Heriot-Watt University

Masters Thesis

Bayesian Reconsruction and Regression over Networks

Author:
John Smith

Supervisor:
Dr. James Smith

A thesis submitted in fulfilment of the requirements for the degree of MSc.

in the

School of Mathematical and Computer Sciences

January 2023

Declaration of Authorship

I, John Smith, declare that this thesis titled, 'Bayesian Reconsruction and Regression over Networks' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: _______________________________________

Date: _______________________________________

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

Abstract

The Thesis Abstract is written here (and usually kept to just this page).

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor :)

Contents

List of Figures

List of Tables

Abbreviations

GSP	Graph Signal Processing
GSR	Graph Signal Reconstruction
KGR	Kernel Graph Regression
RNC	Regression with Network Cohesion
GLS	Generalised Least Squares

Symbols

Unless otherwise specified, the following naming conventions apply.

Integer constants

N	The number of nodes in a graph
T	The number of time points considered
M	The number of explanatory variables
Q	The number of queries

Integer variables

n	The index of a specific node in a graph
t	The index of a specific time point
m	The index of a specific explanatory variable
q	The index of a specific query
i, j, k	Generic indexing variables

Scalar variables

α	An autocorrelation regularisation parameter
β	A hyperparameter characterising a graph filter
γ	A precision parameter
λ	An eigenvalue *or* ridge regression penalty parameter
μ	The mean of a random variable
θ	AR(1) autocorrelation parameter
σ^2	The variance of a random variable

Matrices

$\mathbf{A}$	The graph adjacency matrix
$\mathbf{D}$	A diagonal matrix
$\mathbf{E}$	The prediction residuals
$\mathbf{F}$	A predicted graph signal
$\mathbf{G}$	A graph filter matrix
$\mathbf{H}$	A Hessian matrix
$\mathbf{I}_N$	The $(N \times N)$ identity matrix
$\mathbf{J}_N$	An $(N \times N)$ matrix of ones
$\mathbf{K}$	A kernel (Gram) matrix
$\mathbf{L}$	The graph Laplacian
$\mathbf{S}$	A binary selection matrix
$\mathbf{U}$	Laplacian eigenvector matrix
$\mathbf{V}$	Kernel eigenvector matrix
$\mathbf{X}$	Data matrix of explanatory variables
$\mathbf{Y}$	(Partially) observed graph signal
$\mathbf{\Lambda}$	A diagonal eigenvalue matrix
$\mathbf{\Sigma}$	A covariance matrix
$\mathbf{\Phi}, \mathbf{\Psi}$	Generic eigenvector matrices
$\mathbf{\Omega}$	Log marginal variance matrix

Vectors/tensors

$\mathbf{1}_N$	A length-N vector of ones
$\mathbf{e}$	The prediction residuals
$\mathbf{e}_i$	The i-th unit basis vector
$\mathbf{f}$	The predicted graph signal
$\mathbf{s}$	A binary selection vector/tensor
$\mathbf{x}$	A vector of explanatory variables
$\mathbf{y}$	The observed graph signal
$\boldsymbol{\alpha}$	A flexible intercept vector/tensor
$\boldsymbol{\beta}$	A graph filter parameter vector *or* vector of regression coefficients
$\boldsymbol{\theta}$	A aggregated coefficient vector $[\boldsymbol{\alpha}^\top, \boldsymbol{\beta}^\top]^\top$

Functions

$g(\cdot)$	A graph filter function
$p(\text{statement})$	The probability that a statement is true
$\pi(\cdot)$	A probability density function
$\xi(\cdot)$	Optimisation target function
$\kappa(\cdot, \cdot)$	A kernel function

Operations

$(\cdot)^\top$	Transpose of a matrix/vector
$\lvert\lvert \cdot \rvert\rvert_\text{F}$	The Frobenius norm
$\text{tr}(\,\cdot\,)$	The trace of a square matrix
$\text{vec}(\cdot)$	Convert a matrix to a vector in column-major order
$\text{vec}_\text{RM}(\cdot)$	Convert a matrix to a vector in row-major order
$\text{mat}(\cdot)$	Convert a vector to a matrix in column-major order
$\text{mat}_\text{RM}(\cdot)$	Convert a vector to a matrix in row-major order
$\text{diag}(\cdot)$	Convert a vector to a diagonal matrix
$\text{diag}^{-1}(\cdot)$	Convert the diagonal of a matrix into a vector
$\otimes$	The Kronecker product
$\oplus$	The Kronecker sum
$\circ$	The Hadamard product

