ABSTRACT

ABBA, MOHAMED ABDELKADER. Topics in Bayesian Transfer Learning and scalable MCMC for Gaussian processes. (Under the direction of Brian Reich and Jonathan Williams).

Deep neural networks (DNN) have revolutionized the field of machine learning with significant breakthroughs in a wide variety of applications including computer vision (Voulodimos et al. 2018), natural language processing (Young et al. 2018), and protein folding (Senior et al. 2020). However, DNNs are notoriously dependent on large amounts of training data to successfully learn complex relationship in the data. Recently, *transfer learning* has emerged as a valuable tool in statistics and machine learning, and enabled the application of these highly complex models to problems with limited data availability. Broadly defined, transfer learning describes a machine learning approach for augmenting the training of a learning task on a *target* population data set with a learning algorithm that has already been trained on a closely related data set from a *source* population, particularly for scenarios where the target and source populations are *not* identical; see Weiss et al. (2016) for a recent survey on transfer learning. A large part of this thesis focuses on Bayesian transfer learning methodology for deep neural networks and investigates theoretical guarantees in simplified settings.

In the first project, we build a Bayesian transfer learning method for deep neural networks with an application to material science. A key task in the emerging field of materials informatics is to use machine learning to predict a material's properties and functions. A fast and accurate predictive model allows researchers to more efficiently identify or construct a material with desirable properties. As in many fields, deep learning is one of the state-of-the art approaches, but fully training a deep learning model is not always feasible in materials informatics due to limitations on data availability, computational resources, and time. Accordingly, there is a critical need in the application of deep learning to materials informatics problems to develop efficient transfer learning algorithms. The Bayesian framework is natural for transfer learning because the model trained from the source data can be encoded in the prior distribution for the target task of interest. However, the Bayesian perspective on transfer learning is relatively unaccounted for in the literature, and is complicated for deep learning because the parameter space is large and the interpretations of individual parameters are unclear. Therefore, rather than subjective prior distributions for individual parameters, we propose a new Bayesian transfer learning approach based on the penalized complexity prior on the Kullback–Leibler divergence between the predictive models of the source and target tasks. We show via simulations that the proposed method outperforms other transfer learning methods across a variety of settings.

The proposed method is applied to predict the properties of a molecular crystal based on its structural properties, and we show improved precision for estimating the band gap of a material compared to state-of-the-art methods currently used in materials science.

Although, the Bayesian framework is natural for transfer learning, to our knowledge there are no theoretical guarantees even for the simplest cases. Motivated by the performance of the Bayesian transfer learning method proposed in the first project, the second project focuses on Bayesian transfer learning in high dimensional normal means and regression problems. We propose a Bayes shrinkage estimator with solid theoretical guarantees under mild assumptions. The proposed methodology is tested on synthetic data and outperforms state-of-the-art transfer learning methods. We then use this method to fine-tune the last layer of a neural network model to predict the molecular gap property. We report improved performance compared to classical fine tuning and methods using only the target data.

In the third project of this thesis, we switch our attention to scalable Bayesian inference in spatial data. Gaussian processes (GPs) are commonly used for prediction and inference for spatial data analyses. However, since estimation and prediction tasks have cubic time and quadratic memory complexity in number of locations, GPs are difficult to scale to large spatial datasets. The Vecchia approximation induces sparsity in the dependence structure and is one of several methods proposed to scale GP inference. Our work adds to the substantial research in this area by developing a stochastic gradient Markov chain Monte Carlo (SGMCMC) framework for efficient computation in GPs. At each step, the algorithm subsamples a minibatch of locations and subsequently updates process parameters through a Vecchia-approximated GP likelihood. Since the Vecchia-approximated GP has a time complexity that is linear in the number of locations, this results in scalable estimation in GPs. Through simulation studies, we demonstrate that SGMCMC is competitive with state-of-the-art scalable GP algorithms in terms of computational time and parameter estimation.

Topics in Bayesian Transfer Learning and scalable MCMC for Gaussian processes

by
Mohamed Abdelkader Abba

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina
2024

APPROVED BY:

Brian Reich Co-chair of Advisory Committee	Jonathan Williams Co-chair of Advisory Committee
Emily Hector	Srijan Sengupta

Sarah Khan

DEDICATION

To my parents Memme and Mohamed.

BIOGRAPHY

Mohamed Abdelkader Abba was born in Nouakchott the capital city of Mauritania. He graduated with a bachelor degree in Applied Mathematics from the University of Nouakchott. Before coming to NC state as graduate student, he worked for four years as a data analyst in his home country.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors Dr. Reich and Dr. Williams for their continuous academic, professional and mental support during the five years of my PhD journey. Their patience, knowledge, and guidance helped me during my time at NC state and this work would not have been fulfilled without their help.

I would like to extend my sincere thanks to Dr. Sengupta, Dr. Hector and Dr. Khan for serving on my committee. I appreciate the time and effort they put in as my committee member.

I would also like to thank all the faculty, staff, and fellow students of the Statistics Department at North Carolina State University. My sincere thanks also go to Alison McCoy, Lanakila Alexander, and Terry Byron for their assistance all these years.

Finally, I express my very profound gratitude to my parents and my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

List of Tables . . . . . **vii**

List of Figures . . . . . **ix**

Chapter 1 INTRODUCTION and background material **1**
1.1 Deep Learning 2
1.1.1 Model 2
1.1.2 Training 3
1.2 Transfer Learning 6
1.2.1 Transfer Learning in DNN 7

Chapter 2 A penalized complexity prior for deep Bayesian transfer learning with application to materials informatics **10**
2.1 Introduction 10
2.1.1 Data and motivation 12
2.1.2 Literature review 13
2.2 Background material 14
2.2.1 Deep Learning Model 14
2.2.2 Transfer Learning 15
2.2.3 Transfer Learning with DNN architectures 15
2.3 PCPs for Deep Bayesian Transfer Learning 16
2.3.1 Penalized Complexity Priors 17
2.3.2 PCP for the VI case 18
2.3.3 PCP for the point estimate case 19
2.4 Simulation study 22
2.5 Real data analysis 26
2.6 Discussion 28

Chapter 3 A Bayesian shrinkage estimator for transfer learning **31**
3.1 Introduction 31
3.2 Statistical model 33
3.2.1 Sparse case 34
3.2.2 Bounded norm case 37
3.3 Theoretical properties 38
3.3.1 Risk of the Horseshoe estimator in the sparse case 38
3.3.2 Risk properties in the bounded ℓ_2 norm case 41
3.4 Simulation study 42
3.4.1 Sparse $\boldsymbol{\delta}$ 42
3.4.2 Bounded $\|\boldsymbol{\delta}\|$ 43
3.4.3 Results 44
3.5 Materials informatics example 46

3.6 Discussion 52

Chapter 4 Stochastic gradient MCMC for massive geostatistical data 54
4.1 Introduction 54
4.1.1 Methods to handle large spatial datasets 55
4.1.2 Review of stochastic gradient methods 57
4.2 Matérn Gaussian process model and approximations 58
4.2.1 Vecchia approximation 59
4.3 SG-MCMC algorithm 60
4.3.1 SG Langevin Dynamics 61
4.3.2 Derivation of Gradient and Fisher Information for SGRLD 62
4.4 Simulation study 65
4.4.1 Data generation 65
4.4.2 Competing methods and metrics 66
4.4.3 Results 67
4.5 Discussion 71

Chapter 5 CONCLUSIONS 72

References 75

APPENDICES 85
Appendix A Supplementary material for chapter 2 86
Appendix B Supplementary material for chapter 3 90
Appendix C Supplementary material for chapter 4 95

LIST OF TABLES

Table 2.1 **Simulation study results**: The non-Bayesian methods ("TL1-TL2") use transfer learning by fixing different numbers of layers of the network using source data, the Bayesian methods are the Bayesian model without transfer learning ("BNN"), full Bayesian transfer learning model with point estimate ("BTL-PCP-PE") and VI ("BTL-PCP-VI") in the first stage. Table (a) gives the mean square prediction error (Monte-Carlo standard errors) over the 500 replicated data sets by the difference between the number of active covariates over tasks, k, and size of the target data set, n. Table (b) gives the average coverage (width) for the Bayesian 95% prediction intervals. 24

Table 2.2 Approximate computational times of the methods used in the simulation study for data sets of size 1000. The Bayesian methods take approximately six times more time to train compared to classical methods. 25

Table 2.3 **Simulation study results**: The non-Bayesian methods ("TL1-TL2") use transfer learning by fixing different numbers of layers of the network using source data, the Bayesian methods are the Bayesian model without transfer learning ("BNN"), full Bayesian transfer learning model with point estimate ("BTL-PCP-PE") and VI ("BTL-PCP-VI") in the first stage, and Bayesian transfer learning prior from Zhou et al. (2020) "BTL-VI". Table (a) gives the mean square prediction error by the difference between the number of active covariates over tasks, k, and size of the target data set, n. Table (b) gives the average coverage (width) for the Bayesian 95% prediction intervals. 25

Table 3.1 Mean squared error for $\boldsymbol{\beta}_2$ for the sparse case with $p = o(n_1)$, $\alpha = 0.2$ and $\gamma = 0.2$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods. 45

Table 3.2 Mean squared error for $\boldsymbol{\beta}_2$ for the sparse case with $p = n_1$ and $\alpha = 0.2$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods. 46

Table 3.3 Mean squared error for $\boldsymbol{\beta}_2$ for the sparse case with $n_1 = o(p)$, $\alpha = 0.2$ and $\gamma = -0.2$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods. 47

Table 3.4 Mean squared error for the means $\boldsymbol{\beta}_2$ in the bounded case with $p = o(n_1)$ and $\gamma = 0.2$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods. 48

Table 3.5 Mean squared error for the means $\boldsymbol{\beta}_2$ in the bounded case with $p = n_1$ and $\gamma = 0$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods. 49
Table 3.6 MSE and coverage of 95% for the band gap prediction application with different target data size, n_2. The methods are Bayesian transfer learning using PCP and Horseshoe priors, transfer learning using ordinal least squares (OLS), and a baseline model that uses only target data. 51

Table 4.1 Mean squared error of covariance parameters. Mean (Monte Carlo standard errors) over 100 simulations for each sample size N. The proposed SGRLD method is compared with other SGMCMC methods (pSGLD, ADAMSGLD, MSGLD) and the full likelihood NNGP method. 68
Table 4.2 Coverage of the 95% credible intervals for the covariance parameters. Mean (Monte Carlo standard errors) over 100 simulations for each sample size N. The proposed SGRLD method is compared with other SGMCMC methods (pSGLD, ADAMSGLD, MSGLD) and the full likelihood NNGP method.. 69
Table 4.3 Effective sample size per unit time (in minutes) of covariance parameters. Mean (Monte Carlo standard errors) over 100 simulations for each sample size N. The proposed SGRLD method is compared with other SGMCMC methods (pSGLD, ADAMSGLD, MSGLD) and the full likelihood NNGP method. 70

Table C.1 Mean squared error of covariance parameters from 10^4 locations. Mean (Monte Carlo standard errors) over 100 simulations. The proposed SGFS method is compared with other SG methods (RMSPROP, ADAM, MSGD) and the full likelihood GpGp method. 97

LIST OF FIGURES

Figure 1.1 DNN architecture with two hidden layers. 2
Figure 1.2 Sketch of netwrok transfer, figure reproduced from Tan et al. (2018). . . . 8

Figure 2.1 The form of the PCP distribution on $\tilde{\tau}$ with $\lambda = 1$, $L = 3$, $c_i = 1/L$, $\upsilon_1 = \upsilon_2 = 16$ and $\upsilon_3 = 1$. 19
Figure 2.2 True hazard function of the scaled KL divergence (blue), and cumulative hazard for different generated data sets (gray). The dashed lines represent the median and mean of the empirical cumulative hazard. 21
Figure 2.3 Test set mean squared error (MSE; eV^2) for different transfer learning methods applied to band-gap prediction. The methods are plotted against the size of the target data training set. Each point represents the MSE for one fold in the cross validation. For comparison, the sample variance of the response is 1.12 eV^2. 27
Figure 2.4 Predicted versus band-gap for the training (left) and test (right) sets for the BLT-PCP-PE methods with $n = 5,000$ training observations. The units of the plot are electronvolts. 28

Figure A.1 MCMC evolution for six randomly drawn fitted values $f(\mathbf{x}_i; \boldsymbol{\theta})$ of the training set. 88

CHAPTER

1

INTRODUCTION AND BACKGROUND MATERIAL

In the framework of transfer learning, we are interested in solving a task of interest, that we will call the target task. Typically, we will have few to no labeled samples, hence training a complex model from scratch is probably infeasible. Transfer learning aims to leverage previously acquired *knowledge* from pretrained models on large datasets to facilitate training and improve performance on the target task. The datasets used to fit the pretrianed models will be referred to as source datasets. Using transfer learning the dependence on large training data could be alleviated, this has been crucial for deploying deep neural network models to solve real world problems. This chapter starts by reviewing the main strategies of transfer learning in Deep Neural Networks. Section 2.2.1 describes the mathematical model for a deep neural network, and briefly discusses training strategies, Section 2.2.2 presents transfer learning in the deep learning framework, subsequently reviewing the leading approaches and their differences.

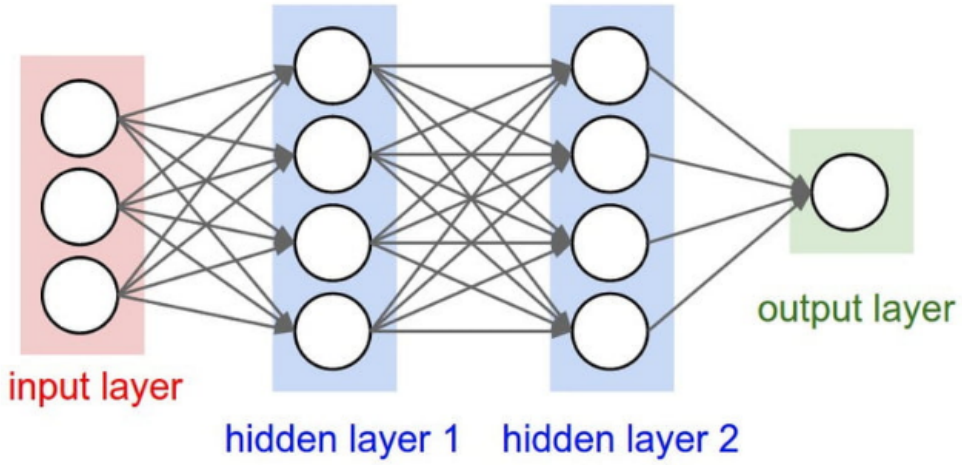

Figure 1.1: DNN architecture with two hidden layers.

1.1 Deep Learning

Deep learning (Goodfellow et al. 2016) refers to the use of deep neural networks (DNN) to accomplish a machine learning task. The basic building block of DNN is called a layer. In general DNNs are comprised of successive layers, starting with an input layer, followed by one or more hidden layer and finally an output layer. Each layer contains a number of *nodes* that are connected to the next layer through a set of weights that control the impact of a given layer on the next, see Figure 1.1.

Over the years, there have been a variety of layer types as well as different architectures for DNNs. In this work, we only discuss the feed forward layers in detail. All other types of layers are more similar than not and details can be found in Goodfellow et al. (2016).

1.1.1 Model

Mathematically, a feed forward dense layer can be viewed as a function $l : \mathbb{R}^d \longrightarrow \mathbb{R}^k$ characterized by a weight matrix $W_l \in \mathbb{R}^{k\times d}$, a bias vector $b_l \in \mathbb{R}^k$, and activation function $a_l(.)$ applied element wise. Hence for $\mathbf{x} \in \mathbb{R}^d$, a feed forward dense layer l with weights W_l, bias b_l and activation function $a(.)$, applies an affine transformation to $\mathbf{x}$ followed by the activation function:

$$l(\mathbf{x}) = a(W_l \mathbf{x} + b_l).$$

Other types of feed forward layers, include but are not limited to Batch Normalization layers (Ioffe and Szegedy 2015; Bjorck et al. 2018), Max-pooling layers (Nagi et al. 2011) and convolutional layers (Voulodimos et al. 2018).

A DNN with $L+1$ layers $l_0, \ldots, l_L$, where l_0 is the input layer $l_0(\mathbf{x}) = \mathbf{x}$, $L-1$ hidden layers and one output layer, represents a function f from the input space $\mathscr{X}$ to the output space $\mathscr{Y}$:

$$\begin{aligned} f_{\boldsymbol{\theta}} &: \mathscr{X} \longrightarrow \mathscr{Y} \\ f_{\boldsymbol{\theta}}(\mathbf{x}) &= l_L(l_{L-1}(\ldots(l_1(\mathbf{x})))). \end{aligned} \tag{1.1}$$

This function is parameterized by the vector $\boldsymbol{\theta}$ which concatenates all the parameters from all the layers. For example if we have one hidden layer with parameters $\{W_1, b_1\}$, and one output layer with parameters $\{W_2, b_2\}$, then $\boldsymbol{\theta}$ is the vector concatenating $\{W_1, W_2, b_1, b_2\}$. Fitting this network requires learning all the entries of $\boldsymbol{\theta}$. Typically, the output space is either the real line for regression task or a vector of probabilities that are constrained to a simplex in $\mathbb{R}^p$ for classification tasks. When the task is regression the output layer performs the usual linear regression on the outputs of the last layer. Likewise, in classification the last layer performs half space learning. This way, DNNs can be viewed as learning both a representation of the data as well as performing a task on the new representation. The deeper the NN, the larger the class of possible representations the model can choose from.

1.1.2 Training

Let $\boldsymbol{\theta}$ be the vector of all the parameters in a DNN model, a dataset $\mathbf{D}_n = \left\{\mathbf{x}_i, y_i\right\}_{1 \le i \le n}$, where $\mathbf{x}_i$ represents the features of the i-th observation and y_i the label or response value. As with any another machine learning method, we could either try to learn a point estimate $\hat{\boldsymbol{\theta}}$ by minimizing some loss function, or follow the Bayesian paradigm to learn a posterior distribution of the parameters given the data.

Point Estimates

Learning point estimates for the parameters of a DNN can be formalized as an optimization problem, through the choice of a loss function and possibly regularization terms. In most cases the loss function is equivalent to the negative log-likelihood of a statistical model (squared loss

for regression corresponds to a MLE of a Gaussian model, and cross entropy loss corresponds to MLE of a logistic model with discrete probabilities, etc...). Likewise, regularization terms correspond to Maximum a Posteriori (MAP) estimation. Without loss of generality, let $\mathscr{L} : \mathscr{Y} \times \mathscr{Y} \longrightarrow \mathbb{R}$ be the loss function and $\boldsymbol{\Omega} : \Theta \longrightarrow \mathbb{R}$ the regularization term. One of the most important assumptions in training DNN is that the loss function decomposes across training samples, in statistical terms, given $\boldsymbol{\theta}$, the observations are *independent*. The optimization problem can thus be written as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \Theta} \mathscr{F}(\boldsymbol{\theta}, \mathscr{D}_n) = \arg\min_{\boldsymbol{\theta} \in \Theta} \sum_{i=1}^{n} \mathscr{L}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \boldsymbol{\Omega}(\boldsymbol{\theta}). \tag{1.2}$$

This formulation is crucial for the training procedure. DNNs require massive datasets to successfully learn complex patterns, so evaluating the model output on the whole dataset is very expensive or simply not possible in some cases. Using (1.2), we can get an unbiased estimate of the gradient of the objective function based on a small size batch of the data $\mathscr{B} \subset \{1, \ldots, n\}$ instead of the whole dataset. So, at the k-th iteration of the optimization procedure, given batch $\mathscr{B}_k$ of the data, the following batch gradient is an unbiased estimate of the gradient of the objective function $\mathscr{F}(\boldsymbol{\theta}, \mathscr{D}_n)$:

$$\tilde{\nabla}_{\boldsymbol{\theta}} \mathscr{F}(\boldsymbol{\theta}, \mathscr{D}_n) = \frac{n}{|\mathscr{B}_k|} \sum_{i \in \mathscr{B}_k} \nabla_{\boldsymbol{\theta}} \mathscr{L}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \nabla_{\boldsymbol{\theta}} \boldsymbol{\Omega}(\boldsymbol{\theta}). \tag{1.3}$$

Using (1.3), Stochastic Gradient Descent SGD (Bottou 2012) methods can be used to reach a local minimum. It should be noted however, that the objective in (1.2) is rarely convex, so there is no guarantee that the optimization algorithm will not be stuck at a saddle point. SGD training simply replaces the usual gradient descent steps with steps in the unbiased direction given in (1.3):

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon_k \tilde{\nabla}_{\boldsymbol{\theta}} \mathscr{F}(\boldsymbol{\theta}, \mathscr{D}_n)$$

where ϵ_k is the learning rate at iteration k. In recent years, many sophisticated flavors of SGD, using momentum acceleration and/or adaptive learning rates have been developed to improve and accelerate training. Recently Liu et al. (2020) extensively studied the effect of using momentum with SGD, their empirical results suggest that SGD with momentum efficiently escapes saddle points, they also presents some theoretical guarantees.

Fully Bayesian Approach

Bayesian learning first requires the choice of a prior distribution on the model parameters. A default choice for DNNs is the multivariate normal distribution $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \Sigma_0)$, with a diagonal covariance matrix Σ_0. Coupled with the likelihood of the data $p(\mathscr{D}_n \mid \boldsymbol{\theta})$, the aim is to learn the posterior distribution:

$$p(\boldsymbol{\theta} \mid \mathscr{D}_n) = \frac{p(\mathscr{D}_n \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int_\Theta p(\mathscr{D}_n \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}},$$

or some functional of this quantity. In general the lack of interpretability of the parameters in neural networks models has shifted the attention from the posterior distribution of the parameters to the induced predictive distribution for a new data observation $(\mathbf{x}^*, y^*)$:

$$p(y^* \mid \mathbf{x}^*, \mathscr{D}_n) = \int_\Theta p(y^* \mid \boldsymbol{\theta}, \mathbf{x}^*) p(\boldsymbol{\theta} \mid \mathscr{D}_n) \mathrm{d}\boldsymbol{\theta}. \tag{1.4}$$

Due to the nonlinearities in DNN models, none of these integrals can be solved analytically, and Markov Chain Monte Carlo (MCMC) methods (Robert and Casella 2013) are used to sample the posterior distribution and approximate (1.4).

Hamiltonian Monte Carlo (HMC) (Brooks et al. 2011), or some flavor of the algorithm is the usual MCMC method used to train Bayesian NN. Although this technique requires the whole dataset for each iteration and is very computationally intensive, it enjoys the theoretical guarantee of asymptotically sampling from the true distribution.

Approximate Bayes

Approximate Bayes methods and specifically Variational Bayes (VB) (Blundell et al. 2015) is generally the main method for training Bayesian neural networks. Instead of sampling, VB learns a parametric approximation of the true posterior distribution. This avoids storing high dimensional parameter samples as is the case in MCMC. One can always generate samples from the approximate distribution if needed. This makes VB well suited for online and active learning.

The simplest version and most widely used flavor of VB for neural networks training is the Mean Field Varriational Bayes (MFVB) (Bhattacharya and Maiti 2021). In this method the true posterior $p_{\mathscr{D}_n} = p(\boldsymbol{\theta} \mid \mathscr{D}_n)$ is approximated by a Gaussian distribution that factorises across all the parameters $q_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = \prod_{w \in \boldsymbol{\theta}} \mathcal{N}(\mu_w, \sigma_w)$. The Kullback divergence metric is typically used to

find the best parameters $\boldsymbol{\alpha} = \{\mu_w, \sigma_w\}_{w \in \boldsymbol{\theta}}$ minimizing the distance from $q_{\boldsymbol{\alpha}}(\boldsymbol{\theta})$ to $p(\boldsymbol{\theta} \mid \mathscr{D}_n)$. This optimization problem can be reformulated as follows:

$$\begin{aligned}
\mathrm{KL}\big(q_{\boldsymbol{\alpha}} \,||\, p_{\mathscr{D}_n}\big) &= \int_{\boldsymbol{\theta}} q_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) \log\left(\frac{q_{\boldsymbol{\alpha}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta} \mid \mathscr{D}_n)}\right) \\
&= \int_{\boldsymbol{\theta}} q_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) \log\left(\frac{q_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) p(\mathscr{D}_n)}{p(\mathscr{D}_n \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}\right) \\
&= \mathrm{KL}\big(q_{\boldsymbol{\alpha}} \,||\, p(\boldsymbol{\theta})\big) - \mathbb{E}_{q_{\boldsymbol{\alpha}}}[\log\big(p(\mathscr{D}_n|\boldsymbol{\theta})\big)] + \log\big(p(\mathscr{D}_n\big). \qquad (1.5)
\end{aligned}$$

The last term in (1.5) is the logarithm of the marginal distribution of the data which depends neither on $\boldsymbol{\theta}$ nor the approximate distribution. Hence minimizing the KL divergence is the same as maximizing the following quantity, which is usually called the evidence lower bound (Tzikas et al. 2008):

$$\mathrm{ELBO}(\boldsymbol{\alpha}) = \mathbb{E}_{q_{\boldsymbol{\alpha}}}[\log\big(p(\mathscr{D}_n|\boldsymbol{\theta})\big)] - \mathrm{KL}\big(q_{\boldsymbol{\alpha}} \,||\, p(\boldsymbol{\theta})\big). \qquad (1.6)$$

From (1.6), we can see that maximizing the lower bound involves two terms. The first term is the expected log-likelihood under the approximate posterior distribution, as we increase this quantity the objective increases. However, the second term appears with a negative sign, hence acts as a penalty that controls the deviation of $q_{\boldsymbol{\alpha}}(\boldsymbol{\theta})$ from the chosen prior distribution.

1.2 Transfer Learning

Before discussing transfer learning strategies used DNN, we will start by reviewing definitions of transfer learning as well as different types of this method. The notations followed will be the same as the ones provided in the influential work of Weiss et al. (2016) and Pan and Yang (2009) whose early work has been widely adopted in the literature.

Definition 1.2.1. *: A domain $\mathscr{D}$ is composed of two parts, a feature space $\mathscr{X}$ and a sampling distribution $p(\mathbf{X})$, where $\mathbf{X}$ denotes a set of instances $\mathbf{X} = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathscr{X}, i = 1, \ldots, n\}$. Thus we can write $\mathscr{D} = \big\{\mathbf{X}, p(\mathbf{X})\big\}$.*

Definition 1.2.2. *: A task $\mathscr{T}$ consists of a label space $\mathscr{Y}$ and a decision function f, that is, $\mathscr{T} = \big\{\mathscr{Y}, f\big\}$. Solving a task can be defined as learning the decision function.*

The decision function can either be a predictive distribution given the inputs $f(\mathbf{x}) = p(y|\mathbf{x})$ or the expected value $f(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}]$. It is important to note that a domain is only observed

through the instances of the data. In other words, for a source domain $\mathscr{D}_S$ and task $\mathscr{T}_S$, we only observe the instances and labels through a dataset $\mathscr{D}_S = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathscr{X}_S, y_i \in \mathscr{Y}_S, 1 \leq i \leq n_S\}$.

Definition 1.2.3. *(Transfer Learning): Given a number t of datasets corresponding to source domains and tasks $\{(\mathscr{D}_{S_i}, \mathscr{T}_{S_i}) \mid i = 1, \ldots, t\}$, and some observations from a target domain and task $(\mathscr{D}_T, \mathscr{T}_T)$, transfer learning takes advantage of the knowledge implied from the source domains to improve the performance of the learned decision function f_T on the target domain.*

In Definition 1.2.3, when $t = 1$ we have single source transfer learning, and multisource transfer learning when $t > 1$. In the case of multisource transfer learning, all the source task are learned simultaneously using a DNN with shared lower layers for all the tasks with the goal to find a common feature representation.

Based on the difference between the domains and/or tasks, transfer learning can be divided into further categories. Pan and Yang (2009) provide an exhaustive survey of all the different categories and approaches in each one.

1.2.1 Transfer Learning in DNN

In the case of DNNs, transfer learning is mainly used in the case where $\mathscr{X}_S = \mathscr{X}_T$, that is the feature space is the same but the domains and tasks are different, this falls under *inductive transfer learning* category. Therefore, in the remainder of the thesis we drop the subscript on the feature space. Note however, that same feature space does not imply same domain, we can still have differences in the marginal distribution $p(\mathbf{X}_S) \neq p(\mathbf{X}_T)$. Furthermore, there is no restriction on the tasks.

What do Neural Networks Learn ?

