is a hypergraph, (4.5) can be used as the observation equation corresponding to that dimension.

3) Incorporation of node attributes. In addition to the dynamic network data, we may also have multivariate data of a set of attributes for each node, $\mathbf{z}_i$, such as age, gender, and education background. The attributes of a node typically do not change over time, so we can incorporate them into the initial state of the NSSM by using them to define the covariance matrix of the initial state vector, i.e., $\mathbf{\Sigma}_{ij} = \kappa\left(\mathbf{z}_i, \mathbf{z}_j\right)$. $\kappa(\cdot)$ is an appropriate kernel function. This is to consider that two nodes with similar attribute profiles should have more correlated initial states.

4) Incorporation of external/environmental factors. External factors may affect the social propensities (i.e., the states) of some or all the nodes in the network. For example, in a social network, external factors may be regulation or political climate; in a biological network, external factors may be exposure to environmental hazards or a disease process. External factors, denoted by $\mathbf{u}_{t-1}$, can be incorporated into the state equation as $\mathbf{x}_t = \mathbf{A}_{t-1}\mathbf{x}_{t-1} + \mathbf{B}_{t-1}\mathbf{u}_{t-1} + \mathbf{q}_{t-1}$.

4.4.4 Parameter Estimation for the NSSM by an EM-BOS Algorithm

In this section, we will discuss parameter estimation for the basic NSSM in (4.1) and (4.4), leaving the parameter estimation for the four extended NSSMs for future work.

The parameters in the NSSM include $\mathbf{A}_{t-1}$, $\mathbf{Q}_{t-1}$, and σ_δ^2. Given that network data have been collected at τ past time points, i.e., $= [\mathbf{y}_1^T, \dots, \mathbf{y}_\tau^T]^T$, our objective is to estimate $\mathbf{A}_{t-1}$, $\mathbf{Q}_{t-1}$, $t = 2, \dots, \tau$, and σ_δ^2 from the data. σ_δ^2 is the variance of measure noise. Because it is just a scalar, we will treat it as a tuning parameter and use a simple line search to select it. This will avoid complicated mathematical estimation. $\mathbf{A}_{t-1}$ and $\mathbf{Q}_{t-1}$ are $n \times n$ matricies that cannot be treated as tuning parameters and need to be estimated from data. Treating the $\mathbf{A}_{t-1}$ and $\mathbf{Q}_{t-1}$ at different time points as different parameters results in a saturated model that is of little use. To tackle this problem, a general principle in SSM is to represent the time-varying parameters as functions of a small set of hyperparameters, $\boldsymbol{\theta}$, i.e., $\mathbf{A}_{t-1} = \mathbf{A}_{t-1}(\boldsymbol{\theta})$ and $\mathbf{Q}_{t-1} = \mathbf{Q}_{t-1}(\boldsymbol{\theta})$. $\mathbf{A}_{t-1}(\cdot)$ and $\mathbf{Q}_{t-1}(\cdot)$ are functions of $\boldsymbol{\theta}$ given by domain knowledge. In this paper, we focus on the functions that are time-invarying, which reduces the parameters to $\mathbf{A}$ and $\mathbf{Q}$. To estimate $\mathbf{A}$ and $\mathbf{Q}$, we adopt the EM framework that treats $\mathbf{y} = [\mathbf{y}_1^T, \dots, \mathbf{y}_\tau^T]^T$ as the observed data and $\mathbf{x} = [\mathbf{x}_1^T, \dots, \mathbf{x}_\tau^T]^T$ as the missing data. EM is an iterative procedure for finding the maximum likelihood estimates of model parameters from data with missing values (Dempster, Laird, and Rubin 1977). It iterates between an E-step that finds the expectation of the complete-data log-likelihood with respect to the missing data given the observed data and the current parameter estimates, and an M-step that finds parameter estimates that maximize the expectation in the E-step. Under the EM framework, the complete-data log-likelihood function of the NSSM is:

$$l(\mathbf{A}, \mathbf{Q}|\mathbf{y}, \mathbf{x}) = log\, p(\mathbf{y}, \mathbf{x}|\mathbf{A}, \mathbf{Q}) = log\, p(\mathbf{x}|\mathbf{A}, \mathbf{Q}) + log\, p(\mathbf{y}|\mathbf{x})$$
$$= \sum_{t=2}^{\tau} \log p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{A}, \mathbf{Q}) + \log p(\mathbf{x}_1) + \sum_{t=1}^{\tau} log\, p(\,\mathbf{y}_t|\mathbf{x}_t). \quad (4.7)$$

Using the state equation and the EP approximation for the observation equation and omitting constants, (4.7) becomes:

$$l(\mathbf{A},\mathbf{Q}|\mathbf{y},\mathbf{x}) = -\frac{(\tau-1)}{2} log|\mathbf{Q}| - \frac{1}{2}\sum_{t=2}^{\tau}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1})^T\mathbf{Q}^{-1}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1}) - \frac{1}{2}\mathbf{x}_1^T\mathbf{\Sigma}^{-1}\mathbf{x}_1 - \frac{1}{2}\sum_{t=1}^{\tau}\mathbf{x}_t{}^T\mathbf{\Pi}_t\ \mathbf{x}_t.$$

In the E-step, we compute the expectation of $l(\mathbf{A},\mathbf{Q}|\mathbf{y},\mathbf{x})$ with respect to $\mathbf{x}$, given $\mathbf{y}$ and the $\mathbf{A}$ and $\mathbf{Q}$ estimated from the previous iteration, $\mathbf{A}^*$and $\mathbf{Q}^*$, i.e.,

$$f(\mathbf{A},\mathbf{Q}) \triangleq \mathrm{E}_{\mathbf{x}|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}\{l(\mathbf{A},\mathbf{Q}|\mathbf{y},\mathbf{x})\}$$

$$= \left\{\begin{matrix} -\frac{(\tau-1)}{2} log|\mathbf{Q}| - \frac{1}{2}\sum_{t=2}^{\tau} tr\left((\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1})^T\mathbf{Q}^{-1}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1})\right) \\ -\frac{1}{2}tr(\mathbf{x}_1^T\mathbf{\Sigma}^{-1}\mathbf{x}_1) - \frac{1}{2}\sum_{t=1}^{\tau} tr\left(\mathbf{x}_t{}^T\mathbf{\Pi}_t\ \mathbf{x}_t\right) \end{matrix}\right\}. \quad (4.8)$$

$tr(\cdot)$ is the trace operator. Using the communicative property of expectation and trace, (4.8) can be further written as:

$$f(\mathbf{A},\mathbf{Q}) = -\frac{(\tau-1)}{2} log|\mathbf{Q}| - \frac{1}{2}\sum_{t=2}^{\tau} tr\left(\mathbf{s}_t\mathbf{Q}^{-1} - \mathbf{A}\mathbf{s}_{t-1,t}\mathbf{Q}^{-1} - \mathbf{s}_{t-1,t}\mathbf{A}^T\mathbf{Q}^{-1} + \mathbf{A}\mathbf{s}_{t-1}\mathbf{A}^T\mathbf{Q}^{-1}\right) - \frac{1}{2}tr(\mathbf{\Sigma}^{-1}\mathbf{s}_1) - \frac{1}{2}\sum_{t=1}^{\tau} tr\left(\mathbf{\Pi}_t\ \mathbf{s}_t\right), \quad (4.9)$$

where $\mathbf{s}_t \triangleq \mathrm{E}_{\mathbf{x}|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t\mathbf{x}_t{}^T)$ and $\mathbf{s}_{t-1,t} \triangleq \mathrm{E}_{\mathbf{x}|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_{t-1}\mathbf{x}_t{}^T)$. In the M-step, we find the $\mathbf{A}$ and $\mathbf{Q}$ that maximize $f(\mathbf{A},\mathbf{Q})$ using the gradient method, i.e.,

$$\widehat{\mathbf{A}} = \left\{\sum_{t=2}^{\tau}\mathbf{s}_{t-1,t}\right\}\left\{\sum_{t=2}^{\tau}\mathbf{s}_{t-1}\right\}^{-1}, \quad (4.10)$$

$$\widehat{\mathbf{Q}} = \frac{1}{(\tau-1)}\sum_{t=2}^{\tau}\left(\mathbf{s}_t - \widehat{\mathbf{A}}\mathbf{s}_{t-1,t} - \mathbf{s}_{t-1,t}\widehat{\mathbf{A}}^T + \widehat{\mathbf{A}}\mathbf{s}_{t-1}\widehat{\mathbf{A}}^T\right). \quad (4.11)$$

The E-step and M-step will be iteratively applied until convergence.

The challenging part in this EM framework is how to compute the expectations, $\mathbf{s}_t$ and $\mathbf{s}_{t-1,t}$, in the E-step. Using the joint posterior distribution of the state vectors at all time points, i.e., $p(\mathbf{x}|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*)$, to derive the expectations is mathematically and

computationally intractable. To tackle this challenge, we develop a BOS algorithm to compute the expectations using recursive equations. A brief introduction to the general concept of BOS is provided in Appendix C. Next, we present the details for the development of a BOS algorithm in our context:

According to the definitions of $\mathbf{s}_t$ and $\mathbf{s}_{t-1,t}$, we get:

$$\mathbf{s}_t \triangleq \mathrm{E}_{\mathbf{x}|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t\mathbf{x}_t{}^T) = \mathrm{E}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t\mathbf{x}_t{}^T)$$

$$= \mathrm{Var}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t) + \mathrm{E}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)\mathrm{E}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)^T, \tag{4.12}$$

$$\mathbf{s}_{t-1,t} \triangleq \mathrm{E}_{\mathbf{x}|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_{t-1}\mathbf{x}_t{}^T)$$

$$= \mathrm{Cov}_{\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_{t-1},\mathbf{x}_t) + \mathrm{E}_{\mathbf{x}_{t-1}|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_{t-1})\mathrm{E}_{\mathbf{x}_t|\mathbf{y}\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)^T. \tag{4.13}$$

It is easy to show that $\mathrm{E}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t) = 0$ because the initial state vector $\mathbf{x}_1$ has a zero mean. Therefore, the key to obtaining $\mathbf{s}_t$ and $\mathbf{s}_{t-1,t}$ is to obtain $\mathrm{Var}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)$ and $\mathrm{Cov}_{\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_{t-1},\mathbf{x}_t)$. We develop an BOS algorithm to compute $\mathrm{Var}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)$ and $\mathrm{Cov}_{\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_{t-1},\mathbf{x}_t)$. The result is presented in Proposition 4.2. The development of the BOS algorithm needs to use the results from a Bayesian optimal prediction (BOP) algorithm and a Bayesian optimal filtering (BOF) algorithm, which are presented in Lemma 1. The detailed derivations for Lemma 4.1 and Proposition 4.2 are given in Appendix C.

<u>Lemma 4.1 (BOP and BOF):</u> Let $\mathbf{P}_t^- \triangleq \mathrm{Var}_{\mathbf{x}_t|\mathbf{y}_{1:t-1},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)$ and $\mathbf{P}_t \triangleq \mathrm{Var}_{\mathbf{x}_t|\mathbf{y}_{1:t},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)$, where $\mathbf{y}_{1:t} = [\mathbf{y}_1^T, \dots, \mathbf{y}_t^T]^T$. Then, the recursive equations for computing $\mathbf{P}_t^-$ and $\mathbf{P}_t$ are given by:

$$\mathbf{P}_t^- = \mathbf{A}^*\mathbf{P}_{t-1}\mathbf{A}^{*T} + \mathbf{Q}^*,$$

$$\mathbf{P}_t = (\mathbf{\Pi}_t + (\mathbf{P}_t^-)^{-1})^{-1},$$

where $\mathbf{\Pi}_t$ is the approximate covariance matrix obtained by EP in Proposition 4.1. The recursions are started from the first time point with $\mathbf{P}_0 = \mathbf{\Sigma}$.

<u>Proposition 4.2 (BOS)</u>: Let $\mathbf{P}_t^S \triangleq \mathrm{Var}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)$ and $\mathbf{P}_{t-1,t}^S \triangleq \mathrm{Cov}_{\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_{t-1},\mathbf{x}_t)$. The backward recursive equations for computing $\mathbf{P}_t^S$ and $\mathbf{P}_{t-1,t}^S$ are given by:

$$\mathbf{P}_t^S = \mathbf{P}_t + \mathbf{H}_t\left(\mathbf{P}_{t+1}^S - \mathbf{A}^*\mathbf{P}_t\mathbf{A}^{*T} - \mathbf{Q}^*\right)\mathbf{H}_t^T,$$

$$\mathbf{P}_{t-1,t}^S = \mathbf{P}_t^S\mathbf{H}_{t-1}^T,$$

where $\mathbf{H}_t = \mathbf{P}_t\mathbf{A}^{*T}(\mathbf{A}^*\mathbf{P}_t\mathbf{A}^{*T} + \mathbf{Q}^*)^{-1}$. The backward recursions are started from the last time point τ, with $\mathbf{P}_\tau^S = \mathbf{P}_\tau$.

Finally in this section, we summarize the steps of the aforementioned algorithm, called the EM-BOS algorithm, for estimating the parameters $\mathbf{A}$ and $\mathbf{Q}$ of the NSSM. The algorithm takes network data collected at τ past time points, $\mathbf{y} = [\mathbf{y}_1^T, \dots, \mathbf{y}_\tau^T]^T$, as input.

<u>Step 1</u>. Specify the covariance matrix of the initial state vector, $\mathbf{\Sigma}$, and let $\mathbf{P}_0 = \mathbf{\Sigma}$. Also specify the initial values for $\mathbf{A}$ and $\mathbf{Q}$, i.e., $\mathbf{A}^*$ and $\mathbf{Q}^*$.

<u>Step 2</u>. Approximate the observation equation at each time point t using the EP result in Proposition 4.1 and obtain the approximate inverse covariance matrix $\mathbf{\Pi}_t$, $t = 1, \dots, \tau$.

<u>Step 3</u>. Use the $\mathbf{\Pi}_t$ obtained from Step 2 together with the $\mathbf{A}^*$ and $\mathbf{Q}^*$ in Step 1 to obtain $\mathrm{Var}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)$ and $\mathrm{Cov}_{\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_{t-1},\mathbf{x}_t)$, i.e., $\mathbf{P}_t^S$ and $\mathbf{P}_{t-1,t}^S$, according to the BOS algorithm in Proposition 4.2, , $t = 2, \dots, \tau$.

Step 4. Use the $\mathbf{P}_t^s$ and $\mathbf{P}_{t-1,t}^s$ obtained from Step 3 to compute $\mathbf{s}_t$ and $\mathbf{s}_{t-1,t}$ according to (4.11) and (4.12), respectively, and to further compute $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{Q}}$ according to (4.9) and (4.10).

Step 5. Update the estimates for $\mathbf{A}$ and $\mathbf{Q}$ by setting $\mathbf{A}^* = \widehat{\mathbf{A}}$ and $\mathbf{Q}^* = \widehat{\mathbf{Q}}$, and repeat Steps 3-4 until convergence.