Graphs

$\mathcal{G}$	A graph
$\mathcal{V}$	A vertex/node set
$\mathcal{E}$	An edge set

Miscellaneous

$\hat{(\cdot)}$	The estimator of a matrix/vector/tensor
$O(\cdot)$	The runtime complexity
x_i	A vector element
$\mathbf{X}_i$	A matrix column
$\mathbf{X}_{ij}$	A matrix element

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Background and Definitions

Graph Signal Processing (GSP) is a rapidly evolving field that sits at the intersection between spectral graph theory, statistics and data science [Shuman et al., 2013]. In this context, a graph is an abstract collection of objects in which any pair may be, in some sense, "related". These objects are referred to as vertices (or nodes) and their connections as edges [Newman, 2018]. GSP is concerned with the mathematical analysis of signals that are defined over the nodes of a graph, referred to simply as *graph signals*.

A graph signal can be thought of as a value that is measured simultaneously at each node in a graph. In practice, it is represented as a vector where each element corresponds to a single node. For example, consider a social network where each node represents an individual and presence of an edge between two nodes indicates that the two individuals have met. An example of a graph signal in this context could be the age of each person in the network. Figure 1.1 shows a graphical depiction of a signal defined over a network.

Graphs and graph signals have proven a useful way to describe data across a broad range of applications owing to their flexibility and relative simplicity. They are able to summarise the of properties large, complex systems within a single easily-digestible structure. Much of the data

The GSP community, in particular, is focused on generalising tools designed for traditional signal processing tasks to irregular graph-structured domains.

[Ortega et al., 2018]

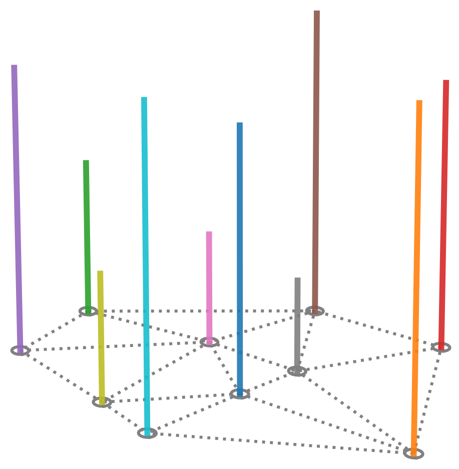

FIGURE 1.1: A graphical depiction of a graph signal. Here, the nodes are represented by circles, the edges as dotted lines, and the value of the signal at each node is represented by the height of its associated bar.

1.2 Thesis overview

Chapter 2

Outline and Fundamentals

2.1 Graph Signal Processing

2.1.1 A broad overview of the field

2.1.2 The graph Laplacian

2.1.3 Graph filters

2.2 Regression and Reconstruction

2.2.1 Graph Signal Reconstruction

Introduce the known work on GSR

2.2.2 Kernel Graph Regression

Introduce the known work on KGR and GPoG

2.2.3 Regression with Network Cohesion

Introduce the known work on RNC

Chapter 3

Kernel Generalized Least Squares Regression for Network Data

3.1 Kernel Graph Regression with Missing Values

3.2 GLS Kernel Graph Regression

3.2.1 A Gauss-Markov estimator

3.2.2 AR(1) processes

3.2.3 Experiments

Chapter 4

Regression and Reconstruction on Cartesian Product Graphs

4.1 Graph Products

In this chapter, we turn our attention to the topic of signal processing on *Cartesian product graphs*. This special class of graph finds applications in numerous areas, such as video, hyper-spectral image processing and network time series problems. However, the Cartesian product is not the only way to consistently define a product between two graphs. In this section we formally introduce the concept of a graph product, examine some prominent examples, and explain why we choose to look specifically at the Cartesian graph product.

4.1.1 Basic definitions

In the general case, consider two undirected graphs $\mathcal{G}_A = (\mathcal{V}_A, \mathcal{E}_A)$ and $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B)$ with vertex sets given by $\mathcal{V}_A = \{a \in \mathbb{N} \,|\, a \leq A\}$ and $\mathcal{V}_B = \{b \in \mathbb{N} \,|\, b \leq B\}$ respectively. (In this context we do not regard zero to be an element of the natural numbers). A new graph $\mathcal{G}$ can be constructed by taking the product between $\mathcal{G}_A$ and $\mathcal{G}_B$. This can be generically written as follows.