As discussed in Section 2.2.1, DNNs are composed of many layers, each performing a transformation of the data, that can be viewed as learned representations. It has been shown that layers learn different concepts relating to the task (Dube et al. 2020). Early layers learn general representation of the data, while the the deeper layers are more task specific and learn specialized representations. The former can thus be used to improve performance on related tasks. Although the potential benefits for transfer learning are significant, especially in limited data availability scenarios, there remains very important and challenging questions to address in order to use this method efficiently. Perhaps, the most important challenge is identifying which layers of the model are general and which are specific to the source tasks.

Classical Approaches

The classical approach of deep transfer learning can be summarised as the process of reusing parts of the neural network architecture pretrained on source domain(s), as the lower layers of the model used on the target task. The sketch map of this general technique is shown in Figure 1.2.

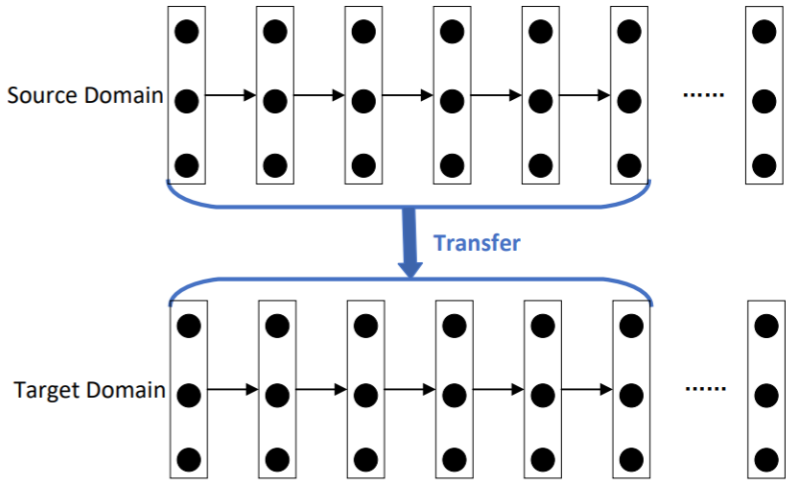

Figure 1.2: Sketch of netwrok transfer, figure reproduced from Tan et al. (2018).

Mathematically, we can summarize this in a more concise notation. Let $g : \mathscr{X} \longrightarrow \mathbb{R}^k$, the function representing shared layers across all source tasks, and let $f^i : \mathbb{R}^k \longrightarrow \mathscr{Y}_{\mathscr{S}_i}, 1 \leq i \leq t$, functions that represent the top layers for each source task, and finally $f^0 : \mathbb{R}^k \longrightarrow \mathscr{Y}_{\mathscr{T}}$, the top part of the model for the target task. Transfer learning in DNN can be thought of as the process of learning simultaneously $g, f^i, 1 \leq i \leq t$, on the source tasks, and then use the learned feature representation $\hat{g}$ on the target task, where this will form the the first layers of the model.

Number of Layers to Transfer

The first question one should ask in any deep transfer learning problem is which parts of the source model should be reused in the target task. Previous works have focused on empirical evaluations of the transferability of some pretrained models in computer vision and natural language processing (NLP) problems. For example, Yosinski et al. (2014) empirically evaluated the transferabilty of pretrained layers trained on Imagenet (Deng et al. 2009), and concluded

that lower convolutional layers are well suited for transfer learning problems in computer vision. This conclusion was based on the model performance compared to random initialization. One surprising result is that initializing a network with transferred features from almost any number of layers can produce a boost to generalization that lingers even after retraining the layers on the target task. Furthermore, they reported an interesting aspect that arises in the target task optimization. Due to the fact that the transferred layers had direct connections with the top parts of the source model, they tend to adapt during the training stage to a common scale. Splitting this model in the middle of co-adapted layers, and randomly initializing the top parts in the target model can cause the target model to perform poorly.

Freezing or Fine-tuning Transferred Layers

Once a specific number of the source model layers are transferred to the target task, the parameters of these layers can either be held fixed or simply treated as starting values during the training of the target task. The former is referred to as freezing of the layers, and the later fine-tuning. If we believe that the tasks are indeed similar and that the transferred layers encode some general representation of source and target features, then the transferred layers carry more importance importance than the rest of the architecture of the target model. While the non transferred layers are randomly initialized and will be learned from the data, the transferred layers hold all the knowledge learned from the source tasks. How to handle this part of the model is a major issue in deep transfer learning. The recent work of Dube et al. (2020) empirically studied the effect of freezing the transferred layers or training them. The conclusion was that freezing pretrained layers, although not optimal, was always better that random initialization. Furthermore, the best performance was obtained when the learning rate for the pretrained layers was significantly lower than the rate of the other layers in the model. They advocate for a factor of 10% as a default good choice.

CHAPTER

2

A PENALIZED COMPLEXITY PRIOR FOR DEEP BAYESIAN TRANSFER LEARNING WITH APPLICATION TO MATERIALS INFORMATICS

2.1 Introduction

Materials informatics has fundamentally changed materials science research (e.g., Himanen et al. 2019). To design or select a material for a particular function, researchers have traditionally relied on intuition and costly experimentation. This process is now supplemented by machine learning to predict a candidate material's properties and triage materials for further experimentation. The addition of machine learning has been shown to improve efficiency, especially when using multiple data sources (e.g., Batra 2021). We aim to build a predictive model for a material's band gap, defined as the energy differential between the lowest unoccupied and highest occupied electronic states (Kittel and McEuen 2019). The band gap governs desirable

properties that are useful in industrial sectors such as electric and photovoltaic conductivity. Traditionally, band gap size is computed using methods in quantum mechanics such as density functional theory (DFT; Kohn and Sham 1965). However, these methods require running costly computer simulations, and so it is *not* feasible to exhaustively search over a broad class of materials. Our objective is to build a statistical model (sometimes called a meta-model, surrogate model, or emulator; O'Hagan 2006) along with an accompanying transfer learning methodology that is both effective at predicting the output of the DFT simulation, and is able to optimize experimentation on future test data sets from related data sources.

Deep neural network (DNN) architectures have emerged as leading models in materials informatics. Beyond materials science, DNNs have revolutionized the field of machine learning with significant breakthroughs in a wide variety of applications including computer vision (Voulodimos et al. 2018), natural language processing (Young et al. 2018), and protein folding (Senior et al. 2020); see Dargan et al. (2019) for further references to modern applications. Their ability to model complex nonlinear processes enables them to handle a large class of prediction problems. Training a neural network, however, is not an easy task, often requiring intensive computational cost, large quantities of training data, and careful hyperparameter selection (Bengio 2012). As an over-parameterized expressive model, DNNs are prone to overfitting, especially in the case of small data sets. Tan et al. (2018) concludes that the number of parameters in a DNN and the size of the data required for good generalization performance have an almost linear relationship. Since these models tend to have thousands (sometimes millions) of parameters, the size of the data required becomes quickly prohibitive. This is where *transfer learning* plays a central role.

Broadly defined, transfer learning describes a machine learning approach for augmenting the training of a learning task on a *target* population data set with a learning algorithm that has already been trained on a closely related data set from a *source* population, particularly for scenarios where the target and source populations are *not* identical; see Weiss et al. (2016) for a recent survey on transfer learning. By developing effective transfer learning strategies, it is possible to reduce the computational burden, and the need for large training data sets, for training DNN models in applications where complex DNNs have already been trained for similar data sets. For example, Raghu et al. (2019) and Ahishakiye et al. (2021) investigated the use of a pre-trained DNN on ImageNet (Deng et al. 2009) to improve the accuracy in medical imaging tasks where labeled data is usually scarce. Although the potential benefits for transfer learning are significant, there are no generally accepted procedures for how to construct or evaluate a transfer learning strategy, and this is especially true from the Bayesian perspective.

Dube et al. (2020) provides a review of the current approaches to address a variety of important questions of concern for developing transfer learning methods. For instance, what aspects of the source tasks can be leveraged to improve performance on the target task? In the case of DNN source models, this commonly boils down to isolating the layers that are task agnostic versus those that are task dependent.

2.1.1 Data and motivation

Motivated by the predictive materials science application of estimating the band gap of a material based on its structural properties, we propose a principled statistical approach to transfer learning within a Bayesian framework. In materials science, the band gap governs desirable properties of candidate materials. The conductivity of a solid is closely related to the band gap size. For example, materials with large band gaps are generally insulators, while materials with low band gaps are semiconductors (Van de Walle 2012). Band gap size (measured in electronvolts, eV) is typically computed using different quantum mechanics methods like the DFT. DFT calculation can be expensive for large molecules and crystals. This, combined with the large possible number of materials, has driven the need for fast screening methods. According to Olsthoorn et al. (2019), there has been a growing interest in the development of accurate machine learning models for material sciences and quantum systems. For small organic molecules, Olsthoorn et al. (2019) reported that DNN models have reached chemical accuracy. For large molecular crystals, however, the task of band gap prediction seems to be much harder. In addition, DFT calculations are more expensive to perform on molecular crystals. For example, Ma et al. (2021) found that the algorithmic complexity of DFT is cubic in the number of particles for a given molecule; hence, it is easier to collect and compute band gap values for molecules than it is for molecular crystals. Given the difficulty of band gap calculation for molecular crystals, we propose to use transfer learning methods to borrow information from molecules (source) to molecular crystals (target) to improve prediction. The target data considered here will consist of 50,000 molecular crystals and their band gap values computed using DFT. This data set was provided by the Material Science Research Group at Carnegie Mellon University and required a sustained effort over several years to produce. The source data will be the OE62 data set (Stuke et al. 2019) which consists of 62,000 molecules and their band gap values. The molecules in the source data were all extracted from organic crystals, and the band gap values were also computed using DFT.

2.1.2 Literature review

Bayesian neural networks have attracted considerable attention in recent years as a result of computational advances (e.g., Wilson 2020). One of the earliest works on Bayesian transfer learning in DNNs was Wohlert et al. (2018), where a single source task is considered and a DNN with feed forward layers is trained using mean field variational Bayes (VB; Zhang et al. 2018). The approximate posterior learned on the source task is used as a prior on the parameters for the target task. Wohlert et al. (2018) did not, however, address the problem of freezing or training the transferred layers and surprisingly gave worse performance than training using only the target data. Chandra and Kapoor (2020) introduced a new method for Bayesian multi-source transfer learning where they proposed a new Markov chain Monte Carlo (MCMC) sampling scheme with parameter proposals for the target task based on proposals for all the source tasks. At first, all source task parameters are sampled, and then a single source for transfer of knowledge is used for the proposals for the target task. This method requires storing all source tasks data in memory together with the target task and does not fall under the two stage inductive transfer learning framework considered in our paper. Bueno et al. (2019) used Bayesian transfer learning for volcano-seismic prediction and uncertainty quantification using variational inference. Yang et al. (2020) focused on Bayesian hyperparameter optimization for transfer learning. Zhou et al. (2020) proposed a new form of regularization that can be viewed as a Gaussian prior on the network weights where the Fisher information from the source task is used as the precision matrix. More generally, Bayesian transfer learning for models other than DNNs has been considered by other articles in the literature, most commonly attributable to the fact that it is natural to borrow strength across data sets via prior distributions (e.g., Karbalayghareh et al. 2018).

The Bayesian transfer learning methodology that we develop sequentially trains a DNN on a target data set by first specifying a prior distribution concentrated on parameter estimates for the DNN trained from a source data set. While this approach is sensible, the primary challenge is that constructing prior densities from pre-trained DNN layers is not trivial because individual parameters are not identifiable nor do they have inherent meaning. Accordingly, to solve this issue, we extend the fundamental notion of *penalized complexity priors* (PCPs) from Simpson et al. (2017b) to the deep learning setting, and specify our *transfer learning prior distribution* in terms of the Kullback-Leibler divergence between the source and target model. This prior is constructed with the intuition that the predictive model is likely shared (to some extent) across tasks, but is flexible enough to disregard the source task if appropriate.

The target data set in our materials science application is small, and so we implement a

fully Bayesian analysis with computations via MCMC sampling strategies. However, since the source data set is large, a fully Bayesian analysis is prohibitive, and so we analyze the source data both using optimization and variational inference (VI; Blundell et al. 2015) methods. Both methods provide point estimates for each parameter, but VI also provides (approximate) uncertainty quantification. Therefore, in addition to applying to cases where data sets from both the target and the source are available, our proposed method can be applied using source data DNNs trained by other users, such as ImageNet (Deng et al. 2009). In our exposition, we develop our transfer learning methods using the optimization-based source model and the VI-based source model, in parallel, and we show that both provide improvements over traditional transfer learning methods for both synthetic and real data.

The remainder of the paper proceeds as follows. Section 2.2 reviews background material on DNNs and transfer learning. Section 2.3 introduces the proposed PCP priors for transfer learning in DNNs. The proposed methods are then evaluated using simulated and real data in Sections 4.4 and 2.5, respectively. Section 2.6 concludes. The code for reproducing our empirical results is available at `https://github.ncsu.edu/mabba/Pybayes.git`.

2.2 Background material

2.2.1 Deep Learning Model

We begin by briefly reviewing the DNN model; for a more comprehesive review see Goodfellow et al. (2016). In general, DNNs are comprised of successive layers, starting with an input layer, followed by one or more hidden layers, and finally an output layer. Each layer contains a number of nodes that are connected to the next layer through weights that control the impact of a given layer on the next. For concreteness, we describe the Bayesian transfer learning model in the simple case of a continuous response and fully-connected feed forward neural network. However, the ideas proposed in this paper translate to other structures, including recurrent (Medsker and Jain 2001) and convolutional (LeCun et al. 1995) networks (see Section 2.6 for further discussion).

A dense feed forward layer is a function $l : \mathbb{R}^d \longrightarrow \mathbb{R}^k$ characterized by a weight matrix $W \in \mathbb{R}^{k\times d}$, a bias vector $b \in \mathbb{R}^k$, and a real-valued activation function a applied element wise to a vector argument. For $\mathbf{x} \in \mathbb{R}^d$, the dense layer applies an affine transformation to $\mathbf{x}$ followed by the activation function, $l(\mathbf{x}) = a(W\mathbf{x} + b)$. A DNN with L hidden layers first applies an input layer, typically $l_0(\mathbf{x}) = \mathbf{x}$, then $L-1$ hidden layers, and one output layer; each layer is assigned

unique weights, biases, and (potentially) activation functions, i.e., $l_i(\mathbf{x}) = a_i(W_i\mathbf{x} + b_i)$ for layer $i \in \{1, ..., L\}$. The composition of these L layers defines a function f from the input space $\mathscr{X}$ to the output space $\mathscr{Y}$:

$$\begin{aligned} f(\cdot;\boldsymbol{\theta}) &: \mathscr{X} \longrightarrow \mathscr{Y} \\ f(\mathbf{x};\boldsymbol{\theta}) &= l_L(l_{L-1}(\cdots(l_1(\mathbf{x})))), \end{aligned} \tag{2.1}$$

where $\boldsymbol{\theta} := \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_L\}$ and $\boldsymbol{\theta}_i := \{b_i, W_i\}$ are the parameters for layer i. For a regression task, we consider $f(\mathbf{x};\boldsymbol{\theta})$ to be the mean response given $\mathbf{x}$, whereas for a classification task, $f(\mathbf{x};\boldsymbol{\theta})$ represents a vector of probabilities.

2.2.2 Transfer Learning

Before discussing transfer learning for DNNs, we review general definitions and concepts of transfer learning. We adopt notation from Pan and Yang (2009) and Weiss et al. (2016) whose early work has been widely adopted in the literature. A *domain* is composed of two parts, a feature space $\mathscr{X}$ and a sampling distribution $p(\mathbf{X})$, where $\mathbf{X}$ denotes a set of instances $\mathbf{X} = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathscr{X}, i = 1, \ldots, n\}$. Accordingly, $\mathscr{D} = \{\mathscr{X}, p\}$. Next, a *task* (e.g., classification or prediction) consists of a label space $\mathscr{Y}$ and a decision function f, that is, $\mathscr{T} = \{\mathscr{Y}, f\}$.

Transfer learning considers two domains: the source and the target. The main objective is to perform the target task in the target domain, and the analysis are supplemented by the related data from the source task (or multiple source tasks, e.g., Maurer et al. (2016)). For a source domain and task, denoted $\mathscr{D}_S$ and $\mathscr{T}_S$, respectively, an observed data set is the collection $\{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathscr{X}_S, y_i \in \mathscr{Y}_S, 1 \le i \le n_S\}$, and f_S is understood as a conditional distribution or decision function for the instances y_i given $\mathbf{x}_i$ for $i \in \{1, \ldots, n_S\}$. Similarly, define $\mathscr{D}_T$, $\mathscr{T}_T$, $\{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathscr{X}_T, y_i \in \mathscr{Y}_T, 1 \le i \le n_T\}$ as the domain, task, and data for the target domain. DNNs typically require *inductive transfer learning* where the feature space is the same across tasks (Pan and Yang 2009), i.e., $\mathscr{X}_S = \mathscr{X}_T$. Therefore, we will drop the subscript on the feature space and simply use $\mathscr{X}$. Note that this does *not* imply that the conditional decision functions, nor the sampling distributions of the features, are the same for the source and target.

2.2.3 Transfer Learning with DNN architectures

As discussed in Section 2.2.1, DNNs are composed of many layers, each performing a transformation of the data. These transformations can be viewed as learned representations. It has been

argued that different layers may learn different concepts relating to the task (Dube et al. 2020). Specifically, there is evidence that early layers learn general representations of the data, while the the deeper layers are more task specific and learn specialized representations. In Zeiler and Fergus (2013), the authors studied the activation of early layers of DNNs in computer vision problems, where convolutional layers are typically used, and noted that early convolutional layers extract high level features with the potential to generalize to different image domains. Using this logic, it follows that transfer learning for DNNs amounts to determining how many early layers to share across tasks and how many to assign as task specific.

As defined in (2.1), in DNNs the tasks are defined by the weight parameters for each layer of the network. Assuming the same network architecture for both tasks, define $\boldsymbol{\theta}_S = \{\boldsymbol{\theta}_{S1}, ..., \boldsymbol{\theta}_{SL}\}$ and $\boldsymbol{\theta}_T = \{\boldsymbol{\theta}_{T1}, ..., \boldsymbol{\theta}_{TL}\}$ as the parameters for the source and target tasks, respectively. A transfer learning strategy with DNNs can be defined as the process of learning simultaneously the parameters $\boldsymbol{\theta}_S$ to augment the learning of $\boldsymbol{\theta}_T$ on the target task. The transferred layers can either be held fixed on the target task (referred to as *freezing*) or fine-tuned. Recently, Dube et al. (2020) investigated the empirical performances of freezing and fine-tuning. In their work, they focused on a computer vision task with pre-trained layers from different architectures that were fit to the Imagenet data set. The conclusion was that freezing pre-trained layers, although not optimal, was always better that random initialization. The best performance was obtained when the learning rate for the pre-trained layers was significantly lower than the rate of the other layers in the model. They advocate for a factor of 10% as a default choice.

2.3 PCPs for Deep Bayesian Transfer Learning

For the target task, let $\boldsymbol{\theta}_T \equiv \boldsymbol{\theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_L\}$ and denote v_i as the number of parameters in $\boldsymbol{\theta}_i$; for the source task, denote $\hat{\boldsymbol{\theta}} = \{\hat{\boldsymbol{\theta}}_1, ..., \hat{\boldsymbol{\theta}}_L\}$ as the estimated values for $\boldsymbol{\theta}$ that are trained from the source data and, when available, let Σ_i be the covariance matrix for $\boldsymbol{\theta}_i$ (e.g., the posterior covariance given the source data). Our goal is to leverage these estimated parameters to construct an informative prior for the target task. If the tasks were the same the natural solution would be to set $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$. On the other hand, if the tasks were not at all related, $\hat{\boldsymbol{\theta}}$ does not encode any information for the target task and would probably be worse than random initialization. In between these two extreme cases lies the core of transfer learning, where there is a reason to assume that tasks are related but we do not know to which degree.

The focus of our analysis is a judicious choice of the prior distribution for $\boldsymbol{\theta}$. Constructing a prior distribution that uses $\hat{\boldsymbol{\theta}}$ requires, first, the choice of a method for computing $\hat{\boldsymbol{\theta}}$ from

the source data, and second, requires specification of the prior distribution for $\boldsymbol{\theta}$ given $\hat{\boldsymbol{\theta}}$. As described in the next subsection, we use the PCP of Simpson et al. (2017b) for the prior distribution of $\boldsymbol{\theta}$ given $\hat{\boldsymbol{\theta}}$. The remainder of this section provides an overview the PCP idea of Simpson et al. (2017b), and then describes two methods we propose for constructing $\hat{\boldsymbol{\theta}}$. The PCP is derived for both methods of constructing $\hat{\boldsymbol{\theta}}$.

2.3.1 Penalized Complexity Priors

In the framework of Simpson et al. (2017b), the problem of prior choice is considered from a model complexity perspective. The prior on the parameter of interest θ is controlled by a flexibility parameter τ. The base (or simplest) model for θ, that we will denote by $p_0(\theta)$ corresponds to the case where $\tau = 0$, and for larger values of τ the prior deviates from the base model. Following Occam's razor, simple models should be preferred until there is evidence for more complex ones; in other words, the hyperprior on τ should favor the base model. PCPs as defined in Simpson et al. (2017b) give a principled way of choosing a prior on τ that controls the deviation from the base model by penalizing a scaled version of the Kullback-Liebler divergence from $p_0(\theta)$ to $p(\theta|\tau)$:

$$d_{\mathrm{KL}}\big(p \,||\, p_0\big) = \sqrt{2\mathrm{KL}\big(p \,||\, p_0\big)} = \left(2\int_\theta p(\theta|\tau)\log\left(\frac{p(\theta|\tau)}{p_0(\theta)}\right)d\theta\right)^{1/2}. \tag{2.2}$$

From (2.2), it is clear that $d_{\mathrm{KL}}\big(p \,||\, p_0\big)$ is a function of τ. That being so, let

$$h(\tau) = d_{\mathrm{KL}}\big(p(\theta|\tau) \,||\, p_0(\theta)\big),$$

and notice that $h(\tau) = 0$ if and only if $\tau = 0$. Furthermore, if $h(\cdot)$ is strictly increasing, penalising $h(\tau)$ for large values induces a prior that penalizes large values of τ, and consequently puts more mass close to the base model. Simpson et al. (2017b) recommend using an exponential distribution, $h(\tau) \sim \mathrm{Exp}(\lambda)$, to ensure a mode at $\tau = 0$ and a constant rate of penalization for larger τ. Let

$$p(\boldsymbol{\theta}|\boldsymbol{\tau}) = \prod_{i=1}^{L} N\left(\boldsymbol{\theta}_i; \hat{\boldsymbol{\theta}}_i, g(\tau_i)\Sigma_i\right),$$

where N is the multivariate Gaussian density function, g is an increasing function, the flexibility parameter τ_i controls the deviation from the base model in layer i and $\boldsymbol{\tau} = (\tau_1, ..., \tau_L)$. The case $\boldsymbol{\tau} = \prime$ is used to define the base model. This layer-wise specification reflects the fact that some layers can be more transferable than others, and hence require more penalization.

If we marginalize over $\boldsymbol{\tau}$, the prior model becomes a scale mixture of Gaussian distributions, where the mixing distribution will be the prior on $\boldsymbol{\tau}$. This latter distribution will be implicitly defined by placing an exponential distribution on $h(\boldsymbol{\tau}) = d_{\text{KL}}\big(p(\boldsymbol{\theta}_T|\boldsymbol{\tau}) \,||\, p_0(\boldsymbol{\theta}_T)\big)$. When $L > 1$, the implicit prior on $\boldsymbol{\tau}$ is not identifiable since the mapping $h : \mathbb{R}_+^L \longrightarrow \mathbb{R}_+$ cannot be bijective and hence is not invertible. To circumvent this issue, Simpson et al. (2017b) extend the PCP prior to a multivariate parameter $\boldsymbol{\tau}$ by having the prior on $\boldsymbol{\tau}$ uniform on each level set of $h(\cdot)$. Instead, we opt for the simpler approach of constraining $\boldsymbol{\tau}$ to the $(L-1)$-simplex given by any weighted combination of its elements. This is equivalent to $\boldsymbol{\tau} = \tilde{\tau}\mathbf{c}$, where $\mathbf{c} \sim \text{Dirichlet}(\infty_L)$ and $\tilde{\tau} \in \mathbb{R}_+$. In this formulation, $\tilde{\tau}$ becomes a global flexibility parameter, and the weights c_i control how much deviation each layer is allowed since $\tau_i = \tilde{\tau} c_i$. Thus, given the vector of weights $\mathbf{c}$, we have a one-to-one mapping $h_{\mathbf{c}}(\tilde{\tau})$, through which we have an explicit penalising prior on the global parameter $\tilde{\tau}$.

2.3.2 PCP for the VI case

If the source data are analyzed using VB methods, then $\hat{\boldsymbol{\theta}}_i$ is an approximate posterior mean and Σ_i is an approximate $v_i \times v_i$ posterior covariance matrix. Because standard VB methods approximate the posterior as independent across parameters, Σ_i is a diagonal matrix for each i. We incorporate this information in the prior by setting $g(\tau) = 1 + \tau$ so that the base model with $\boldsymbol{\tau} = \prime$ uses the posterior of the source data directly as the prior for the target data, which would be optimal Bayesian learning if the two tasks are the same. If $\tau_i > 0$, then the prior variance increases to reflect uncertainty about the relationship between source and target tasks. Under this prior distribution, the KL divergence is

$$\text{KL}\big(p(\boldsymbol{\theta}|\tilde{\tau}, \mathbf{c}) \,||\, p_0(\boldsymbol{\theta})\big) = \frac{1}{2}\left(\sum_{i=1}^{L} v_i c_i \tilde{\tau} - v_i \log(1 + c_i \tilde{\tau})\right).$$

So if $d = \sqrt{2\text{KL}\big(p(\boldsymbol{\theta}|\tilde{\tau}, \mathbf{c}) \,||\, p_0(\boldsymbol{\theta})\big)} \sim \text{Exp}(\lambda)$, the one-to-one mapping between d and $\tilde{\tau}$ is

$$d = \sqrt{\sum_{i=1}^{L} v_i \phi(c_i \tilde{\tau})} \tag{2.3}$$

where $\phi(x) = x - \log(1+x)$. Furthermore, ϕ is continuously differentiable and strictly increasing, hence given $\mathbf{c}$ we have a unique prior on $\tilde{\tau}$ corresponding to the desired prior on the KL

divergence from the base model. The induced prior is

$$p(\tilde{\tau}|\mathbf{c}) = p(h_{\mathbf{c}}(\tilde{\tau})) \left| \frac{d\, h_{\mathbf{c}}(\tilde{\tau})}{d\,\tilde{\tau}} \right| = \frac{\lambda e^{-\lambda h_{\mathbf{c}}(\tilde{\tau})}}{2\left(\sum_{i=1}^{L} v_i \phi(c_i \tilde{\tau})\right)^{1/2}} \left(\sum_{i=1}^{L} v_i \frac{c_i^2 \tilde{\tau}}{1 + c_i \tilde{\tau}} \right). \tag{2.4}$$

with $h_{\mathbf{c}}(\tilde{\tau})^2 = \sum_{i=1}^{L} v_i \phi(c_i \tilde{\tau})$. Equation (2.4) coupled with the Dirichlet prior on the weight $\mathbf{c}$ completely specifies a prior distribution on the vector of scales $\boldsymbol{\tau}$. Using numerical inversion of the mapping in (2.3), sampling from this prior is straightforward.

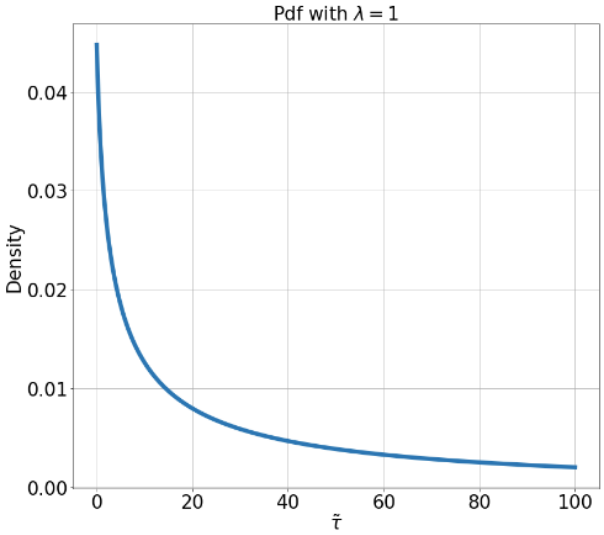

Figure 2.1: The form of the PCP distribution on $\tilde{\tau}$ with $\lambda = 1$, $L = 3$, $c_i = 1/L$, $v_1 = v_2 = 16$ and $v_3 = 1$.