4.5 Change Detection in Dynamic Networks by Integrating NSSM and SPC

Change detection by SPC has two phases: At phase I, a monitoring statistic is defined and in-control data are used to establish a control chart with control limits for the monitoring statistic. At phase II, new data are collected, for which the monitoring statistics are computed and compared with the control limits established at phase I, and a change is declared if the monitoring statistics of the new data exceed the control limits. The key to successfully applying this general SPC procedure to our specific problem is to define a proper monitoring statistic. When the data are univariate or multivariate time series, a typical approach is to define the monitoring statistic to be a "residual" that reflects the distance between the observed data at time t and the prediction using a time series model. We follow a similar idea and define a distance metric as the monitoring statistic for dynamic network data in the following way:

Once the network at time t is observed, we can estimate the covariance matrix of the state vector based on this network alone, i.e., $\mathrm{Var}_{\mathbf{x}_t|\mathbf{y}_t}(\mathbf{x}_t)$, by EP. Meanwhile, we can obtain a predicted covariance matrix of the state vector using an NSSM that has been developed based on dynamic network data with natural evolution but no change (i.e., the in-control data), i.e., $\mathrm{Var}_{\mathbf{x}_t|\mathbf{y}_{1:t-1},\widehat{\mathbf{A}},\widehat{\mathbf{Q}}}(\mathbf{x}_t)$. The intuition is that if there is no change, the

"distance" between $\mathrm{Var}_{\mathbf{x}_t|\mathbf{y}_{1:t-1},\hat{\mathbf{A}},\hat{\mathbf{Q}}}(\mathbf{x}_t)$ and $\mathrm{Var}_{\mathbf{x}_t|\mathbf{y}_t}(\mathbf{x}_t)$ should be small; a large distance implies an abnormal change. To develop a meaningful distance metric between two covariance matrices, there are a number of options, which were mainly discussed in the SPC literature for covariance monitoring. We choose to apply singular value decomposition (SVD) to each covariance matrix, keep the first singular vector of each matrix, which is the most informative in terms of characterizing the structure of a covariance matrix, and compute the Euclidean distance between the two singular vectors. Specifically, we propose the following distance metric:

$$w_t \triangleq \log \left\| \mathbf{u}_1 \left(\mathrm{Var}_{\mathbf{x}_t|\mathbf{y}_{1:t-1},\hat{\mathbf{A}},\hat{\mathbf{Q}}}(\mathbf{x}_t) \right) - \mathbf{u}_1 \left(\mathrm{Var}_{\mathbf{x}_t|\mathbf{y}_t}(\mathbf{x}_t) \right) \right\|_2^2, \tag{4.14}$$

where $\mathbf{u}_1(\cdot)$ denotes the first singular vector from an SVD on a covariance matrix and $\|\cdot\|_2^2$ denotes the Euclidean distance. The natural logarithm is used to transform the Euclidean distance to be approximate Gaussian. Other transformations could be adopted for Gaussian approximation and a normality check using a QQ plot is recommended to choose a proper transformation. We found that the natural logarithm transformation worked reasonably well in both the simulation studies and the real-data application that will be presented in the next section.

Once a monitoring statistic is defined, the choice of a control chart depends on the magnitude or type of changes that are targeted for detection. For example, CUSUM and EWMA charts are proper choices for detecting small changes, whereas Shewhart charts may be used for detecting greater changes. The focus of this paper is not to design various types of control charts, but instead lays out a general framework for change detection on dynamic network data. Therefore, we focus on a Shewhart chart for the

remaining of the discussion, whereas other types of control charts can be easily "plugged in" this framework.

To establish control limits for a Shewhart chart with the monitoring statistic defined in (14), we can obtain an w_t for each time point during the in-control time period except the first time point, i.e., $t = 2, \dots, \tau$. Then, we can compute the mean and standard deviation over the set of $(w_{1,}, \dots, w_\tau)$, μ_w and σ_w. The upper control limit (UCL) is: $UCL = \mu_w + k\sigma_w$. An lower control limit (LCL) would not be necessary because the monitoring statistic is a distance. k is chosen to satisfy a pre-defined type I error, α. Because the monitoring statistic is approximately Gaussian, $k = \phi^{-1}(1 - \alpha)$. $\phi^{-1}(\cdot)$ is the inverse CDF of the standard normal distribution. This completes the developmental work at phase I. At phase II, i.e., when a new network is observed at time $t + 1$, w_{t+1} can be computed using (14) and compared with the UCL. A change is declared if $w_{t+1} > UCL$.

4.6 Case Studies

4.6.1 Simulation Studies

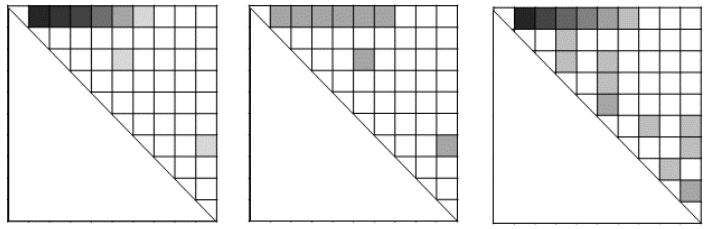
We perform simulation studies to serve three purposes: 1) revealing insights on the NSSM in modeling natural evolution of dynamic networks; 2) assessing the accuracy of the NSSM in modeling natural evolution of dynamic networks; 3) assessing the performance of the change detection method in Section 4.5 in detecting various types of changes. These studies are presented in Sections 4.6.1.1-3, respectively.

4.6.1.1 Insights on the NSSM

We focus on two typical types of natural evolution of dynamic networks: hub forming and community forming. To generate network data that reflect hub forming, we

adopt the following approach: The network includes 10 nodes with v_1 being a hub node. To mimic the evolution process that v_1 is becoming a hub, we start with a network (i.e., the network at $t = 1$) in which v_1 is only connected with one other node, and add one more node to be connected with v_1 in the network at each of five subsequent time points. To mimic the reality that non-hub nodes can also interact with each other, we randomly add two edges between the non-hub nodes at each time point. In this way, we generate networks at six time points. Then, we fit an NSSM on the six networks, based on which we further apply the BOF algorithm in Lemma 1 to estimate the covariance matrix of the state vector at $t = 6$, i.e., $\mathbf{P}_6$. We plot $\mathbf{P}_6^{-1}$ as a color matrix in Figure 7(a). Each row/column corresponds to a node. Only the upper triangular part of the symmetric matrix is shown. The level of darkness reflects the magnitude of the entries in $\mathbf{P}_6^{-1}$. It can be clearly seen that v_1 is a hub node. Furthermore, the varying levels of darkness on the first row reveal the evolution process of how other nodes became connected with v_1, i.e., the darker a node, the earlier it became connected with v_1. For comparison, we estimate $\mathbf{P}_6^{-1}$ by two intuitive methods: One method uses the network at $t = 6$ alone to estimate $\mathbf{P}_6^{-1}$, i.e., no network data in the previous time points are used. The other method estimates each entry in $\mathbf{P}_6^{-1}$ by counting the frequency of occurrence for the edge corresponding to that entry in networks at the six time points. The results by the two methods are shown in Figure 7(b) and (c), respectively. The limitations of the two methods are obvious: Using the network at $t = 6$ alone sheds little light on the evolution process of how other nodes became connected with v_1. The result is only able to show which nodes are connected with the hub but not the time sequence of the connections. The other method that counts the frequency of occurrence makes identification of the hub

node difficult, because it is not able to "forget" the edges randomly appeared between non-hub nodes.

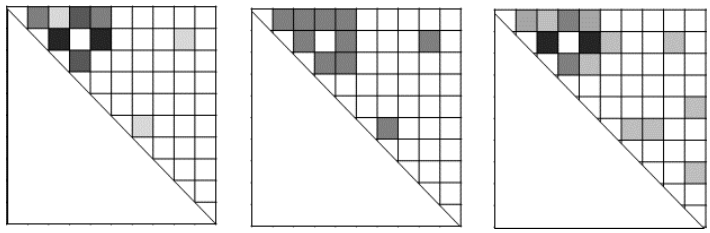


(a) (b) (c)

Figure 7. Estimated $\mathbf{P}_6^{-1}$ to reflect a hub forming process in dynamic networks by (a) BOF under NSSM, (b) a method using the network data at $t = 6$ alone, and (c) a method counting frequency of occurrence of edges.

Furthermore, we generate another set of network data that reflect a community forming process. The network includes 10 nodes with $\{v_1, v_2, v_3, v_4, v_5\}$ being a community. To mimic the evolution process of the community forming, we start with a network in which there are only two edges in the community, and add two more edges in the network at each of three subsequent time points. To mimic the reality that nodes outside the community can also interact with each other and with the nodes in the community, we randomly add two edges not inside the community at each time point. In this way, we generate networks at four time points. Then, we fit an NSSM on the four networks, based on which we further apply the BOF algorithm in Lemma 1 to estimate the covariance matrix of the state vector at $t = 4$, i.e., $\mathbf{P}_4$. We plot $\mathbf{P}_4^{-1}$ as a color matrix in Figure 8(a). For comparison, we also estimate $\mathbf{P}_4^{-1}$ by the two methods used in the previous hub forming experiment, and show the results in Figure 8(b) and (c). We can draw a similar conclusion to the previous experiment that the NSSM is able to capture the evolution process of how the nodes in the community became connected, whereas the time sequence of the connections is lost by the method in Figure 8(b) and the method in

Figure 8(c) cannot distinguish between the connections in a community that are more persistent and other non-community related random connections, making it difficult to identify the community.



(a) (b) (c)

Figure 8. Estimated $\mathbf{P}_4^{-1}$ to reflect a community forming process in dynamic networks by (a) BOF under NSSM, (b) a method using the network data at $t = 4$ alone, and (c) a method counting frequency of occurrence of edges.

4.6.1.2 Accuracy of NSSM

To simulate naturally-evolving (i.e., no-change) dynamic networks, we must find an approach to link the probability that two nodes have an edge at time t, i.e., $p(y_{ij,t} = 1)$, with the existence/non-existence of an edge between the two nodes in all previous time points. We choose to link $p(y_{ij,t} = 1)$ with an exponentially weighted average (EWA) of the edge variables in all previous time points, i.e.,

$$p(y_{ij,t} = 1) = \alpha_0 + \sum_{s=1}^{t-1} \lambda(1-\lambda)^s y_{ij,t-s}. \tag{4.15}$$

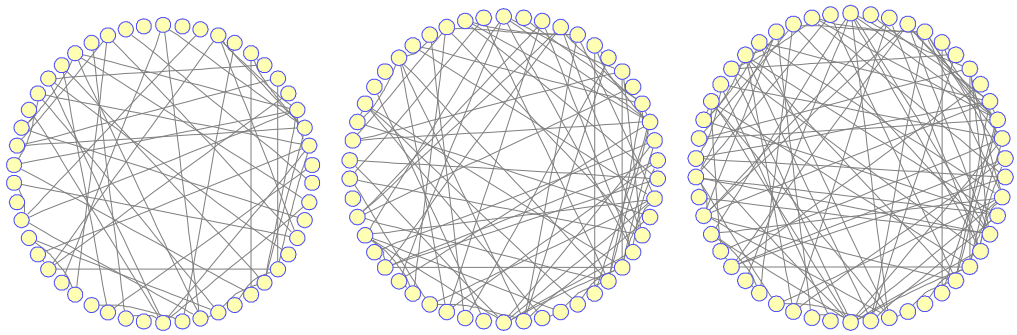
$\lambda \in (0,1)$ is a smoothing parameter. The EWA approach weights the edge variables at different past time points in geometrically decreasing order so that the most recent edge variables are weighted most highly while the most distant edge variables contribute very little. α_0 is a small constant probability for generating "noise" edges. A noise edge in dynamic networks is one that appears at time t even though there is no edge between the

two nodes at any previous time point. There are always noise edges in real-world dynamic networks.

In the first experiment, we simulate dynamic networks of 20 nodes. First, the networks at the first five time points are independently generated with each including 10 randomly selected edges. Then, (4.15) with $\lambda = 0.1$ and $\alpha_0 = 0.1$ is recursively applied to generate dynamic networks at $t = 6, \ldots, 15$. Next, at each time point $t = 6, \ldots, 15$, we fit an NSSM using the networks at all previous time points. Based on the NSSM, we can further obtain a prediction for the probability that two nodes will have an edge at t, i.e.,

$$p\left(y_{ij,t} \middle| \mathbf{y}_{1:t-1}\right) = \frac{1}{2} + \frac{arcsin\rho_{ij}}{\pi}, \tag{4.16}$$

where ρ_{ij} is the entry at the i-th row and j-th column of $\mathbf{P}_t^-$. The derivation for (4.16) is skipped. We use (4.16) for every pair of nodes in the network, compare the predicted probability with the true existence of the edge, and assess the prediction accuracy for the network at t by Area Under the Curve (AUC). For comparison, we also compute the AUC of two completing methods: One method computes the predicted probability using the network at the immediate previous network; the other method uses the frequency of edge occurrence in all the previous networks as an estimate for the predicted probability. Table 8 summarizes the AUC performance of the three methods. In the second experiment, we simulate dynamic networks of a larger size, i.e., with 50 nodes. Figure 9 shows the networks at three time snapshots. The AUC performances of the three methods are shown in Table 9.



$t = 10$ $t = 12$ $t = 14$

Figure 9. Dynamic networks with natural evolution at three time snapshots

In the third experiment, we simulate dynamic networks with two evolving communities, with each community consisting of 20 nodes. The natural evolution within each community is simulated in the same way as the first experiment. Furthermore, we add two random edges between the two communities at each time point to account for between-community interaction. The AUC performances of the three methods are shown in Table 10. It can be seen that the NSSM outperforms the two competing methods in all three experiments.