$$\mathcal{G} = \mathcal{G}_A \diamond \mathcal{G}_B = (\mathcal{V}, \, \mathcal{E}) \tag{4.1}$$

For all definitions of a graph product, the new vertex set $\mathcal{V}$ is given by the Cartesian product of the vertex sets of the factor graphs, that is

$$\mathcal{V} = \mathcal{V}_A \times \mathcal{V}_B = \{(a,\, b) \in \mathbb{N}^2 \,|\, a \leq A \text{ and } b \leq B\} \tag{4.2}$$

Typically, vertices are are arranged in lexicographic order, in the sense that $(a, b) \leq (a', b')$ iff $a < a'$ or $(a = a'$ and $b \leq b')$ [Harzheim, 2005]. Each consistent rule for constructing the new edge set $\mathcal{E}$ corresponds to a different definition of a graph product. In general, there are eight possible conditions for deciding whether two nodes (a, b) and (a', b') are to be connected in the new graph.

1. $[a,\, a'] \in \mathcal{E}_A$ and $b = b'$
2. $[a,\, a'] \notin \mathcal{E}_A$ and $b = b'$
3. $[a,\, a'] \in \mathcal{E}_A$ and $[b,\, b'] \in \mathcal{E}_B$
4. $[a,\, a'] \notin \mathcal{E}_A$ and $[b,\, b'] \in \mathcal{E}_B$
5. $[a,\, a'] \in \mathcal{E}_A$ and $[b,\, b'] \notin \mathcal{E}_B$
6. $[a,\, a'] \notin \mathcal{E}_A$ and $[b,\, b'] \notin \mathcal{E}_B$
7. $a = a'$ and $[b,\, b'] \in \mathcal{E}_B$,
8. $a = a'$ and $[b,\, b'] \notin \mathcal{E}_B$

Each definition of a graph product corresponds to the union of a specific subset of these conditions, thus, there exist 256 different types of graph product [Barik et al., 2015]. Of these, the Cartesian product (conditions 1 or 7), the direct product (condition 3), the strong product (conditions 1, 3 or 7) and the lexicographic product (conditions 1, 3, 5 or 7) are referred to as the standard products and are well-studied [Imrich and Klavžar, 2000]. A graphical depiction of the standard graph products is shown in figure 4.1. In each of these four cases, the adjacency and Laplacian matrices of the product graph can be described in terms of matrices relating to the factor graphs [Barik et al., 2018, Fiedler, 1973]. This is shown in table 4.1.

	Adjacency matrix	Laplacian
Cartesian	$\mathbf{A}_A \oplus \mathbf{A}_B$	$\mathbf{L}_A \oplus \mathbf{L}_B$
Direct	$\mathbf{A}_A \otimes \mathbf{A}_B$	$\mathbf{D}_A \otimes \mathbf{L}_B + \mathbf{L}_A \otimes \mathbf{D}_B - \mathbf{L}_A \otimes \mathbf{L}_B$
Strong	$\mathbf{A}_A \otimes \mathbf{A}_B + \mathbf{A}_A \oplus \mathbf{A}_B$	$\mathbf{D}_A \otimes \mathbf{L}_B + \mathbf{L}_A \otimes \mathbf{D}_B - \mathbf{L}_A \otimes \mathbf{L}_B + \mathbf{L}_A \oplus \mathbf{L}_B$
Lexicographic	$\mathbf{I}_A \otimes \mathbf{A}_B + \mathbf{A}_A \otimes \mathbf{J}_A$	$\mathbf{I}_A \otimes \mathbf{L}_B + \mathbf{L}_A \otimes \mathbf{J}_B + \mathbf{D}_A \otimes (B\mathbf{I}_B - \mathbf{J}_B)$

Table 4.1: The adjacency and Laplacian matrices for the standard graph products. Here, $\mathbf{D}_A$ and $\mathbf{D}_B$ are the diagonal degree matrices, i.e $\mathbf{D}_A = \text{diag}(\mathbf{A}_A\mathbf{1})$. $\mathbf{I}_A$ and $\mathbf{J}_A$ are the $(A \times A)$ identity matrix and matrix of ones respectively.