It can be shown that the induced prior on the global flexibility parameter in (2.4) has a mode at zero with a strictly decreasing density as $\tilde{\tau}$ increases. Also, the tail of the prior behaves like a modified Weibull distribution (Almalki and Nadarajah 2014) with rate $\lambda \sum v_i c_i$ and shape 0.5. Figure 2.1 plots the density when $\lambda = 1$.

2.3.3 PCP for the point estimate case

Assume that a point estimate $\hat{\boldsymbol{\theta}}$ is derived from the source data but no measure of uncertainty Σ is provided. In this case, the source data does not define a base distribution for $\boldsymbol{\theta}$ and thus we cannot compute the KL divergence between base and full models for $\boldsymbol{\theta}$ without

further assumptions. Since DNNs are primarily used for prediction, we place a prior on the KL divergence between the predictive model under the source and target models. Let $f(\mathbf{x}|\boldsymbol{\theta})$ denote the target DNN output for a given input $\mathbf{x}$ and parameter values ($\boldsymbol{\theta}$). Let the regression model likelihood be $p(y) \sim \text{Normal}(f(\mathbf{x}|\boldsymbol{\theta}), \sigma^2)$ and $\boldsymbol{\theta}_i \sim \text{Normal}(\hat{\boldsymbol{\theta}}_i, \tau_i \mathscr{I})$ which induces a prior distribution on the function f. Unfortunately, for the DNN model it is not available in analytical form. To overcome this, we approximate the DNN output using a first-order Taylor approximation around $\hat{\boldsymbol{\theta}}$

$$f(\mathbf{x}|\boldsymbol{\theta}) \approx f(\mathbf{x}|\hat{\boldsymbol{\theta}}) + \left(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\right)^{\mathrm{T}} \nabla_{\hat{\boldsymbol{\theta}}} f(\mathbf{x}|\hat{\boldsymbol{\theta}}). \tag{2.5}$$

For small values of $\tilde{\tau}$ (i.e., the most relevant values for the prior, assuming the target and source are similar) a first-order approximation is arguably tight enough. Furthermore, if the activation functions used in the model are piece-wise linear like the widely used ReLU function (Agarap 2018), the output function $f(\mathbf{x}|\boldsymbol{\theta})$ will also be piece-wise linear and we cannot go beyond a first-order Taylor expansion.

Based on the approximation, for the base model with $\tau = \prime$ we get the prior distribution

$$p_0\left(y\right) \sim \text{Normal}\left(f(\mathbf{x}|\hat{\boldsymbol{\theta}},), \sigma^2\right),$$

and for the flexible model the prior on $f(\mathbf{y}|\tilde{\tau}, \mathbf{c})$ is

$$p\left(y|\tilde{\tau}, \mathbf{c}\right) \dot{\approx} \text{Normal}\left(f(\mathbf{x}|\hat{\boldsymbol{\theta}}), \sigma^2 + \tilde{\tau}\mathbf{c}^{\mathrm{T}}\alpha(\mathbf{x})\right),$$

where $\alpha(\mathbf{x}) = \left(\|\nabla_{\hat{\boldsymbol{\theta}}_i} f(\mathbf{x}|\hat{\boldsymbol{\theta}},)\|^2\right)_{1 \le i \le L}$. Now that we have two normal models, an analytic expression for the KL distance is

$$\text{KL}\left(p\left(y|\tilde{\tau}, \mathbf{c}\right) || \, p_0\right) = \frac{1}{2}\phi\left[\tilde{\tau}\tilde{\alpha}_{\mathbf{c}}(\mathbf{x})\right] \quad \text{where} \quad \tilde{\alpha}_{\mathbf{c}}(\mathbf{x}) = \frac{\mathbf{c}^{\mathrm{T}}\alpha(\mathbf{x})}{\sigma^2}. \tag{2.6}$$

In equation (2.6), $\tilde{\alpha}(\mathbf{x})$ depends on the gradient of $f(\mathbf{x}|\hat{\boldsymbol{\theta}})$ for a given input point $\mathbf{x}$. Assuming that the inputs are identically distributed and we have N target data points, the marginal KL divergence can be approximated by

$$\text{KL}\left(p \, || \, p_0\right) \approx \frac{1}{2N}\sum_{i=1}^{N}\phi\left[\tilde{\tau}\tilde{\alpha}_{\mathbf{c}}(\mathbf{x}_i)\right]. \tag{2.7}$$

Using (2.7) we have a one-to-one mapping between $d = \sqrt{2\text{KL}\left(p \, || \, p_0\right)}$ and the global flexibility

parameter $\tilde{\tau}$. Hence, an exponential distribution on d induces a prior on $\tilde{\tau}$, which can be obtained with the same simple change of variables as in (2.4). Let $d_{\mathbf{c}}(\tilde{\tau}) = \sum_{i=1}^{N} \phi\left[\tilde{\tau}\tilde{\alpha}_{\mathbf{c}}(\mathbf{x}_i)\right]/N$. Then

$$p(\tilde{\tau}|\mathbf{c}) = p(d_{\mathbf{c}}(\tilde{\tau}))\left|\frac{d_{\mathbf{c}}(\tilde{\tau})}{d\tilde{\tau}}\right| = \frac{\lambda e^{-\lambda d_{\mathbf{c}}(\tilde{\tau})}}{2\left(\frac{1}{N}\sum_{i=1}^{N}\phi\left[\tilde{\tau}\tilde{\alpha}_{\mathbf{c}}(\mathbf{x}_i)\right]\right)^{1/2}}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\tilde{\alpha}_{\mathbf{c}}(\mathbf{x}_i)^2\tilde{\tau}}{1+\tilde{\alpha}_{\mathbf{c}}(\mathbf{x}_i)\tilde{\tau}}\right). \tag{2.8}$$

The form of the prior in (2.8) is similar to (2.4): in both cases the prior induced on $\tilde{\tau}$ has mode at zero, is a strictly decreasing density as $\tilde{\tau}$ increases, and has the same tail behavior.

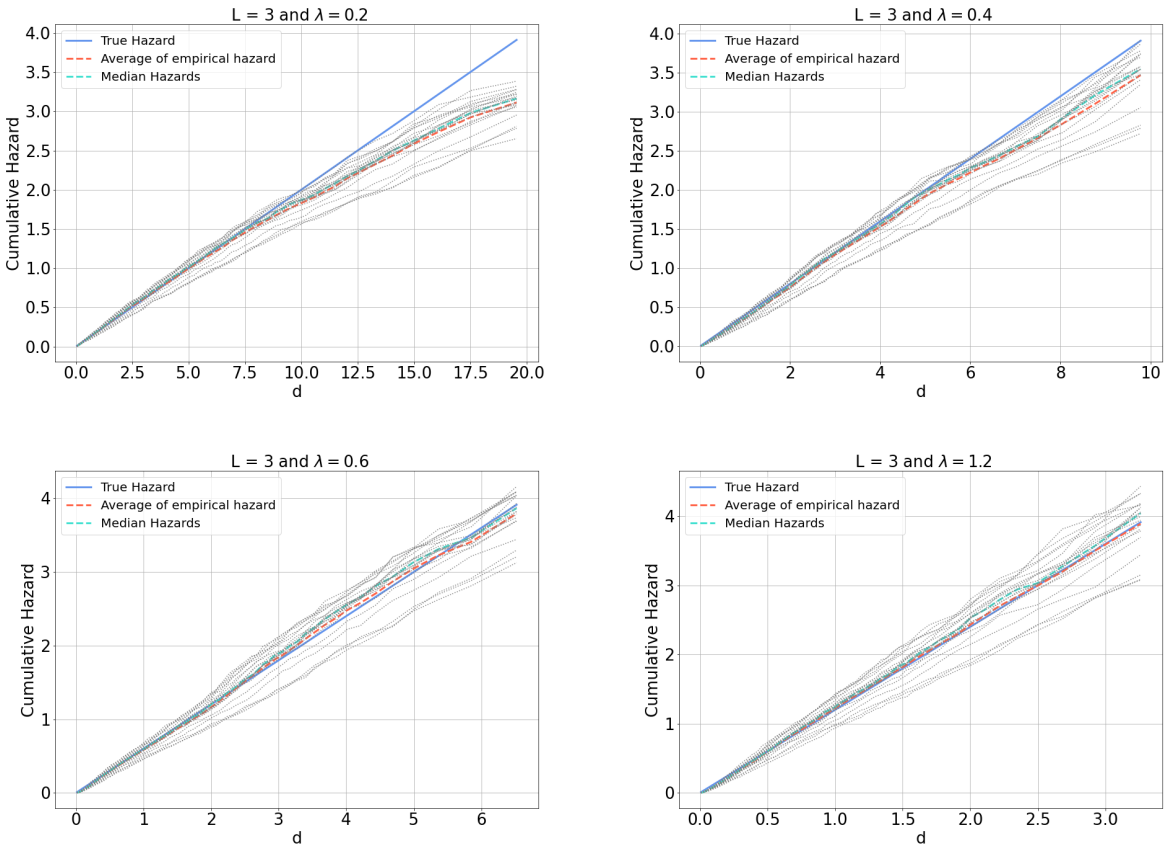

Figure 2.2: True hazard function of the scaled KL divergence (blue), and cumulative hazard for different generated data sets (gray). The dashed lines represent the median and mean of the empirical cumulative hazard.

We evaluate the approximation used in (2.5) to verify that the derived prior on $\boldsymbol{\tau}$ will result in an approximately exponential distribution on the KL divergence between the two models. We assume $L = 3$ layers with $\{64, 32, 1\}$ nodes respectively . The source-data estimates $\hat{\boldsymbol{\theta}}_i$ are generated following the Glorot initialization scheme (Glorot and Bengio 2010). Since we cannot compute the KL divergence analytically, the distribution of KL divergence is approximated with Monte Carlo sampling. The approximation is evaluated for 50 different target data sets generated as described in Section 4. For each of those data sets we compute the gradients in (2.5) and first draw $\mathbf{c}$ and $\tilde{\tau}$ from their priors for different values of the rate parameter λ, then sample $\boldsymbol{\theta}_i|\boldsymbol{\tau} \sim \text{Normal}(\hat{\boldsymbol{\theta}}_i, \tau_i \mathscr{I})$. We then sample 20,000 replications of the model output $f(\mathbf{x}; \boldsymbol{\theta})$ and approximate the KL divergence from the base model $f(\mathbf{x}; \hat{\boldsymbol{\theta}})$ for each replication to get the empirical distribution of the KL divergence. Figure 2.2 plots the cumulative hazard function of the scaled KL divergence d which should be linear with slope λ if the distribution is Exponential(λ). The approximation holds near the origin for all λ, and is tighter for the entire distribution for larger values of λ.

2.4 Simulation study

We perform the following simulation experiments to evaluate the predictive performance of the proposed Bayesian transfer learning methods. The data are generated as follows. For observation i, the covariates are generated as $x_{i1} \overset{iid}{\sim} \text{Normal}(0, 1)$ and $x_{ij}|x_{ij-1} \sim \text{Normal}(\rho\, x_{ij-1}, 1-\rho^2)$ for $j \in \{2, ..., p\}$. The source data are generated for constants, c, σ, and k_1, as

$$Y_i \sim \text{Normal}\left\{c\,\mu_1(\mathbf{x}_i), \sigma^2\right\} \text{ with } \mu_1(\mathbf{x}_i) = \cos\left(2\sum_{j=1}^{k_1} x_{ij}/k_1\right),$$

and the target data are generated, for constant k_2, as

$$Y_i \sim \text{Normal}\left\{c\,\mu_2(\mathbf{x}_i), \sigma^2\right\} \text{ with } \mu_2(\mathbf{x}_i) = \cos\left(2\sum_{j=1}^{k_2} x_{ij}/k_2\right).$$

We set $p = 30$ covariates so $\mathbf{x}_i \in \mathbb{R}^p$, $\rho = 0.5$, $\sigma = 1$, $k_1 = 15$, and $k_2 = k_1 + k$ with $k \in \{0, 5, 10, 15\}$, and we generate 1,000 synthetic source training observations with a varying number of target training observations $n \in \{30, 50, 70\}$. The constant c is set so that the signal-to-noise ratio is 4:1 for the target data (i.e., so the variance of $c\,\mu_2(\mathbf{x}_i)$ is 4). If $k = 0$ then the two data-generating processes are identical, and if k is large they are dissimilar. For each scenario, we simulate 500

data sets and compare predictions on a test set of size $n_{\text{test}} = 200$.

Both the source and target data are analyzed using the fully-connected model with four hidden layers having widths 24, 16, 12, and 8, respectively. The source data are fit using the optimization scheme given in Appendix A.2, which produces estimates $\hat{\boldsymbol{\theta}}$ and their variances in the case of VI. We then fit several models to each target data set. The first group are non-Bayesian optimization-based transfer learning "TL" methods. We consider two methods that fix some layers using the source fit and tune the remaining layers using the target data. The first model "TL1" shares no layers and is fit to the target data with no connection to the source data. The second model "TL2" proceeds in two stages; first, the output layer only is trained on the target data while the others are frozen, and next, in the second stage all layers are fine tuned.

The second group of methods are Bayesian. The first Bayesian model "BNN" ignores the source data by setting the prior $\boldsymbol{\theta}_i|\boldsymbol{\tau} \sim \text{Normal}(\prime, \tau_{\rangle}\mathscr{I})$ and $\tau_i \sim \text{InvGamma}(2,1)$ so that the marginal standard deviation is 1. Although, there is no consensus on the default prior choice in Bayesian deep learning, priors with zero mean and unit standard deviation are the most common (Fortuin 2021). The second method "BTL-PCP-PE" fits the Bayesian transfer learning model with the priors centered at point estimates $\hat{\boldsymbol{\theta}}$ alone, as described in Section 2.3.3. The third model "Bayes-PCP-VI" uses the PCP prior based on VI, as described in Section 2.3.2. For both PCP methods we assume $\mathbf{c} \sim \text{Dirichlet}(\infty_{\mathscr{L}})$ and set λ so that the correlation between $f(\mathbf{x}_i;\boldsymbol{\theta})$ and Y_i is 0.5. All Bayesian method are fit using MCMC as described in Appendix A.3.

The results are compared using MSE, and for the Bayesian methods the coverage and width of 95% prediction intervals for $\mu_2(\mathbf{x}_i)$ over a target data test set of 200 observations is provided. The results are reported in Table 2.1. BLT-PCP-PE outperforms TL1 and TL2 in all cases, perhaps due to the added stability of prior distributions. When the source and target are similar (i.e., $k \in \{0,5\}$), all the transfer learning methods work well, especially if the number of training samples is small. When the discrepancy between the source and target domain is high, however, the transfer learning methods do not improve the performance of the models. In fact, the point estimate method "TL-1" outperforms "TL-2", and the Bayesian methods are comparable. An interesting result is the poor performance of the PCP prior for the VI case compared to the other Bayesian methods. It seems that building a PCP prior based on the mean-field VI posterior approximation does not encode enough information for the model to learn from small data sets. However, while VI does not perform as well as MCMC, it is much faster than MCMC and still outperforms TL1 and TL2 in most cases and is thus a viable option for large data sets. Furthermore, when k is large, "BNN" outperforms all the other methods. This is expected since the prior distribution in the PCP has a sharp mode at zero which results in a strong penalizing

effect. Thus, when the data sets are dissimilar and the number of target training samples is small, BNN performs better. Perhaps choosing a different distribution for the prior, with heavier tails, on the scaled Kullback–Leibler divergence could reduce this dependence on the source in such cases. That being said, this could also lead to a less sharp mode at the origin and hence less concentration around the true solution when the data sets are similar.

Table 2.1: **Simulation study results**: The non-Bayesian methods ("TL1-TL2") use transfer learning by fixing different numbers of layers of the network using source data, the Bayesian methods are the Bayesian model without transfer learning ("BNN"), full Bayesian transfer learning model with point estimate ("BTL-PCP-PE") and VI ("BTL-PCP-VI") in the first stage. Table (a) gives the mean square prediction error (Monte-Carlo standard errors) over the 500 replicated data sets by the difference between the number of active covariates over tasks, k, and size of the target data set, n. Table (b) gives the average coverage (width) for the Bayesian 95% prediction intervals.

(a) Mean squared prediction error (Monte Carlo standard errors)

k	n	TL1	TL2	BNN	BTL-PCP-PE	BTL-PCP-VI
0	30	1.98 (0.06)	1.36 (0.03)	1.87 (0.06)	1.04 (0.03)	1.48 (0.05)
0	50	1.82 (0.09)	1.16 (0.03)	1.60 (0.03)	1.01 (0.04)	1.14 (0.09)
0	70	1.59 (0.08)	1.02 (0.04)	1.06 (0.03)	0.70 (0.02)	0.94 (0.05)
5	30	2.10 (0.05)	1.49 (0.09)	1.81 (0.05)	1.09 (0.02)	1.47 (0.05)
5	50	1.82 (0.05)	1.26 (0.04)	1.64 (0.03)	1.07 (0.04)	1.20 (0.03)
5	70	1.65 (0.07)	1.02 (0.05)	1.01 (0.04)	0.86 (0.02)	0.93 (0.01)
10	30	1.89 (0.10)	2.11 (0.05)	1.85 (0.09)	1.37 (0.03)	1.69 (0.03)
10	50	1.84 (0.06)	1.79 (0.05)	1.61 (0.08)	1.49 (0.06)	1.83 (0.07)
10	70	1.74 (0.10)	1.83 (0.04)	1.05 (0.04)	1.04 (0.05)	1.12 (0.06)
15	30	2.18 (0.09)	2.61 (0.03)	1.89 (0.04)	1.94 (0.02)	2.21 (0.04)
15	50	1.86 (0.07)	2.12 (0.05)	1.46 (0.08)	1.49 (0.07)	1.85 (0.08)
15	70	1.69 (0.05)	1.81 (0.06)	1.19 (0.03)	1.32 (0.02)	1.70 (0.03)

(b) Average coverage (Monte Carlo standard errors, average width) of 95% prediction intervals

k	n	BNN	BTL-PCP-PE	BTL-PCP-VI
0	30	0.86 (0.0089, 0.71)	0.94 (0.0076, 0.65)	0.78 (0.0118, 0.52)
0	50	0.88 (0.0081, 0.69)	0.95 (0.0057, 0.51)	0.82 (0.0095, 0.49)
0	70	0.93 (0.0072, 0.55)	0.98 (0.0044, 0.41)	0.90 (0.0099, 0.38)
5	30	0.86 (0.0096, 0.73)	0.91 (0.0099, 0.68)	0.76 (0.0102, 0.51)
5	50	0.90 (0.0083, 0.70)	0.95 (0.0064, 0.49)	0.83 (0.0094, 0.50)
5	70	0.92 (0.0064, 0.52)	0.97 (0.0056, 0.39)	0.90 (0.0087, 0.40)
10	30	0.85 (0.0099, 0.70)	0.90 (0.0092, 0.67)	0.73 (0.0134, 0.56)
10	50	0.91 (0.0090, 0.61)	0.94 (0.0065, 0.54)	0.89 (0.0106, 0.53)
10	70	0.95 (0.0081, 0.47)	0.97 (0.0057, 0.45)	0.91 (0.0096, 0.43)
15	30	0.82 (0.0097, 0.72)	0.78 (0.0096, 0.56)	0.65 (0.0127, 0.57)
15	50	0.91 (0.0082, 0.64)	0.90 (0.0091, 0.48)	0.80 (0.0119, 0.51)
15	70	0.95 (0.0068, 0.50)	0.94 (0.0074, 0.46)	0.88 (0.0081, 0.41)

To mimic the the real data analysis in Section 2.5, we conduct a second set of simulations with the same generative model as above but with $p = 1,000$ covariates, $\rho = 0.5$, $\sigma = 0.32$ and the constant $c = 1$ to approximate the 3:1 signal-to-noise in the data analysis. The number of active covariates is determined by $k_1 = 900$, and $k_2 = k_1 + k$ with $k \in \{0, 100\}$; if $k = 0$ then the two data-generating processes are identical, and if k is large they similar but not identical. We generate 10,000 synthetic source training observations and $n = 500$ target training observations. For each scenario, we simulate 100 data sets and compare predictions on a test set of size $n_{\text{test}} = 10,000$. For these larger data sets we also consider the VI approach of Zhou et al. (2020). The results Table 3.2 resemble those in Table 2.1, i.e., the BTL-PCP-PE approach has the smallest MSE and coverage near the nominal level.

Table 2.2: Approximate computational times of the methods used in the simulation study for data sets of size 1000. The Bayesian methods take approximately six times more time to train compared to classical methods.

Computational time per method					
	TL1	TL2	BNN	BTL-PCP-PE	BTL-PCP-VI
Time in minutes	15.12	15.09	98.04	103.41	21.05

Table 2.3: **Simulation study results**: The non-Bayesian methods ("TL1-TL2") use transfer learning by fixing different numbers of layers of the network using source data, the Bayesian methods are the Bayesian model without transfer learning ("BNN"), full Bayesian transfer learning model with point estimate ("BTL-PCP-PE") and VI ("BTL-PCP-VI") in the first stage, and Bayesian transfer learning prior from Zhou et al. (2020) "BTL-VI". Table (a) gives the mean square prediction error by the difference between the number of active covariates over tasks, k, and size of the target data set, n. Table (b) gives the average coverage (width) for the Bayesian 95% prediction intervals.

(a) Mean squared prediction error (standard errors)

k	n	TL1	TL2	BNN	BTL-PCP-PE	BTL-PCP-VI	BTL-VI
0	500	0.68 (0.12)	0.65 (0.08)	0.76 (0.11)	0.44 (0.08)	0.68 (0.09)	0.70 (0.15)
100	500	0.70 (0.08)	0.72 (0.09)	0.72 (0.13)	0.47 (0.11)	0.71 (0.12)	0.73 (0.13)

(b) Average coverage (average width) of 95% prediction intervals

k	n	BNN	BTL-PCP-PE	BTL-PCP-VI	BTL-VI
0	500	0.90 (0.41)	0.94 (0.35)	0.79 (0.22)	0.82 (0.19)
100	500	0.89 (0.35)	0.92 (0.38)	0.73 (0.18)	0.80 (0.24)

2.5 Real data analysis

We apply the proposed transfer learning methods for band gap prediction of the molecular crystal target data set. For training, we consider only subsets of the data with cardinality 1,000, 2,500, or 5,000 samples to represent a typical data set size, a validation set of 10,000 samples and finally a testing set of 20,000 samples. Since we are dealing with molecules, we first need to preprocess the data and compute descriptors $\mathbf{x}$ that can be fed into the DNN. We use the MBTR descriptor method (Huo and Rupp 2018) to compute $p = 1,260$ descriptors for every data point in both the source and target data sets. These descriptors are functions of the structure of the material, e.g., the types and configuration of its atoms. The architecture of the model will have two hidden layers with 128 and 64 neurons, respectively. First the model is trained on the source data with 40,000 samples for training set and 10,000 for the validation. The parameters of the best performing model in terms of mean squared error (MSE) are saved and transferred to the target task as $\hat{\boldsymbol{\theta}}$.

On the target task, we consider four different transfer learning methods. The non-Bayesian methods differ in the treatment of the hidden layers of the model; we can either freeze the hidden layers and treat them as feature extractors and only fit the output layer to the target task "TL-freeze", or we can fine-tune the hidden layers along with the output layer on the target task "TL-fine-tune". For both Bayesian methods, the priors are $\boldsymbol{\theta}_i|\boldsymbol{\tau} \sim \text{Normal}(\hat{\boldsymbol{\theta}}_i, \tau_i \mathscr{I})$. The two methods differ in the priors on $\boldsymbol{\tau} = (\tau_1, \tau_2, \tau_3)$; we compare the PCP with point estimate approach "BTL-PCP-PE" from Section 2.3.3 with $\mathbf{c} \sim \text{Dirichlet}(1,1,1)$ to the uninformative prior approach with $\tau_i \sim \text{InvGamma}(2,1)$ "BTL-BNN". Since the point estimate approach outperformed the VI approach in the simulation study we consider only the point estimate approach in this data analysis. To verify that transfer learning provides benefit over using only the target data, we compare to the state of the art DNN model used in materials science where DNNs have been developed as end-to-end models in the sense that they take as input the molecular representation and learn their own embedding. Two of the leading methods are SHNET (Schütt et al. 2017) and MEGNET (Chen et al. 2019).

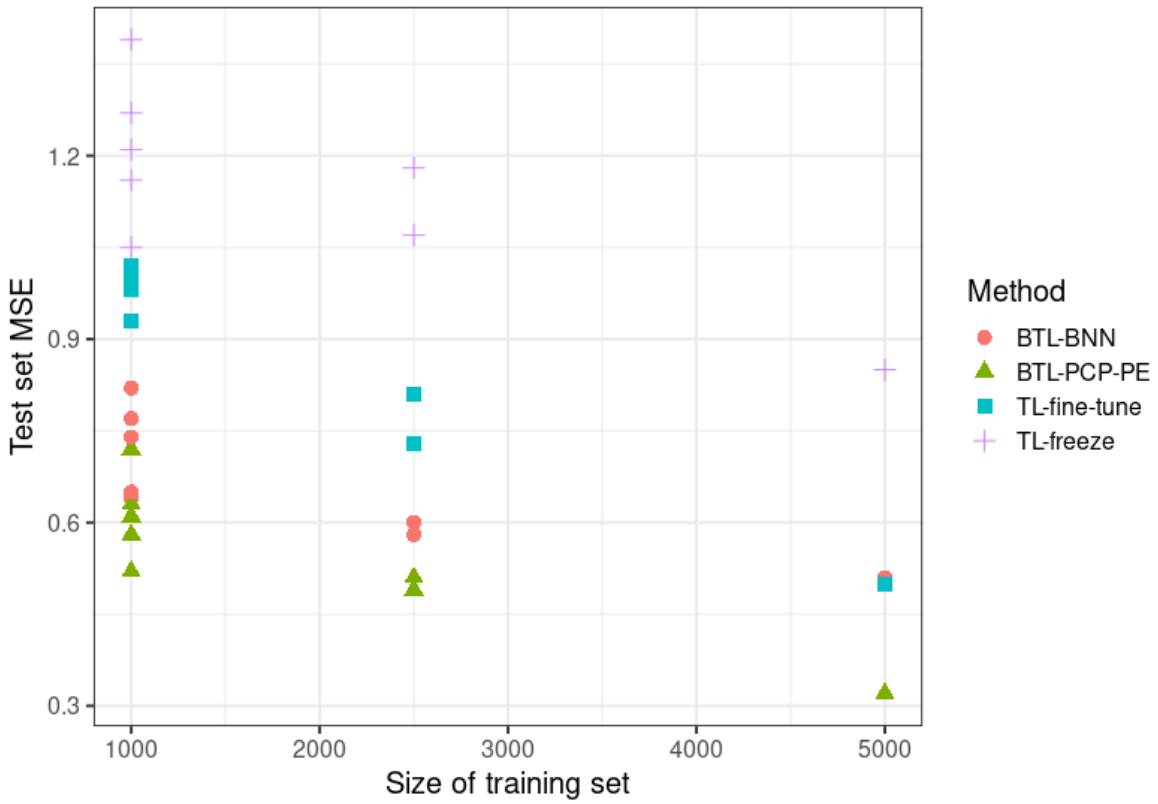

Figure 2.3: Test set mean squared error (MSE; eV^2) for different transfer learning methods applied to band-gap prediction. The methods are plotted against the size of the target data training set. Each point represents the MSE for one fold in the cross validation. For comparison, the sample variance of the response is 1.12 eV^2.

The target data are split into groups of size 5,000, 10,000, and 20,000 for training, so that each group contains materials of different classes and the band gap distribution stays roughly the same. We train the model for samples of size $n \in \{1000, 2500, 5000\}$. Figure 2.3 plots the MSE on a test set of size 20,000 for each method and training set, and Figure 2.4 plots the observed and predicted values for the BTL-PCP-PE method. As expected, MSE decreases for each method as the size training set increases, and in all cases the BTL-PCP-PE method gives the smallest MSE.