Table 8. Average (standard deviation) AUC of prediction over the time range of the dynamic networks with 20 nodes

NSSM	Competing method 1 (use the immediate previous network)	Competing method 2 (use the frequency of edge occurrence)
0.86 (0.02)	0.77 (0.03)	0.56 (0.01)

Table 9. Average (standard deviation) AUC of prediction over the time range of the dynamic networks with 50 nodes

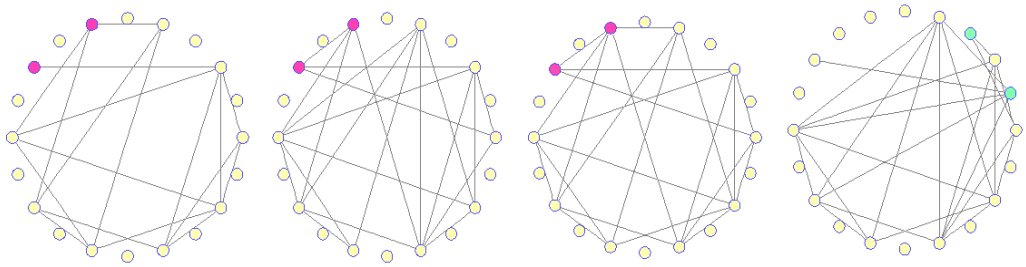
NSSM	Competing method 1 (use the immediate previous network)	Competing method 2 (use the frequency of edge occurrence)
0.84 (0.02)	0.72 (0.02)	0.55 (0.005)

Table 10. Average (standard deviation) AUC of prediction over the time range of the dynamic networks with two communities (20 nodes in each community)

NSSM	Competing method 1 (use the immediate previous network)	Competing method 2 (use the frequency of edge occurrence)
0.84 (0.02)	0.76 (0.03)	0.52 (0.002)

4.6.1.3 Performance of the Change Detection Method

The dynamic networks generated in the first experiment of Section 4.6.1.2 are treated as in-control data, based on which we can obtain the UCL according to the change detection method in Section 4.5. Starting from $t = 16$, the network has a structural shift that $\delta\%$ of the nodes belonging to the community of the in-control networks drop from the community and the same number of new nodes are added into the community. We try $\delta\% = 20\%, 40\%, 60\%, 80\%$. The larger the $\delta\%$, the more dramatic the structural shift. Note that we focus on "structural shifts" that are not easily detected by visual inspection; nor can they be detected by monitoring summary statistics of the networks like the network density. Figure 10 shows the networks at four time points: the first three networks are from the in-control time period; the last network is immediately after a shift with $\delta\% = 20\%$. The shift is one that two new nodes (green nodes at $t = 16$) join the community of the in-control networks, while two old nodes who have been in the community (red nodes at $t = 10,12,14$) are swapped out.



$t = 10$ $t = 12$ $t = 14$ $t = 16$

Figure 10. Dynamic networks with a structural shift at $t = 16$

For each value of $\delta\%$, we compute the monitoring statistic for $t = 16$ using (4.14) and compare it with the UCL. We repeat this experiment for ten times and record the proportion of times that the monitoring statistic at $t = 16$ excceds the UCL. This proportion reflects the probability that the structural shift is successfully detected on the first time point after it happens. We report this probability for type I error $\alpha = 0.005$ and $\alpha = 0.05$ in Table 11. It can be seen that our approach is able to detect the shift with probability one at all shift magnitudes when $\alpha = 0.05$. With a smaller type I error, i.e, $\alpha = 0.005$, the detection probability decreases as the shift magnitude decreases. Finally, we check the validity of the assumption that the monitoring statistic during the in-control time period follows a Gaussian distribution by generating a QQ plot on the monitoring statistics derived from all the in-control networks and performing a Kolmogorov-Smirnov (KS) test. The close-to-straightline pattern in the QQ plot in Figure 11 and the large p-value of the KS test (p=0.97) provides strong evidence that the Gaussian assumption is valid.

Table 11. Probability of detecting the structural shift at the first time point after the shift with $\alpha = 0.005$ ($\alpha = 0.05$)

Structural shift magnitude δ%			
20%	40%	60%	80%
0.7 (1)	0.8 (1)	1 (1)	1 (1)

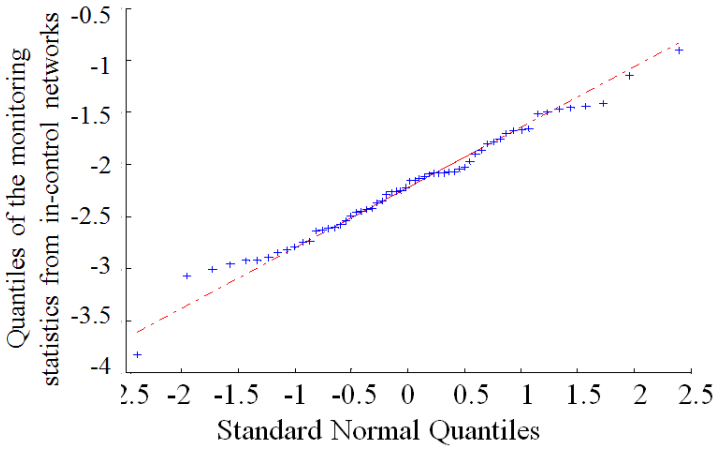


Figure 11. QQ plot on the monitoring statistics derived from in-control networks

4.6.2 Application to Change Detection on Enron Dynamic Email Networks

Enron Corporation was an energy and trading company ranked as the seventh largest in the US in 2000. On 12/1/2001, Enron filed for bankruptcy. This sudden collapse cast suspicions and promoted federal investigation. During the investigation, the courts subpoenaed extensive email logs from most of Enron's employees, and the Federal Energy Regulatory Commission (FERC) published the database online. In this section, we use the dynamic email communication networks between the Enron employees(McCallum, Corrada-Emmanuel, and Wang 2005). We focus on a small subset of the network that consists of 16 employees associated with the Transwestern Pipeline

Division within Enron. The networks with natural evolution, i.e., the in-control data, include monthly email communications between the 16 employees within eight months from 09/2000 to 04/2001.

First, we would like to assess the accuracy of the NSSM in fitting the in-control data. We adopt a similar approach to Section 4.6.1.1. That is, at each of the eight time point, t, we fit an NSSM using the networks at all previous time points. Based on the NSSM, we further obtain a prediction for the probability that two nodes will have an edge at t using (4.16). We use (4.16) for every pair of nodes in the network, compare the predicted probability with the true existence of the edge, and assess the prediction accuracy for the network at t by AUC. The mean and standard deviation of the AUC over the in-control time period are 0.88 and 0.1, respectively. We consider this accuracy to be satisfactory and proceed to use the fitted NSSM to detect changes. To establish a control limit, UCL, we compute the monitoring statistic defined in (4.14) for each time point during the in-control time period except the first time point. The UCL based on these monitoring statistics with a type I error $\alpha = 0.005$ is found to be $UCL = -0.09$.

Furthermore, we extract the network data for two new months, 05/2001 and 10/2001, in which two known changes occurred. The change in 05/2001 was that the CEO of the Division had more communication with the employees although his "in-control" pattern in previous networks was more communication with the CFO and VP of the Division. This change was probably due to the launch of a new initiative or project. Another change was in 10/2001 when the Enron's scandal was revealed and every division of the corporation experienced changes. We compute the monitoring statistic using (4.14) for each new network and compare it with the UCL. Figure 12 shows the

results of change detection for the two new networks. It is clear that the monitoring statistics at the in-control time period all fall below the UCL, while those corresponding to the two changes are far above the UCL. That is, both changes can be successfully detected.

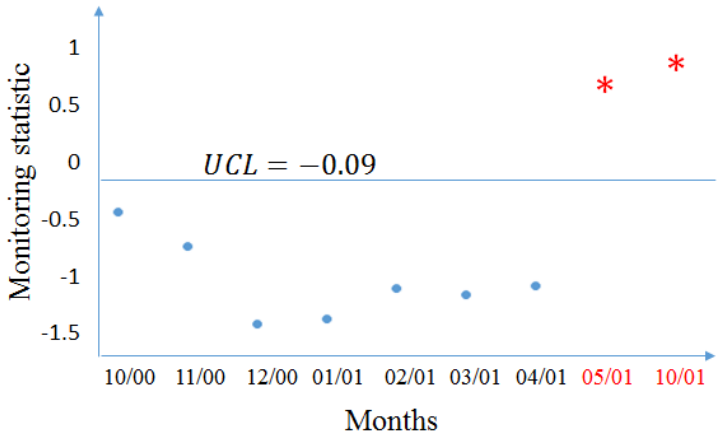


Figure 12. Monitoring and change detection of the Enron dynamic email networks

Conclusion:

In my dissertation research, I developed new transfer learning methods and demonstrated the utility of the methods in real-world applications. For spatial transfer learning across different domains, I developed a predictive model that can flexibly incorporate the data or knowledge of the source domains, whichever available, to the modeling of the target domain. I developed a computationally efficient algorithm in model estimation and performed theoretical analysis. For temporal transfer learning, I developed an NSSM for characterizing the temporal evolution of dynamic network data. I developed an EP algorithm and an EM-BOS algorithm for tractable parameter estimation of the NSSM. Furthermore, I applied the proposed spatial transfer learning approach to modeling of degenerate biological systems, and applied the NSSM to change detection in dynamic social network data.

The research can be extended in several directions:

- For spatial transfer learning, the proposed method was formulated under a Bayesian framework but solved from an optimization point of view to gain efficiency. A Bayesian estimation approach such as empirical Bayes and hierarchical Bayes could allow better characterization of the uncertainty. Second, a similar approach may be developed for predictive modeling of nonlinear relationships. Third, future engineering system design may adopt biological principles like degeneracy in order to be more robust and adaptive to unpredictable environmental situations. By that time, it will be very interesting to study how to migrate the proposed approach to engineering systems.

- For temporal transfer learning or NSSM, Section 4.4.3 discussed several extended NSSM, for which parameter estimation and change detection can be further pursued. We adopted an SVD-based covariance monitoring approach to compare the predicted and estimated state vectors at each time point, which is suitable for detecting structural changes in the dynamic network data. Other covariance monitoring approaches can be adopted to detect other types of changes. Computational efficiency of the parameter estimation could be further improved by taking advantage of modern machine learning developments such as sparse learning.

References

Airoldi, Edoardo M, David M Blei, Stephen E Fienberg, and Eric P Xing. 2009. "Mixed Membership Stochastic Blockmodels." In *Advances in Neural Information Processing Systems*, 33–40.

Alwan, Layth C., and Harry V. Roberts. 1988. "Time-Series Process Modeling for Statistical Control." *Journal of Business Economics and Statistics* 6 (1): 87–95.

Apley, Daniel W, and Jianjun Shi. 1999. "The GLRT for Statistical Process Control of Autocorrelated Processes." *IIE Transactions* 31 (12): 1123–34.

Argyriou, Andreas, Charles Micchelli, Massimiliano Pontil, and Yiming Ying. 2008. "A Spectral Regularization Framework for Multi-Task Structure Learning." *Advances in Neural Information Processing Systems*, 1–8.

Azarnoush, B., K. Paynabar, J. Bekki, and G. C. Runger. 2015. "Monitoring Temporal Homogeneity in Network Streams." *Journal of Quality Technology*.

Bakker, Bart, and Tom Heskes. 2003. "Task Clustering and Gating for Bayesian Multitask Learning." *Journal of Machine Learning Research* 4 (1): 83–99.

Berthouex, P M, W G Hunter, and L Pallesen. 1978. "Monitoring Sewage-Treatment Plants-Some Quality-Control Aspects." *Journal of Quality Technology* 10: 139–49.

Bonilla, Edwin, Kian Ming Chai, and Christopher Williams. 2008. "Multi-Task Gaussian Process Prediction." *Advances in Neural Information Processing Systems* 20: 153–60.

Caruana, Rich. 1997. "Multitask Learning." *Machine Learning* 28 (1): 41–75.

Cevher, Volkan, D Kahle, K Tsianos, and T Saleem. 2008. "Variational Bayes Approximation." *Annals of Mathematical Statistics*, no. 1: 1–7.

Chatterjee, A, and S Lahiri. 2010. "Asymptotic Properties of the Residual Bootstrap for Lasso Estimators." *Proceedings of the American Mathematical Society* 138 (12): 4497–4509.

Chung, F R K. 1999. "Spectral Graph Theory." *American Methematical Society* 92.

Crassidis, John L, and John L Junkins. 2011. *Optimal Estimation of Dynamic Systems*. CRC press.

Dai, Wenyuan, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2007. "Boosting for Transfer Learning." In *Proceedings of the 24th International Conference on Machine Learning*, 193–200.

Dempster, Arthur P, Nan M Laird, and Donald B Rubin. 1977. “Maximum Likelihood from Incomplete Data via the EM Algorithm.” *Journal of the Royal Statistical Society. Series B (methodological)*, 1–38.

Dooley, K J, and S G Kapoor. 1990. “An Enhanced Quality Evaluation System for Continuous Manufacturing Processes, Part 2: Application.” *Journal of Manufacturing Science and Engineering* 112: 63–68.

Dooley, K J, S G Kapoor, M I Dessouky, and R E DeVor. 1986. “An Integrated Quality Systems Approach to Quality and Productivity Improvement in Continuous Manufacturing Processes.” *Journal of Manufacturing Science and Engineering* 108: 322–27.

Edelman, G M, and J a Gally. 2001. “Degeneracy and Complexity in Biological Systems.” *Proceedings of the National Academy of Sciences of the United States of America* 98 (24): 13763–68.

Ernst, Jason, Pouya Kheradpour, Tarjei S Mikkelsen, Noam Shoresh, Lucas D Ward, Charles B Epstein, Xiaolan Zhang, et al. 2011. “Mapping and Analysis of Chromatin State Dynamics in Nine Human Cell Types.” *Nature* 473 (7345): 43–49.

Evgeniou, A, and Massimiliano Pontil. 2007. “Multi-Task Feature Learning.” *Advances in Neural Information Processing Systems* 19: 41.

Evgeniou, Theodoros, and M Pontil. 2004. “Regularized Multi – Task Learning.” *International Conference on Knowledge Discovery and Data Mining*, 109–17.

Genkin, Alexander, David D Lewis, and David Madigan. 2007. “Large-Scale Bayesian Logistic Regression for Text Categorization.” *Technometrics* 49 (3): 291–304.

Globerson, A, G Chechik, F Pereira, N Tishby, M S Handcock, A E Raftery, J M Tantrum, and A Series. 2007. “Euclidean Embedding of Co-Occurrence Data.” *In Advances in Neural Information Processing Systems* 170: 497–504.

Handcock, Mark S., Adrian E. Raftery, and Jeremy M. Tantrum. 2007. “Model-Based Clustering for Social Networks.” *Journal of the Royal Statistical Society. Series A: Statistics in Society* 170 (2): 301–54.

Hanneke, S, W Fu, and E P Xing. 2010. “Discrete Temporal Models of Social Networks.” *Electronic Journal of Statistics*, 585–605.

Ho, Q, L Song, and E P Xing. 2011. “Evolving Cluster Mixed-Membership Blockmodel for Time-Varying Networks.” *Proceedings of the 14th International Conference on Artifical Intelligence and Statistics*, 342–50.

Hoff, P D. 2011. “Hierarchical Multilinear Models for Multiway Data.” *Computational Statistics Data Analysis* 55: 530–43.

Hoff, P D, A E Raftery, and M S Handcock. 2002. “Latent Space Approaches to Social Network Analysis.” *Journal of the American Statistical Association* 97: 1090–98.

Huang, Shuai, Jing Li, Liang Sun, Jieping Ye, Adam Fleisher, Teresa Wu, Kewei Chen, and Eric Reiman. 2010. “Learning Brain Connectivity of Alzheimer’s Disease by Sparse Inverse Covariance Estimation.” *NeuroImage* 50 (3): 935–49.