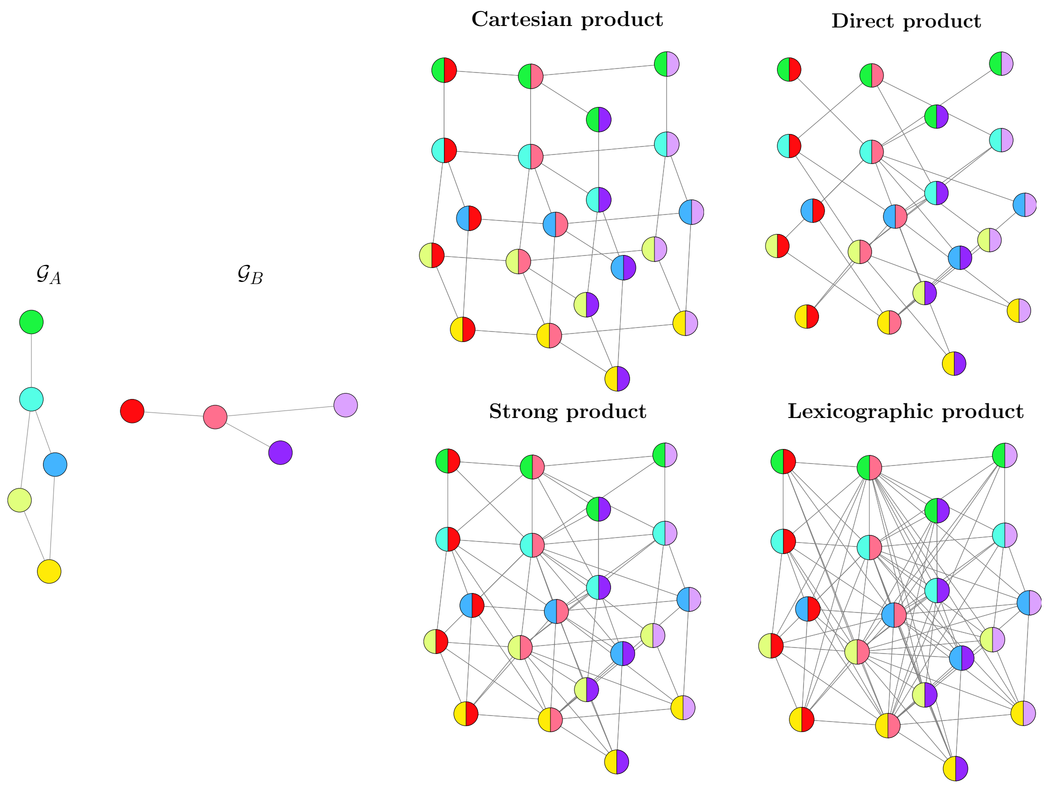

FIGURE 4.1: A graphical depiction of the four standard graph products

Given these definitions, it may seem that all the standard graph products are non-commutative in the sense that $\mathbf{A}_A \oplus \mathbf{A}_B \neq \mathbf{A}_B \oplus \mathbf{A}_A$ etc. However, the graphs $\mathcal{G}_A \diamond \mathcal{G}_B$ and $\mathcal{G}_B \diamond \mathcal{G}_A$ are in fact isomorphically identical in the case of the Cartesian, direct and strong products. This is not the case for the Lexicographic product [Imrich and Klavžar, 2000].

4.1.2 The spectral properties of graph products

In the field of graph signal processing, we are often concerned with analysing the properties of graphs via eigendecomposition of the graph Laplacian [Mieghem, 2010]. In the case of product graphs, it is greatly preferable if we are able to fully describe the spectrum of $\mathcal{G}_A \diamond \mathcal{G}_B$ in terms of the spectra of $\mathcal{G}_A$ and $\mathcal{G}_B$ alone. This is because direct decomposition of a dense $\mathbf{L}$ has time-complexity $O(A^3B^3)$, whereas decomposition of the factor Laplacians individually has complexity $O(A^3 + B^3)$. As the graphs under considerations become medium to large, this fact quickly makes direct decomposition of the product graph Laplacian intractable. However, in the general case, only the spectra of the Cartesian and lexicographic graph products can be described in this way [Barik et al., 2018]. In the case of the direct and strong product, it is possible to estimate the spectra without performing the full decomposition (see [Sayama, 2016]). However, in general, the full eigendecomposition of the product graph Laplacian can only be described in terms of the factor eigendecompositions when both factor graphs are regular.