With a fully Bayesian treatment of all the parameters, we can look at the summary statistics of $\mathbf{c}$, the parameter that controls which layers contain the useful information from the source data and which are irrelevant. For training set size 5,000, the posterior means (standard deviation) for c_1, c_2 and c_3 are 0.05 (0.02), 0.05 (0.03) and 0.90 (0.15) respectively. Therefore, the first two

layers are shrunk towards the source data fits while the output layer varies from the source-model fit. The Bayesian methods also provide prediction intervals. The empirical coverage and average width of 95% prediction intervals for the test set are, respectively, 0.91 and 1.24 for BTL-BNN and 0.94 and 1.16 for BTL-PCP-PE.

These results are competitive with state-of-the-art methods in the materials community. For a training set of size 5,000 (the same 5,000 as in Figure 2.3) the MSE for SHNET and MEGNET are 0.48 eV^2 and 1.01 eV^2, respectively. In fact, using the same data except with a much larger training set of 30,000 observations, the methods achieved MSE of 0.59 eV^2 for SHNET and 0.34 eV^2 for MEGNET.

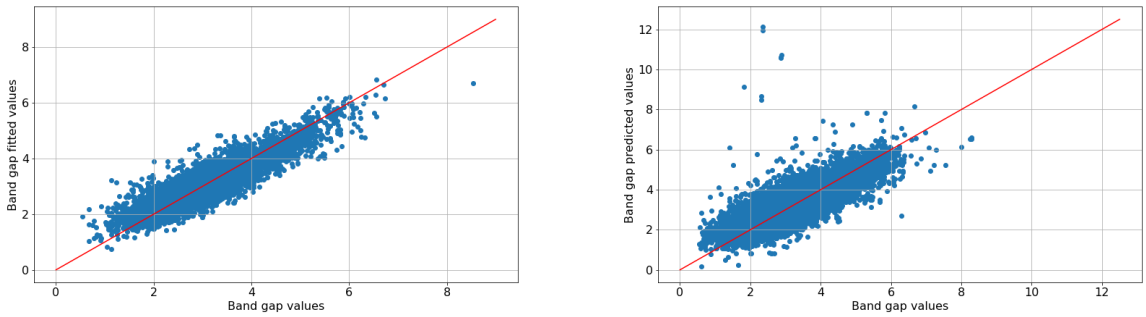

Figure 2.4: Predicted versus band-gap for the training (left) and test (right) sets for the BLT-PCP-PE methods with $n = 5,000$ training observations. The units of the plot are electronvolts.

2.6 Discussion

For molecular crystals, the computational cost of band gap values can be orders of magnitude higher in comparison to organic molecules. Recently, DNN models reached chemical accuracy levels for band gap prediction on small organic molecules, and can now be reliably used as a fast screening method. However, these models often require large data sets for training and can have unstable performance on small data sets. Using transfer learning, we leverage the accuracy of neural networks models on organic molecules to build a data efficient model with accurate predictions and calibrated uncertainty quantification for molecular crystals band gap prediction. Our new transfer learning methods provide regularization by centering the prior distribution on estimates from an auxiliary source data set, *i.e.*, organic molecules. We use the PCP framework to ensure that the prior concentrates around the source model but is allowed

to deviate if appropriate. We develop two methods for cases where the source data analysis does and does not provide uncertainty measures for the parameter estimates. We show via simulation that the proposed methods reduce prediction error compared to standard transfer learning methods, and unlike standard methods the Bayesian approach gives reasonable coverage for prediction intervals.

The accuracy of our method for large molecular crystals matches state of the art models while only requiring a sixth of the training data. The performance on both simulated and real data sets indicate that the methods can be used to guide and optimize experimentation and selection of materials with specific properties without the need for a large training sample, hence reducing the cost associated with DFT calculations. The potential scope of the application of the developed transfer learning methods goes beyond material science and includes areas of applied statistical learning where data labelling is costly and auxiliary source tasks are available.

There are many areas of future work. We have developed our method in the simplest case of a continuous response and feed forward network. The prior based on VI in Section 2.3.2 would apply directly to other networks and response distributions as the prior is only a function of the approximate Gaussian posterior distribution from the source data analysis. Similarly, the prior based on point estimates in Section 2.3.3 can be applied without modification to non-Gaussian responses and richer architecture. Changing the network architecture would change the form of the gradients in (2.5) but the method itself can be used without modification. The method also applies for non-continuous responses if we view f as a process that spans the real line that is related to the response distribution. For example, for binary data, f in (2.5) could be the logistic function of the success probability, which spans the real line and thus the proposed methods could be applied on this scale. It may also be possible to extend the PCP prior from neural networks to other frameworks such as tree-based methods (e.g., Chipman et al. 2010; Chen and Guestrin 2016).

Next, our method applies an informative prior on the overall variance parameter, but a uniform prior on the distribution of the variance across levels, $\mathbf{c}$. If prior information about the transferability of different layers is available, it could be incorporated in the prior for $\mathbf{c}$. For example, it may be reasonable to assume the prior mean of c_i increases from the input to the output layer. Lastly, another interesting area of future work is to compare the two-stage analysis that sequentially analyzes the source then target data with a simultaneous analysis of all data sources. Here we have assumed that building a hierarchical model for all data sources is computationally prohibitive, but it would be interesting to compare the efficiency of these two

approaches when possible, especially when the source data set is small and thus the parameter estimates are uncertain.

CHAPTER

3

A BAYESIAN SHRINKAGE ESTIMATOR FOR TRANSFER LEARNING

3.1 Introduction

In the field of statistical inference and modeling, the effective utilization of available data is vital for achieving accurate and robust results, but in many real-world scenarios, obtaining sufficient and high-quality data can be challenging. *Transfer learning*, a concept originally popularized in the domain of deep learning (e.g., Yosinski et al. 2014; Abba et al. 2023), has gained prominence for its potential to address data scarcity and enhance model performance of various machine learning tasks (Weiss et al. 2016). The fundamental idea of transfer learning is to leverage a task trained from a source domain (associated with a population for which data is readily available) and apply it in training a task in a target domain (associated with a related but distinct population for which limited data are available) to improve predictive performance in the target domain (Pan and Yang 2010).

Our paper establishes theoretical guarantees for transfer learning in the classical *normal means problem*, and naturally extends to high-dimensional linear regression. We develop a

two-stage Bayesian method based on shrinkage priors (e.g., van Erp et al. 2019) in two settings. In the first setting, we assume that the difference between the source and target means is *sparse* and show under mild conditions that our two-stage method achieves considerable risk reduction compared to using the target data only. In the second setting, we focus on the case where the difference in means is bounded in norm, and provide the form of the minimax linear estimator.

For high-dimensional regression problems, penalized approaches have been shown to provide a safeguard against over-fitting and result in estimators with optimal predictive risk in specific scenarios (McNeish 2015). Further, penalization fits well within the Bayesian framework, and it is demonstrated in Li and Lin (2010); van der Pas et al. (2014) and van der Pas et al. (2017) that the Bayesian formulation of penalized regression via shrinkage priors leads to at least as good performance as frequentist analogues. Accordingly, Bayesian shrinkage priors have become a popular tool for inference in high dimensional problems; see (van Erp et al. 2019) for a complete survey. A notable advantage of the Bayesian approach is the natural uncertainty quantification inherent in posterior probabilistic inference.

Approaches to Bayesian transfer learning for linear regression have been considered previously in the literature, leveraging the convenience to borrow strength across data sets via prior distributions. To our knowledge, however, there has not been any theoretical analysis of Bayesian transfer learning methods for the normal means or high-dimensional linear regression problems. The recent Bayesian transfer learning approach developed in Hickey et al. (2022) applies to the normal means and high-dimensional linear regression problems, but focuses on the calibration of prediction sets. A prior distribution constructed using semi-definite programming is presented in Raina et al. (2006) to build a multivariate Gaussian prior on the target parameters; significant empirical improvement in the target task is reported. A Bayesian transfer learning framework is proposed in Karbalayghareh et al. (2018), in the case where the source and target domains are modeled through a joint prior density on the model parameters. A prior distribution is developed in Dzyabura et al. (2019) to link the source task outcomes to the target, and the method is applied to the online and offline behaviors of customers; this approach could be viewed as a ridge penalty. Other than that of Hickey et al. (2022), all of these methods require access to the source data, which presents difficulties in many domains with privacy restriction. Our proposed method only requires access to the estimates from the source task.

Outside of the Bayesian paradigm, a variety of statistical methods for transfer learning in linear regression have been proposed in recent years. Two-sample hypothesis testing problems

are investigated in multiple testing scenarios with a large number of tests in Xia et al. (2019) and Cai et al. (2019). High-dimensional linear models are focused on in Bastani (2021) with one source task, and a two-stage estimator is developed with bounded excess risk when the sample size of the source task is larger than the number of covariates. A nearly optimal method for selection of informative source task samples is constructed in Li et al. (2022) in the case where the model parameters are sparse for both the source and target tasks. Lastly, estimators were developed in Lei et al. (2022) that achieve near-minimax linear risk for linear regression problems under covariate shift. Aside from our Bayesian construction, another key difference between our method and the aforementioned papers is that we assume no structure on the source task parameters.

The remainder of the paper proceeds as follows. Sections 3.2 and 3.3 introduce the statistical method for the normal-means problem and establish its theoretical properties. A simulation study is presented in Section 4.4 to empirically evaluate the performance of our proposed methods compared to other approaches. In Section 3.5 we develop an extension to the linear regression case and apply our method to fine-tuning the last layer of a neural network for material informatics prediction. Our paper concludes in Section 4.5 with a few final remarks.

3.2 Statistical model

Denote the source domain data as $\mathbf{Y}_{11},\ldots,\mathbf{Y}_{1n_1} \in \mathbb{R}^p$ and the target domain data as $\mathbf{Y}_{21},\ldots,\mathbf{Y}_{2n_2} \in \mathbb{R}^p$. We assume the following data generating model:

$$\begin{aligned}
&\mathbf{Y}_{11},\ldots,\mathbf{Y}_{1n_1}|\boldsymbol{\beta}_1,\sigma^2 \overset{iid}{\sim} \text{Normal}(\boldsymbol{\beta}_1,\sigma^2\mathbf{I}_p), \text{ independent of}\\
&\mathbf{Y}_{21},\ldots,\mathbf{Y}_{2n_2}|\boldsymbol{\beta}_2,\sigma^2 \overset{iid}{\sim} \text{Normal}(\boldsymbol{\beta}_2,\sigma^2\mathbf{I}_p),
\end{aligned}$$

where $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are the p-dimensional vectors of means for the source and target data, respectively, and the noise level σ^2 is assumed to be the same across data sets. Our primary inferential goal is to estimate the target mean vector $\boldsymbol{\beta}_2$ using target data and augmented information from the source data.

In the case where the two data sets are similar and $n_2 \ll n_1$ we will argue that the inferential goal is achieved by choosing a prior that penalizes the difference between the means, $\boldsymbol{\delta} := \boldsymbol{\beta}_2 - \boldsymbol{\beta}_1$. If $\boldsymbol{\delta} = \prime$ the most informative prior for $\boldsymbol{\beta}_2$ is the posterior of $\boldsymbol{\beta}_1$ given $\mathbf{Y}_1 = \{\mathbf{Y}_{11},\ldots,\mathbf{Y}_{1n_1}\}$. However, if $\boldsymbol{\delta}$ is different from zero, this prior would be concentrated around the wrong region of the parameter space. We consider two priors for two different cases for the structure of $\boldsymbol{\delta}$.

In the first setting (Section 3.2.1), we assume that $\boldsymbol{\delta}$ is sparse with most entries being exactly zero, implying most elements of $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are equal. In the second case (Section 3.2.2), the ℓ_2-norm of $\boldsymbol{\delta}$ is assumed to be bounded by some constant, implying none of the elements of $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are equal but that their differences are small.

We consider a two-stage approach, where we first fit the source model given the data $\mathbf{Y}_1$ and an improper flat prior for $\boldsymbol{\beta}_1$, resulting in the posterior for $\boldsymbol{\beta}_1 \mid \mathbf{Y}_1, \sigma^2 \sim \text{Normal}(\overline{\mathbf{Y}}_1, \sigma^2/n_1\mathbf{I}_p)$, where $\overline{\mathbf{Y}}_1 := \sum_{i=1}^{n_1} \mathbf{Y}_{1i}/n_1$. The second stage is to estimate of the target mean vector by $\hat{\boldsymbol{\beta}}_2 := \hat{\boldsymbol{\beta}}_1 + \hat{\boldsymbol{\delta}}$, where $\hat{\boldsymbol{\beta}}_1 := \mathbb{E}[\boldsymbol{\beta}_1 \mid \mathbf{Y}_1]$, i.e., the posterior mean from the source data, and the estimator $\hat{\boldsymbol{\delta}}$ is to be determined in the following sections. We assume that the source data sample size n_1 is sufficiently large so that the quality of our target estimate will mainly depend on $\hat{\boldsymbol{\delta}}$ and its assumed structure. The two structures we assume are sparse $\boldsymbol{\delta}$ and $\boldsymbol{\delta}$ with bounded norm, and in both cases we are able to evaluate the *risk* of our estimator and compare it to the best possible estimators based on minimax optimality.

3.2.1 Sparse case

In the case of sparse $\boldsymbol{\delta}$, the prior must have considerable mass for large component values of $\boldsymbol{\delta}$ while still having a spike at the origin. This can be achieved using a horseshoe prior (Carvalho et al. 2009). The horseshoe prior belongs to the family of global-local shrinkage priors which have been widely applied for sparse Bayesian learning. The horseshoe prior is characterized by an infinite mode at the origin and Cauchy-like, heavy tails. This translates in robust, large signal recovery and adaptation to unknown sparsity (Carvalho et al. 2009). Furthermore, the theoretical properties of the horseshoe prior have been well-studied (Datta and Ghosh 2013). van der Pas et al. (2014) show that when the true parameter vector is sufficiently sparse, the posterior mean under a horseshoe prior achieves the minimax rate and that the posterior contracts at the minimax rate.

Denote the observed mean of the target data by $\overline{\mathbf{Y}}_2 := \sum_{i=1}^{n_2} \mathbf{Y}_{2i}/n_2$, and note that $\overline{\mathbf{Y}}_1$ and $\overline{\mathbf{Y}}_2$ are sufficient statistics for $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$, respectively.That being true, the data generating models for these sufficient statistics, for every $j \in \{1, \ldots, p\}$, are

$$\overline{Y}_{1j} \sim \text{Normal}(\beta_{1j}, \sigma^2/n_1)$$
$$\overline{Y}_{2j} \sim \text{Normal}(\beta_{1j} + \delta_j, \sigma^2/n_2).$$

Next, let $Z_j := \overline{Y}_{2j} - \overline{Y}_{1j}$ and $\sigma_n^2 := \sigma^2(\frac{1}{n_1} + \frac{1}{n_2})$ to reparameterize the above model as

$$\begin{aligned} \overline{Y}_{1j} &\sim \text{Normal}(\beta_{1j}, \sigma^2/n_1) \\ Z_j &\sim \text{Normal}(\delta_j, \sigma_n^2). \end{aligned} \tag{3.1}$$

In this parameterization, Y_{1j} and Z_j are sufficient statistics, respectively, for β_{1j} and δ_j. We specify a horseshoe-like priors δ_j for each $j \in \{1, \ldots, p\}$, as

$$\begin{aligned} \delta_j &\sim \text{Normal}(0, \lambda_j^2 \tau^2 \sigma_n^2) \\ \lambda_j &\sim \text{Cauchy}_+(0, 1). \end{aligned} \tag{3.2}$$

This prior specification on δ_j depends on two scale parameters. The *global shrinkage* parameter τ can be tailored to be small to encourage sparsity in the posterior concentration for $\boldsymbol{\delta}$, while the *local* parameters λ_j allow large signals to escape the global shrinkage. In the edge case that $\tau = 0$ then $\delta_j = 0$ and $\beta_{2j} = \beta_{1j}$. In a fully Bayesian specification, a non informative prior is placed on the *global shrinkage* parameter, $\tau \sim g(\tau)$. Carvalho et al. (2009) argue for a Half-Cauchy prior. Similarly, the error variance is given an uninformative prior, e.g., $\sigma^2 \sim \text{IG}(a, b)$. Alternatively, we can view τ as a hyperparameter and estimate it using empirical Bayes methods or marginal maximum-likelihood. van der Pas et al. (2014) noticed that marginal maximum-likelihood estimates run the risk of collapsing to zero, and propose to constrain possible values of τ to $(1/p, \infty)$.

With the formulation in (3.1) and (3.2),

$$\delta_j \mid Z_j, \lambda_j, \tau, \sigma^2 \sim \text{Normal}\{(1-\kappa_j)Z_j, (1-\kappa_j)\sigma_n^2\}, \tag{3.3}$$

where $\kappa_j = 1/(1+\lambda_j^2\tau^2)$. The interpretation of the posterior shrinkage parameter κ_j is discussed in Carvalho et al. (2009). If λ_j is large then $\kappa_j \approx 0$, δ_j follows a normal distribution with mean Z_j and variance σ_n^2, and there is effectively no transfer learning. On the other hand, if $\lambda_j \approx 0$ then $\kappa_j \approx 1$ and $\delta_j \approx 0$, corresponding to strong transfer learning.

Using the conditional posterior mean $\tilde{\delta}_j := \mathbb{E}[\delta_j \mid Z_j, \kappa_j, \tau, \sigma^2]$ results in the point estimate for β_{2j}:

$$\hat{\beta}_{2j} = \tilde{\delta}_j + \hat{\beta}_{1j} = (1-\kappa_j)Z_j + \hat{\beta}_{1j}.$$

In Section 3.3, we focus on the properties of the Bayes estimator $\hat{\delta}_j = \mathbb{E}[\delta_j | Z_j, \tau, \sigma_n^2]$ and derive its asymptotic properties under assumptions of suitable values of τ. Hence we need to integrate

out κ_j in (3.3). The marginal posterior of κ_j conditional on τ and σ^2 has density function

$$p(\kappa_j|Z_j,\tau,\sigma^2)=\frac{(1-\kappa_j)^{\frac{-1}{2}}\frac{1}{1-(1-\tau^2)\kappa_j}e^{-\frac{Z_j^2}{2\sigma_n^2}\kappa_j}}{\int_0^1(1-\kappa)^{\frac{-1}{2}}\frac{1}{1-(1-\tau^2)\kappa}e^{-\frac{Z_j^2}{2\sigma_n^2}\kappa}d\kappa},$$

and so with reference to equation (3.3),

$$\begin{aligned}
\hat{\delta}_j &= (1-\mathbb{E}[\kappa_j|Z_j,\tau,\sigma_n^2])Z_j \\
&= Z_j\cdot\frac{\int_0^1(1-\kappa)^{\frac{1}{2}}\frac{1}{1-(1-\tau^2)\kappa}e^{-\frac{Z_j^2}{2\sigma_n^2}\kappa}d\kappa}{\int_0^1(1-\kappa)^{-\frac{1}{2}}\frac{1}{1-(1-\tau^2)\kappa}e^{-\frac{Z_j^2}{2\sigma_n^2}\kappa}d\kappa} \\
&= Z_j\cdot\frac{\int_0^1 u^{\frac{1}{2}}\frac{1}{\tau^2+(1-\tau^2)u}e^{-\frac{Z_j^2}{2\sigma_n^2}(1-u)}du}{\int_0^1 u^{-\frac{1}{2}}\frac{1}{\tau^2+(1-\tau^2)u}e^{-\frac{Z_j^2}{2\sigma_n^2}(1-u)}du}, \text{ where } u=1-\kappa \\
&= Z_j\cdot w_{\tau,\sigma_n^2}(Z_j), \qquad (3.4)
\end{aligned}$$

where

$$w_{\tau,\sigma_n^2}(Z_j)=\frac{\int_0^1 u^{\frac{1}{2}}g(u)e^{\frac{Z_j^2}{2\sigma_n^2}u}du}{\int_0^1 u^{-\frac{1}{2}}g(u)e^{\frac{Z_j^2}{2\sigma_n^2}u}du}, \text{ with } g(u)=\frac{1}{\tau^2+(1-\tau^2)u}. \qquad (3.5)$$

In other words, $\hat{\delta}_j$ shrinks the observation $Z_j=\overline{Y}_{2j}-\overline{Y}_{1j}$ by a weight $w_{\tau,\sigma_n^2}\in(0,1)$. Furthermore, the shrinkage weights are symmetric and strictly increasing in terms of the magnitude of the difference $|Z_j|$. If $|Z_j|$ is sufficiently small $w_{\tau,\sigma_n^2}(Z_j)$ is almost zero. On the other hand, $w_{\tau,\sigma_n^2}(Z_j)\to 1$ as $|Z_j|\to\infty$. Hence the posterior mean is asymptotically unbiased for sufficiently large signals, and shrinks small signals to zero.

For a fully Bayesian treatment of the model, we need to integrate over τ and σ^2 in (3.4) to get the marginal posterior mean $\mathbb{E}[\delta_j \mid Z_j]$. However, this is analytically intractable and we rely on Markov chain Monte Carlo (MCMC) sampling to approximate this quantity. van der Pas et al. (2017) showed that the horseshoe prior retains its optimal risk properties under both full Bayesian estimation and marginal maximum likelihood plug-in estimates as long as the latter are truncated to the left by $1/p$.

With the sparsity assumption, and the theoretical guarantees for the horseshoe prior, our second stage estimation procedure is optimal in recovering $\boldsymbol{\delta}$. Furthermore, abundant source

task data guarantees accurate first stage estimates. Thus we expect our two stage approach estimator to achieve a low error rate on the target task parameters.

3.2.2 Bounded norm case

In this section, we relax to the assumption that $\boldsymbol{\delta}$ is sparse to the assumption that it is bounded in norm, i.e., $\|\boldsymbol{\delta}\| := \left(\sum_1^p \delta_j^2\right)^{1/2} < C$ for some unknown constant C. In the absence of a sparsity assumption only global shrinkage is warranted. Therefore, we consider the following global shrinkage prior:

$$\boldsymbol{\delta} \sim \text{Normal}(\prime, \tau^2 \mathbf{I}_p). \tag{3.6}$$

The global shrinkage parameter has prior $\tau^2 \sim g(\cdot)$.

After the first-stage we have the posterior $\beta_{1j} \mid \overline{Y}_{1j} \overset{ind}{\sim} \text{Normal}(\overline{Y}_{1j}, \sigma^2/n_1)$. This posterior forms the prior in the second-stage estimation. Hence model for the second stage is

$$\begin{aligned} \overline{Y}_{2j} &\sim \text{Normal}(\beta_{2j}, \sigma^2/n_2) \\ \beta_{2j} &\sim \text{Normal}(\overline{Y}_{1j}, \sigma^2/n_1 + \tau^2) \end{aligned} \tag{3.7}$$

The choice of the prior on τ^2 will determine how heavy the tails are as well how sharp the mode is around $\overline{\mathbf{Y}}_1$. To examine the effect of g on the posterior of $\boldsymbol{\beta}_2$ we first look at the conditional posterior:

$$\beta_{2j} \mid \overline{Y}_{2j}, \overline{Y}_{1j}, \tau, \sigma^2 \sim \text{Normal}(\mu_j(\tau^2), v_j(\tau^2)) \tag{3.8}$$

$$\mu_j(\tau^2) = \overline{Y}_{Tj} + \frac{\tau^2/\sigma_n^2}{1+\tau^2/\sigma_n^2}(1-q_n)(\overline{Y}_{2j} - \overline{Y}_{1j}) \tag{3.9}$$

$$v_j(\tau^2) = \frac{\sigma^2}{n_2}\frac{q_n + \tau^2/\sigma_n^2}{1+\tau^2/\sigma_n^2} \tag{3.10}$$

for $\overline{Y}_{Tj} = (n_1\overline{Y}_{1j} + n_2\overline{Y}_{2j})/n_T$, $n_T = n_1 + n_2$ and $q_n = \frac{n_2}{n_T}$. Let $\kappa = \frac{\tau^2/\sigma_n^2}{1+\tau^2/\sigma_n^2}$. If $\tau^2 = 0$ then $\kappa = 0$ and the model collapses to pooling the two data sets. On the other hand, if $\tau^2 = \infty$ then $\kappa = 1$ and the model is equivalent to the analysis with target data only. Hence τ^2 controls the influence of the source model in the analysis. A prior penalizing large values of τ^2 will in turn penalize deviation from the source estimates.

We use the Penalized complexity prior (PCP) (Simpson et al. 2017a) on τ^2. In the PCP framework, we first start with a base model which in our case is $\tau^2 = 0$ *i.e.,* $\boldsymbol{\beta}_2 = \boldsymbol{\beta}_1$, and then penalize the deviance of the extended model $\tau^2 > 0$ from the base model through an

exponential prior on the square root of the Kullback-Leibler (KL) divergence between the two. Let π_0 and π_τ denote the prior distribution on $\boldsymbol{\beta}_2$ in (3.7) for the base and extended models respectively. Let $\tilde{\tau}^2 = \tau^2/(\sigma^2/n_1)$, and define the scaled KL divergence between the two models

$$d(\tilde{\tau}^2) = \sqrt{\frac{2}{p} D_{\mathrm{KL}}(\pi_{\tilde{\tau}} || \pi_0)} = \sqrt{\tilde{\tau}^2 - \log(1+\tilde{\tau}^2)}. \tag{3.11}$$

Clearly d is a strictly increasing function of $\tilde{\tau}^2$. Furthermore, if $\tilde{\tau}^2 = 0$ then $d(\tilde{\tau}^2) = 0$ and the model collapses to the base model. Assigning an exponential distribution on $d(\tilde{\tau}^2) \sim \mathrm{Exp}(\lambda)$ induces a prior g on $\tilde{\tau}^2$ with density

$$g(\tilde{\tau}^2) = p(d(\tilde{\tau}^2)) \times \left| d'(\tilde{\tau}^2) \right| = \frac{\lambda \tilde{\tau}^2}{1+\tilde{\tau}^2} \frac{e^{-\lambda\sqrt{\tilde{\tau}^2 - \log(1+\tilde{\tau}^2)}}}{2\sqrt{\tilde{\tau}^2 - \log(1+\tilde{\tau}^2)}}. \tag{3.12}$$

It can be shown that the induced prior on $\tilde{\tau}^2$ in (3.12) has a mode at zero with a strictly decreasing density as $\tilde{\tau}^2$ increases. Also, the tail of the prior behaves like a modified Weibull distribution (Almalki and Nadarajah 2014) with rate λ and shape 0.5. The hyper-parameter λ controls the magnitude of the penalty for deviating from the base model and can be set using prior belief on the difference between the two tasks. Simpson et al. (2017a) studied the properties of PCP prior in estimating the a vector of normal means and concluded that PCP priors will result in more mass at intermediate shrinkage and concluded that this is appropriate for cases when the signals are small. Under our assumption $\|\boldsymbol{\delta}\|$ is bounded, so when p is large, we expect all the entries δ_j to be small, hence our choice of the PCP prior.

3.3 Theoretical properties

3.3.1 Risk of the Horseshoe estimator in the sparse case

In what follows, for notational clarity the subscripts of the expectation operators specifies that we are taking the expectation with respect to the data distribution given the value of the subscript, e.g.,

$$\mathbb{E}_{\theta^o}[\|x - \theta^o\|^2] = \int (x - \theta^o) f(x \mid \theta^o) dx.$$

In this section we show the theoretical properties of the proposed transfer learning method. Let $\boldsymbol{\beta}_1^o$, and $\boldsymbol{\beta}_2^o$ be the true values of source and target means respectively, and let $\boldsymbol{\delta}^o = \boldsymbol{\beta}_2^o - \boldsymbol{\beta}_1^o$. First we consider the case where $p = o(n_1)$ and n_2 is fixed to reflect the setting with large source

data and sparse target data. Without loss of generality, assume $n_2 = 1$. In this case the MLE of $\boldsymbol{\beta}_1$ is $\hat{\boldsymbol{\beta}}_1 = \overline{\mathbf{Y}}_1$ and we have:

$$\mathbb{E}_{\boldsymbol{\beta}_1^o}[||\overline{\mathbf{Y}}_1 - \boldsymbol{\beta}_1^o||^2] = \frac{p}{n_1}\sigma^2 \longrightarrow 0 \text{ as } p, n_1 \longrightarrow \infty. \tag{3.13}$$

So the MLE for the source task estimates is consistent. Furthermore, assume $\boldsymbol{\delta}^o$ is a 'nearly black object', that is all but $q = \|\boldsymbol{\delta}^o\|_0$ elements of $\boldsymbol{\delta}^o$ are exactly zero with $q = o(p)$. We will show that the risk for estimating $\boldsymbol{\beta}_2$ using our two-stage approach attains the minimax risk for estimating a sparse vector of means. Recall that $\hat{\boldsymbol{\beta}}_2 = \overline{\mathbf{Y}}_1 + \hat{\boldsymbol{\beta}}$, where $\hat{\boldsymbol{\delta}} = \mathbb{E}[\boldsymbol{\delta} \mid \mathbf{Z}, \tau, \sigma_n^2]$.