Huang, Shuai, Jing Li, Jieping Ye, Adam Fleisher, Kewei Chen, Teresa Wu, and Eric Reiman. 2012. “A Sparse Structure Learning Algorithm for Gaussian Bayesian Network Identification from High-Dimensional Data.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (6): 1328–42.

Idier, Jérôme. 2013. *Bayesian Approach to Inverse Problems*. John Wiley & Sons.

Jacob, Laurent, Jean-philippe Vert, Francis R Bach, and Jean-philippe Vert. 2009. “Clustered Multi-Task Learning: A Convex Formulation.” In *Advances in Neural Information Processing Systems*, 745–52.

Jebara, Tony. 2004. “Multi-Task Feature and Kernel Selection for SVMs.” In *Proceedings of the Twenty-First International Conference on Machine Learning*, 55.

Kalman, R.E. 1960. “A New Approach to Linear Filtering and Prediction Problems.” *Journal of Basic Engineering* 82 (1): 35–45.

Kitagawa, Genshiro. 1996. “Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models.” *Journal of Computational and Graphical Statistics* 5 (1): 1–25.

Knight, Keith, and Wenjiang Fu. 2000. “Asymptotics for Lasso-Type Estimators.” *Annals of Statistics* 28 (5): 1356–78.

Lawrence, Dr Neil, and John Platt. 2004. “Learning to Learn with the Informative Vector Machine.” *Twentyfirst International Conference on Machine Learning*, 65.

Lee, N H, and C E Priebe. 2011. “A Latent Process Model for Time Series of Attributed Random Graphs.” *Statistical Inference for Stochastic Processes* 14 (3). Springer: 231–53.

Li, Fan, and Nancy R. Zhang. 2010. “Bayesian Variable Selection in Structured High-Dimensional Covariate Spaces With Applications in Genomics.” *Journal of the American Statistical Association* 105 (491): 1202–14.

Li, Xuejing, Casandra Panea, Chris H Wiggins, Valerie Reinke, and Christina Leslie. 2010. “Learning ‘Graph-Mer’ Motifs That Predict Gene Expression Trajectories in Development.” *PLoS Computational Biology* 6 (4): e1000761.

Liao, Xuejun, Ya Xue, and Lawrence Carin. 2005. “Logistic Regression with an Auxiliary Data Source.” In *International Conference on Machine Learning*, 505–12.

Liu, Jun, Shuiwang Ji, and Jieping Ye. 2009. “Multi-Task Feature Learning Via Efficient L2,1-Norm Minimization.” *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 339–48.

Marchette, D. 2012. “Scan Statistics on Graphs.” *Computational Statistics* 4: 466–73.

McCallum, A, A Corrada-Emmanuel, and X Wang. 2005. “Topic and Role Discovery in Social Networks.” *Computer Science Department Faculty Publication Series* 30: 3.

McCulloh, I, and K M Carley. 2011. “Detecting Change in Longitudinal Social Networks.” *Journal of Social Structure* 12: 1–37.

Miller, Kurt T., Thomas L. Griffiths, and Michael I. Jordan. 2009. “Nonparametric Latent Feature Models for Link Prediction.” *Advances in Neural Information Processing Systems* 10 (1): 1–9.

Minka, Thomas P. 1999. “Expectation Propagation for Approximate Bayesian Inference.” *Statistics* 17 (2): 362–69.

Minka, Tp. 2008. “EP: A Quick Reference.” *Citeseer* 2008 (17): 1–7.

Neil, J, C Hash, A Brugh, M Fisk, and C B Storlie. 2013. “Scan Statistics for the Online Detection of Locally Anomalous Subgraphs.” *Technometrics* 55: 403–14.

Nowicki, Krzysztof, and Tom a. B Snijders. 2001. “Estimation and Prediction for Stochastic Blockstructures.” *Journal of the American Statistical Association* 96 (455): 1077–87.

Park, Y, C E Priebe, A Youssef, and IEEE. 2013. “Anomaly Detection in Time Series of Graphs Using Fusion of Graph Invariants.” *Selected Topics in Signal Processing of* 7: 67–75.

Priebe, C E, J M Conroy, D J Marchette, and Y Park. 2005. “Scan Statistics on Enron Graphs.” *Computational Mathematical Organization Theory* 11: 229–47.

Rückert, Ulrich, and Stefan Kramer. 2008. “Kernel-Based Inductive Transfer.” *Machine Learning and Knowledge Discovery in Databases*, 220–33.

Sarkar, P, A W Moore, A C M, and SIGKDD. 2005. “Dynamic Social Network Analysis Using Latent Space Models.” 7: 31–40.

Schwaighofer, Anton, V. Tresp, and K. Yu. 2005. “Learning Gaussian Process Kernels via Hierarchical Bayes.” *Advances in Neural Information Processing Systems* 17: 1209–16.

Shi, Jianjun. 2006. *Stream of Variation Modeling and Analysis for Multistage Manufacturing Processes*. CRC press.

Shun, Zhenming, and PETER McCULLAGH. 1995. "Laplace Approximation of High Dimensional Integrals." *Journal of the Royal Statistical Society. Series B (Methodological)*. JSTOR, 749–60.

Smith, Richard L. 1998. "Bayesian and Frequentist Approaches to Parametric Predictive Inference." In *Bayesian Statistics*, 589–612.

Tibshirani, Robert. 1994. "Regression Selection and Shrinkage via the Lasso." *Journal of the Royal Statistical Society B*.

Vogelstein, Bert, and Kenneth W Kinzler. 2004. "Cancer Genes and the Pathways They Control." *Nature Medicine* 10 (8): 789–99.

Wang, Hua, Feiping Nie, Heng Huang, Shannon L. Risacher, Andrew J. Saykin, and Li Shen. 2012. "Identifying Disease Sensitive and Quantitative Trait-Relevant Biomarkers from Multidimensional Heterogeneous Imaging Genetics Data via Sparse Multimodal Multitask Learning." *Bioinformatics* 28 (12): 127–36.

Wu, Pengcheng, and Thomas G. Dietterich. 2004. "Improving SVM Accuracy by Training on Auxiliary Data Sources." *Proceedings of the Twenty-First International Conference on Machine Learning*, 110.

Xing, E P, W Fu, and L Song. 2010. "A State-Space Mixed Membership Blockmodel for Dynamic Network Tomography." *The Annals of Applied Statistics* 4: 535–66.

Xu, Kevin S., and Alfred O. Hero. 2014. "Dynamic Stochastic Blockmodels for Time-Evolving Social Networks." *IEEE Journal on Selected Topics in Signal Processing* 8 (4): 552–62.

Xue, Ya, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. 2007. "Multi-Task Learning for Classification with Dirichlet Process Priors." *The Journal of Machine Learning Research* 8: 35–63.

Yang, T, Y Chi, S Zhu, Y Gong, and R Jin. 2011. "Detecting Communities and Their Evolutions in Dynamic Social Networks—A Bayesian Approach." *Machine Learning* 82: 157–89.

Zhang, Yu, and Dit-yan Yeung. 2010. "A Convex Formulation for Learning Task Relationships in Multi-Task Learning." *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence*, 733–442.

APPENDIX A

DERIVATION IN CHAPTER 2

I: Derivation for (2.6).

f(W), according to its definition in Case II, is

$$f(W) = \sum_{k=1}^{K} \|y_k - X_k w_k\|_2^2 + \lambda_1 \|W\|_1 + \lambda_2 \left(Q\log|\Omega| + K\log|\Phi| + tr(\Phi^{-1} W \Omega^{-1} W^T) \right). \tag{A-1}$$

Furthermore, $|\Omega| = |\widetilde{\Omega}| \left(\varsigma_K - \varpi_K^T \widetilde{\Omega}^{-1} \varpi_K \right)$. Then,

$$\begin{aligned} & Q\log|\Omega| + K\log|\Phi| \\ &= Q\log|\widetilde{\Omega}| + Q\log\left(\varsigma_K - \varpi_K^T \widetilde{\Omega}^{-1} \varpi_K\right) + (K-1)\log|\Phi| + \log|\Phi| \\ &= Q\log|\widetilde{\Omega}| + (K-1)\log|\Phi| + \log\left\{ \left(\varsigma_K - \varpi_K^T \widetilde{\Omega}^{-1} \varpi_K\right)^Q |\Phi| \right\} \\ &= Q\log|\widetilde{\Omega}| + (K-1)\log|\Phi| + \log|\Sigma_K|. \end{aligned} \tag{A-2}$$

Σ_K is defined in (2.10).

Also,

$$\begin{aligned} & tr(\Phi^{-1} W \Omega^{-1} W^T) \\ &= tr\left(\Phi^{-1} \left(\widetilde{W}, w_K\right) \begin{bmatrix} \widetilde{\Omega}^{-1} + \frac{\widetilde{\Omega}^{-1} \varpi_K \varpi_K^T \widetilde{\Omega}^{-1}}{\varsigma_K - \varpi_K^T \widetilde{\Omega}^{-1} \varpi_K} & -\frac{\widetilde{\Omega}^{-1} \varpi_K}{\varsigma_K - \varpi_K^T \widetilde{\Omega}^{-1} \varpi_K} \\ -\frac{\varpi_K^T \widetilde{\Omega}^{-1}}{\varsigma_K - \varpi_K^T \widetilde{\Omega}^{-1} \varpi_K} & \frac{1}{\varsigma_K - \varpi_K^T \widetilde{\Omega}^{-1} \varpi_K} \end{bmatrix} \left(\widetilde{W}, w_K\right)^T \right). \end{aligned} \tag{A-3}$$

Expanding the block matrix multiplication within the trace and simplifying the result, we can get:

$$tr(\boldsymbol{\Phi}^{-1} \mathbf{W} \boldsymbol{\Omega}^{-1} \mathbf{W}^T) = tr\left(\boldsymbol{\Phi}^{-1} \widetilde{\mathbf{W}} \widetilde{\boldsymbol{\Omega}}^{-1} \widetilde{\mathbf{W}}^T\right) + (\mathbf{w}_K - \boldsymbol{\mu}_K)^T \boldsymbol{\Sigma}_K^{-1} (\mathbf{w}_K - \boldsymbol{\mu}_K). \tag{A-4}$$

$\boldsymbol{\mu}_K$ is defined in (2.9). Inserting (A-4) and (A-2) into (A-1) and re-organizing the terms, (2.6) can be obtained. Δ

II: Proof of Theorem 2.1.

(2.5) is a convex optimization, which can be solved by a Block Coordinate Descent (BCD) algorithm. We consider two coordinates in our problem, source domains $1, \ldots, \mathrm{K}-1$ as a whole and the target domain, respectively. Then, BCD works by alternately optimizing each coordinate. Specifically, at the n-the iteration, $\mathrm{n} = 1,2,3, \ldots$, BCD solves the following two optimizations:

$$w_K^{(n)} = \underset{w_K}{argmin}\ f\left(\left(\widetilde{W}^{(n-1)}, w_K\right)\right), \tag{A-5}$$

$$\widetilde{W}^{(n)} = \underset{\widetilde{W}}{argmin}\ f\left(\left(\widetilde{W}, w_K^{(n)}\right)\right). \tag{A-6}$$

(A-5) is to optimize the target domain, $\mathrm{w_K}$, treating source domains as fixed by using estimates from the previous iteration, $\widetilde{\mathrm{W}}^{(\mathrm{n}-1)}$. (A-6) then optimizes the source domains, $\widetilde{\mathrm{W}}$, treating the target domain as fixed by using the estimate from (A-5), $\mathrm{w}_\mathrm{K}^{(\mathrm{n})}$.

The objective function in (2.5), i.e., $f(\mathbf{W})$, consists of a non-differentiable term, $\|\mathbf{W}\|_1$. According to the seminal work by Tseng (2001), when a convex objective function includes a non-differentiable term, BCD will converge to the optimal solution if the term is separable according to the coordinates. This is exactly our case, i.e., $\|\mathbf{W}\|_1 = \|\widetilde{\mathbf{W}}\|_1 + \|\mathbf{w}_K\|_1$. Therefore, the BCD in (A-5) and (A-6) will converge to the global optimal solution $\widehat{\mathbf{W}}^{\mathrm{I}}$ (i.e., the solution to (2.5) in Case I). Furthermore, the convergence enjoys a monotone property (Tseng 2001), i.e.,

$$f\left(\left(\widetilde{\boldsymbol{W}}^{(0)}, \boldsymbol{w}_K^{(1)}\right)\right) \geq f\left(\left(\widetilde{\boldsymbol{W}}^{(1)}, \boldsymbol{w}_K^{(1)}\right)\right) \geq f\left(\left(\widetilde{\boldsymbol{W}}^{(1)}, \boldsymbol{w}_K^{(2)}\right)\right) \geq f\left(\left(\widetilde{\boldsymbol{W}}^{(2)}, \boldsymbol{w}_K^{(2)}\right)\right) \geq \cdots \geq f\left(\widehat{\boldsymbol{W}}^{I}\right). \tag{A-7}$$

Let the initial values, $\widetilde{\mathrm{W}}^{(0)}$, be the knowledge of source domains in Case II, i.e., $\widetilde{\mathrm{W}}^{(0)} = \widetilde{\mathrm{W}}^{*}$. Then, (A-7) gives:

$$\mathrm{f}\left(\left(\widetilde{\mathrm{W}}^{*}, \mathrm{w}_{\mathrm{K}}^{(1)}\right)\right) \geq \mathrm{f}\left(\widehat{\mathrm{W}}^{\mathrm{I}}\right). \tag{A-8}$$

Next, according to (A-5), $\mathrm{w}_{\mathrm{K}}^{(1)}$ is

$\mathbf{w}_K^{(1)} = \operatorname{argmin}_{\mathbf{w}_K} f\left(\left(\widetilde{\mathbf{W}}^{(0)}, \mathbf{w}_K\right)\right) = \operatorname{argmin}_{\mathbf{w}_K}\left\{f\left(\widetilde{\mathbf{W}}^{(0)}\right) + g\left(\mathbf{w}_K | \widetilde{\mathbf{W}}^{(0)}\right)\right\} =$

$\operatorname{argmin}_{\mathbf{w}_K} g\left(\mathbf{w}_K | \widetilde{\mathbf{W}}^{(0)}\right)$. The second "=" follows from (2.6). $f\left(\widetilde{\mathbf{W}}^{(0)}\right)$ is dropped in the last equation because it is a constant. Comparing (A-8) and (2.11), we get $\mathbf{w}_K^{(1)} = \widehat{\mathbf{w}}_K^{\mathrm{II}}$. Therefore, (A-8) becomes $f\left(\left(\widetilde{\mathbf{W}}^{*}, \widehat{\mathbf{w}}_K^{\mathrm{II}}\right)\right) \geq f\left(\widehat{\mathbf{W}}^{\mathrm{I}}\right)$. When $\widetilde{\mathbf{W}}^{*} = (\widehat{\mathbf{w}}_1^{\mathrm{I}}, \ldots, \widehat{\mathbf{w}}_{K-1}^{\mathrm{I}})$, it means that BCD attains the optimal solution in one coordinate (the source domains). Then, it must attain the optimal solution in the other coordinate (the target domain), i.e., $\widehat{\mathbf{w}}_K^{\mathrm{II}} = \widehat{\mathbf{W}}^{\mathrm{I}}$. This completes the proof for Theorem 2.1. Δ

III: Proof of Theorem 2.2.