Consider the eigendecompositions of $\mathbf{L}_A$ and $\mathbf{L}_B$.

$$\mathbf{L}_A = \mathbf{U}_A \mathbf{\Lambda}_A \mathbf{U}_A^\top, \quad \text{and} \quad \mathbf{L}_B = \mathbf{U}_B \mathbf{\Lambda}_B \mathbf{U}_B^\top \tag{4.3}$$

where $\mathbf{U}_A$ and $\mathbf{U}_B$ are the respective orthonormal eigenvector matrices, and $\mathbf{\Lambda}_A$ and $\mathbf{\Lambda}_B$ are the diagonal eigenvalue matrices given by

$$\mathbf{\Lambda}_A = \begin{bmatrix} \lambda_1^{(A)}, & & & \\ & \lambda_2^{(A)} & & \\ & & \ddots & \\ & & & \lambda_A^{(A)} \end{bmatrix} \quad \text{and} \quad \mathbf{\Lambda}_B = \begin{bmatrix} \lambda_1^{(B)}, & & & \\ & \lambda_2^{(B)} & & \\ & & \ddots & \\ & & & \lambda_B^{(B)} \end{bmatrix} \tag{4.4}$$

Given these definitions, table 4.2 gives information about the spectral decomposition of the standard graph products.

	Eigenvalues	Eigenvectors
Cartesian	$\lambda_a^{(A)} + \lambda_b^{(B)}$	$(\mathbf{U}_A)_a \otimes (\mathbf{U}_B)_b$
Direct*	$r_A \lambda_b^{(B)} + r_B \lambda_a^{(A)} - \lambda_a^{(A)} \lambda_b^{(B)}$	$(\mathbf{U}_A)_a \otimes (\mathbf{U}_B)_b$
Strong*	$(1 + r_A)\lambda_b^{(B)} + (1 + r_B)\lambda_a^{(A)} - \lambda_a^{(A)} \lambda_b^{(B)}$	$(\mathbf{U}_A)_a \otimes (\mathbf{U}_B)_b$
Lexicographic†	$B\lambda_a^{(A)}$	$(\mathbf{U}_A)_a \otimes \mathbf{1}_B$
	$\lambda_b^{(B)} + B\deg(a)$	$\mathbf{e}_a \otimes (\mathbf{U}_B)_b$

Table 4.2: Eigendecomposition of the Laplacian of the standard graph products. Here, a and b are understood to run from 1 to A and 1 to B respectively. $\star$ only for r_A and r_B-regular factor graphs. † note that the lower row forms a full spanning set, but the upper row can also be a useful parametrisation for a subspace.

4.1.3 GSP with Cartesian product graphs

While both the direct and strong products do find uses in certain applications (see [Kaveh and Alinejad, 2011]), they are both less common and more challenging to work with in a graph signal processing context due to their spectral properties described in the previous subsection. In practice, being limited to regular factor graphs means the majority of practical GSP applications are ruled out. The lexicographic product does not share this drawback, however it is also significantly less common than the Cartesian product in real-world applications. For this reason, in the following, we focus primarily on the Cartesian product.

Given the spectral decomposition of the Cartesian graph product stated in table 4.2, we can write the Laplacian eigendecomposition in matrix form as follows.

$$\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top, \quad \text{where} \quad \mathbf{U} = \mathbf{U}_A \otimes \mathbf{U}_B \quad \text{and} \quad \boldsymbol{\Lambda} = \boldsymbol{\Lambda}_A \oplus \boldsymbol{\Lambda}_B \tag{4.5}$$

This motivates the following definitions for the Graph Fourier Transform (GFT) and its inverse (IGFT). Consider a signal defined over the nodes of a Cartesian product graph expressed as a matrix $\mathbf{Y} \in \mathbb{R}^{B \times A}$. We can perform the GFT as follows.

$$\text{GFT}(\mathbf{Y}) = \text{mat}\Big(\big(\mathbf{U}_A^\top \otimes \mathbf{U}_B^\top\big)\,\text{vec}(\mathbf{Y})\Big) = \mathbf{U}_B^\top \mathbf{F} \mathbf{U}_A \tag{4.6}$$

Correspondingly, we can define the IGFT acting on a matrix of spectral components $\mathbf{Z} \in \mathbb{R}^{B \times A}$ as follows.

$$\text{IGFT}(\mathbf{Z}) = \text{mat}\Big(\big(\mathbf{U}_A \otimes \mathbf{U}_B\big)\,\text{vec}(\mathbf{Y})\Big) = \mathbf{U}_B \mathbf{Z} \mathbf{U}_A^\top \tag{4.7}$$