If $q = o(p)$ then $\hat{\boldsymbol{\delta}}$ attains the minimax risk for a suitable choice of $\tau = (\frac{q}{p})^\alpha$, with $\alpha \geq 1$ (van der Pas et al. 2014) :

$$\mathbb{E}_{\boldsymbol{\delta}^o}[||\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}^o||^2] \leq \sigma_n^2 q \log(\frac{p}{q})(1 + o(1)). \tag{3.14}$$

Using (3.13) and(3.14), we can get an upper bound on the risk of the Bayes estimator for the target task $\hat{\boldsymbol{\beta}}_2 = \overline{\mathscr{Y}}_1 + \hat{\boldsymbol{\delta}}$. The risk can be written as

$$\mathbb{E}_{\boldsymbol{\beta}_2^o}[||\hat{\boldsymbol{\beta}}_2 - \boldsymbol{\beta}_2^o||^2] = \mathbb{E}_{\boldsymbol{\beta}_1^o}[||\overline{\mathscr{Y}}_1 - \boldsymbol{\beta}_1^o||^2] + \mathbb{E}_{\boldsymbol{\delta}^o}[||\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}^o||^2] + 2\,\mathbb{E}_{\boldsymbol{\beta}_1^o, \boldsymbol{\delta}^o}[(\overline{\mathscr{Y}}_1 - \boldsymbol{\beta}_1^o)^T(\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}^o)]. \tag{3.15}$$

The third term in (3.15) is the most complex, but Lemma 3.3.1 shows it is strictly negative and can thus be controlled.

Lemma 3.3.1. *Assuming* $\overline{\mathscr{Y}}_1 \sim Normal(\boldsymbol{\beta}_1^o, \sigma_1^2 I_p)$ *independent of* $\overline{\mathscr{Y}}_2 \sim Normal(\boldsymbol{\beta}_2^o, \sigma_2^2 I_p)$ *with* $\overline{\boldsymbol{\delta}} = \overline{\mathscr{Y}}_2 - \overline{\mathscr{Y}}_1$ *and* $\boldsymbol{\delta}_0 = \boldsymbol{\beta}_2^o - \boldsymbol{\beta}_1^o$, *then*

$$\mathbb{E}_{\boldsymbol{\beta}_1^o, \boldsymbol{\delta}^o}[(\overline{\mathscr{Y}}_1 - \boldsymbol{\beta}_1^o)^T(\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}^o)] < 0.$$

Before discussing the properties of the Bayesian estimator, we summarize the assumptions on the model parameters and asymptotic growth of p and n_1.

(A1) The MLE on the source task is a consistent estimator of $\boldsymbol{\beta}_1^o$ *i.e.*, $p = o(n_1)$.

(A2) The proportion of non-zero elements of $\boldsymbol{\delta}^o$ vanishes asymptotically $q = o(p)$.

(A3) $\tau = (\frac{q}{p})^\alpha$, with $\alpha \geq 1$.

(A4) The non-zero entries in $\boldsymbol{\delta}^0$ are greater than the minimum threshold necessary for signal recovery $\sqrt{2\sigma_n^2 \log(p/q)}$.

Lemma 3.3.2. *Under assumptions (A2)-(A4) and as $p \to \infty$. If $\delta_j^o = 0$:*

$$\mathbb{E}_{\delta_j^o}[(Z_j - \delta_j^o)(w_{\tau,\sigma_n^2}(Z_j)Z_j - \delta_j^o)] \approx \sigma_n^2 \tau \sqrt{\log \frac{1}{\tau}}.$$

If $\delta_j^o \neq 0$:

$$\mathbb{E}_{\delta_j^o}[(Z_j - \delta_j^o)(w_{\tau,\sigma_n^2}(Z_j)Z_j - \delta_j^o)] \approx \sigma_n^2.$$

Theorem 3.3.3. *Under the assumptions of Lemma 3.3.2,*

$$\mathbb{E}_{\boldsymbol{\beta}_2^o}[||\hat{\boldsymbol{\beta}}_2 - \boldsymbol{\beta}_2^o||^2] < \sigma^2 \frac{p}{n_1} + \sigma_n^2 q \log(\frac{p}{q})(1 + o(1)) - 2\frac{\sigma^2}{n_1}\left(q + (p-q)\tau\sqrt{\log \frac{1}{\tau}}\right). \tag{3.16}$$

Furthermore, if assumption (A1) is satisfied, then

$$\mathbb{E}_{\boldsymbol{\beta}_2^o}[||\hat{\boldsymbol{\beta}}_2 - \boldsymbol{\beta}_2^o||^2] < \sigma_n^2 q \log(\frac{p}{q})(1 + o(1)). \tag{3.17}$$

We can see from (3.16) that the overall risk of estimating $\boldsymbol{\beta}_2$ is less than the sum of the risks from the two stages. The first term on the right is the risk of the MLE of $\boldsymbol{\beta}_1$. If $n_1 = O(p)$ this term is asymptotically irrelevant and the risk is dominated by the Minimax risk for the estimation of the sparse vector of differences $\boldsymbol{\delta}$. We can compare the risk properties of our proposed estimator to the best possible estimator using only the target task. According to Pinsker (1980), the *minimax* risk for $\boldsymbol{\beta}_2$ using only the target data is

$$r^* = p\frac{\sigma^2 M^2}{\sigma^2 + M^2}, \text{ with } M \text{ s.t.} \|\boldsymbol{\beta}_2^o\| \leq M.$$

From (3.16), we can see that even in the case where $n_1 = o(p)$

$$\sup_{\boldsymbol{\beta}_2^o} \mathbb{E}[||\boldsymbol{\beta}_2^o - \hat{\boldsymbol{\beta}}_2||^2] = o(r^*). \tag{3.18}$$

So integrating source task information and using an appropriate shrinkage prior leads to an estimator with negligible risk relative to estimators using the target data only.

If the MLE is consistent *i.e.* $p = o(n_1)$ then (3.17) shows that our estimator for $\boldsymbol{\beta}_2$ asymptotically attains the minimax risk for a sparse normal means vector.

3.3.2 Risk properties in the bounded ℓ_2 norm case

In the non-sparse case the assumptions are modified as

(B1) $\boldsymbol{\delta} \in \Delta_p = \left\{ \boldsymbol{\delta} \in \mathbb{R}^p : \|\boldsymbol{\delta}\| = \sum_1^p \delta_i^2 \leq M \right\}$ bounded in norm by some constant M for all n_2, p.

(B2) $p = o(n_1)$.

Recall that according to Pinsker's theorem (Pinsker 1980) under assumptions (B1)-(B2) the *minimax risk* for $\boldsymbol{\beta}_1^0$ is $p\sigma^2/n_1$ and $\overline{Y_1}$ is minimax. Furthermore the *minimax risk* for $\boldsymbol{\delta}$ is

$$R_p = \liminf_{p\to\infty} \inf_{\hat{\boldsymbol{\delta}}} \sup_{\boldsymbol{\delta}\in\Delta_p} \mathbb{E}[\|\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}\|^2] = \frac{\sigma^2 M^2}{\sigma^2 + M^2}. \tag{3.19}$$

The following theorem from Pinsker (1980) states that linear minimax estimators are asymptotically minimax for $\boldsymbol{\delta}$.

Theorem 3.3.4. *(Pinsker 1980) Define the minimax linear risk as*

$$R_p^L = \inf_{\hat{\boldsymbol{\delta}}\in\mathscr{L}} \sup_{\boldsymbol{\delta}\in\Delta_p} \mathbb{E}[\|\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}\|^2]$$

where $\mathscr{L}$ denotes the set of all linear estimators of the form

$$\hat{\hat{\boldsymbol{\delta}}} = (w_1 Z_1, \ldots, w_p Z_p).$$

Then, minimax linear estimators are asymptotically minimax i.e., $R_p^L \sim R_p$ as $p \to \infty$.

To derive the form of the minimax linear estimator of $\boldsymbol{\beta}_2$, we restrict the parameter space of $\boldsymbol{\beta}_2$ to also be bounded in norm. Let γ such that $\|\boldsymbol{\beta}_2\| \leq \gamma$. Then the following result gives the form of a linear estimator that achieves the minimax rate up to a multiplicative constant.

Theorem 3.3.5. *(Lei et al. 2022)The best linear estimator for $\boldsymbol{\beta}_2$ based on the source and target data has the form:*

$$\boldsymbol{\beta}_2^{op} = \mathbf{A}\overline{Z} + \mathbf{B}\overline{Y}_1 \tag{3.20}$$

where $\mathbf{A}$ *and* $\mathbf{B}$ *are diagonal matrices and achieves the minimax rate up to a multiplicative constant.*

Theorem 3.3.5 shows that a minimax linear estimator takes the form of a linear combination of the sufficient statistics $\overline{Z}$ and $\overline{Y}_1$. As shown above, the proposed two-step estimator has this form, although we cannot verify that the estimator achieves the minimax rate.

3.4 Simulation study

Here we apply the methods described in Section 3.2 to simulated data to shed light on the performance of the proposed two-stage transfer learning method. We illustrate the performance of the methods in the normal means case for both sparse and bounded settings.

3.4.1 Sparse $\boldsymbol{\delta}$

Data generation

The dimension of the mean vectors will take values $p \in \{100, 500, 10^3, 5 \cdot 10^3, 10^4\}$ and the number of observations in the target task will take values $n_2 \in \{1, 5, 30\}$. The noise level is $\sigma = 1$ and for the source-task true means $\boldsymbol{\beta}^0$, the entries will be independently simulated from a uniform distribution on the interval $(-3, 3)$. Let q the number of non-zero entries in $\boldsymbol{\delta}^0$. We set $q = \lceil p^{1-\alpha} \rceil$ for some $\alpha \in (0,1)$ so $q = o(p)$. The entries of $\boldsymbol{\delta}^0$ follow:

$$\begin{cases} \delta_j^0 \sim \text{Normal}(5,1) & 1 \leq j \leq q \\ \delta_j^0 = 0 & q < j \leq p \end{cases}$$

We then set the target means to $\boldsymbol{\beta}_2^0 = \boldsymbol{\beta}_1^0 + \boldsymbol{\delta}^0$. In other words, the source and target model means are the same except for the first q entries. The source task sample size will also vary according to the parameter dimension. We set $n_1 = \lceil p^{1+\gamma} \rceil$, where $\gamma \in (-1, 1)$ so $p = o(n_1)$ when γ is positive , $n_1 = o(p)$ when γ is negative and $p = O(n_1)$ when $\gamma = 0$. Finally, given $\left(\boldsymbol{\beta}_1^0, \boldsymbol{\delta}, n_1, n_2, \sigma\right)$ we generate the source and target sample means using the following distribution:

$$\overline{Y}_{1j} \sim \text{Normal}(\beta_{1j}^0, \frac{\sigma^2}{n_1})$$
$$\overline{Y}_{2j} \sim \text{Normal}(\beta_{1j}^0 + \delta_j^0, \frac{\sigma^2}{n_2}).$$

For each combination of p, n_2 and q we simulation 1000 datasets.

Competing methods and metrics

To compare the performance of our proposed estimator, we consider three different estimation methods, two of which fall within the two stage framework. In the first stage, we estimate the source data means vector $\hat{\boldsymbol{\beta}}_1$, and in the second stage an estimate of the means difference $\hat{\boldsymbol{\delta}}$ is

computed using $\hat{\boldsymbol{\beta}}_1$. For the two-stage methods, we will further examine the effect of the choice of the first-stage estimate. Depending on the value of γ, we will consider two different choices. If $\gamma > 0$, we will only use the MLE $\hat{\boldsymbol{\beta}}_1^{ML} = \overline{Y}_1$ since it is consistent and unbiased. When $\gamma \leq 0$, $\overline{Y}_1$ will no longer be consistent, in this case we consider both $\hat{\boldsymbol{\beta}}_1^{ML}$ and $\hat{\boldsymbol{\beta}}_1^{JS}$ the James-Stein estimator defined in (3.21). The James-Stein estimator, although biased, uniformly dominates the MLE and is asymptotically minimax under the assumptions of the Pinsker theorem.

$$\hat{\boldsymbol{\beta}}_1^{JS} = \left(1 - \frac{(p-2)\frac{\sigma^2}{n_1}}{\|\hat{\boldsymbol{\beta}}_1^{ML}\|^2}\right)\hat{\boldsymbol{\beta}}_1^{ML} \tag{3.21}$$

For the second stage *i.e.*, the target data, the three different methods, including our proposed estimator for $\boldsymbol{\delta}$ are

$$\begin{aligned}
\hat{\hat{\boldsymbol{\delta}}}^{ML} &= Z \\
\hat{\hat{\boldsymbol{\delta}}}^{JS} &= \left(1 - \frac{(p-2)\sigma_n^2}{\|Z\|^2}\right)\overline{Z} \\
\hat{\hat{\boldsymbol{\delta}}}^{HS} &= \left((1 - \mathbb{E}[\kappa_j | Z_j])Z_j\right)_{1 \leq j \leq p}.
\end{aligned}$$

Where $Z_j = \overline{Y}_{2j} - \hat{\boldsymbol{\beta}}_{1j}$, in other words $\hat{\hat{\boldsymbol{\delta}}}$ depends on the first stage estimation method.

The main competing method is the Trans-Lasso algorithm (Li et al. 2022). The Trans-Lasso algorithm proceeds by first combining the two tasks and estimating a common means vector using the Lasso (Tibshirani 1996) operator. Then uses the Lasso again to estimate the means of the residuals in the target task only. The final Trans-Lasso estimate is the sum of the two-stage estimates. We compare all the different methods using mean squared error (MSE) for $\boldsymbol{\beta}_2$.

3.4.2 Bounded $\|\boldsymbol{\delta}\|$

Data generation

For the non-sparse case, the data generation is the same as in Section 3.4.1 except for $\boldsymbol{\delta}^o$. The true vales of $\boldsymbol{\delta}^o$ are simulated to satisfy $\|\boldsymbol{\delta}^0\| \leq C = 3\sqrt{p}$. To generate the true values that fall within this p-dimensional ball, we first generate $W = (w_i)_{1 \leq i \leq p}$ with $w_i \overset{iid}{\sim}$ Normal(0,1) then set $\boldsymbol{\delta}^o = C U^{1/p} \frac{W}{\|W\|}$, where $U \sim$ Uniform(0,1). We then set the target means to $\boldsymbol{\beta}_2^0 = \boldsymbol{\beta}_1^0 + \boldsymbol{\delta}^0$.

Competing methods

In the bounded case, we will consider two competing transfer learning methods along with a target data only estimator using the James-Stein method. The main competing transfer learning method is the Soft Parameter Sharing (SPS) method (Yang et al. 2023). The SPS method applies a ridge penalty to $\boldsymbol{\delta}$ and optimizes the squared error loss for the two tasks jointly. Yang et al. (2023) studied the properties of SPS under the bounded difference case and showed that when $\|\boldsymbol{\delta}\| = O(\sigma^2 p/n_2)$ then the SPS risk is negligible compared to minimax risk using the target only data. The second method we consider uses the JS estimator in the second stage.

3.4.3 Results

Sparse $\boldsymbol{\delta}$

Table 3.1 summarizes the results for all methods when the source data sample size n_1 is larger that the dimension p (*i.e.*, $\gamma > 0$). In this case the first-stage estimate is exactly the MLE $\overline{Y}_1$. We see that as p increases the two-stage Horseshoe method outperforms all the other candidates including the Trans-Lasso method.

In Table 3.2 we consider the case where $n_1 = p$. In this scenario, the MLE is not a consistent estimator of $\boldsymbol{\beta}_1$, hence we also consider the James-Stein estimator for $\boldsymbol{\beta}_1$ for the source task. From the MSE results, it is clear that using an unbiased estimator in the first stage results in a lower MSE for the target data means vector for all the methods. The Horseshoe estimator has a lower MSE compared to the other methods, with comparable performance to Trans-Lasso in the very large p cases.

Finally, Table 3.3 shows the same results when $n_1 = o(p)$. Again, we see that the Horseshoe estimator based on the MLE in the first stage achieves the best MSE, despite the latter being inconsistent for $\boldsymbol{\beta}_1$ and having a expected risk linear in p. The results from the different scenarios indicate that in terms of MSE the two-stage approach primarily requires an unbiased estimate of the source data means vector, the risk in the first stage estimate tends to have a mitigated effect on the target data risk due to the negative correlation between $\hat{\boldsymbol{\delta}}$ and $\hat{\boldsymbol{\beta}}_1$ (Lemma 1).

Bounded $\|\boldsymbol{\delta}\|$

Table 3.4 summarizes the results for all methods when the source data sample size n_1 is larger that the dimension p (*i.e.*, $\gamma > 0$). In this case the first stage estimate $\overline{Y}_1$ is consistent for $\boldsymbol{\beta}_1$. We

Table 3.1: Mean squared error for $\boldsymbol{\beta}_2$ for the sparse case with $p = o(n_1)$, $\alpha = 0.2$ and $\gamma = 0.2$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods.

			Methods				
p	q	n_2	MLE	James-Stein	Horseshoe	Trans-Lasso	Max SE
100	40	1	0.99	0.91	0.76	0.82	$1.6 \cdot 10^{-3}$
500	145	1	1.01	0.88	0.59	0.74	$1.2 \cdot 10^{-3}$
10^3	252	1	1.00	0.86	0.53	0.68	$1.0 \cdot 10^{-3}$
$5 \cdot 10^3$	911	1	1.00	0.82	0.41	0.47	$0.9 \cdot 10^{-3}$
10^4	1585	1	1.00	0.80	0.36	0.40	$0.9 \cdot 10^{-3}$
100	40	5	0.22	0.19	0.11	0.16	$1.1 \cdot 10^{-3}$
500	145	5	0.20	0.19	0.09	0.13	$1.0 \cdot 10^{-3}$
10^3	252	5	0.20	0.18	0.08	0.12	$0.8 \cdot 10^{-3}$
$5 \cdot 10^3$	911	5	0.19	0.18	0.07	0.10	$0.8 \cdot 10^{-3}$
10^4	1585	5	0.20	0.17	0.06	0.10	$0.7 \cdot 10^{-3}$
100	40	30	0.03	0.03	0.02	0.03	$5.2 \cdot 10^{-4}$
500	145	30	0.03	0.03	0.02	0.02	$4.7 \cdot 10^{-4}$
10^3	252	30	0.03	0.03	0.02	0.02	$4.8 \cdot 10^{-4}$
$5 \cdot 10^3$	911	30	0.03	0.03	0.01	0.02	$4.1 \cdot 10^{-4}$
10^4	1585	30	0.03	0.03	0.01	0.02	$2.6 \cdot 10^{-4}$

notice that the two-stage PCP method outperforms all the other candidates, with comparable performance to SPS when p is large. In Table 3.5 we consider the case where $n_1 = p$. In this scenario, we also notice that the PCP estimator outperforms the SPS, and is comparable to the JS estimator based on target only data. From the MSE results, we notice that the PCP estimator outperforms the other transfer learning methods, but is substantially worse than the JS estimator based on the target data only especially for the large p cases. So, unlike the sparse case setting, abundant source task data is a requirement for all transfer learning methods in the bounded case.

Table 3.2: Mean squared error for $\boldsymbol{\beta}_2$ for the sparse case with $p = n_1$ and $\alpha = 0.2$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods.

			First stage MLE			First stage JS				
p	q	n_2	MLE	JS	HS	MLE	JS	HS	Trans-Lasso	Max SE
100	40	1	0.99	0.95	0.62	0.99	7.67	13.03	0.85	0.77
500	145	1	1.00	0.93	0.52	1.00	7.82	10.25	0.71	0.82
10^3	252	1	1.00	0.92	0.48	1.00	7.51	8.70	0.61	0.69
$5 \cdot 10^3$	911	1	1.00	0.90	0.39	1.00	6.44	5.41	0.58	0.73
10^4	1585	1	1.00	0.88	0.34	1.00	5.90	4.27	0.44	0.64
100	40	5	0.22	0.20	0.11	0.20	7.24	12.44	0.18	0.98
500	145	5	0.20	0.20	0.09	0.20	7.46	9.68	0.15	0.94
10^3	252	5	0.20	0.20	0.09	0.20	7.21	8.12	0.13	0.86
$5 \cdot 10^3$	911	5	0.19	0.20	0.07	0.20	6.21	4.71	0.12	0.57
10^4	1585	5	0.20	0.20	0.06	0.20	5.71	3.55	0.09	0.41
100	40	30	0.03	0.03	0.02	0.03	7.12	12.26	0.03	1.02
500	145	30	0.03	0.03	0.02	0.03	7.38	9.55	0.02	0.73
10^3	252	30	0.03	0.03	0.01	0.03	7.13	7.97	0.02	0.59
$5 \cdot 10^3$	911	30	0.03	0.03	0.01	0.03	6.15	4.55	0.02	0.53
10^4	1585	30	0.03	0.03	0.01	0.03	4.53	1.50	0.01	0.58

3.5 Materials informatics example

We apply the proposed transfer learning methods for fine-tuning the last layer of neural network used for band gap prediction of a molecular crystal target data set. The data are described in Abba et al. (2023). Band gap, defined as the energy differential between the lowest unoccupied and highest occupied electronic states (Kittel and McEuen 2019), determines many aspects of interest in industrial sectors such as electric and photovoltaic conductivity. We use transfer learning methods to borrow information from molecules (source) to molecular crystals (target)

Table 3.3: Mean squared error for $\boldsymbol{\beta}_2$ for the sparse case with $n_1 = o(p)$, $\alpha = 0.2$ and $\gamma = -0.2$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods.

			First stage MLE			First stage JS				
p	q	n_2	MLE	JS	HS	MLE	JS	HS	Trans-Lasso	Max SE
100	40	1	0.99	0.95	0.63	1.00	7.74	13.16	0.91	0.77
500	145	1	1.00	0.94	0.52	1.00	7.81	10.23	0.82	0.87
1000	252	1	1.00	0.93	0.49	1.00	7.51	8.69	0.73	0.71
5000	911	1	1.00	0.90	0.39	1.00	6.43	5.40	0.56	0.69
10000	1585	1	1.00	0.89	0.35	1.00	5.90	4.27	0.43	0.56
100	40	5	0.20	0.20	0.12	0.20	7.24	12.44	0.17	0.96
500	145	5	0.20	0.20	0.10	0.20	7.47	9.70	0.15	0.85
1000	252	5	0.20	0.20	0.09	0.20	7.20	8.11	0.13	0.71
5000	911	5	0.20	0.20	0.07	0.20	6.21	4.72	0.10	0.68
10000	1585	5	0.20	0.19	0.06	0.20	5.71	3.55	0.10	0.61
100	40	30	0.03	0.03	0.02	0.03	7.12	12.26	0.03	1.01
500	145	30	0.03	0.03	0.02	0.03	7.38	9.55	0.03	0.63
1000	252	30	0.03	0.03	0.01	0.03	7.13	7.97	0.02	0.62
5000	911	30	0.03	0.03	0.01	0.03	6.15	4.55	0.02	0.56
10000	1585	30	0.03	0.03	0.01	0.03	4.53	1.50	0.01	0.47

to improve prediction. The full target data consist of 50,000 molecular crystals and their band gap values computed using Density Functional Theory (DFT; Kohn and Sham 1965). This data set was provided by the Material Science Research Group at Carnegie Mellon University and required several years to collect. The source data will be the OE62 data set (Stuke et al. 2020) which consists of $n_1 = 62,000$ molecules and their band gap values. The molecules in the source data were all extracted from organic crystals, and the band gap values were also computed using DFT. For training, we consider only subsets of the target data with cardinality $n_2 \in \{500, 1000, 2000\}$ to represent a typical matrials data set sizes, a validation set of 10,000

Table 3.4: Mean squared error for the means $\boldsymbol{\beta}_2$ in the bounded case with $p = o(n_1)$ and $\gamma = 0.2$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods.

		TL Methods			Target only	
p	n_2	SPS	JS	PCP	James-Stein	Max SE
100	1	1.12	1.95	0.61	0.92	0.11
500	1	1.07	1.94	0.58	0.88	0.09
10^3	1	0.96	1.93	0.52	0.89	0.13
$5 \cdot 10^3$	1	1.02	1.91	0.42	0.91	0.10
10^4	1	0.99	1.82	0.29	0.84	0.09
100	5	0.42	1.11	0.19	0.19	0.04
500	5	0.45	1.09	0.19	0.21	0.06
10^3	5	0.39	1.08	0.18	0.20	0.08
$5 \cdot 10^3$	5	0.31	0.97	0.18	0.21	0.08
10^4	5	0.25	0.96	0.17	0.19	0.06
100	30	0.08	1.02	0.03	0.18	0.01
500	30	0.09	0.82	0.03	0.03	0.01
10^3	30	0.04	0.91	0.02	0.03	$2.4 \cdot 10^{-3}$
$5 \cdot 10^3$	30	0.03	0.81	0.02	0.03	$1.8 \cdot 10^{-3}$
10^4	30	0.03	0.74	0.02	0.03	$1.6 \cdot 10^{-3}$

Table 3.5: Mean squared error for the means $\boldsymbol{\beta}_2$ in the bounded case with $p = n_1$ and $\gamma = 0$. The scenarios vary by the size of the vector of means (p), the number of non-zero means (q) and the number of target data samples (n_2). The final column gives the largest Monte Carlo standard error in each row to gauge statistical significance between methods.

		TL Methods			Target only	
p	n_2	SHS	JS	PCP	James-Stein	Max SE
100	1	1.64	1.99	1.08	0.92	0.14
500	1	1.71	2.01	1.16	0.88	0.13
10^3	1	1.86	2.13	1.24	0.89	0.15
$5 \cdot 10^3$	1	1.81	1.98	1.12	0.91	0.11
10^4	1	1.79	2.02	1.09	0.84	0.16
100	5	1.09	1.21	0.99	0.19	0.09
500	5	1.15	1.14	1.01	0.21	0.09
10^3	5	1.13	1.18	0.98	0.20	0.07
$5 \cdot 10^3$	5	1.19	1.27	0.94	0.21	0.09
10^4	5	1.25	1.16	0.97	0.19	0.10
100	30	0.88	0.97	0.66	0.18	0.07
500	30	0.79	0.92	0.48	0.03	0.07
10^3	30	0.82	0.84	0.52	0.03	0.07
$5 \cdot 10^3$	30	0.93	0.72	0.51	0.03	0.07
10^4	30	0.74	0.76	0.38	0.03	0.06

samples and finally a testing set of 20,000 samples.

Since we are dealing with molecules, we first need to preprocess the data and compute descriptors. We use the MBTR descriptor method (Huo and Rupp 2018) to compute $p = 1,260$ descriptors for every data point in both the source and target data sets. These descriptors are functions of the structure of the material, e.g., the types and configuration of its atoms. A possible approach is to use these features as covariates in a linear regression. However, the relationship between these features and band gap is likely not additive or linear. To capture these complexities, we follow Abba et al. (2023) and use a neural network model. The architecture of the neural network model has two blocks of the MEGNET (Chen et al. 2019) followed by two hidden layers with 1024 and 512 neurons, respectively. MEGNET is a feature-extraction layer designed specifically for materials problems that exploits physical laws to derive meaningful features from the molecule's structure.

We apply the full deep learning model to the source data, and refit only the output layer parameters using target data. First the model is trained on the source data with 40,000 samples for training set and 10,000 for the validation. The parameters of the best performing model in terms of the MSE on the validation set of the source task are saved and transferred to the target task. All the layers except the linear output layer will be fixed at the source task estimates. In other words, the neural network up to the last hidden layer is used to extract a feature set of size $p = 512$ for the target task.