Both (2.12) and (2.13) can be solved analytically, i.e., $\widehat{\mathrm{w}}_{\mathrm{K}} = \left(\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{X}_{\mathrm{K}} + \lambda\mathrm{I}\right)^{-1}\left(\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{y}_{\mathrm{K}} + \lambda\mu_{\mathrm{K}}\right)$ and $\breve{\mathrm{w}}_{\mathrm{K}} = \left(\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{X}_{\mathrm{K}}\right)^{-1}\left(\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{y}_{\mathrm{K}}\right)$ Let $\mathrm{B} \triangleq \left(\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{X}_{\mathrm{K}} + \lambda\mathrm{I}\right)^{-1}$ and $\mathrm{Z} \triangleq \left(\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{X}_{\mathrm{K}} + \lambda\mathrm{I}\right)^{-1}\left(\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{X}_{\mathrm{K}}\right)$. Then, it can be derived that $\mathrm{Z} = \mathrm{I} - \lambda\mathrm{B}$. Using B and Z, we can show that $\widehat{\mathrm{w}}_{\mathrm{K}} = \mathrm{Z}\breve{\mathrm{w}}_{\mathrm{K}} + \lambda\mathrm{B}\mu_{\mathrm{K}}$. Therefore,

$$\mathrm{MSE}\left(\widehat{\mathrm{w}}_{\mathrm{K}}\right) = \mathrm{E}\left\{\left(\widehat{\mathrm{w}}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right)^{\mathrm{T}}\left(\widehat{\mathrm{w}}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right)\right\}$$

$$= \mathrm{E}\left\{\left(\mathrm{Z}\breve{\mathrm{w}}_{\mathrm{K}} + \lambda\mathrm{B}\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right)^{\mathrm{T}}\left(\mathrm{Z}\breve{\mathrm{w}}_{\mathrm{K}} + \lambda\mathrm{B}\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right)\right\}$$

$$= \mathrm{E}\left\{\left(\mathrm{Z}\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{Z}\mathrm{w}_{\mathrm{K}} + \mathrm{Z}\mathrm{w}_{\mathrm{K}} + \lambda\mathrm{B}\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right)^{\mathrm{T}}\left(\mathrm{Z}\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{Z}\mathrm{w}_{\mathrm{K}} + \mathrm{Z}\mathrm{w}_{\mathrm{K}} + \lambda\mathrm{B}\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right)\right\}$$

$$= \mathrm{E}\left\{\left(\left(\mathrm{Z}\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{Z}\mathrm{w}_{\mathrm{K}}\right) + \left(\mathrm{Z}\mathrm{w}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}} + \lambda\mathrm{B}\mu_{\mathrm{K}}\right)\right)^{\mathrm{T}}\left(\left(\mathrm{Z}\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{Z}\mathrm{w}_{\mathrm{K}}\right) + \left(\mathrm{Z}\mathrm{w}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}} + \lambda\mathrm{B}\mu_{\mathrm{K}}\right)\right)\right\}$$

$$= \mathrm{E}\left\{\left(\mathrm{Z}\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{Zw}_{\mathrm{K}}\right)^{\mathrm{T}}\left(\mathrm{Z}\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{Zw}_{\mathrm{K}}\right)\right\} + (\mathrm{Zw}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}} + \lambda \mathrm{B}\mu_{\mathrm{K}})^{\mathrm{T}}(\mathrm{Zw}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}} + \lambda \mathrm{B}\mu_{\mathrm{K}}). \tag{A-9}$$

In the last equation in (A-9), the cross-product, $2(\mathrm{Zw}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}} + \lambda \mathrm{B}\mu_{\mathrm{K}})^{\mathrm{T}}\mathrm{Z}\,\mathrm{E}\left(\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right)$, is omitted. This is because $\breve{\mathrm{w}}_{\mathrm{K}}$, as an ordinary least squares estimator, is unbiased, and therefore $\mathrm{E}\left(\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right) = 0$. Continuing the derivation in (A-9), we can obtain:

$$\mathrm{MSE}\left(\widehat{\mathrm{w}}_{\mathrm{K}}\right) = \mathrm{E}\left\{\left(\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right)^{\mathrm{T}}\mathrm{Z}^{\mathrm{T}}\mathrm{Z}\left(\breve{\mathrm{w}}_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}\right)\right\} + \lambda^2(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}})^{\mathrm{T}}\mathrm{B}^{\mathrm{T}}\mathrm{B}(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}})$$

$$= \sigma^2\ \mathrm{tr}\left\{\left(\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{X}_{\mathrm{K}}\right)^{-1}\mathrm{Z}^{\mathrm{T}}\mathrm{Z}\right\} + \lambda^2(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}})^{\mathrm{T}}\mathrm{B}^{\mathrm{T}}\mathrm{B}(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}})$$

$$= \sigma^2\ \mathrm{tr}\{\mathrm{B}(\mathrm{I} - \lambda \mathrm{B})\} + \lambda^2(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}})^{\mathrm{T}}\mathrm{B}^{\mathrm{T}}\mathrm{B}(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}})$$

$$= \sigma^2\mathrm{tr}(\mathrm{B}) - \sigma^2\lambda\mathrm{tr}(\mathrm{B}^2) + \lambda^2(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}})^{\mathrm{T}}\mathrm{B}^{\mathrm{T}}\mathrm{B}(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}). \tag{A-10}$$

Perform an eigen-decomposition for $\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{X}_{\mathrm{K}}$, i.e., $\mathrm{X}_{\mathrm{K}}^{\mathrm{T}}\mathrm{X}_{\mathrm{K}} = \mathrm{P}^{\mathrm{T}}\Lambda\mathrm{P}$. Λ is a diagonal matrix of eigenvalues $\gamma_1, \dots, \gamma_{\mathrm{Q}}$. P^{T} consists of corresponding eigenvectors. Then the $\mathrm{tr}(\cdot)$ in (A-10) can be shown to be:

$$\mathrm{tr}(\mathrm{B}) = \sum_{\mathrm{i}=1}^{\mathrm{Q}} \frac{1}{\gamma_{\mathrm{i}} + \lambda} \text{ and } \mathrm{tr}(\mathrm{B}^2) = \sum_{\mathrm{i}=1}^{\mathrm{Q}} \frac{1}{(\gamma_{\mathrm{i}} + \lambda)^2}. \tag{A-11}$$

Furthermore, let $\boldsymbol{\alpha} \triangleq \mathbf{P}(\boldsymbol{\mu}_K - \mathbf{w}_K)$ and denote the elements of $\boldsymbol{\alpha}$ by $\alpha_1, \dots, \alpha_Q$. Then, the last term in (A-10) can be shown to be:

$$(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}})^{\mathrm{T}}\mathrm{B}^{\mathrm{T}}\mathrm{B}(\mu_{\mathrm{K}} - \mathrm{w}_{\mathrm{K}}) = \sum_{\mathrm{i}=1}^{\mathrm{Q}} \frac{\alpha_{\mathrm{i}}^2}{(\gamma_{\mathrm{i}} + \lambda)^2}. \tag{A-12}$$

Inserting (A-12) and (A-11) into (A-10),

$$\mathrm{MSE}\left(\widehat{\mathrm{w}}_{\mathrm{K}}\right) = \sigma^2 \sum_{\mathrm{i}=1}^{\mathrm{Q}} \frac{1}{\gamma_{\mathrm{i}} + \lambda} - \sigma^2\lambda \sum_{\mathrm{i}=1}^{\mathrm{Q}} \frac{1}{(\gamma_{\mathrm{i}} + \lambda)^2} + \lambda^2 \sum_{\mathrm{i}=1}^{\mathrm{Q}} \frac{\alpha_{\mathrm{i}}^2}{(\gamma_{\mathrm{i}} + \lambda)^2}$$

$$= \sigma^2 \sum_{\mathrm{i}=1}^{\mathrm{Q}} \frac{1}{\gamma_{\mathrm{i}} + \lambda} - \sigma^2\lambda \sum_{\mathrm{i}=1}^{\mathrm{Q}} \frac{1}{(\gamma_{\mathrm{i}} + \lambda)^2} + \lambda^2 \sum_{\mathrm{i}=1}^{\mathrm{Q}} \frac{\alpha_{\mathrm{i}}^2}{(\gamma_{\mathrm{i}} + \lambda)^2}$$

$$= \sum_{i=1}^{Q} \frac{\sigma^2\gamma_i+\lambda^2\alpha_i^2}{(\gamma_i+\lambda)^2}. \tag{A-13}$$

When $\lambda = 0$, $\mathrm{MSE}(\widehat{w}_K) = \mathrm{MSE}(\breve{w}_K)$. To show that $\mathrm{MSE}(\widehat{w}_K) < \mathrm{MSE}(\breve{w}_K)$ at some $\lambda > 0$, we only need to show that there exists a λ^* such that $\frac{\partial\,\mathrm{MSE}(\widehat{w}_K)}{\partial\lambda} < 0$ for $0 < \lambda < \lambda^*$. To make $\frac{\partial\,\mathrm{MSE}(\widehat{w}_K)}{\partial\lambda} = 2\sum_{i=1}^{Q} \frac{\gamma_i(\lambda\alpha_i^2-\sigma^2)}{(\gamma_i+\lambda)^3} < 0$, a sufficient condition is to make every term in the summation smaller than zero, i.e., $\lambda < \frac{\sigma^2}{\alpha_i^2}$, or equivalently, $\lambda < \frac{\sigma^2}{\max_i(\alpha_i^2)}$. This proves the existence of $\lambda^* = \frac{\sigma^2}{\max_i(\alpha_i^2)}$ and thereby proves Theorem 1. Δ

IV: Proof of Theorem 2.3.

According to (A-10), for a fixed λ, $\mathrm{MSE}(\widehat{w}_K)$ changes only with respect to $(\mu_K - w_K)^T B^T B(\mu_K - w_K)$. The smaller the $(\mu_K - w_K)^T B^T B(\mu_K - w_K)$, the smaller the $\mathrm{MSE}(\widehat{w}_K)$. According to Definition 1, $(\mu_K - w_K)^T B^T B(\mu_K - w_K)$ is the transfer learning distance $d(\mu_K;\lambda)$. Therefore, the smaller the transfer learning distance, the smaller the $\mathrm{MSE}(\widehat{w}_K)$. This gives

$$MSE\left(\widehat{\mathbf{w}}_K^{(1)};\lambda\right) \le MSE\left(\widehat{\mathbf{w}}_K^{(2)};\lambda\right). \tag{A-14}$$

Let $\lambda^{(1)*} = \mathrm{argmin}_\lambda\, \mathrm{MSE}\left(\widehat{w}_K^{(1)}\right)$ and $\lambda^{(2)*} = \mathrm{argmin}_\lambda\, \mathrm{MSE}\left(\widehat{w}_K^{(2)}\right)$. Then, $\mathrm{MSE}\left(\widehat{w}_K^{(1)};\lambda^{(1)*}\right) \le \mathrm{MSE}\left(\widehat{w}_K^{(1)};\lambda^{(2)*}\right) \le \mathrm{MSE}\left(\widehat{w}_K^{(2)};\lambda^{(2)*}\right)$. The second inequality follows from (A-14). This completes the proof for Theorem 2.3. Δ

V: Obtaining (2.17) by the Gradient method.

Given $\mathbf{w}_K$, the optimization problem in (2.17) with respect to ς_K and $\boldsymbol{\varpi}_K$ is:

$$\min_{\varsigma_K,\boldsymbol{\varpi}_K} \varphi(\varsigma_K,\boldsymbol{\varpi}_K)$$

$$= \min_{\varsigma_K,\boldsymbol{\varpi}_K} \left\{ log\left(\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K\right) + \frac{1}{\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K} (\mathbf{w}_K - \boldsymbol{\mu}_K)^T \boldsymbol{\Phi}^{-1} (\mathbf{w}_K - \boldsymbol{\mu}_K) \right\}$$

Using the gradient method, set the partial derivatives of $\varphi(\varsigma_K,\boldsymbol{\varpi}_K)$ to be zero:

$$\begin{cases} \partial\, \varphi(\varsigma_K,\boldsymbol{\varpi}_K)/\partial \varsigma_K = 0 \\ \partial\, \varphi(\varsigma_K,\boldsymbol{\varpi}_K)/\partial \boldsymbol{\varpi}_K = 0' \end{cases}$$

i.e.,

$$\partial\, \varphi(\varsigma_K,\boldsymbol{\varpi}_K)/\partial \varsigma_K = \frac{Q}{\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K} - \frac{\left(\mathbf{w}_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K\right)^T \boldsymbol{\Phi}^{-1} \left(\mathbf{w}_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K\right)}{\left(\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K\right)^2} = 0, \quad \text{(A-15)}$$

$$\partial \varphi(\varsigma_K,\boldsymbol{\varpi}_K)/\partial \boldsymbol{\varpi}_K =$$

$$-\frac{2Q\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K}{\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K} - \frac{2\widetilde{\boldsymbol{\Omega}}^{-1}\widetilde{\mathbf{W}}^T \boldsymbol{\Phi}^{-1}\left(\mathbf{w}_K - \widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\right)}{\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K} + \frac{2\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\left(\mathbf{w}_K - \widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\right)^T \boldsymbol{\Phi}^{-1}\left(\mathbf{w}_K - \widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\right)}{\left(\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K\right)^2} = 0.$$

(A-16)

From (A-15), we can get:

$$Q\left(\varsigma_K - \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K\right) = \left(\mathbf{w}_K - \widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\right)^T \boldsymbol{\Phi}^{-1}\left(\mathbf{w}_K - \widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\right). \quad \text{(A-17)}$$

Inserting (A-17) into the third term of (A-16) and through some algebra, we can get:

$$\widetilde{\boldsymbol{\Omega}}^{-1}\widetilde{\mathbf{W}}^T \boldsymbol{\Phi}^{-1}\mathbf{w}_K - \widetilde{\boldsymbol{\Omega}}^{-1}\widetilde{\mathbf{W}}^T \boldsymbol{\Phi}^{-1}\widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K = 0. \quad \text{(A-18)}$$

According to (2.15), $\widetilde{\mathbf{W}}^T \boldsymbol{\Phi}^{-1}\widetilde{\mathbf{W}} = Q\widetilde{\boldsymbol{\Omega}}$. Using this in (A-18),

$$\widehat{\boldsymbol{\varpi}}_K = \widetilde{\mathbf{W}}^T \boldsymbol{\Phi}^{-1}\mathbf{w}_K/Q. \quad \text{(A-19)}$$

Furthermore, according to (A-17),

$$\varsigma_K = \frac{1}{Q}\left(\mathbf{w}_K - \widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\right)^T \Phi^{-1}\left(\mathbf{w}_K - \widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K\right) + \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K$$

$$= \frac{1}{Q}\mathbf{w}_K{}^T \Phi^{-1}\mathbf{w}_K - \frac{2}{Q}\mathbf{w}_K{}^T \Phi^{-1}\widetilde{\mathrm{W}}\widetilde{\Omega}^{-1}\varpi_{\mathrm{K}} + \frac{1}{\mathrm{Q}}\varpi_{\mathrm{K}}^{\mathrm{T}}\widetilde{\Omega}^{-1}\widetilde{\mathrm{W}}^{\mathrm{T}}\Phi^{-1}\widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K + \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\varpi}_K$$

$= \frac{1}{Q} \mathrm{w_K}^{\mathrm{T}} \Phi^{-1} \mathrm{w_K} - \frac{2}{\mathrm{Q}} \mathrm{w_K}^{\mathrm{T}} \Phi^{-1} \widetilde{\mathrm{W}} \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K + 2 \boldsymbol{\varpi}_K^T \widetilde{\boldsymbol{\Omega}}^{-1} \boldsymbol{\varpi}_K.$

Using (A-19) in the second term, we can get $\hat{\varsigma}_K = \frac{1}{Q} \mathbf{w}_K{}^T \boldsymbol{\Phi}^{-1} \mathbf{w}_K$.