Product graph signals: repseprentation and vectorisation

It is natural to assume that signals defined on the nodes of a Cartesian product graph $\mathcal{G}_A \,\square\, \mathcal{G}_B$ can be represented by matrices (order two tensors) of shape $(A \times B)$. Since product graph operators, such as the Laplacian $\mathbf{L}_A \oplus \mathbf{L}_B$, act on vectors of length AB, we must define a consistent function to map matrix graph signals $\in \mathbb{R}^{A \times B}$ to vector graph signals $\in \mathbb{R}^{AB}$. The standard mathematical operator for this purpose is the vec$(\cdot)$ function, along with its reverse operator mat$(\cdot)$. However, this is somewhat problematic since vec$(\cdot)$ is defined to act in *column-major* order, that is

$$\text{vec}\left(\begin{bmatrix} \mathbf{Y}_{(1,1)} & \mathbf{Y}_{(1,2)} & \cdots & \mathbf{Y}_{(1,B)} \\ \mathbf{Y}_{(2,1)} & \mathbf{Y}_{(2,2)} & \cdots & \mathbf{Y}_{(2,B)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Y}_{(A,1)} & \mathbf{Y}_{(A,2)} & \cdots & \mathbf{Y}_{(A,B)} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{Y}_{(1,1)} \\ \mathbf{Y}_{(2,1)} \\ \vdots \\ \mathbf{Y}_{(A-1,B)} \\ \mathbf{Y}_{(A,B)} \end{bmatrix}$$

As is visible, this does not result in a lexicographic ordering of the matrix elements when the graph signal has shape $(A \times B)$. Therefore, to avoid this issue and to be consistent with standard mathematical notation, we will assume that graph signals are represented by matrices of shape $(B \times A)$ when considering the product between two graphs. This ensures that matrix elements are correctly mapped to vector elements when using the column-major vec$(\cdot)$ function.

4.2 Graph Signal Reconstruction on Cartesian Product Graphs

One task that is of particular interest

4.2.1 A stationary iterative method

Hello

4.2.2 A conjugate gradient method

Hello

4.2.3 Convergence properties

Hello

4.2.4 Image processing experiments

Hello

4.3 Kernel Graph Regression with Unrestricted Missing Data Patterns

Hello

4.3.1 Cartesian product graphs and KGR

Hello

4.3.2 Convergence properties

Hello

4.4 Regression with Network Cohesion

Hello

4.4.1 Regression with node-level covariates

Hello

4.4.2 Convergence properties

Hello

4.5 Multi-Dimensional Cartesian Product Graphs

Hello

4.5.1 Fast computation with d-dimensional Kronecker products

Hello

4.5.2 Signal reconstruction

Hello

4.5.3 Kernel Graph Regression

Hello

4.5.4 Regression with Network Cohesion

Chapter 5

Signal Uncertainty: Estimation and Sampling

5.1 Introduction

5.2 Posterior Estimation

5.2.1 Log-variance prediction

5.2.2 Estimation models

5.2.3 Query strategies

5.2.4 Comparison and analysis

5.3 Posterior Sampling

5.3.1 Perturbation optimization

5.4 Estimation vs Sampling

5.4.1 Experiments

Chapter 6

Working with Binary-Valued Graph Signals

6.1 Logistic Graph Signal Reconstruction

6.2 Logistic Kernel Graph Regression

6.3 Logistic Regression with Network Cohesion

6.4 Approximate Sampling via the Laplace Approximation

Chapter 7

Conclusions

7.1 Main Section 1

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

Barik, S., Bapat, R. B., and Pati, S. (2015). On the laplacian spectra of product graphs. *Applicable Analysis and Discrete Mathematics*, 9:39–58.

Barik, S., Kalita, D., Pati, S., and Sahoo, G. (2018). Spectra of graphs resulting from various graph operations and products: a survey. *Special Matrices*, 6:323 – 342.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305.

Harzheim, E. (2005). Chapter 4 - products of orders. In *Ordered Sets*, volume 7 of *Advances in Mathematics*. Springer-Verlag, New York.

Imrich, W. and Klavžar, S. (2000). *Product Graphs: Structure and Recognition*. A Wiley-Interscience publication. Wiley.

Kaveh, A. and Alinejad, B. (2011). Laplacian matrices of product graphs: applications in structural mechanics. *Acta Mechanica*, 222:331–350.

Mieghem, P. v. (2010). *Graph Spectra for Complex Networks*. Cambridge University Press.

Newman, M. (2018). *Networks*. Oxford University Press.

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M. F., and Vandergheynst, P. (2018). Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828.

Sayama, H. (2016). Estimation of laplacian spectra of direct and strong product graphs. *Discrete Applied Mathematics*, 205:160–170.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98.