Let $\hat{\boldsymbol{\beta}}_1$ and $\hat{\boldsymbol{\Sigma}}_1$ be the estimated output layer weights and their estimated variance from the source task, and let $\boldsymbol{\beta}_2$ the weights of the output layer in the target task. We test both the horseshoe and the PCP prior on $\boldsymbol{\delta} = \boldsymbol{\beta}_2 - \boldsymbol{\beta}_1$ and compare their performance to classical ordinary least square estimate using prediction MSE and coverage on the test set. Let $\mathbf{Y}$, the true gap values for the target data. Let $\mathbf{X}$ the extracted features from the last hidden layer of the neural net. For the target data, we fit the following model

$$\mathbf{Y} \sim \text{Normal}(\mathbf{X}\boldsymbol{\beta}_2, \sigma^2 \mathbf{I}_{n_2}). \tag{3.22}$$

The OLS estimator fits the model in (3.22). For the PCP and horseshoe methods, we first compute the predicted gap on the target data given $\hat{\boldsymbol{\beta}}_1$, denote $\tilde{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}}_1$. Given the first-stage estimates and their uncertainty, we have $\tilde{\mathbf{Y}} \sim \text{Normal}(\mathbf{X}\hat{\boldsymbol{\beta}}_1, \sigma^2 \mathbf{X}\hat{\boldsymbol{\Sigma}}\mathbf{X}^{\mathrm{T}})$. Define $\mathbf{Z} = \mathbf{Y} - \tilde{\mathbf{Y}}$ the difference between the true target data values and the predicted values given the first stage estimates. We reparameterize (3.22) in terms $\boldsymbol{\delta}$ and fit the following second-stage regression model

$$\mathbf{Z} \sim \text{Normal}(\mathbf{X}\boldsymbol{\delta}, \sigma^2(\mathbf{I}_{n_2} + \mathbf{X}\hat{\boldsymbol{\Sigma}}\mathbf{X}^{\mathrm{T}})). \tag{3.23}$$

We compare the Bayesian transfer learning methods with simply fitting $\boldsymbol{\beta}_2$ on the training data using ordinary least squares (OLS). Furthermore, a baseline model for comparison that fits the full neural network model using the target data only is also included. Due to the difficulty of training an end-to-end neural network model on small amounts of data, when $n_2 = 500$ we first use 5000 samples for 5 epochs to get reasonable starting values, then continue training the baseline model on the 500 target data points.

For the horseshoe and PCP, we run $5 \cdot 10^5$ iterations of MCMC sampling algorithms described in Appendix B.1 and compute the prediction intervals. For the OLS method we use the predicted confidence intervals. For the baseline model, the predicted confidence intervals were obtained using a drop-out layer (Srivastava et al. 2014) before the last hidden layer of the model.

Table 3.6: MSE and coverage of 95% for the band gap prediction application with different target data size, n_2. The methods are Bayesian transfer learning using PCP and Horseshoe priors, transfer learning using ordinal least squares (OLS), and a baseline model that uses only target data.

	Method			
n_2	PCP	Horseshoe	OLS	Baseline
500	0.91(83%)	0.94(87%)	1.08(89%)	1.02(71%)
1000	0.83(92%)	0.90(88%)	0.94(88%)	0.94(61%)
2000	0.81(91%)	0.88(90%)	0.87(91%)	0.82(63%)

For small target datasets with $n_2 = 500$, the Bayesian transfer learning methods provide substantial improvements over OLS and the baseline method. As expected, as n_2 increases, the advantage of transfer learning dissipates and the prediction accuracy improves for all methods. For this application, the PCP method generally outperforms the horseshoe method. This could be because all the covariates used the second stage are actually functions of the same original molecular features, and so all of their effects vary with similar magnitudes between the target and source datasets.

3.6 Discussion

In this paper, we developed a Bayesian estimator for transfer learning in high dimensional settings. We propose a two-stage procedure based on estimates from the source task, our proposed methodology obviates the need for source task data and only requires the estimates and their uncertainty. This makes our method attractive for cases where the source data is abundant but not available due to privacy rules. In this case, our proposed methodology can leverage prior beliefs on the relationships between similar tasks.

In the Bayesian framework, assumption on model parameters can naturally be translated into a prior distribution. Hence, depending on the assumed similarity between source and target tasks, we propose two different estimators based on two different prior distributions. Although we focused on the normal means model, this work can be step towards more theoretical examination of Bayesian transfer learning methods in linear regression and more complex models.

When the difference of the means vectors between the two tasks is sparse, a global local type prior is proposed and the theoretical guarantees for the normal means model are provided. We observe through theoretical analysis of the risk that the first and second stage of the estimation procedure enjoy a synergy property. Meaning that the total risk of the estimator is strictly less than the sum of the risks from the first and second stage estimators. Further, using minimax type results, we showed that our estimator results in negligible risk compared to the best estimator based on the target data only. From a transfer learning point of view, this not only means that we run no risk of negative transfer, but also, the proposed estimator vastly outperforms any estimator based solely on the target data. We choose the horseshoe prior for the sparse regression case because it is computationally convenient and has theoretical support. However, other choices of global local shrinkage priors (Bhattacharya et al. 2015; Bhadra et al. 2017; Zhang et al. 2022) could result in comparable performance.

In the second case, we only assume that the difference between the mean vectors is bounded. Here, we use the principled prior construction approach of Simpson et al. (2017b) to penalize the deviation of the target estimates from the source estimates. The PCP framework helps to ensure that the prior concentrates around the source estimates and is allowed to deviate if appropriate. Although, we do not provide theoretical guarantees for the proposed estimator, we derive the form of the linear minimax estimator for the transfer learning task. Our proposed estimator takes the same form.

Using simulation studies we showed that our proposed method outperforms state-of-the-art transfer learning methods in both scenarios. An interesting, empirical finding from the

simulation study, is that an unbiased first stage estimate is crucial even when the number of parameters is larger than the size of the source task data. In fact, using a biased source task estimate (with a lower risk) results in considerably larger MSE for the target task parameters particularly in the sparse case. This can be explained by noticing that the original formulation of the priors involves the difference between the true parameter vectors. Hence, when an efficient but biased estimate is used in the first stage, most of the prior mass ends in the wrong region of the parameter space.

There are many areas of potential future work. A in depth theoretical investigation of the regression case is naturally the first extension. In addition, the proposed can be extended to the non Gaussian model and possibly non-continuous response case.

CHAPTER

4

STOCHASTIC GRADIENT MCMC FOR MASSIVE GEOSTATISTICAL DATA

4.1 Introduction

Gaussian process (GP) modeling is a powerful statistical and machine learning tool used to tackle a variety of tasks including regression, classification and optimization. Particularly in spatial statistics, GPs have become the primary tool for inference (Gelfand and Schliep 2016). In spatial regression and classification problems, the response variable is assumed to have a spatially-correlated structure. GPs model this spatial dependence by specifying a form of the correlation between any two points in the spatial domain. In this paper we focus on the regression setting under the Matérn correlation with large amounts of data. Formally, GPs place a prior on the spatial process using a parameterized correlation function, this allows to estimate a posteriori the parameters given the observed data.

One of the main advantages of GPs is their ability to provide predictions at unobserved locations along with uncertainty quantification. Spatial interpolation, commonly known as kriging (Woodard 2000), provides optimal predictions at unobserved sites based on the correla-

tion between a given location and its observed neighbours (Cressie 1988). However, handling large datasets with GPs poses computational challenges due to the cubic time complexity and quadratic memory requirements in the evaluation of the joint likelihood. This prohibitive computational cost mainly results from the evaluation of the covariance matrix and computing its inverse. Several methods have been proposed to address this issue and make GPs more scalable for large datasets. In this work, we combine stochastic gradient (SG) methods along with the Vecchia (Vecchia 1988) approximation to develop an efficient algorithm for scalable Bayesian inference in massive spatial data settings. In the following section we review some of the main methods used to scale GPs (see Heaton et al. (2019) for a full survey), and briefly discuss applications of SG methods in correlated and dependent data settings.

4.1.1 Methods to handle large spatial datasets

The main computational bottleneck in GP regression is the inverse of the covariance matrix. To overcome this problem, a large body of literature has been proposed over the last decades, including but not limited to low rank approximation, covariance tapering, divide-and-conquer strategies and Vecchia-type methods. Although these approaches differ significantly, they all tend to result in an amenable structure on the covariance or its inverse.

A low-rank approximation of the GP can be used to overcome the covariance inverse cost, (*e.g.*, Cressie and Johannesson (2008); Katzfuss and Cressie (2011); Kang et al. (2009)). Low-rank approximations project the spatial process on a low-dimensional space and use the low-rank representation as a surrogate to approximate the original process. Banerjee et al. (2008) used predictive process methods, where first a certain number of knots are placed in the spatial domain, then used as a conditioning set for the expectation of the original process. Fixed rank kriging (Cressie and Johannesson 2008) approximates the original process using a small number of basis functions, which results in a precision matrix that can be obtained by inversion of a matrix with a much smaller dimension.

Instead of approximating the original process, one can impose fixed structures on the covariance or precision matrices directly. This method, also known as covariance tapering (Furrer et al. 2006; Kaufman et al. 2008), imposes a compact support on the correlation function, hence correlation between a site and distant neighbours is shrunk to zero. This induces a sparse structure on the covariance that is leveraged to speed up the computation. Instead of imposing a structure on the covariance, Rue et al. (2009) directly impose a sparse structure on the precision matrix using a Gaussian Markov random field approximation to the true process.

Divide-and-conquer approaches have also been proposed to scale GPs inference. Barbian

and Assunção (2017) and Guhaniyogi and Banerjee (2018) propose splitting the spatial domain in subsets, perform the analysis in parallel on each subset then combine the results. This strategy distributes the workload into smaller parts. Another option, is to divide the spatial region into independent sub-regions and perform the analysis on the whole dataset under this assumption (Sang et al. 2011). Unlike the former, the later uses the whole dataset but reduces the computational cost using independence between subregions.

One of the earliest and most influential methods for scalable GPs is the Vecchia approximation (Vecchia 1988; Stein et al. 2004). In the Vecchia framework, the full likelihood is factorized into a series of conditional distributions. This factorization is then simplified by reducing the conditioning sets to include a small number of neighbours, which in turn results in a sparse precision matrix. Guinness (2018) showed that Vecchia's method is an accurate approximation to the true Gaussian process model in terms of the Kullback-Leibler divergence. This approach is also well suited for parallel computing due to the factorization of the likelihood. Recent works have built upon and extended the Vecchia approximation. Katzfuss and Guinness (2021) propose a generalization of the Vecchia's framework and show that many existing approaches to Gaussian process approximation can be viewed as a special case of the extended method. Datta et al. (2016) proposed the nearest-neighbor Gaussian process as an extension of the Vecchia approximation, later, Finley et al. (2019) outline an efficient Markov Chain Monte Carlo (MCMC) algorithm for scalable full Bayesian inference using this method.

In general, all the aforementioned methods reduce the computational cost from cubic to linear in the number of observations. However, in the Bayesian framework we are mostly interested in posterior sampling through MCMC methods in order to get uncertainty estimates of the model parameters as well predictive credible intervals for certain locations. Typically, MCMC methods require thousands of iterations to accurately approximate the posterior distribution. Hence, even when the cost per iteration is linear, the total time can still be prohibitive. Recent work has therefore also focused on subsampling approaches for spatial data to reduce the computational cost associated with posterior sampling. Saha and Bradley (2023) have developed an efficient composite sampling scheme for posterior inference. Similarly, Heaton and Johnson (2023) use minibatches to approximate the complete conditional distribution of conjugate parameters, and provide an approximate Metropolis-Hastings (MH) acceptance probabilities for non-conjugate parameters. While Heaton and Johnson (2023) use a Vecchia approximation to define the minibatches, neither of these two works use any gradient information when drawing samples from the posterior, and are therefore fundamentally different from the gradient-based approach we will employ in our study.

4.1.2 Review of stochastic gradient methods

When dealing with large datasets, stochastic gradient (SG) methods (Robbins and Monro 1951) have become the default choice in machine learning (Hardt et al. 2016). To avoid a costly gradient based on the full dataset, SG methods only require an unbiased and possibly noisy estimate using a subsample of the data. When the data is independent and identically distributed (*iid*) a proper scaling of the gradient based on a given subsample of the data yields an unbiased gradient estimate. The popularity and success of SG methods in optimization eventually lead to their adoption for scalable Bayesian inference (Nemeth and Fearnhead 2021). Scalable SG Markov Chain Monte Carlo (SGMCMC) methods for posterior sampling in the *iid* setting have been proposed (Welling and Teh 2011; Chen et al. 2015; Ma et al. 2015; Dubey et al. 2016; Baker et al. 2019). Convergence of SGMCMC methods has also received considerable attention. Under mild conditions, SGMCMC methods produce approximate samples from the posterior (Teh et al. 2016; Durmus and Moulines 2017; Dalalyan and Karagulyan 2019).

Although SG methods are widely used in the *iid* setting, their possible use in the correlated setting is still new. A naive application of SGMCMC methods in the correlated setting would overlook critical dependencies in the data during subsampling. Moreover, the gradient estimate from the subsamples cannot be guaranteed to be unbiased. To the best of our knowledge, subsampling methods for spatial data that result in unbiased gradient estimates has not been addressed. Chen et al. (2020) studied the performance and theoretical guarantees for SG optimization for GP models. Although, the gradient based on a minibatch of the data leads to biased estimates of the full gradient of the log-likelihood, Chen et al. (2020) established convergence guarantees for recovering recovering noise variance and spatial process variance in the case of the exponential covariance function. In their work, the length scale parameter, which controls the degree of correlation between distinct points is assumed known, and no convergence result is provided. Recent works have considered other types of dependent data. In the case of network data, Li et al. (2016b) developed an SGMCMC algorithm for the mixed-member stochastic block models. Ma et al. (2017) leveraged the short-term dependencies in Hidden Markov models to construct an estimate of the gradient with a controlled bias using non-overlapping subsequences of the data. This approach was extended to linear and non-linear state space models (Aicher et al. 2019, 2021).

SGMCMC methods can be divided in two main groups based on either Hamilton dynamics (Chen et al. 2014) or Langevin dynamics (Welling and Teh 2011). In this work we use the Langevin dynamics (LD) method due to its lower number of hyperparameters, our approach can be extended to the Hamiltonian dynamics with minor modifications. We extend the SGLD

method to the case of non-iid data using the Vecchia approximation and provide a method that takes account of the local curvature to improve convergence.

In the remainder of this paper, Section 4.2 discusses the Matérn Gaussian process model and the Vecchia approximation used to obtain unbiased gradients. Section 4.3 presents the derived SGMCMC algorithm for Gaussian process learning. In Section 4.4 we test our proposed method using a simulation study and Section 4.5 concludes. We also discuss the performance of stochastic optimization algorithm for Maximum likelihood estimation for GP in the supplementary material.

4.2 Matérn Gaussian process model and approximations

Let Y_i for $i \in \{1,...,n\}$ be the observation at spatial location $\mathbf{s}_i = (s_{i1}, s_{i2})$ and $\mathbf{X}_i = (X_{i1},...,X_{ip})$ be a corresponding vector of covariates. The data-generation model for Gaussian process regression in the case of Gaussian data is

$$Y_i = \mathbf{X}_i\boldsymbol{\beta} + Z(\mathbf{s}_i) + \varepsilon_i \tag{4.1}$$

with fixed effects $\boldsymbol{\beta}$, spatial process $Z(\mathbf{s}_i)$ and $\varepsilon_i \overset{iid}{\sim} \text{Normal}(0,\tau^2)$ is measurement error with nugget variance τ^2. The process $Z(\mathbf{s})$ is an isotropic spatial Gaussian process with mean $\text{E}\{Z(\mathbf{s})\} = 0$, spatial variance $\text{Var}\{Z(\mathbf{s})\} = \sigma^2$ and spatial correlation $\text{Cor}\{Z(\mathbf{s}_i), Z(\mathbf{s}_j)\} = \text{K}(d_{ij})$ for distance $d_{ij} = ||\mathbf{s}_i - \mathbf{s}_j||$. Specifically, we assume the correlation function is the Matérn (Stein 1999) correlation function with range ρ and smoothness ν

$$\text{K}(d) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{d}{\rho}\right)^{\nu}\mathscr{K}_{\nu}\left(\frac{d}{\rho}\right), \tag{4.2}$$

where $\mathscr{K}_{\nu}$ is the modified Bessel function of the second kind. Let $\boldsymbol{\theta} = (\sigma^2, \rho, \nu, \tau^2)$ be the collection of covariance parameters.

The marginal distribution (over Z) of $\mathbf{Y} = \{Y(\mathbf{s}_1),\ldots,Y(\mathbf{s}_n)\}$ is multivariate normal with mean $\mathbb{E}[\mathbf{Y}] = \mathbf{X}\boldsymbol{\beta}$, for $\mathbf{X} \in \mathscr{R}^{n\times p}$ covariate matrix with the i$^{\text{th}}$ row $\mathbf{X}_i$, and covariance matrix $\mathbb{E}[(\mathbf{Y}-\mathbf{X}\boldsymbol{\beta})(\mathbf{Y}-\mathbf{X}\boldsymbol{\beta})^{\text{T}} \mid \boldsymbol{\theta}] = \Sigma(\boldsymbol{\theta})$ with

$$\begin{aligned} \Sigma(\boldsymbol{\theta}) &= \sigma^2\mathbf{K} + \tau^2\mathscr{I}_n \\ \mathbf{K}_{i,j} &= \text{K}(d_{ij}). \end{aligned} \tag{4.3}$$

The full log-likelihood then becomes

$$\ell_{\text{full}}(\boldsymbol{\beta}, \boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma|(\boldsymbol{\theta}) - \frac{1}{2}(\mathbf{Y}-\mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}\Sigma(\boldsymbol{\theta})^{-1}(\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}). \tag{4.4}$$

Evaluating (4.4) involves computing the determinant and inverse of $\Sigma(\boldsymbol{\theta})$ which generally requires $O(n^3)$ operations. This cost becomes prohibitive for large spatial datasets. The remainder of this section discusses the computationally-efficient Vecchia approximation.

4.2.1 Vecchia approximation

For any set of spatial locations, the joint distribution of $\mathbf{Y}$ can be written as a product of univariate conditional distributions, which can then be approximated by a Vecchia approximation (Vecchia 1988; Stein et al. 2004; Datta et al. 2016; Katzfuss and Guinness 2021):

$$f(Y(\mathbf{s}_1), ..., Y(\mathbf{s}_n)) = \prod_{i=1}^{n} f(Y(\mathbf{s}_i)|\, Y(\mathbf{s}_1), ..., Y(\mathbf{s}_{i-1})) \approx \prod_{i=1}^{n} f_i(Y(\mathbf{s}_i)|\, Y(\mathbf{s}_{(i)})) \tag{4.5}$$

for $Y(\mathbf{s}_{(i)}) = \{Y(\mathbf{s}_j);\, j \in \mathcal{N}_i\}$ and conditioning set $\mathcal{N}_i \subseteq \{1, ..., i-1\}$, e.g., the indices of the $m_i \leq m$ locations in $\mathcal{N}_i$ that are closest to $\mathbf{s}_i$ according to some ordering of the data. Here, we use the notation that the collection of variables over the conditioning set of Y_i is denoted $Y_{(i)} = \{Y_j;\, j \in \mathcal{N}_i\}$. Of course, not all locations that are dependent with location i need be included in $\mathcal{N}_i$ because distant observations may be approximately independent after conditioning on more local observations. Conditioning on only $\mathcal{N}_i$ leads to substantial computational savings when m is small, *i.e.*, $m << n$. The Vecchia approximation is attractive for GPs in particular since the conditional densities are Normally distributed. Stein (2002) proved that a screening effect exists in this context which ensures that the Vecchia approximation is a good approximation, and Stein (2011) provided conditions for a variety of situations when the screening effect might hold. Also, while we have motivated the Vecchia likelihood as an approximation, it is in fact a valid joint probability density function (PDF) which permits a standard Bayesian analysis and interpretation.

Let $p(\boldsymbol{\beta}, \boldsymbol{\theta})$ be the prior distribution on the regression and covariance parameters. Using (4.5) we can write the posterior as (ignoring a constant that does not depend on the parameters)

$$\ell(\boldsymbol{\beta}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \log f(Y(\mathbf{s}_i) \mid Y(\mathbf{s}_{(i)}), \boldsymbol{\beta}, \boldsymbol{\theta})$$

$$\log p(\boldsymbol{\beta}, \boldsymbol{\theta} \mid \mathbf{Y}) = \ell(\boldsymbol{\beta}, \boldsymbol{\theta}) + \log p(\boldsymbol{\beta}, \boldsymbol{\theta}). \tag{4.6}$$

Hence the log-likelihood and log-posterior of the parameters $\{\boldsymbol{\beta}, \boldsymbol{\theta}\}$ can be written as a sum of conditional normal log-densities, where the conditioning set is at most of size m. The cost of computing the log-posterior in (4.6) is linear in n and cubic in m.

Although the Vecchia approximation reduces the complexity cost from $O(n^3)$ for the full likelihood to $O(nm^3)$, this can still pose challenges for very large n. We can further reduce the cost of Bayesian inference by using subsampling strategies which have had substantial success in SG methods (Newton et al. 2018). Although we are still in correlated data setting, sampling the summands of (4.6) with equal probability and without replacement leads to an unbiased estimate of the gradient. Let $\mathscr{B} \subset \{1, \ldots, n\}$ be a subsample, *i.e.*, a minibatch index set of size $n_{\mathscr{B}}$, and let

$$\bar{\ell}_{\mathscr{B}}(\boldsymbol{\beta}, \boldsymbol{\theta}) = \frac{n}{n_{\mathscr{B}}} \sum_{i \in \mathscr{B}} \log f(Y(\mathbf{s}_i) \mid Y(\mathbf{s}_{(i)}), \boldsymbol{\beta}, \boldsymbol{\theta}). \tag{4.7}$$

Theorem 4.2.1. *The gradient of $\bar{\ell}_{\mathscr{B}}$ is an unbiased estimator of the gradient of the Vecchia posterior $\ell(\boldsymbol{\beta}, \boldsymbol{\theta})$.*

Proof.

$$\begin{aligned}
\mathbb{E}_{\mathscr{B}}[\nabla \bar{\ell}_{\mathscr{B}}(\boldsymbol{\beta}, \boldsymbol{\theta})] &= \nabla \mathbb{E}_{\mathscr{B}} \left[\frac{n}{n_{\mathscr{B}}} \sum_{i=1}^{n} \log f(Y(\mathbf{s}_i) \mid Y(\mathbf{s}_{(i)}), \boldsymbol{\beta}, \boldsymbol{\theta}) \delta_{i \in \mathscr{B}} \right] \\
&= \nabla \sum_{i=1}^{n} \log f(Y(\mathbf{s}_i) \mid Y(\mathbf{s}_{(i)}), \boldsymbol{\beta}, \boldsymbol{\theta}) \\
&= \nabla \ell(\boldsymbol{\beta}, \boldsymbol{\theta}).
\end{aligned} \tag{4.8}$$

□

Using (4.8), we can construct an unbiased estimate of the gradient of the Vecchia log-posterior based on a minibatch of the data:

$$\bar{g}_{\mathscr{B}}(\boldsymbol{\beta}, \boldsymbol{\theta}) = \nabla \bar{\ell}_{\mathscr{B}}(\boldsymbol{\beta}, \boldsymbol{\theta}) + \nabla \log p(\boldsymbol{\beta}, \boldsymbol{\theta}), \tag{4.9}$$

hence reducing the cost of learning iterations to be linear in $n_{\mathscr{B}}$ instead of n, *i.e.*, $O(m^3 n_{\mathscr{B}})$.

4.3 SG-MCMC algorithm

In this section we first review the general SG Langevin dynamics method and then present the proposed algorithm based on the Vecchia approximation.

4.3.1 SG Langevin Dynamics

SG Markov chain Monte Carlo (Ma et al. 2015) is a popular method for scalable Bayesian inference. SGMCMC proceeds by simulating continuous dynamics of a potential energy, namely the negative log-posterior $-\log p(\boldsymbol{\beta}, \boldsymbol{\theta} \mid \mathbf{Y})$, such that the dynamics generate samples from the posterior distribution. Let $\boldsymbol{\phi} = (\boldsymbol{\beta}^{\mathrm{T}}, \boldsymbol{\theta}^{\mathrm{T}})^{\mathrm{T}}$ be the parameter concatenating the regression and covariance parameters of the Gaussian process regression model. The Langevin diffusion over $\log p(\boldsymbol{\phi} \mid \mathbf{Y})$ is given by the stochastic differential equation (SDE)

$$d(\boldsymbol{\phi}_t) = \nabla \log p(\boldsymbol{\phi}_t \mid \mathbf{Y}) dt + \sqrt{2} dW_t, \tag{4.10}$$

where dW_t is Brownian motion and the index t represents time. The distribution of samples $\boldsymbol{\phi}_t$ converges to the true posterior as $t \to \infty$ (Roberts and Rosenthal 1998).

Since simulating a continuous time process is infeasible, in practice a discretized numerical approximation is used. Here we use the Euler discretization method. Let h_t the step size at time t, and let $\boldsymbol{\phi}_t$ the current value of the parameter. The Euler approximation of the Langevin dynamics is given by

$$\boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}_t + h_t \nabla \log p(\boldsymbol{\phi}_t \mid \mathbf{Y}) + \sqrt{2h_t} e_t \tag{4.11}$$

where e_t is random white noise. This recursive sampling approach is known as the Langevin Monte Carlo algorithm. Often, a Metropolis-Hastings (MH) correction step is added to account for the discretization error.

When the size of the dataset is large, computing the log-posterior gradient represents a computational bottleneck. To overcome this problem, the key idea of SGLD is to replace $\nabla \log p(\boldsymbol{\phi} \mid \mathbf{Y})$ with an unbiased gradient estimate, *i.e.*, $\bar{g}_{\mathscr{B}}(\boldsymbol{\phi})$ in (4.9) that is computationally cheaper to compute, and use a decreasing step size h_t to avoid the costly M-H correction steps.

$$\text{SGLD}: \qquad \boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}_t + h_t \bar{g}_{\mathscr{B}}(\boldsymbol{\phi}_t) + \sqrt{2h_t} e_t. \tag{4.12}$$

In order to assure convergences to the true posterior the step sizes must satisfy

$$0 < h_{t+1} < h_t, \sum_{t=1}^{\infty} h_t = \infty \text{ and } \sum_{t=1}^{\infty} h_t^2 < \infty.$$

The SGLD step in (4.12) updates all the parameters using the same step size. This can cause slow mixing when different parameters have different curvature or scales. SG Riemannian Langevin Dynamics (SGRLD) takes account of the difference in curvature and scale by using an

appropriate Riemannian metric $G(\boldsymbol{\phi})$, and simulates the diffusion by preconditioning the unbiased gradient and noise in (4.12) using $G^{-1}(\boldsymbol{\phi})$. SGRLD achieves better mixing by incorporating geometric information of the posterior. Commonly used metrics for $G(\boldsymbol{\phi})$ include the Fisher information matrix and estimates of the Hessian of the log-posterior. Given a preconditioning matrix $G(\boldsymbol{\phi})$, the SGRLD step is

$$\text{SGRLD} \qquad \boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}_t + h_t\left(G^{-1}(\boldsymbol{\phi}_t)\bar{g}_{\mathscr{B}}(\boldsymbol{\phi}_t) + \Gamma(\boldsymbol{\phi}_t)\right) + \sqrt{2h_t}G^{-1/2}(\boldsymbol{\phi}_t)e_t, \tag{4.13}$$

where the term $\Gamma(\boldsymbol{\phi}_t)$ represents the drift term that describes how the preconditioner $G(\boldsymbol{\phi}_t)$ changes with respect to $\boldsymbol{\phi}_t$. The drift term is given by

$$\Gamma(\boldsymbol{\phi}_t)_i = \sum_j \frac{\partial\, G(\boldsymbol{\phi}_t)^{-1}_{ij}}{\partial\, \boldsymbol{\phi}_{tj}}. \tag{4.14}$$

The drift term vanishes in the SGLD step since the preconditioner is assumed to be the identity matrix. The SGRLD algorithm in (4.13) takes steps in the steepest ascent on the manifold defined by the metric $G(\boldsymbol{\phi}_t)$. For many statistical models, the Fisher information matrix is intractable, however we will show in the next section that using the Vecchia's approximation we can compute the Fisher information and its inverse without incurring a high computational cost. Therefore, we use the Fisher information matrix, denoted $\mathscr{I}(\boldsymbol{\phi})$, for $G(\boldsymbol{\phi})$.