Finally, in order to prove that the $\widehat{\varpi}_{\mathrm{K}}$ and $\hat{\varsigma}_{\mathrm{K}}$ are optimal solutions for a minimization problem, we will need to show $\begin{cases} \partial\, \varphi^2(\varsigma_{\mathrm{K}}, \varpi_{\mathrm{K}}) / \partial \varsigma_{\mathrm{K}}^2 \,|_{\hat{\varsigma}_{\mathrm{K}}, \widehat{\varpi}_{\mathrm{K}}} \succ 0 \\ \partial\, \varphi^2(\varsigma_{\mathrm{K}}, \varpi_{\mathrm{K}}) / \partial \varpi_{\mathrm{K}}^2 \,|_{\hat{\varsigma}_{\mathrm{K}}, \widehat{\varpi}_{\mathrm{K}}} \succ 0 \end{cases}$. " $\succ$" denotes a matrix being positive definite. It can be derived that:

$\partial\, \varphi^2(\varsigma_{\mathrm{K}}, \varpi_{\mathrm{K}}) / \partial \varsigma_{\mathrm{K}}^2 \,|_{\hat{\varsigma}_{\mathrm{K}}, \widehat{\varpi}_{\mathrm{K}}} = \frac{\mathrm{Q}}{\left(\varsigma_{\mathrm{K}} - \varpi_{\mathrm{K}}^{\mathrm{T}} \widetilde{\Omega}^{-1} \varpi_{\mathrm{K}}\right)^2} \succ 0.$

Furthermore,

$\partial\, \varphi^2(\varsigma_{\mathrm{K}}, \varpi_{\mathrm{K}}) / \partial \varpi_{\mathrm{K}}^2 \,|_{\hat{\varsigma}_{\mathrm{K}}, \widehat{\varpi}_{\mathrm{K}}} = \frac{2\mathrm{Q}}{\left(\varsigma_{\mathrm{K}} - \varpi_{\mathrm{K}}^{\mathrm{T}} \widetilde{\Omega}^{-1} \varpi_{\mathrm{K}}\right)^2} \widetilde{\Omega}^{-1} \varpi_{\mathrm{K}} \varpi_{\mathrm{K}}^{\mathrm{T}} \widetilde{\Omega}^{-1} + \frac{2\mathrm{Q}}{\varsigma_{\mathrm{K}} - \varpi_{\mathrm{K}}^{\mathrm{T}} \widetilde{\Omega}^{-1} \varpi_{\mathrm{K}}} \widetilde{\Omega}^{-1},$

where $\widetilde{\Omega}^{-1} \varpi_{\mathrm{K}} \varpi_{\mathrm{K}}^{\mathrm{T}} \widetilde{\Omega}^{-1} \succ 0$ and $\widetilde{\Omega}^{-1} \succ 0$. So $\partial\, \varphi^2(\varsigma_{\mathrm{K}}, \varpi_{\mathrm{K}}) / \partial \varpi_{\mathrm{K}}^2 \,|_{\hat{\varsigma}_{\mathrm{K}}, \widehat{\varpi}_{\mathrm{K}}} \succ 0.$ Δ

APPENDIX B
DERIVATION IN CHAPTER 3

To prove Lemma 3.1 is to prove $(\mathbf{w}_K - \boldsymbol{\mu}_K)^T \mathbf{L}(\mathbf{w}_K - \boldsymbol{\mu}_K) = \sum_{X_i \sim X_j} a_{ij} \left((w_{iK} - \mu_{iK}) - \left(w_{jK} - \mu_{jK}\right)\right)^2$. Start from the left-hand side. Write $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix of the nodes' degrees, i.e., $\mathbf{D} = diag\left(d_1, \dots, d_Q\right)$. $\mathbf{A}$ is matrix of the edge weights, i.e., $\mathbf{A} = \left\{a_{ij}\right\}$. The diagonal elements of $\mathbf{A}$ are zero. Then,

$$(\mathbf{w}_K - \boldsymbol{\mu}_K)^T \mathbf{L}(\mathbf{w}_K - \boldsymbol{\mu}_K)$$

$$= (\mathbf{w}_K - \boldsymbol{\mu}_K)^T \mathbf{D}(\mathbf{w}_K - \boldsymbol{\mu}_K) - (\mathbf{w}_K - \boldsymbol{\mu}_K)^T \mathbf{A}(\mathbf{w}_K - \boldsymbol{\mu}_K) = \sum_{i=1}^{Q} (w_{iK} - \mu_{iK})^2 d_i - \sum_{X_i \sim X_j} (w_{iK} - \mu_{iK}) \left(w_{jK} - \mu_{jK}\right) a_{ij}. \quad \text{(B-1)}$$

Plugging in the definition that $d_i = \sum_{X_i \sim X_j} a_{ij}$, (B-1) becomes

$$(\mathbf{w}_K - \boldsymbol{\mu}_K)^T \mathbf{L}(\mathbf{w}_K - \boldsymbol{\mu}_K)$$

$$= \sum_{i=1}^{Q} (w_{iK} - \mu_{iK})^2 \sum_{X_i \sim X_j} a_{ij} - \sum_{X_i \sim X_j} (w_{iK} - \mu_{iK}) \left(w_{jK} - \mu_{jK}\right) a_{ij}$$

$$= \sum_{i=1}^{Q} \sum_{X_i \sim X_j} (w_{iK} - \mu_{iK})^2 a_{ij} - \sum_{X_i \sim X_j} (w_{iK} - \mu_{iK}) \left(w_{jK} - \mu_{jK}\right) a_{ij}$$

$$= \frac{1}{2}\left(\sum_{X_i \sim X_j} (w_{iK} - \mu_{iK})^2 a_{ij} + \sum_{X_j \sim X_i} \left(w_{jK} - \mu_{jK}\right)^2 a_{ji}\right) - \sum_{X_i \sim X_j} (w_{iK} - \mu_{iK}) \left(w_{jK} - \mu_{jK}\right) a_{ij}. \quad \text{(B-2)}$$

Because the graph is unidirectional, $a_{ji} = a_{ij}$, (B-2) becomes

$$(\mathbf{w}_K - \boldsymbol{\mu}_K)^T \mathbf{L}(\mathbf{w}_K - \boldsymbol{\mu}_K)$$

$$= \frac{1}{2}\left(\sum_{X_i \sim X_j} (w_{iK} - \mu_{iK})^2 a_{ij} + \sum_{X_i \sim X_j} \left(w_{jK} - \mu_{jK}\right)^2 a_{ij}\right) - \sum_{X_i \sim X_j} (w_{iK} - \mu_{iK}) \left(w_{jK} - \mu_{jK}\right) a_{ij}$$

$$= \frac{1}{2}\left(\sum_{X_i \sim X_j} (w_{iK} - \mu_{iK})^2 a_{ij} + \sum_{X_i \sim X_j} \left(w_{jK} - \mu_{jK}\right)^2 a_{ij} - 2 \sum_{X_i \sim X_j} (w_{iK} - \mu_{iK}) \left(w_{jK} - \mu_{jK}\right) a_{ij}\right)$$

$$= \frac{1}{2}\left(\sum_{X_i \sim X_j} a_{ij}\left((w_{iK} - \mu_{iK}) - (w_{jK} - \mu_{jK})\right)^2\right).$$

The $1/2$ can be absorbed by λ_2.

Next, we prove Theorem 3.1. Denote the objective function in (3.5) by $\psi(\mathbf{w}_K)$, i.e.,

$$\psi(\mathbf{w}_K) = \|\mathbf{y}_K - \mathbf{X}_K\mathbf{w}_K\|_2^2 + \lambda_1\|\mathbf{w}_K\|_1 + \lambda_2 \sum_{X_l \sim X_h} a_{lh}\left((w_{lK} - \mu_{lK}) - (w_{hK} - \mu_{hK})\right)^2.$$

Because $\widehat{w}_{ik}^{VI}$ and $\widehat{w}_{jk}^{VI}$ are solutions to the optimization problem in (3.5) and they are non-zero, they should satisfy: $\left.\frac{\partial f(\mathbf{w}_K)}{\partial w_{iK}}\right|_{\widehat{\mathbf{w}}_K^{VI}} = 0$ and $\left.\frac{\partial f(\mathbf{w}_K)}{\partial w_{jK}}\right|_{\widehat{\mathbf{w}}_K^{VI}} = 0$, i.e.,

$$\left.\frac{\partial\|\mathbf{y}_K - \mathbf{X}_K\mathbf{w}_K\|_2^2}{\partial w_{iK}}\right|_{\widehat{\mathbf{w}}_K^{\mathrm{VI}}} + \lambda_1 sgn(\widehat{w}_{ik}^{VI}) + 2\lambda_2 \sum_{X_i \sim X_h} a_{ih}\left((\widehat{w}_{ik}^{VI} - \mu_{iK}) - (\widehat{w}_{hk}^{VI} - \mu_{hK})\right) = 0, \tag{B-3}$$

$$\left.\frac{\partial\|\mathbf{y}_K - \mathbf{X}_K\mathbf{w}_K\|_2^2}{\partial w_{jK}}\right|_{\widehat{\mathbf{w}}_K^{\mathrm{II}}} + \lambda_1 sgn(\widehat{w}_{jk}^{VI}) - 2\lambda_2 \sum_{X_l \sim X_j} a_{lj}\left((\widehat{w}_{lk}^{VI} - \mu_{lK}) - (\widehat{w}_{jk}^{VI} - \mu_{jK})\right) = 0. \tag{B-4}$$

Focusing on (B-3), the third term on the left-hand side can be written into:

$$2\lambda_2 \sum_{X_i \sim X_h} a_{ih}\left((\widehat{w}_{ik}^{VI} - \mu_{iK}) - (\widehat{w}_{hk}^{VI} - \mu_{hK})\right)$$

$$= 2\lambda_2 a_{ij}\left((\widehat{w}_{ik}^{VI} - \mu_{iK}) - (\widehat{w}_{jk}^{VI} - \mu_{jK})\right) + 2\lambda_2 \sum_{X_i \sim X_h, h \neq j} a_{ih}\left((\widehat{w}_{ik}^{VI} - \mu_{iK}) - (\widehat{w}_{hk}^{VI} - \mu_{hK})\right)$$

$$\approx 2\lambda_2 a_{ij}\left((\widehat{w}_{ik}^{VI} - \mu_{iK}) - (\widehat{w}_{jk}^{VI} - \mu_{jK})\right). \tag{B-5}$$

The last step follows from the given assumption that $a_{ij} \gg a_{ih}$. Similarly, the third term on the left-hand side of (B-4) can be written into

$$2\lambda_2 \sum_{X_l \sim X_j} a_{lj}\left((\widehat{w}_{lk}^{VI} - \mu_{lK}) - (\widehat{w}_{jk}^{VI} - \mu_{jK})\right) = 2\lambda_2 a_{ij}\left((\widehat{w}_{ik}^{VI} - \mu_{iK}) - (\widehat{w}_{jk}^{VI} - \mu_{jK})\right).$$

(B-6)

Considering (B-5) and (B-6) and taking the difference between (B-3) and (B-4), $\lambda_1 sgn(\widehat{w}_{ik}^{VI})$ cancels with $\lambda_1 sgn(\widehat{w}_{jk}^{VI})$ because it is known that $\widehat{w}_{ik}^{VI}\widehat{w}_{jk}^{VI} > 0$, and we get:

$$\left.\frac{\partial\|\mathbf{y}_K-\mathbf{X}_K\mathbf{w}_K\|_2^2}{\partial w_{iK}}\right|_{\widehat{\mathbf{w}}_K^{VI}} - \left.\frac{\partial\|\mathbf{y}_K-\mathbf{X}_K\mathbf{w}_K\|_2^2}{\partial w_{jK}}\right|_{\widehat{\mathbf{w}}_K^{VI}} + 4\lambda_2 a_{ij}\left((\widehat{w}_{ik}^{VI}-\mu_{iK})-(\widehat{w}_{jk}^{VI}-\mu_{jK})\right) = 0.$$

(B-7)

$$-2\left(\boldsymbol{x}_{iK}^T-\boldsymbol{x}_{jK}^T\right)(\mathbf{y}_K-\mathbf{X}_K\widehat{\mathbf{w}}_K^{VI}) + 4\lambda_2 a_{ij}\left((\widehat{w}_{ik}^{VI}-\mu_{iK})-(\widehat{w}_{jk}^{VI}-\mu_{jK})\right) = 0. \qquad \text{(B-8)}$$

Furthermore, we can get:

$$\left|(\widehat{w}_{ik}^{VI}-\mu_{iK})-(\widehat{w}_{jk}^{VI}-\mu_{jK})\right| \le \frac{\|\boldsymbol{x}_{iK}-\boldsymbol{x}_{jK}\|_2 \times\|\mathbf{y}_K-\mathbf{X}_K\widehat{\mathbf{w}}_K^{VI}\|_2}{2\lambda_2}\frac{1}{a_{ij}}. \qquad \text{(B-9)}$$