4.3.2 Derivation of Gradient and Fisher Information for SGRLD

Given an index set for a mini-batch subset of the data $\mathscr{B}$, the log-likelihood in (4.7) decomposes as the sum of log-conditional densities of the $Y(\mathbf{s}_i)$ given the conditioning points $Y(\mathbf{s}_{(i)})$. Computing the gradient of these conditional densities is analytically complicated and not computationally tractable. We follow Guinness (2019) and first rewrite the log-conditional densities in terms of marginal densities then compute the gradients and Fisher information. Let $u_i = Y(\mathbf{s}_{(i)})$, the set of neighbours, and $v_i = (Y(\mathbf{s}_{(i)}), Y(\mathbf{s}_i))$, the vector of concatenating the i^{th} observation and its neighbours. Let $\mathbf{Q}_i$ and $\mathbf{R}_i$ be the covariate matrices for u_i and v_i respectively, and let $\mathbf{A}_i$ and $\mathbf{B}_i$ denote the covariance matrices of u_i and v_i. The minibatch

log-likelihood in (4.7) can thus be written as

$$\begin{aligned}\bar{\ell}_{\mathscr{B}}(\boldsymbol{\phi}) &= \sum_{i\in\mathscr{B}} \log f(v_i \mid \boldsymbol{\phi}) - \log f(u_i \mid \boldsymbol{\phi}) \\ &= -\frac{1}{2}\sum_{i\in\mathscr{B}} \log|\mathbf{B}|_i - \log|\mathbf{A}|_i \\ &\quad -\frac{1}{2}\sum_{i\in\mathscr{B}} [(v_i - \mathbf{R}_i\boldsymbol{\beta})^{\mathrm{T}}\mathbf{B}_i^{-1}(v_i - \mathbf{R}_i\boldsymbol{\beta}) - (u_i - \mathbf{Q}_i\boldsymbol{\beta})^{\mathrm{T}}\mathbf{A}_i^{-1}(u_i - \mathbf{Q}_i\boldsymbol{\beta})] - \frac{n_{\mathscr{B}}}{2}\log(2\pi).\end{aligned} \tag{4.15}$$

In order to compute the log-likelihood, we need the following quantities

$$p^1_{\mathscr{B}}(\boldsymbol{\theta}) = \sum_{i\in\mathscr{B}} \log|\mathbf{B}|_i - \log|\mathbf{A}|_i \tag{4.16}$$

$$p^2_{\mathscr{B}}(\boldsymbol{\theta}) = \sum_{i\in\mathscr{B}} (v_i^{\mathrm{T}}\mathbf{B}_i^{-1}v_i - u_i^{\mathrm{T}}\mathbf{A}_i^{-1}u_i) \tag{4.17}$$

$$p^3_{\mathscr{B}}(\boldsymbol{\theta}) = \sum_{i\in\mathscr{B}} (\mathbf{R}_i^{\mathrm{T}}\mathbf{B}_i^{-1}v_i - \mathbf{Q}_i^{\mathrm{T}}\mathbf{A}_i^{-1}u_i) \tag{4.18}$$

$$p^4_{\mathscr{B}}(\boldsymbol{\theta}) = \sum_{i\in\mathscr{B}} (\mathbf{R}_i^{\mathrm{T}}\mathbf{B}_i^{-1}\mathbf{R}_i - \mathbf{Q}_i^{\mathrm{T}}\mathbf{A}_i^{-1}\mathbf{Q}_i). \tag{4.19}$$

The quantities in (4.16) - (4.19) only depend on the covariance parameters $\boldsymbol{\theta}$ via $\mathbf{A}_i$ and $\mathbf{B}_i$ and not the mean parameters $\boldsymbol{\beta}$. We can now write the minibatch log-likelihood as

$$\bar{\ell}_{\mathscr{B}}(\boldsymbol{\phi}) = -\frac{n_{\mathscr{B}}}{2}\log(2\pi) - \frac{1}{2}\left[p^1_{\mathscr{B}}(\boldsymbol{\theta}) + p^2_{\mathscr{B}}(\boldsymbol{\theta}) - 2\boldsymbol{\beta}^{\mathrm{T}}p^3_{\mathscr{B}}(\boldsymbol{\theta}) + \boldsymbol{\beta}^{\mathrm{T}}p^4_{\mathscr{B}}(\boldsymbol{\theta})\boldsymbol{\beta}\right]. \tag{4.20}$$

Mean parameters

The gradient of the minibatch log-likelihood with respect to the mean parameters $\boldsymbol{\beta}$ is

$$\frac{\partial\bar{\ell}_{\mathscr{B}}(\boldsymbol{\beta},\boldsymbol{\theta})}{\partial\boldsymbol{\beta}} = p^3_{\mathscr{B}}(\boldsymbol{\theta}) - p^4_{\mathscr{B}}(\boldsymbol{\theta})\boldsymbol{\beta}. \tag{4.21}$$

For the Fisher information, recall that if a random vector follows multivariate normal model with mean and variance parameterized by two different parameter vectors, *i.e.*, $W \sim \mathscr{N}(\mu(\boldsymbol{\beta}), \Sigma(\boldsymbol{\theta}))$, then the Fisher information is block diagonal $\mathscr{I}(\boldsymbol{\phi}) = \mathrm{diag}(\mathscr{I}(\boldsymbol{\beta}), \mathscr{I}(\boldsymbol{\theta}))$. Furthermore, let $J_{\boldsymbol{\beta}}$ be the Jacobian of $\mu(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$. Then the Fisher information matrix is analytically

available (Mardia and Marshall 1984) and takes the form

$$\mathscr{I}(\boldsymbol{\beta}) = J_{\boldsymbol{\beta}} \Sigma^{-1} J_{\boldsymbol{\beta}}^{\mathrm{T}} \tag{4.22}$$

$$\mathscr{I}(\boldsymbol{\theta})_{jk} = \frac{1}{2} \mathrm{Tr}\left(\Sigma^{-1} \frac{\partial \Sigma}{\partial \boldsymbol{\theta}_j} \Sigma^{-1} \frac{\partial \Sigma}{\partial \boldsymbol{\theta}_k} \right). \tag{4.23}$$

Using (4.22) and the chain rule property of the Fisher information, $\mathscr{I}_{Y(s_i)|u_i}(\boldsymbol{\phi}) = \mathscr{I}_{v_i}(\boldsymbol{\phi}) - \mathscr{I}_{u_i}(\boldsymbol{\phi})$ and summing over the components of the log-likelihood we get

$$\mathscr{I}_{\mathscr{B}}(\boldsymbol{\beta}) = \sum_{i \in \mathscr{B}} (\mathbf{R}_i^{\mathrm{T}} \mathbf{B}_i^{-1} \mathbf{R}_i - \mathbf{Q}_i^{\mathrm{T}} \mathbf{A}_i^{-1} \mathbf{Q}_i) = p^4_{\mathscr{B}}(\boldsymbol{\theta}). \tag{4.24}$$

Hence the Fisher information of $\boldsymbol{\beta}$ is constant with respect to the mean parameters. In addition, since $\mathscr{I}(\boldsymbol{\phi})$ is block diagonal, the drift term which represents how $\mathscr{I}(\boldsymbol{\beta})$ changes with respect to $\boldsymbol{\phi}$ is $\Gamma_{\mathscr{B}}(\boldsymbol{\beta}) = 0_p$. The SGRLD step for regression parameters is thus

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + h_t p^4_{\mathscr{B}}(\boldsymbol{\theta}_t)^{-1} \left(p^3_{\mathscr{B}}(\boldsymbol{\theta}_t) - p^4_{\mathscr{B}}(\boldsymbol{\theta}_t) \boldsymbol{\beta}_t \right) + \sqrt{2h_t} p^4_{\mathscr{B}}(\boldsymbol{\theta})^{-1/2} e_t. \tag{4.25}$$

Covariance parameters

For the covariance parameters, we first start by computing the partial derivatives of the quantities defined in (4.16)-(4.19) with respect to the components of $\boldsymbol{\theta}$, $p^k_j(\boldsymbol{\theta}) = \partial p^k_{\mathscr{B}}(\boldsymbol{\theta}) / \partial \theta_j$ for $j \in \{1, \ldots, 4\}$

$$p^1_j(\boldsymbol{\theta}) = \sum_{i \in \mathscr{B}} \left(\mathrm{Tr}(\mathbf{B}_i^{-1} \frac{\partial \mathbf{B}_i}{\partial \theta_j}) - \mathrm{Tr}(\mathbf{A}_i^{-1} \frac{\partial \mathbf{A}_i}{\partial \theta_j}) \right) \tag{4.26}$$

$$p^2_j(\boldsymbol{\theta}) = \sum_{i \in \mathscr{B}} \left(v_i^{\mathrm{T}} \mathbf{B}_i^{-1} \frac{\partial \mathbf{B}_i}{\partial \theta_j} \mathbf{B}_i^{-1} v_i - u_i^{\mathrm{T}} \mathbf{A}_i^{-1} \frac{\partial \mathbf{A}_i}{\partial \theta_j} \mathbf{A}_i^{-1} u_i \right) \tag{4.27}$$

$$p^3_j(\boldsymbol{\theta}) = \sum_{i \in \mathscr{B}} \left(\mathbf{R}_i^{\mathrm{T}} \mathbf{B}_i^{-1} \frac{\partial \mathbf{B}_i}{\partial \theta_j} \mathbf{B}_i^{-1} v_i - \mathbf{Q}_i^{\mathrm{T}} \mathbf{A}_i^{-1} \frac{\partial \mathbf{A}_i}{\partial \theta_j} \mathbf{A}_i^{-1} u_i \right) \tag{4.28}$$

$$p^4_j(\boldsymbol{\theta}) = \sum_{i \in \mathscr{B}} \left(\mathbf{R}_i^{\mathrm{T}} \mathbf{B}_i^{-1} \frac{\partial \mathbf{B}_i}{\partial \theta_j} \mathbf{B}_i^{-1} \mathbf{R}_i - \mathbf{Q}_i^{\mathrm{T}} \mathbf{A}_i^{-1} \frac{\partial \mathbf{A}_i}{\partial \theta_j} \mathbf{A}_i^{-1} \mathbf{Q}_i \right) \tag{4.29}$$

$$\frac{\partial \bar{\ell}_{\mathscr{B}}(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \theta_j} = -\frac{1}{2} \left[p^1_j(\boldsymbol{\theta}) + p^2_j(\boldsymbol{\theta}) - 2 p^3_j(\boldsymbol{\theta}) \boldsymbol{\beta} + \boldsymbol{\beta}^{\mathrm{T}} p^4_j(\boldsymbol{\theta}) \boldsymbol{\beta} \right]. \tag{4.30}$$

Using (4.23) and the chain rule decomposition of the Fisher information, we derive the analytic form of the Fisher information and drift term for the covariance parameters

$$\mathscr{I}_{\mathscr{B}}(\boldsymbol{\theta})_{jk} = \frac{1}{2}\sum_{i\in\mathscr{B}} \mathrm{Tr}\left(\mathbf{B}_i^{-1}\frac{\partial \mathbf{B}_i}{\partial \theta_j}\mathbf{B}_i^{-1}\frac{\partial \mathbf{B}_i}{\partial \theta_k}\right) - \mathrm{Tr}\left(\mathbf{A}_i^{-1}\frac{\partial \mathbf{A}_i}{\partial \theta_j}\mathbf{A}_i^{-1}\frac{\partial \mathbf{A}_i}{\partial \theta_k}\right) \tag{4.31}$$

$$\begin{aligned}\frac{\partial \mathscr{I}_{\mathscr{B}}(\boldsymbol{\theta})_{jk}}{\partial \theta_k} &= \sum_{i\in\mathscr{B}} \mathrm{Tr}\left(\mathbf{B}_i^{-1}\frac{\partial^2 \mathbf{B}_i}{\partial \theta_j \partial \theta_k}\mathbf{B}_i^{-1}\frac{\partial \mathbf{B}_i}{\partial \theta_k}\right) - \mathrm{Tr}\left(\mathbf{B}_i^{-1}\frac{\partial \mathbf{B}_i}{\partial \theta_j}\mathbf{B}_i^{-1}\frac{\partial \mathbf{B}_i}{\partial \theta_k}\mathbf{B}_i^{-1}\frac{\partial \mathbf{B}_i}{\partial \theta_k}\right) \\ &\quad - \sum_{i\in\mathscr{B}} \mathrm{Tr}\left(\mathbf{A}_i^{-1}\frac{\partial^2 \mathbf{A}_i}{\partial \theta_j \partial \theta_k}\mathbf{A}_i^{-1}\frac{\partial \mathbf{A}_i}{\partial \theta_k}\right) - \mathrm{Tr}\left(\mathbf{A}_i^{-1}\frac{\partial \mathbf{A}_i}{\partial \theta_j}\mathbf{A}_i^{-1}\frac{\partial \mathbf{A}_i}{\partial \theta_k}\mathbf{A}_i^{-1}\frac{\partial \mathbf{A}_i}{\partial \theta_k}\right)\end{aligned} \tag{4.32}$$

$$\Gamma_{\mathscr{B}}(\boldsymbol{\theta})_j = -\sum_k \mathscr{I}_{\mathscr{B}}(\boldsymbol{\theta})^{-1}_{j\cdot}\frac{\partial \mathscr{I}_{\mathscr{B}}(\boldsymbol{\theta})}{\partial \theta_k}\mathscr{I}_{\mathscr{B}}(\boldsymbol{\theta})^{-1}_{\cdot k}. \tag{4.33}$$

4.4 Simulation study

In this section, we test our proposed SGRLD method in (4.13) method on synthetic data and assess its performance against state-of-the-art Bayesian methods. We use Mean Squared Error (MSE) and coverage of credible intervals of posterior MCMC estimators to evaluate estimation of the spatial covariance parameters, and we use the Effective sample sizes (ESS) (Heidelberger and Welch 1981) per minute to gauge computational efficiency of MCMC algorithms. We present results only for the spatial covariance parameters because the results are similar across methods for $\boldsymbol{\beta}$.

4.4.1 Data generation

We generate data on a regular rectangular grid formed with n_1 locations on the x-axis and n_2 on the y-axis, with a total number of point $N = n_1 \cdot n_2$ and grid spacing one. We consider $N = \{10^4, 10^5, 10^6\}$ for $n_1 = \{100, 300, 1000\}$ and $n_2 = N/n_1$. We generate the Gaussian process $Z(\mathbf{s})$ from a Matérn kernel with possible smoothness values $\nu \in \{0.5, 1.0, 1.5\}$. The length-scale parameter is chosen such that the correlation function is approximately 10^{-4} for the maximum distance between two points in the grid. We fix the spatial variance $\sigma^2 = 5$, and consider different scenarios for the observation noise based on the proportion of variance $\kappa = \tau^2/\sigma^2 \in \{0.2, 1.0, 5.0\}$. Let $\mathbf{X}_i = (1, x_i)$, the covariate for the i^{th} site, the mean of the Gaussian process will take the form $\mathbb{E}[Y(\mathbf{s}_i)] = \beta_0 + \beta_1 \cos(x_i)$, where $\beta_0 = -3$, and $\beta_1 = 5$, and $x_i \overset{iid}{\sim}$ Uniform$(-3, 3)$. For $N = 10^6$, generating a Gaussian process is computationally infeasible, thus we generate a Vecchia approximated Gaussian process with $m = 120$ neighbors for each site. For each

N, we randomly sample ν and κ we generate 100 datasets and record the posterior mean and posterior credible intervals for each parameter.

4.4.2 Competing methods and metrics

We compare our SGRLD method with four different MCMC methods. The first three are SG methods with adaptive drifts. The last method uses the full dataset to sample the posterior distribution using the Vecchia approximation. The three SGMCMC methods all use momentum and past gradient information to estimate the curvature and accelerate the convergence. These methods extend the momentum methods used in SG optimization methods for faster exploration of the posterior. The first method is Preconditioned SGLD (pSGLD) of Li et al. (2016a) that uses the Root Mean Square Propagation (RMSPROP) (Hinton et al. 2012) algorithm to estimate a diagonal preconditioner for the minibatch gradient and injected noise. The second method is ADAMSGLD (Kim et al. 2022) that extends the widely used ADAM optimizer (Kingma and Ba 2014) to the SGLD setting. ADAMSGLD approximates the first-order and second-order moments of the minibatch gradients to construct a preconditioner. Finally, we also include the performance of Momentum SGLD (MSGLD) where no preconditioner is used but past gradient information is used to accelerate the exploration of the posterior. The details of the above algorithms are included in the Appendix C.1. The last method we consider is the Nearest Neighbor Gaussian Process (NNGP) method (Datta et al. 2016). This method is the standard MCMC method based on the Vecchia approximation and is implemented in the R package spNNGP (Finley et al. 2022).

For the SGMCMC methods, the batch size is set to 250 when the number of location is 10^4 and 500 for the other two cases. We noticed during our experiments that batch sizes in the order of 200 perform better than smaller size ones, with very similar performance to larger ones. The number of epochs will depend on the size of data, and is chosen such that the total number of iterations is 20000, of which a quarter are discarded as burn-in. The learning rate is divided by a factor of 2 every 5 epochs, so the final learning rate is set at 1% of the initial value. A first tentative value of the learning rate is set at $1/N$, then reduced until the norm of the first step is less than one. We noticed that the appropriate learning rate for our SGRLD method is within one to two orders of magnitude large than the learning rate for the other SG sampling methods. For the NNGP method, we take 2000 samples when $N < 10^5$ and 1000 otherwise. For all the methods we use a non-informative flat prior on the regression parameters. For the covariance parameters, the set the following priors

$$\begin{aligned}\rho &\sim \text{Gamma}(9.0, 2.0)\\ \nu &\sim \text{Log-Normal}(1.0, 1.0)\\ \sigma^2 &\sim \text{Gamma}(0.1, 0.1)\\ \tau^2 &\sim \text{Gamma}(0.1, 0.1)\end{aligned}$$

The prior 90% credible intervals for ρ and ν are $(2.06, 7.88)$ and $(0.52, 14.08)$ respectively. Which represent weakly informative priors.

For all the methods, the size of the conditioning set is fixed at $m = 15$. The conditioning sets were selected using the max-min ordering (Katzfuss and Guinness 2021) for $N < 10^6$, and random ordering otherwise. Katzfuss and Guinness (2021) showed that the max-min ordering results in significant improvements over other coordinate based orderings. However, when N is very large, the cost of max-min ordering becomes prohibitive.

4.4.3 Results

Table 4.1 gives the MSE results. Our SGRLD method outperforms all the others with very low MSE across parameters. In particular, the SGMCMC methods all outperform the NNGP method. In our experiments, we noticed that the NNGP method suffers from very slow mixing due to the M-H step necessary for sampling the covariance parameters. In fact, even if we start the NNGP sampling process at true values of the covariance parameters, and reduce the variance of the proposal distribution, the acceptance rate of the M-H step stays below 15%. None of the SGMCMC methods requires any such step as long as the learning rate is kept small.

Table 4.1: Mean squared error of covariance parameters. Mean (Monte Carlo standard errors) over 100 simulations for each sample size N. The proposed SGRLD method is compared with other SGMCMC methods (pSGLD, ADAMSGLD, MSGLD) and the full likelihood NNGP method.

N	Algorithm	Variance, σ^2	Range, ρ	Smoothness, ν	Nugget, τ^2
10^4	pSGLD	0.074(0.013)	0.039(0.008)	0.103(0.017)	$0.002(4\cdot10^{-4})$
	ADAMSGLD	0.075(0.017)	0.036(0.008)	0.129(0.023)	$0.002(6\cdot10^{-4})$
	MSGLD	0.066(0.014)	0.034(0.008)	0.108(0.0196)	$0.002(6\cdot10^{-4})$
	NNGP	0.414(0.131)	0.095(0.071)	0.162(0.106)	$0.093(2.4\cdot10^{-2})$
	SGRLD	0.056(0.016)	0.031(0.006)	0.077(0.013)	$0.001(10^{-4})$
10^5	pSGLD	0.008(0.001)	0.002(0.0003)	0.011(0.0019)	$1\cdot10^{-4}(2\cdot10^{-5})$
	ADAMSGLD	0.014(0.005)	0.008(0.002)	0.031(0.008)	$1\cdot10^{-4}(2\cdot10^{-4})$
	MSGLD	0.017(0.001)	$0.003(5\cdot10^{-4)}$	0.019(0.002)	$2\cdot10^{-4}(4\cdot10^{-5})$
	NNGP	0.116(0.030)	0.024(0.01)	0.118(0.08)	$4\cdot10^{-2}(0.01)$
	SGRLD	$0.005(8\cdot10^{-4})$	$0.001(1.0\cdot10^{-4})$	$0.008(1.8\cdot10^{-3})$	$10^{-4}(2\cdot10^{-}5)$
10^6	pSGLD	0.003(0.001)	0.003(0.0008)	0.002(0.0014)	$3.1\cdot10^{-4}(6\cdot10^{-5})$
	ADAMSGLD	0.009(0.002)	0.006(0.002)	0.026(0.007)	$2\cdot10^{-4}(9\cdot10^{-5})$
	MSGLD	$0.011(1.8\cdot10^{-3})$	$0.003(5\cdot10^{-4)}$	0.019(0.002)	$1\cdot10^{-5}(3\cdot10^{-5})$
	NNGP	0.078(0.055)	0.016(0.009)	0.126(0.086)	0.08(0.049)
	SGRLD	$0.002(3\cdot10^{-4})$	$0.001(1\cdot10^{-4})$	$0.004(6.1\cdot10^{-3})$	$0.4\cdot10^{-4}(1\cdot10^{-}5)$

Table 4.2 summarizes the results for the coverage of the 95% credible intervals. Our SGRLD method again outperforms the other methods. One exception is that the pSGLD algorithm surpasses the SGRLD in the coverage of the variance parameter. Across methods, the smoothness parameter consistently has the lowest coverage, followed by the range parameter. Even for $N = 10^6$, MSGLD, ADAMSGLD and NNGP fail to attain attain a 90% coverage rate. Whilst the SGRLD coverage rate for both parameters is higher than 90% even for $N = 10^4$.

Table 4.2: Coverage of the 95% credible intervals for the covariance parameters. Mean (Monte Carlo standard errors) over 100 simulations for each sample size N. The proposed SGRLD method is compared with other SGMCMC methods (pSGLD, ADAMSGLD, MSGLD) and the full likelihood NNGP method..

N	Algorithm	Variance, σ^2	Range, ρ	Smoothness, ν	Nugget, τ^2
10^4	pSGLD	0.977(0.02)	0.845(0.06)	0.815(0.06)	0.931(0.05)
	ADAMSGLD	0.886(0.05)	0.791(0.08)	0.647(0.08)	0.636(0.05)
	MSGLD	0.793(0.03)	0.847(0.07)	0.709(0.07)	0.683(0.05)
	NNGP	0.783(0.06)	0.776(0.05)	0.614(0.07)	0.812(0.01)
	SGRLD	0.955(0.03)	0.924(0.05)	0.909(0.04)	0.935(0.01)
10^5	pSGLD	0.991(0.03)	0.913(0.04)	0.862(0.05)	0.965(0.02)
	ADAMSGLD	0.861(0.03)	0.754(0.07)	0.814(0.03)	0.738(0.05)
	MSGLD	0.896(0.04)	0.881(0.07)	0.774(0.08)	0.872(0.07)
	NNGP	0.826(0.05)	0.758(0.04)	0.714(0.03)	0.872(0.02)
	SGRLD	0.957(0.01)	0.964(0.01)	0.948(0.01)	$0.932(5\cdot10^{-3})$
10^6	pSGLD	$0.987(6\cdot10^{-3})$	0.934(0.02)	0.901(0.03)	0.961(0.01)
	ADAMSGLD	0.902(0.01)	$0.824(10^{-3})$	0.838(0.02)	0.781(0.03)
	MSGLD	$0.884(10^{-3})$	0.918(0.02)	0.846(0.01)	0.926(0.01)
	NNGP	0.866(0.03)	0.818(0.06)	0.834(0.04)	0.862(0.01)
	SGRLD	$0.968(6\cdot10^{-3})$	$0.941(8\cdot10^{-3})$	$0.929(5\cdot10^{-3})$	$0.941(2\cdot10^{-3})$

For the ESS results in Table 4.3, the SGRLD method offers superior effective samples per unit time for all the parameters. The pSGLD and MSGLD method seem to adapt to the curvature of the variance parameter, with pSGLD offering higher effective samples than SGRLD. This suggests that the computed preconditioner in pSGLD adapts mainly to the curvature of the variance term, but fails to measure the curvature of the smoothness and range. A similar behavior is also observed in the other two methods, MSGLD and ADAMSGLD. On the other hand, the ESS for SGRLD is of the same order for all the parameters. We believe this indicates that using the Fisher information matrix as a Riemannian metric provides an accurate measure of the curvature and results in higher effective samples for all the parameters. The NNGP method provides low effective sample sizes compared to the other three methods, this can be largely attributed to the low acceptance rate from the MH correction step.

Table 4.3: Effective sample size per unit time (in minutes) of covariance parameters. Mean (Monte Carlo standard errors) over 100 simulations for each sample size N. The proposed SGRLD method is compared with other SGMCMC methods (pSGLD, ADAMSGLD, MSGLD) and the full likelihood NNGP method.

N	Algorithm	Variance, σ^2	Range, ρ	Smoothness, ν	Nugget, τ^2
10^4	pSGLD	42.97(1.57)	8.43(0.54)	4.33(0.26)	9.82(0.79)
	ADAMSGLD	9.12(0.45)	4.22(0.33)	2.85(0.28)	3.80(0.48)
	MSGLD	15.68(0.95)	6.48(0.70)	3.65(0.44)	5.11(0.78)
	NNGP	1.02(0.33)	0.99(0.24)	1.11(0.75)	0.51(0.14)
	SGRLD	23.8(1.15)	23.9(1.19)	25.2(1.25)	30.5(1.55)
10^5	pSGLD	66.87(2.09)	10.06(0.65)	3.59(0.21)	11.3(0.79)
	ADAMSGLD	7.87(0.38)	2.37(0.27)	1.15(0.13)	1.64(0.24)
	MSGLD	12.92(0.67)	3.15(0.36)	1.206(0.11)	1.71(0.13)
	NNGP	0.89(0.08)	0.75(0.31)	1.02(0.14)	0.47(0.07)
	SGRLD	22.7(0.33)	22.44(0.27)	22.69(0.13)	23.23(0.34)
10^6	pSGLD	96.49(3.37)	13.68(0.81)	3.04(0.11)	9.74(0.42)
	ADAMSGLD	6.17(0.13)	4.56(0.52)	1.98(0.62)	2.36(0.83)
	MSGLD	15.07(1.01)	3.78(0.81)	2.06(0.30)	5.01(0.97)
	NNGP	0.81(0.16)	1.01(0.34)	0.28(0.05)	0.52(0.03)
	SGRLD	25.8(0.14)	26.05(0.18)	29.62(0.28)	24.07(0.27)

Given the performance of the SG based methods in this simulation study, especially the SGRLD, we conducted an additional simulation study where we focus on point estimates instead of fully Bayesian inference. In Appendix C.2, we tweak the SGRLD method and turn it into a SG Fisher scoring (SGFS) algorithm for point estimates. We compare this method to the full data gradient Fisher scoring method (Guinness 2019) already implemented in the GPGP R package (Guinness et al. 2018). We find imroved speed and estimation precision compared to the GpGp package.

4.5 Discussion

In this paper, we developed a new stochastic gradient based MCMC algorithm for scalable Bayesian inference in large spatial data settings. SG methods offer considerable speed-ups when the data size is very large. In fact, one can take hundreds or even thousands of steps in one pass through the whole datasets compared to only one step if the full dataset is used. This enables fast exploration of the posterior in significantly less time. GPs however fall within the correlated setting case where SGMCMC methods have received limited attention. Spatial correlation is a critical component of Gaussian process methods and naive subsampling would lead to random divisions of the spatial domain at each iteration. By leveraging the form of the Vecchia approximation we derive unbiased gradient estimates based on minibatches of the data. Without the Vecchia approximation subsampling strategies would always lead to biased gradient estimates. The proposed method also uses the exact Fisher information to speed up convergence and explore the parameter space efficiently. Our work contributes to the literature on scalable methods for Gaussian process, and can be extended to non Gaussian models *i.e.* classification.

CHAPTER

5

CONCLUSIONS

This dissertation introduces two main contributions to the statistical literature on Bayesian transfer learning, and proposes a scalable MCMC algorithm for massive Gaussian process regression for spatial data. The following summarizes these contributions and identifies several open areas of research related to this work.

Motivated by the computational challenges in determining band gap values for molecular crystals compared to organic molecules, Chapter 2 develops a principled Bayesian transfer learning method for deep neural networks. Deep neural network models have achieved chemical accuracy for small organic molecules but require large datasets for training and can be unstable with small datasets. The proposed transfer learning method leverages the accuracy of neural networks on organic molecules to create an efficient model for predicting band gaps in molecular crystals. The method uses a Bayesian approach by penalizing the deviation from the model output using the source task estimates. The regularization through model complexity is particularly suited for neural networks, since the individual parameters are not interpretable. The proposed Bayesian method results in interpretable variance distribution among transferred layers. This allows practitioners to identify a posteriori which layers of the model encode general representations for both source and target tasks. Simulation results show

reduced prediction errors compared to standard transfer learning methods, with reasonable prediction interval coverage. Our transfer learning method matches state-of-the-art models with only a fraction of the training data, making it cost-effective for material selection without extensive calculations. The potential applications extend beyond material science to areas where data labeling is costly, such as drug discovery.