We would like to have an upper bound that does not include $\widehat{\mathbf{w}}_K^{VI}$. To achieve this, we adopt the following strategy: Because $\widehat{\mathbf{w}}_K^{VI}$ is the optimal solution, $\psi(\widehat{\mathbf{w}}_K^{VI})$ should be the smallest. Therefore, $\psi(\widehat{\mathbf{w}}_K^{VI}) \le \psi(0)$, i.e.,

$$\psi(\widehat{\mathbf{w}}_K^{VI}) = \|\mathbf{y}_K-\mathbf{X}_K\widehat{\mathbf{w}}_K^{VI}\|_2^2 + \lambda_1\|\widehat{\mathbf{w}}_K^{VI}\|_1 + \lambda_2\sum_{X_l\sim X_h} a_{lh}\left((\widehat{w}_{lk}^{VI}-\mu_{lK})-(\widehat{w}_{hk}^{VI}-\mu_{hK})\right)^2 \le f(0) = \|\mathbf{y}_K\|_2^2 + \lambda_2\sum_{X_l\sim X_h} a_{lh}(\mu_{lK}-\mu_{hK})^2 . \qquad \text{(B-10)}$$

Then,

$\|\mathbf{y}_K-\mathbf{X}_K\widehat{\mathbf{w}}_K^{VI}\|_2^2 \le \|\mathbf{y}_K\|_2^2 + \lambda_2\sum_{X_l\sim X_h} a_{lh}(\mu_{lK}-\mu_{hK})^2 \approx \|\mathbf{y}_K\|_2^2 + 2\lambda_2 a_{ij}(\mu_{iK}-\mu_{jK})^2$, and

$$\|\mathbf{y}_K-\mathbf{X}_K\widehat{\mathbf{w}}_K^{VI}\|_2^2 \le \sqrt{\|\mathbf{y}_K\|_2^2 + 2\lambda_2 a_{ij}(\mu_{iK}-\mu_{jK})^2} . \qquad \text{(B-11)}$$

Inserting (B-11) into (B-9), we get

$$\left|\left(\widehat{w}_{ik}^{VI}-\mu_{iK}\right)-\left(\widehat{w}_{jk}^{VI}-\mu_{jK}\right)\right| \leq \frac{\left\|\mathbf{x}_{iK}-\mathbf{x}_{jK}\right\|_2}{2\lambda_2} \times \sqrt{\frac{\|\mathbf{y}_K\|_2^2}{a_{ij}^2}+\frac{2\lambda_2\left(\mu_{iK}-\mu_{jK}\right)^2}{a_{ij}}}. \qquad \Delta$$

APPENDIX C

DERIVATION IN CHAPTER 4

I. Introduction to EP

Expectation Propagation (EP) is a method in approximate Bayesian inference that uses a deterministic algorithm to approximate the true posterior distribution with an exponential-family distribution (T. P. Minka 1999). It is applicable when the true posterior distribution takes the form of a factorization, i.e.,

$$p(\theta|\mathbf{z}) = \frac{1}{p(\mathbf{z})}\prod_{k=0}^{n} f_k(\theta). \tag{C-1}$$

$\mathbf{z}$ is the observational data. θ is the parameter whose distribution is to be infer. $f_0(\theta)$ is the prior. $f_k(\theta) = p(z_k|\theta), k \geq 1$, is the likelihood corresponding to the k -th observation. When $f_k(\theta)$ takes a complicated form, computing the posterior $p(\theta|\mathbf{z})$ is difficult. EP approximates $p(\theta|\mathbf{z})$ with a mathematically tractable distribution $q(\theta) = \frac{1}{C}\prod_{k=0}^{n} \tilde{f}_k(\theta)$, with each $\tilde{f}_k(\theta)$ being a member of an exponential family, so that the resulting $q(\theta)$ belongs to the same exponential family. The approximation by EP is known to be better when the number of factors in the factorization, n, gets larger.

The best approximation $q(\theta)$ found by EP is one that minimizes the Kullback–Leibler (KL) divergence $KL(p||q)$ (T. Minka 2008). To find the $q(\theta)$, a plausible algorithm would be to approximate each factor in $p(\theta|\mathbf{z})$ using one factor in $q(\theta)$ by matching the moments between $f_k(\theta)$ and $\tilde{f}_k(\theta)$. However, this algorithm limits itself to a small subset of feasible solutions, i.e., it eliminates many candidate solutions that effectively minimize $KL(p||q)$, but for which the individual moments of f do not match those of $\tilde{f}$. This requires a more sophisticated algorithm than simply matching moments factor-by-factor, which leads to EP. Specifically, at each iteration of EP, we pick some l, and include all the factors except $f_l(\theta)$, and then match moments for the distributions

with $f_l(\theta)$ and $\tilde{f}_l(\theta)$ omitted. Because the moment matching includes a large number of factors (all factors except one), a better approximation is guaranteed. EP iterates with one factor omitted at each iteration until convergence.

II. Introduction to Bayesian optimal filtering and smoothing

Bayesian optimal filtering and smoothing refers to the methodology that can be used for estimating the states $(\mathbf{x}_1, \cdots, \mathbf{x}_\tau)$ of a time-varying system, which are indirectly observed through noisy measurements $(\mathbf{y}_1, \cdots, \mathbf{y}_\tau)$ (Kitagawa 1996; Crassidis and Junkins 2011). It computes three marginal posterior distributions to avoid inefficient and unnecessary computing of the joint posterior distribution of the states at all time points, $p(\mathbf{x}_1, \cdots, \mathbf{x}_\tau | \mathbf{y}_1, \cdots, \mathbf{y}_\tau)$. The three marginal posterior distributions are: 1) the filtering distribution, i.e., the marginal posterior distribution of the current state given the previous measurements, $p(\mathbf{x}_t | \mathbf{y}_1, \cdots, \mathbf{y}_t), t = 1, \ldots, \tau$; 2) the prediction distribution, i.e., the posterior marginal distribution of a future state given the previous measurements, $p(\mathbf{x}_{t+n} | \mathbf{y}_1, \cdots, \mathbf{y}_t)$, $n \geq 1$; and 3) the smoothing distribution, i.e., the posterior marginal distribution of a state given a certain interval of measurements, i.e., $p(\mathbf{x}_k | \mathbf{y}_1, \cdots, \mathbf{y}_t)$, $k < t$. Bayesian optimal filtering and smoothing uses recursive equations to compute the three marginal posterior distributions under the assumption that the states at successive time points have a Markovian property.

III. Proof of Proposition 4.1

Our use of EP in this paper has a small twist from its original purpose as illustrated in (I). We use EP to find an approximate Gaussian distribution of the state vector $\mathbf{x}_t$ based on the observation equation $p(\mathbf{y}_t | \mathbf{x}_t)$ in (4*). Because the same EP algorithm is applied to every time point, we omit the subscript "t" in the rest of this proof

for notation simplicity. Mapping to the generic notations of EP defined in (C-1), this is to consider the state vector $\mathbf{x}$ as the "θ", the network $\mathbf{y}$ as the "$\mathbf{z}$", and "$f_0(\theta)$" and "$p(\mathbf{z})$" being constants. Then, the EP in our case is to find a distribution $q(\mathbf{x}) = \prod_{ij} \tilde{f}(\mathbf{x}_{ij})$ to approximate $p(\mathbf{y}_t|\mathbf{x}_t) = \prod_{ij} f(\mathbf{x}_{ij})$, where $\mathbf{x}_{ij} = [x_i, x_j]^{\mathrm{T}}$, $f(\mathbf{x}_{ij})$ is the edge-specific observation equation given in (4), and $\tilde{f}(\mathbf{x}_{ij})$ is zero-mean Gaussian, i.e., $\tilde{f}(\mathbf{x}_{ij}) = s_{ij} exp(-\frac{1}{2}\mathbf{x}_{ij}^T \boldsymbol{\pi}_{ij} \mathbf{x}_{ij})$. Consequently, $q(\mathbf{x})$ is zero-mean Gaussian, i.e., $q(\mathbf{x}) = C\ exp(-\frac{1}{2}\mathbf{x}^T \boldsymbol{\Pi} \mathbf{x})$, where $\boldsymbol{\Pi} = \sum_{\boldsymbol{ij}} \boldsymbol{\Pi}_{ij}$ and $\boldsymbol{\Pi}_{ij}$ is an $n \times n$ matrix with four non-zero entries augmented from the 2×2 matrix $\boldsymbol{\pi}_{ij}$. The goal of EP is to find $\boldsymbol{\Pi}$**,** or equivalently, $\mathbf{V} = \boldsymbol{\Pi}^{-1}$.

Before deriving the EP, we define some additional notations and relations that will be used throughout the derivation. Let $\mathbf{v}_{ij} = \boldsymbol{\pi}_{ij}^{-1}$ and $\mathbf{V}_{ij}$ be an $n \times n$ matrix with four non-zero entries augmented from the 2×2 matrix $\mathbf{v}_{ij}$. It is easy to show that the following relations between $\mathbf{v}_{ij}$ and $\mathbf{V}_{ij}$ hold:

$$\mathbf{V}_{ij}^{-1} = \mathbf{d}_{ij} \mathbf{v}_{ij}^{-1} \mathbf{d}_{ij}^T, \tag{C-2}$$

$$\mathbf{v}_{ij} = \mathbf{d}_{ij}^T \mathbf{V}_{ij} \mathbf{d}_{ij}, \tag{C-3}$$

where $\mathbf{d}_{ij}$ is a $n \times 2$ vector whose first column has one at the i-th position and zeros otherwise, and second column has one at the j-th position and zeros otherwise.

At each iteration of EP, one factor is omitted from $q(\mathbf{x})$. Let $q^{\backslash ij}(\mathbf{x})$ denote the distribution where factor $\tilde{f}(\mathbf{x}_{ij})$ is omitted, i.e., $q^{\backslash ij}(\mathbf{x}) = \frac{q(\mathbf{x})}{\tilde{f}(\mathbf{x}_{ij})}$. The covariance matrix of $q^{\backslash ij}(\mathbf{x})$ can be derived as follows:

$$\mathbf{V}^{\backslash ij} = \left((\mathbf{V}^*)^{-1} - \mathbf{V}_{ij}^{-1}\right)^{-1} = \mathbf{V}^* + \left(\mathbf{V}^* \mathbf{d}_{ij}\right)\left(\mathbf{v}_{ij} - \mathbf{d}_{ij}^T \mathbf{V}^* \mathbf{d}_{ij}\right)^{-1}\left(\mathbf{d}_{ij}^T \mathbf{V}^*\right),$$

where $\mathbf{V}^*$ is the approximation for $\mathbf{V}$ obtained in the previous iteration. Using (C-3), we can get:

$$\mathbf{v}^{\backslash ij} = \mathbf{d}_{ij}^T \mathbf{V}^{\backslash ij} \mathbf{d}_{ij}.$$

Next, we define $q^*(\mathbf{x})$ to be the distribution that combines $q^{\backslash ij}(\mathbf{x})$ and the true factor $f(\mathbf{x}_{ij})$, i.e., $q^*(\mathbf{x}) = \frac{1}{Z_{ij}} q^{\backslash ij}(\mathbf{x}) f(\mathbf{x}_{ij})$. Z_{ij} is a normalizing constant. Then, we can obtain an updated $q(\mathbf{x})$, $q^{new}(\mathbf{x})$, by minimizing the KL divergence between $q^*(\mathbf{x})$ and $q(\mathbf{x})$, i.e.,

$$q^{new}(\mathbf{x}) = argmin\ KL\big(q^*(\mathbf{x}) || q(\mathbf{x})\big).$$

Since we know that $q^{new}(\mathbf{x})$ is zero-mean Gaussian, the key is to find the covariance matrix of $q^{new}(\mathbf{x})$, $\mathbf{V}^{new}$. $\mathbf{V}^{new}$ is related to the normalizing constant Z_{ij} through (C-4):

$$\mathbf{V}^{new} = \mathbf{V}^{\backslash ij} - \mathbf{V}^{\backslash ij} \mathbf{d}_{ij} (-2\nabla_{\mathbf{v}^{\backslash ij}} logZ_{ij}) \mathbf{d}_{ij}^T \mathbf{V}^{\backslash ij}\ , \qquad \text{(C-4)}$$

where $\nabla_{\mathbf{v}^{\backslash ij}} logZ_{ij}$ denotes the derivative of $logZ_{ij}$ with respect to $\mathbf{v}^{\backslash ij}$. This means that the EP is complete as long as we can derive $\nabla_{\mathbf{v}^{\backslash ij}} logZ_{ij}$. The rest of this section is devoted to the derivation for $\nabla_{\mathbf{v}^{\backslash ij}} logZ_{ij}$:

According to the definition of Z_{ij},

$$Z_{ij} = \int f(\mathbf{x}_{ij}) q^{\backslash ij}(\mathbf{x}) d\mathbf{x}$$

$$= \int \left\{ \phi\left(\frac{x_i}{\sigma_\delta}\right) \phi\left(\frac{x_j}{\sigma_\delta}\right) + \left(1 - \phi\left(\frac{x_i}{\sigma_\delta}\right)\right)\left(1 - \phi\left(\frac{x_j}{\sigma_\delta}\right)\right) \right\} \times N\big(\mathbf{x} | \mathbf{0}, \mathbf{V}^{\backslash ij}\big) d\mathbf{x}$$

$$= \int \left\{ \phi\left(\frac{x_i}{\sigma_\delta}\right) \phi\left(\frac{x_j}{\sigma_\delta}\right) + \left(1 - \phi\left(\frac{x_i}{\sigma_\delta}\right)\right)\left(1 - \phi\left(\frac{x_j}{\sigma_\delta}\right)\right) \right\} \times N\big(\mathbf{x}_{ij} | \mathbf{0}, \mathbf{v}^{\backslash ij}\big) d\mathbf{x}_{ij}$$

$=$

$$\int\left\{\int_{-\infty}^{\frac{x_i}{\sigma_\delta}} N(v|0,1)dv \int_{-\infty}^{\frac{x_j}{\sigma_\delta}} N(u|0,1)du + \left(1-\int_{-\infty}^{\frac{x_i}{\sigma_\delta}} N(v|0,1)dv\right)\left(1-\int_{-\infty}^{\frac{x_j}{\sigma_\delta}} N(u|0,1)du\right)\right\} \times N\left(\mathbf{x}_{ij}|\mathbf{0},\mathbf{v}^{\backslash ij}\right)d\mathbf{x}_{ij}$$

$$= \int\left\{1+2\int_{-\infty}^{\frac{x_i}{\sigma_\delta}} N(v|0,1)dv \int_{-\infty}^{\frac{x_j}{\sigma_\delta}} N(u|0,1)du - \int_{-\infty}^{\frac{x_i}{\sigma_\delta}} N(v|0,1)dv - \int_{-\infty}^{\frac{x_j}{\sigma_\delta}} N(u|0,1)du\right\} \times N\left(\mathbf{x}_{ij}|\mathbf{0},\mathbf{v}^{\backslash ij}\right)d\mathbf{x}_{ij}$$

$$= \int\left\{1+2\int_{-\infty}^{0} N\left(v\middle|-\frac{x_i}{\sigma_\delta},1\right)dv \int_{-\infty}^{0} N\left(u\middle|-\frac{x_j}{\sigma_\delta},1\right)du - \int_{-\infty}^{0} N\left(v\middle|-\frac{x_i}{\sigma_\delta},1\right)dv - \int_{-\infty}^{0} N\left(u\middle|-\frac{x_j}{\sigma_\delta},1\right)du\right\} \times N\left(\mathbf{x}_{ij}|\mathbf{0},\mathbf{v}^{\backslash ij}\right)d\mathbf{x}_{ij}. \quad \text{(C-5)}$$

Let

$g\left(\mathbf{v}^{\backslash ij}\right) = \int\left\{\int_{-\infty}^{0} N\left(v\middle|-\frac{x_i}{\sigma_\delta},1\right)dv \int_{-\infty}^{0} N\left(u\middle|-\frac{x_j}{\sigma_\delta},1\right)du\right\} \times N\left(\mathbf{x}_{ij}|\mathbf{0},\mathbf{v}^{\backslash ij}\right)d\mathbf{x}_{ij}$,

$l\left(\mathbf{v}^{\backslash ij}\right) = \int\left\{\int_{-\infty}^{0} N\left(v\middle|-\frac{x_i}{\sigma_\delta},1\right)dv\right\} \times N\left(\mathbf{x}_{ij}|\mathbf{0},\mathbf{v}^{\backslash ij}\right)d\mathbf{x}_{ij}$, and

$h\left(\mathbf{v}^{\backslash ij}\right) = \int\left\{\int_{-\infty}^{0} N\left(u\middle|-\frac{x_j}{\sigma_\delta},1\right)du\right\} \times N\left(\mathbf{x}_{ij}|\mathbf{0},\mathbf{v}^{\backslash ij}\right)d\mathbf{x}_{ij}$. Then, (C-5) becomes $Z_{ij} = 1 + 2g(\mathbf{v}^{\backslash ij}) - l\left(\mathbf{v}^{\backslash ij}\right) - h(\mathbf{v}^{\backslash ij})$. It is not difficult to find that $l\left(\mathbf{v}^{\backslash ij}\right) = h\left(\mathbf{v}^{\backslash ij}\right) = 0.5$. Therefore,

$$Z_{ij} = 2g(\mathbf{v}^{\backslash ij}). \quad \text{(C-6)}$$

Next, we derive $g(\mathbf{v}^{\backslash ij})$. Define $\mathbf{w} = [v, u]^T$. Then,

$g\left(\mathbf{v}^{\backslash ij}\right) =$

$$\int\left\{\int_{-\infty}^{0} \frac{1}{2\pi} exp(-\frac{1}{2}\left(\mathbf{x}_{ij}+\mathbf{w}\sigma_\delta\right)^T\left(\sigma_\delta^2\right)^{-1}\left(\mathbf{x}_{ij}+\mathbf{w}\sigma_\delta\right))d\mathbf{w}\right\} \times N\left(\mathbf{x}_{ij}|\mathbf{0},\mathbf{v}^{\backslash ij}\right)d\mathbf{x}_{ij}$$

$$= \sigma_\delta^2 \int_{-\infty}^{0}\int\left\{\frac{1}{2\pi\sigma_\delta^2} exp(-\frac{1}{2}\left(\mathbf{x}_{ij}+\mathbf{w}\sigma_\delta\right)^T\left(\sigma_\delta^2\right)^{-1}\left(\mathbf{x}_{ij}+\mathbf{w}\sigma_\delta\right)) \times N\left(\mathbf{x}_{ij}|\mathbf{0},\mathbf{v}^{\backslash ij}\right)\right\} d\mathbf{x}_{ij}d\mathbf{w}$$

$=$

$\sigma_\delta^2 \int_{-\infty}^{0} \int N(\mathbf{x}_{ij}|-\left(\frac{1}{\sigma_\delta^2}\mathbf{I}+\left(\mathbf{v}^{\backslash ij}\right)^{-1}\right)^{-1}\left(\sigma_\delta^2\right)^{-1}\mathbf{w}\sigma_\delta, \left(\frac{1}{\sigma_\delta^2}\mathbf{I}+\right.$

$\left.\left(\mathbf{v}^{\backslash ij}\right)^{-1}\right)^{-1})N(-\mathbf{w}\sigma_\delta|0, \sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij})\ d\mathbf{x}_{ij}d\mathbf{w}$

$= \sigma_\delta^2 \int_{-\infty}^{0} N(-\mathbf{w}\sigma_\delta|0, \sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij})d\mathbf{w}.$ (C-7)

Inserting (C-7) into (C-6), we further derive $\nabla_{\mathbf{v}^{\backslash ij}} logZ_{ij}$:

$\nabla_{\mathbf{v}^{\backslash ij}} logZ_{ij} = \frac{1}{Z_{ij}}(2\nabla_{\mathbf{v}^{\backslash ij}} g\left(\mathbf{v}^{\backslash ij}\right))$

$= \sigma_\delta^2 \int_{-\infty}^{0} \frac{1}{2\pi\left|\sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij}\right|^{1/2}} exp\left(-\frac{1}{2}\mathbf{w}^T\sigma_\delta(\sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij})^{-1}\sigma_\delta\mathbf{w}\right)\left(-\frac{\sigma_\delta^2}{2}\right)\left(-(\sigma_\delta^2\mathbf{I}+\right.$

$\left.\mathbf{v}^{\backslash ij})^{-1}\mathbf{w}\mathbf{w}^T(\sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij})^{-1}\right) + exp\left(-\frac{1}{2}\mathbf{w}^T\sigma_\delta(\sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij})^{-1}\sigma_\delta\mathbf{w}\right)\frac{1}{2\pi}\left(-\frac{1}{2}\left|\sigma_\delta^2\mathbf{I}+\right.\right.$

$\left.\left.\mathbf{v}^{\backslash ij}\right|^{-3/2}\left|\sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij}\right|(\sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij})^{-1}\right)d\mathbf{w}$

$=$

$\frac{\sigma_\delta^2}{2}\sigma_\delta^2(\sigma_\delta^2\mathbf{I}+$

$\mathbf{v}^{\backslash ij})^{-1}\int_{-\infty}^{0}\frac{1}{2\pi\left|\sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij}\right|^{1/2}} exp\left(-\frac{1}{2}\mathbf{w}^T\sigma_\delta(\sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij})^{-1}\sigma_\delta\mathbf{w}\right)\begin{pmatrix} v^2 & uv \\ uv & u^2\end{pmatrix}d\mathbf{w}(\sigma_\delta^2\mathbf{I}+$

$\mathbf{v}^{\backslash ij})^{-1} - \frac{1}{2}(\sigma_\delta^2\mathbf{I}+\mathbf{v}^{\backslash ij})^{-1}g(\mathbf{v}^{\backslash ij})$ (C-8)

Using (C-8), $\nabla_{\mathbf{v}^{\backslash ij}} logZ_{ij}$ can be obtained numerically. Δ

IV. Proof of Lemma 1

We first derive the prediction distribution $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}, \mathbf{A}^*, \mathbf{Q}^*)$:

$P(\mathbf{x}_t|\mathbf{y}_{1:t-1}, \mathbf{A}^*, \mathbf{Q}^*) = \int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{A}^*, \mathbf{Q}^*)P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}, \mathbf{A}^*, \mathbf{Q}^*)d\mathbf{x}_{t-1}$

$= \int N(\mathbf{x}_t|\mathbf{A}^*\mathbf{x}_{t-1}, \mathbf{Q}^*)N(\mathbf{x}_{t-1}|\mathbf{0}, \mathbf{P}_{t-1})d\mathbf{x}_{t-1}$

$$= \int N\left(((\mathbf{x}_{t-1}{}^T, \mathbf{x}_t{}^T)^T | \mathbf{y}_{1:t-1}, \mathbf{A}^*, \mathbf{Q}^*) \middle| \mathbf{0}, \begin{pmatrix} \mathbf{P}_{t-1} & \mathbf{P}_{t-1}\mathbf{A}^{*T} \\ \mathbf{A}^*\mathbf{P}_{t-1} & \mathbf{A}^*\mathbf{P}_{t-1}\mathbf{A}^{*T} + \mathbf{Q}^* \end{pmatrix} \right) d\mathbf{x}_{t-1}$$

$$= N(\mathbf{x}_t | \mathbf{0}, \mathbf{A}^*\mathbf{P}_{t-1}\mathbf{A}^{*T} + \mathbf{Q}^* \qquad \text{(C-9)}$$

Let $\mathbf{P}_t^- = \mathbf{A}^*\mathbf{P}_{t-1}\mathbf{A}^{*T} + \mathbf{Q}^*$. Next, we derive the distribution $P(\mathbf{x}_t, \mathbf{y}_t | \mathbf{y}_{1:t-1}, \mathbf{A}^*, \mathbf{Q}^*)$:

$$P(\mathbf{x}_t, \mathbf{y}_t | \mathbf{y}_{1:t-1}, \mathbf{A}^*, \mathbf{Q}^*) = P(\mathbf{y}_t | \mathbf{x}_t) P(\mathbf{x}_t | \mathbf{y}_{1:t-1}, \mathbf{A}^*, \mathbf{Q}^*). \qquad \text{(C-10)}$$

Inserting (C-9) into the right hand side of (C-10), we get:

$$P(\mathbf{x}_t, \mathbf{y}_t | \mathbf{y}_{1:t-1}, \mathbf{A}^*, \mathbf{Q}^*) \approx C_t exp(-\tfrac{1}{2}\mathbf{x}_t{}^T \mathbf{\Pi}_t \mathbf{x}_t) \times S_t exp(-\tfrac{1}{2}\mathbf{x}_t{}^T (\mathbf{P}_t^-)^{-1} \mathbf{x}_t)$$

$$= C_t S_t \, exp\left(-\tfrac{1}{2}\mathbf{x}_t{}^T (\mathbf{\Pi}_t + (\mathbf{P}_t^-)^{-1}) \mathbf{x}_t\right).$$

Furthermore, because $p(\mathbf{x}_t | \mathbf{y}_{1:t}, \mathbf{A}^*, \mathbf{Q}^*) \propto p(\mathbf{x}_t, \mathbf{y}_t | \mathbf{y}_{1:t-1}, \mathbf{A}^*, \mathbf{Q}^*)$, we get $\mathbf{P}_t = (\mathbf{\Pi}_t + (\mathbf{P}_t^-)^{-1})^{-1}$. Δ

V. Proof of Theorem 2

According to the third equation in (C-8), we can get the joint posterior distribution of the states at time t and $t+1$ as $p((\mathbf{x}_t{}^T, \mathbf{x}_{t+1}{}^T)^T | \mathbf{y}_{1:t}, \mathbf{A}^*, \mathbf{Q}^*) = N\left((\mathbf{x}_t{}^T, \mathbf{x}_{t+1}{}^T)^T | \mathbf{0}, \dot{\mathbf{P}}_1\right)$, where $\dot{\mathbf{P}}_1 = \begin{pmatrix} \mathbf{P}_t & \mathbf{P}_t\mathbf{A}^{*T} \\ \mathbf{A}^*\mathbf{P}_t & \mathbf{A}^*\mathbf{P}_t\mathbf{A}^{*T} + \mathbf{Q}^* \end{pmatrix}$. Then, the conditional posterior distribution of $\mathbf{x}_t | \mathbf{x}_{t+1}$ can be found to be:

$$p(\mathbf{x}_t | \mathbf{x}_{t+1}, \mathbf{y}_{1:t}, \mathbf{A}^*, \mathbf{Q}^*) = N\left(\mathbf{x}_t | \mathbf{0}, \dot{\mathbf{P}}_2\right),$$

where $\dot{\mathbf{P}}_2 = \mathbf{P}_t - \mathbf{H}_t\left(\mathbf{A}^*\mathbf{P}_t\mathbf{A}^{*T} + \mathbf{Q}^*\right)\mathbf{H}_t^T$ and $\mathbf{H}_t = \mathbf{P}_t\mathbf{A}^{*T}(\mathbf{A}^*\mathbf{P}_t\mathbf{A}^{*T} + \mathbf{Q}^*)^{-1}$.

Furthermore, due to the Markovian property of the SSM, we can get:

$$p(\mathbf{x}_t | \mathbf{x}_{t+1}, \mathbf{y}, \mathbf{A}^*, \mathbf{Q}^*) = p(\mathbf{x}_t | \mathbf{x}_{t+1}, \mathbf{y}_{1:t}, \mathbf{A}^*, \mathbf{Q}^*) = N\left(\mathbf{x}_t | \mathbf{0}, \dot{\mathbf{P}}_2\right). \qquad \text{(C-11)}$$

Using (C-11), we can further derive:

$$p((\mathbf{x}_t{}^T, \mathbf{x}_{t+1}{}^T)^T|\mathbf{y}, \mathbf{A}^*, \mathbf{Q}^*) = p(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}, \mathbf{A}^*, \mathbf{Q}^*)p(\mathbf{x}_{t+1}|\mathbf{y}, \mathbf{A}^*, \mathbf{Q}^*) =$$

$$N\left(\mathbf{x}_t\middle|\mathbf{0}, \dot{\mathbf{P}}_2\right)N(\mathbf{x}_{t+1}|\mathbf{0}, \mathbf{P}_{t+1}^s) = N\left(\mathbf{0}, \dot{\mathbf{P}}_3\right),$$

where $\dot{\mathbf{P}}_3 = \begin{pmatrix} \dot{\mathbf{P}}_2 + \mathbf{H}_t\mathbf{P}_{t+1}^s\mathbf{H}_t^T & \mathbf{P}_{t+1}^s\mathbf{H}_t^T \\ \mathbf{H}_t\mathbf{P}_{t+1}^s & \mathbf{P}_{t+1}^s \end{pmatrix}$. In $\dot{\mathbf{P}}_3$, $\dot{\mathbf{P}}_2 + \mathbf{H}_t\mathbf{P}_{t+1}^s\mathbf{H}_t^T$ is $\widehat{\mathrm{Var}}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t)$ and $\mathbf{P}_{t+1}^s\mathbf{H}_t^T$ is $\widehat{\mathrm{Cov}}_{\mathbf{x}_t,\mathbf{x}_{t+1}|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t, \mathbf{x}_{t+1})$. That is, $\mathbf{P}_t^s \triangleq \widehat{\mathrm{Var}}_{\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_t) = \dot{\mathbf{P}}_2 + \mathbf{H}_t\mathbf{P}_{t+1}^s\mathbf{H}_t^T = \mathbf{P}_t + \mathbf{H}_t\left(\mathbf{P}_{t+1}^s - \mathbf{A}^*\mathbf{P}_t\mathbf{A}^{*T} - \mathbf{Q}^*\right)\mathbf{H}_t^T$ and $\mathbf{P}_{t-1,t}^s \triangleq \widehat{\mathrm{Cov}}_{\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{y},\mathbf{A}^*,\mathbf{Q}^*}(\mathbf{x}_{t-1}, \mathbf{x}_t) = \mathbf{P}_t^s\mathbf{H}_{t-1}^T$. Δ