Potential future work includes adapting the method to different network architectures and response distributions, exploring its application in binary data, incorporating prior information on layer transferability, and comparing sequential and simultaneous analyses of data sources. In addition, despite the wide success and application of transfer learning methods in deep neural network model, a lack of theoretical guarantees persists, particularly for Bayesian methods.

Following the findings in Chapter 2, Chapter 3 introduces a Bayesian estimator for transfer learning in high-dimensional settings. The simplest version of transfer learning in neural networks freezes the hidden layers, then fine-tunes the output layer. In the case of regression tasks, this can be reformulated into a linear regression on the extracted features from the last hidden layer. Based on this observation, we propose and study the theoretical guarantees of a Bayes estimator in the simple normal means model.

The proposed two-stage procedure relies only on estimates and their uncertainties from the source task. Two different estimators are presented based on different prior distributions, curated to the assumed similarity between source and target tasks. We establish frequentist-properties of Bayes estimator when the difference of mean vectors is sparse using a global local shrinkage prior. The synergy property of the estimation procedure's first and second stages is theoretically analyzed, demonstrating a total risk lower than the sum of individual risks. The estimator is shown to have negligible risk compared to the best estimator based solely on target data, indicating superior performance in transfer learning without negative transfer risk. Using simulation studies, we showcased the performance of our method compared to state-of-the-art transfer learning methods in different scenarios. The empirical finding suggests that an unbiased first-stage estimate is crucial, even when the number of parameters is larger than the size of the source task data.

Although, this work represents an important step in the theoretical analysis of Bayesian transfer learning, many open questions remain. Avenues for future work include extending the method to regression cases, non-Gaussian models, and non-continuous response cases. In addition, given that our proposed method is Bayesian, we can also investigate the properties of the posterior distribution as a whole. An interesting question is how the posterior distribution contracts around the true parameter and at what rate.

In chapter 4, our focus switches to scalable Bayesian inference for Gaussian processes in large spatial datasets. Handling large datasets with GPs poses computational challenges due to the cubic time complexity and quadratic memory requirements in the evaluation of the joint likelihood. We developed a new MCMC algorithm with considerable speed-ups compared to standard Bayesian methods when the data size is very large. We leverage the Vecchia approximation to decompose the likelihood into a sum of independent terms and use stochastic gradient MCMC methods to sample the posterior. Our method provides hundreds or even thousands of steps in one pass through the whole datasets compared to only one step if the full dataset is used. This work contributes to the recent advances in stochastic MCMC methods for correlated data settings.

In this last chapter we focused on stationary covariance functions, however, the method can be straightforwardly extended to other choices of the covariance including non-stationary and space-time covariance structures. One of our immediate future goals for this project is to publish a full R package for large spatial data analysis using GPs.

REFERENCES

Abba, M. A., Williams, J. P., and Reich, B. J. (2023). A penalized complexity prior for deep Bayesian transfer learning with application to materials informatics. *Annals of Applied Statistics*, 17(4):3241–3256.

Agarap, A. F. (2018). Deep learning using rectified linear units (RELU). *arXiv:1803.08375*.

Ahishakiye, E., Van Gijzen, M. B., Tumwiine, J., Wario, R., and Obungoloch, J. (2021). A survey on deep learning in medical image reconstruction. *Intelligent Medicine*.

Aicher, C., Ma, Y.-A., Foti, N. J., and Fox, E. B. (2019). Stochastic gradient mcmc for state space models. *SIAM Journal on Mathematics of Data Science*, 1(3):555–587.

Aicher, C., Putcha, S., Nemeth, C., Fearnhead, P., and Fox, E. B. (2021). Stochastic gradient mcmc for nonlinear state space models. *arXiv preprint arXiv:1901.10568*.

Almalki, S. J. and Nadarajah, S. (2014). Modifications of the Weibull distribution: A review. *Reliability Engineering & System Safety*, 124:32–55.

Baker, J., Fearnhead, P., Fox, E. B., and Nemeth, C. (2019). Control variates for stochastic gradient mcmc. *Statistics and Computing*, 29:599–615.

Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008). Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70:825–848.

Barbian, M. H. and Assunção, R. M. (2017). Spatial subensemble estimator for large geostatistical data. *Spatial Statistics*, 22:68–88.

Bastani, H. (2021). Predicting with proxies: Transfer learning in high dimension. *Management Science*, 67(5):2964–2984.

Batra, R. (2021). Accurate machine learning in materials science facilitated by using diverse data sources. *Nature*, 589:524–525.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer.

Bhadra, A., Datta, J., Polson, N. G., and Willard, B. (2017). The horseshoe+ estimator of ultra-sparse signals. *Bayesian Analysis*, 12(4):1105–1131.

Bhattacharya, A., Pati, D., Pillai, N. S., and Dunson, D. B. (2015). Dirichlet–Laplace priors for optimal shrinkage. *Journal of the American Statistical Association*, 110(512):1479–1490.

Bhattacharya, S. and Maiti, T. (2021). Statistical foundation of variational bayes neural networks. *Neural Networks*, 137:151–173.

Bjorck, J., Gomes, C., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. *arXiv preprint arXiv:1806.02375*.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

Bueno, A., Benitez, C., De Angelis, S., Moreno, A. D., and Ibáñez, J. M. (2019). Volcano-seismic transfer learning and uncertainty quantification with bayesian neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 58(2):892–902.

Cai, T. T., Sun, W., and Wang, W. (2019). Covariate-assisted ranking and screening for large-scale two-sample inference. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 81(2):187–234.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2009). Handling sparsity via the horseshoe. In *Artificial intelligence and statistics*, pages 73–80. PMLR.

Chandra, R. and Kapoor, A. (2020). Bayesian neural multi-source transfer learning. *Neurocomputing*, 378:54–64.

Chee, J. and Toulis, P. (2018). Convergence diagnostics for stochastic gradient descent with constant learning rate. In *International Conference on Artificial Intelligence and Statistics*, pages 1476–1485. PMLR.

Chen, C., Ding, N., and Carin, L. (2015). On the convergence of stochastic gradient mcmc algorithms with high-order integrators. In *Neural Information Processing Systems*.

Chen, C., Ye, W., Zuo, Y., Zheng, C., and Ong, S. P. (2019). Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572.

Chen, H., Zheng, L., Al Kontar, R., and Raskutti, G. (2020). Stochastic gradient descent in correlated settings: A study on gaussian processes. *Advances in neural information processing systems*, 33:2722–2733.

Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298.

Cressie, N. (1988). Spatial prediction and ordinary kriging. *Mathematical geology*, 20:405–421.

Cressie, N. and Johannesson, G. (2008). Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70:209–226.

Dalalyan, A. S. and Karagulyan, A. (2019). User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 129(12):5278–5311.

Dargan, S., Kumar, M., Ayyagari, M. R., and Kumar, G. (2019). A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, pages 1–22.

Datta, A., Banerjee, S., Finley, A. O., and Gelfand, A. E. (2016). Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111:800–812.

Datta, J. and Ghosh, J. K. (2013). Asymptotic Properties of Bayes Risk for the Horseshoe Prior. *Bayesian Analysis*, 8(1):111–132.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Dube, P., Bhattacharjee, B., Petit-Bois, E., and Hill, M. (2020). Improving transferability of deep neural networks. In *Domain Adaptation for Visual Understanding*, pages 51–64. Springer.

Dubey, K. A., J Reddi, S., Williamson, S. A., Poczos, B., Smola, A. J., and Xing, E. P. (2016). Variance reduction in stochastic gradient langevin dynamics. *Advances in neural information processing systems*, 29.

Durmus, A. and Moulines, E. (2017). Nonasymptotic convergence analysis for the unadjusted langevin algorithm.

Dzyabura, D., Jagabathula, S., and Muller, E. (2019). Accounting for discrepancies between online and offline product evaluations. *Marketing Science*, 38.

Finley, A. O., Datta, A., and Banerjee, S. (2022). spNNGP R package for nearest neighbor Gaussian process models. *Journal of Statistical Software*, 103(5):1–40.

Finley, A. O., Datta, A., Cook, B. D., Morton, D. C., Andersen, H. E., and Banerjee, S. (2019). Efficient algorithms for bayesian nearest neighbor gaussian processes. *Journal of Computational and Graphical Statistics*, 28(2):401–414.

Fortuin, V. (2021). Priors in bayesian deep learning: A review.

Furrer, R., Genton, M. G., and Nychka, D. (2006). Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15:502–523.

Gelfand, A. E. and Schliep, E. M. (2016). Spatial statistics and gaussian processes: A beautiful marriage. *Spatial Statistics*, 18:86–104. Spatial Statistics Avignon: Emerging Patterns.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Guhaniyogi, R. and Banerjee, S. (2018). Meta-kriging: Scalable bayesian modeling and inference for massive spatial datasets. *Technometrics*.

Guinness, J. (2018). Permutation and grouping methods for sharpening gaussian process approximations. *Technometrics*, 60(4):415–429. PMID: 31447491.

Guinness, J. (2019). Gaussian process learning via fisher scoring of vecchia's approximation.

Guinness, J., Katzfuss, M., and Fahmy, Y. (2018). Gpgp: fast Gaussian process computation using Vecchia's approximation. *R package version 0.1. 0*.

Hardt, M., Recht, B., and Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pages 1225–1234. PMLR.

Heaton, M. J., Datta, A., Finley, A. O., Furrer, R., Guinness, J., Guhaniyogi, R., Gerber, F., Gramacy, R. B., Hammerling, D., Katzfuss, M., et al. (2019). A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological and Environmental Statistics*, 24:398–425.

Heaton, M. J. and Johnson, J. A. (2023). Minibatch Markov chain Monte Carlo algorithms for fitting Gaussian processes. *arXiv preprint arXiv:2310.17766*.

Heidelberger, P. and Welch, P. D. (1981). A spectral method for confidence interval generation and run length control in simulations. *Communications of the ACM*, 24(4):233–245.

Hickey, J., Williams, J. P., and Hector, E. C. (2022). Transfer learning with uncertainty quantification: Random effect calibration of source to target (RECaST). *arXiv preprint arXiv:2211.16557*.

Himanen, L., Geurts, A., Foster, A. S., and Rinke, P. (2019). Data-driven materials science: status, challenges, and perspectives. *Advanced Science*, 6(21):1900808.

Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2.

Hoffman, M. D., Gelman, A., et al. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623.

Huo, H. and Rupp, M. (2018). Unified representation of molecules and crystals for machine learning. arXiv:1704.06439.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Kang, E., Liu, D., and Cressie, N. (2009). Statistical analysis of small-area data based on independence, spatial, non-hierarchical, and hierarchical models. *Computational Statistics & Data Analysis*, 53:3016–3032.

Karbalayghareh, A., Qian, X., and Dougherty, E. R. (2018). Optimal Bayesian transfer learning. *IEEE Transactions on Signal Processing*, 66(14):3724–3739.

Katzfuss, M. and Cressie, N. (2011). Spatio-temporal smoothing and em estimation for massive remote-sensing data sets. *Journal of Time Series Analysis*, 32:430–446.

Katzfuss, M. and Guinness, J. (2021). A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*, 36(1):124–141.

Kaufman, C. G., Schervish, M. J., and Nychka, D. W. (2008). Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, 103:1545–1555.

Kim, S., Song, Q., and Liang, F. (2022). Stochastic gradient langevin dynamics with adaptive drifts. *Journal of statistical computation and simulation*, 92(2):318–336.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kittel, C. and McEuen, P. (2019). Introduction to solid state physics.

Kohn, W. and Sham, L. J. (1965). Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133–A1138.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Lei, Q., Hu, W., and Lee, J. (2022). Near-optimal linear regression under distribution shift. In *International Conference on Machine Learning*, pages 6164–6174. PMLR.

Li, C., Chen, C., Carlson, D., and Carin, L. (2016a). Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Li, Q. and Lin, N. (2010). The Bayesian elastic net. *Bayesian Analysis*, 5(1):151.

Li, S., Cai, T. T., and Li, H. (2022). Transfer learning for high-dimensional linear regression: Prediction, estimation and minimax optimality. *Journal of the Royal Statistical Society Series B*, 84(1):149–173.

Li, W., Ahn, S., and Welling, M. (2016b). Scalable mcmc for mixed membership stochastic blockmodels. In *Artificial Intelligence and Statistics*, pages 723–731. PMLR.

Liu, Y., Gao, Y., and Yin, W. (2020). An improved analysis of stochastic gradient descent with momentum. *arXiv preprint arXiv:2007.07989*.

Ma, S., Ma, Y., Zhang, B., Tian, Y., and Jin, Z. (2021). Forecasting system of computational time of dft/tddft calculations under the multiverse ansatz via machine learning and cheminformatics. *ACS Omega*, 6(3):2001–2024.

Ma, Y., Ma, Y.-A., Chen, T., and Fox, E. B. (2015). A complete recipe for stochastic gradient mcmc. In *Neural Information Processing Systems*.

Ma, Y.-A., Foti, N. J., and Fox, E. B. (2017). Stochastic gradient mcmc methods for hidden markov models. In *International Conference on Machine Learning*, pages 2265–2274. PMLR.

Makalic, E. and Schmidt, D. F. (2015). A simple sampler for the horseshoe estimator. *IEEE Signal Processing Letters*, 23(1):179–182.

Mardia, K. V. and Marshall, R. J. (1984). Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika*, 71(1):135–146.

Maurer, A., Pontil, M., and Romera-Paredes, B. (2016). The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32.

McNeish, D. M. (2015). Using lasso for predictor selection and to assuage overfitting: A method long overlooked in behavioral sciences. *Multivariate behavioral research*, 50(5):471–484.

Medsker, L. R. and Jain, L. (2001). Recurrent neural networks. *Design and Applications*, 5:64–67.

Nagi, J., Ducatelle, F., Di Caro, G. A., Cireşan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J., and Gambardella, L. M. (2011). Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347. IEEE.

Nemeth, C. and Fearnhead, P. (2021). Stochastic gradient markov chain monte carlo. *Journal of the American Statistical Association*, 116(533):433–450.

Newton, D., Yousefian, F., and Pasupathy, R. (2018). Stochastic gradient descent: Recent trends. *Recent advances in optimization and modeling of contemporary problems*, pages 193–220.

Olsthoorn, B., Geilhufe, R. M., Borysov, S. S., and Balatsky, A. V. (2019). Band gap prediction for large organic crystal structures with machine learning. *Advanced Quantum Technologies*, 2(7-8):1900023.

O'Hagan, A. (2006). Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering & System Safety*, 91(10-11):1290–1300.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Pinsker, M. S. (1980). Optimal filtering of square-integrable signals in gaussian noise. *Problemy Peredachi Informatsii*, 16(2):52–68.

Raghu, M., Zhang, C., Kleinberg, J. M., and Bengio, S. (2019). Transfusion: Understanding transfer learning with applications to medical imaging. *CoRR*, abs/1902.07208.

Raina, R., Ng, A. Y., and Koller, D. (2006). Constructing informative priors using transfer learning. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 713–720, New York, NY, USA. Association for Computing Machinery.

Robbins, H. and Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407.

Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71:319–392.

Saha, S. and Bradley, J. R. (2023). Incorporating subsampling into Bayesian models for high-dimensional spatial data. *arXiv preprint arXiv:2305.13221*.

Sang, H., Jun, M., and Huang, J. Z. (2011). Covariance approximation for large multivariate spatial data sets with an application to multiple climate model errors. *The Annals of Applied Statistics*, pages 2519–2548.

Schütt, K. T., Kindermans, P.-J., Sauceda, H. E., Chmiela, S., Tkatchenko, A., and Müller, K.-R. (2017). Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *arXiv preprint arXiv:1706.08566*.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W., Bridgland, A., et al. (2020). Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710.

Simpson, D., Rue, H., Riebler, A., Martins, T. G., and Sørbye, S. H. (2017a). Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors. *Statistical Science*, 32(1):1 – 28.

Simpson, D., Rue, H., Riebler, A., Martins, T. G., and Sørbye, S. H. (2017b). Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors. *Statistical Science*, 32(1):1 – 28.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Stein, C. M. (1981). Estimation of the Mean of a Multivariate Normal Distribution. *The Annals of Statistics*, 9(6):1135 – 1151.

Stein, M. L. (1999). *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media.

Stein, M. L. (2002). The screening effect in Kriging. *The Annals of Statistics*, 30(1):298 – 323.

Stein, M. L. (2011). 2010 Rietz lecture: When does the screening effect hold? *The Annals of Statistics*, 39(6):2795–2819.

Stein, M. L., Chi, Z., and Welty, L. J. (2004). Approximating likelihoods for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(2):275–296.

Stuke, A., Kunkel, C., Golze, D., Todorović, M., Margraf, J. T., Reuter, K., Rinke, P., and Oberhofer, H. (2020). Atomic structures and orbital energies of 61,489 crystal-forming organic molecules. *Scientific data*, 7(1):58.

Stuke, A., Kunkel, C., Golze, D., Todorović, M., Margraf, J. T., Reuter, K., Rinke, P., and Oberhofer, H. a. (2019). "oe62-dataset" of molecular orbital energies.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer.

Teh, Y., Thiéry, A., and Vollmer, S. (2016). Consistency and fluctuations for stochastic gradient langevin dynamics. *Journal of Machine Learning Research*, 17.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288.

Tzikas, D. G., Likas, A. C., and Galatsanos, N. P. (2008). The variational approximation for bayesian inference. *IEEE Signal Processing Magazine*, 25(6):131–146.

Van de Walle, C. G. (2012). *Wide-band-gap Semiconductors*. Elsevier.

van der Pas, S., Szabó, B., and van der Vaart, A. (2017). Adaptive posterior contraction rates for the horseshoe. *Electronic Journal of Statistics*, 11(2):3196 – 3225.

van der Pas, S. L., Kleijn, B. J. K., and van der Vaart, A. W. (2014). The horseshoe estimator: Posterior concentration around nearly black vectors. *Electronic Journal of Statistics*, 8(2):2585 – 2618.

van Erp, S., Oberski, D. L., and Mulder, J. (2019). Shrinkage priors for bayesian penalized regression. *Journal of Mathematical Psychology*, 89:31–50.

Vecchia, A. V. (1988). Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):297–312.

Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2018.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(1):1–40.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning*.

Wilson, A. G. (2020). The case for Bayesian deep learning. *arXiv preprint arXiv:2001.10995*.

Wohlert, J., Munk, A., Sengupta, S., and Laumann, F. (2018). Bayesian transfer learning for deep networks. *viXra*.

Woodard, R. (2000). Interpolation of spatial data: some theory for kriging. *Technometrics*, 42(4):436.

Xia, Y., Cai, T. T., and Sun, W. (2019). GAP: a general framework for information pooling in two-sample sparse inference. *Journal of the American Statistical Association*.

Yang, F., Zhang, H. R., Wu, S., Su, W. J., and Ré, C. (2023). Analysis of information transfer from heterogeneous sources via precise high-dimensional asymptotics. *arXiv preprint arXiv:2010.11750*.

Yang, H., Jiao, S., and Sun, P. (2020). Bayesian-convolutional neural network model transfer learning for image detection of concrete water-binder ratio. *IEEE Access*, 8:35350–35367.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*.

Young, T., Hazarika, D., Poria, S., and Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75.

Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. (2018). Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026.

Zhang, Y. D., Naughton, B. P., Bondell, H. D., and Reich, B. J. (2022). Bayesian regression using a prior on the model fit: The R2-D2 shrinkage prior. *Journal of the American Statistical Association*, 117(538):862–874.

Zhou, C., Zhang, J., Liu, J., Zhang, C., Shi, G., and Hu, J. (2020). Bayesian transfer learning for object detection in optical remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 58(11):7705–7719.

APPENDICES

APPENDIX

A

SUPPLEMENTARY MATERIAL FOR CHAPTER 2

Appendix A.1: Checking the approximation

The complete procedure to evaluate the how the approximation in (2.5) will result in the appropriate distribution on the scaled KL divergence is detailed in the following steps:

- Let the architecture of the network be $\{64, 32, 1\}$
- Generate $K = 50$ target data sets $\mathbf{X}_k$ (no need for the output);
- For each data set generate random parameters using the Glorot method (Glorot and Bengio 2010) and do the following.
  - Compute the gradient of model output for each entry in each data set and specify λ value for τ distribution;
  - Sample $S = 20000\,\tau$'s; and for each sampled τ do:

1. Sample $M = 1000$ parameters $\boldsymbol{\theta}_i | \widehat{\boldsymbol{\theta}}, \boldsymbol{\tau}$ and compute corresponding NN output for each target data set $\mathbf{X}_k$;
2. Now we have $N * S$ outputs from model $f_{\boldsymbol{\theta}_i}$ for each target $\mathbf{X}_k$;
3. We want to approximate for each $k = 1 \dots K$:

$$KL(p||q) = \mathscr{E}\left[\log\left(\frac{p(Y)}{q(Y)}\right)\right] \text{ where } \quad Y \sim p$$

so we use Monte Carlo integration,

$$\widehat{KL}(p||q) = N^{-1}\sum_{i=1}^{N}\log\left(\frac{p(f(X_i))}{q(f(X_i))}\right) \text{ where } \quad f(X_i) \overset{iid}{\sim} p();$$

4. We also need to approximate $p(Y_i)$ using MC integration,

$$\log\big(p(f(X_i))\big) \approx \log\left[M^{-1}\sum_{j=1}^{M}\mathscr{N}(f(X_i); f(X_i; \boldsymbol{\theta}_j), \boldsymbol{\sigma})\right]$$

5. Compute the KL-divergence empirically using steps 3 and 4;

– Now we have a $\widehat{\text{KL}}$ value for each sample $\boldsymbol{\tau}$;

– Plot the empirical cumulative survival function against theoretical one for pre-specified value of λ.

Appendix A.2: Optimization Details

For the training of all the non-Bayesian models we used stochastic gradient optimization (Bottou 2012) with the Adam optimizer (Kingma and Ba 2014) implemented in `Tensorflow` and `Keras` python libraries. We use default tuning parameters, e.g., the learning rate is initialized at 0.001 and decreased by a factor of 0.7 when the loss function reaches a plateau. Furthermore, a combination of L1 and L2 regularization for the parameters of the models. The choice of the batch size was performed using cross-validation for all data sets and models. We considered $\{16, 32, 64, 128\}$ possible choices. Furthermore, to avoid overfitting early stopping was used. Training is terminated when the validation loss does not decrease for 6 successive epochs.

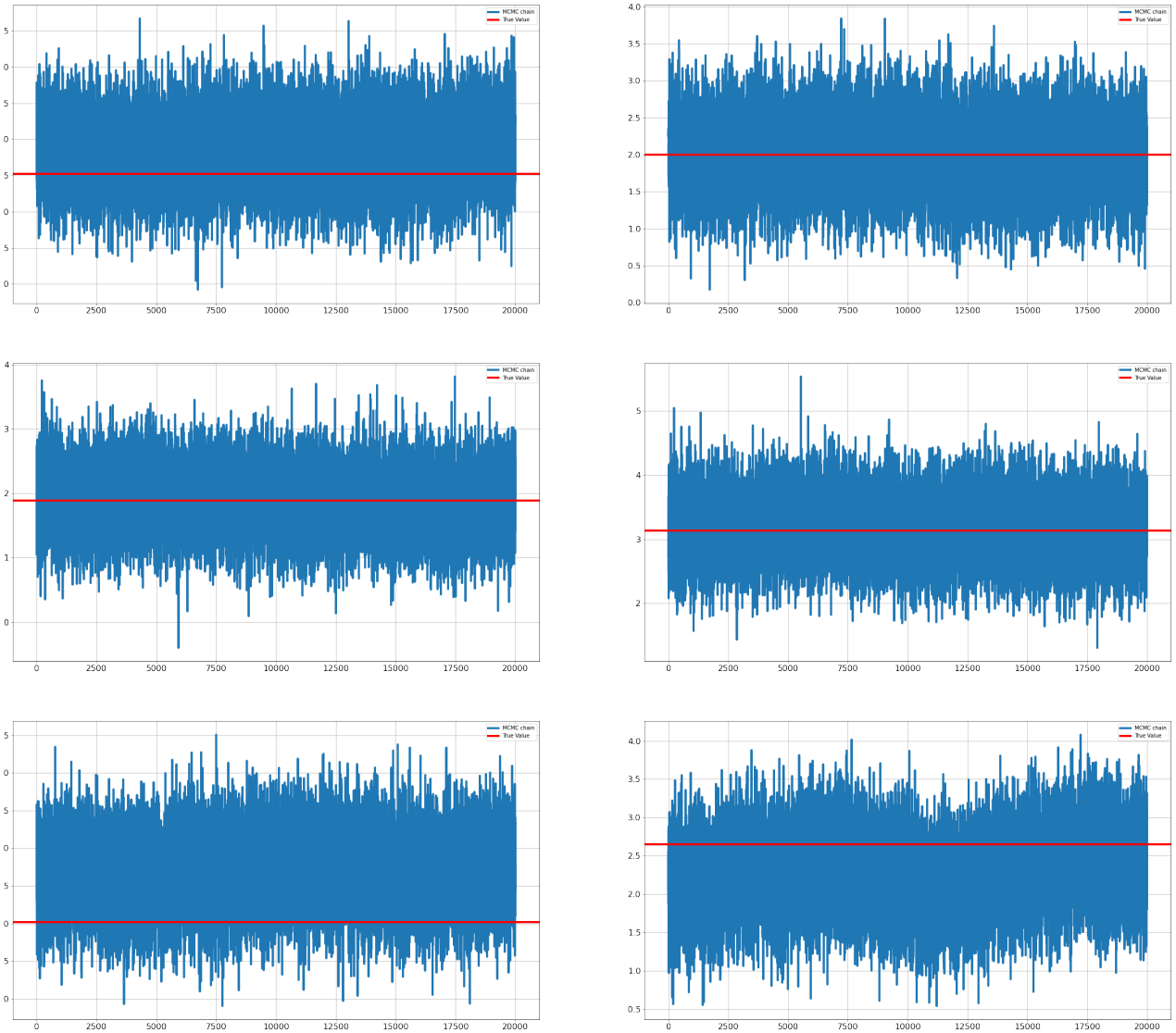

Figure A.1: MCMC evolution for six randomly drawn fitted values $f(\mathbf{x}_i; \boldsymbol{\theta})$ of the training set.

The following hyperparameters were used for MegNet: the number of blocks was 4, the

embedding dimension was 32, the L2 coefficient was 0, the r cutoff was 4 and nfeat_bond was 150. The following hyperparameters were used for SchNet: n_atom_basis was 127, **n**_filters was 32, **n**_interactions was 2, cutoff was 5 and n_gaussians was 25.

Appendix A.3: MCMC Details

All the Bayesian methods in this work were fit using Hamiltonian Monte Carlo (Brooks et al. 2011) method to sample the posterior distribution of all the parameters using the probabilistic `Tensorflow-Probability` library in Python. For the simulation study, the step size was tuned to approximately get an acceptance probability of 0.65, while the number of leapfrog steps was fixed at 50. The number of burn-in samples for every chain was 100,000 and 10,000 samples were collected with a thinning of 4. For the case study of Band Gap prediction, we used the No-UTurn-Sampler (Hoffman et al. 2014) to automatically determine the step size and the integration time. Twenty preliminary runs of 5000 samples each were first used to estimate the mass matrix, then 300,000 samples were discarded as burn-in and 20,000 samples were collected after thinning with 10 samples. Figure A.1 shows the evolution of the Markov chain during the sampling stage.

APPENDIX

B

SUPPLEMENTARY MATERIAL FOR CHAPTER 3

Appendix B.1: Computational details

Horseshoe Gibbs sampling algorithm

Sampling from the horseshoe posterior based from the hierarchical model in (3.2) can be complex, since the Half-Cauchy prior on the local and global scales is not conjugate to the normal model. However we can get a conditionally conjugate sampling scheme by introducing data augmentation variables from an inverse gamma distribution as described in Makalic and Schmidt (2015). Recall that if $U^2|W \sim$ Inverse-Gamma$(1, 1/W)$ and $W \sim$ Inverse-Gamma$(1/2, 1)$ then $U \sim$ Cauchy$_+(0, 1)$. For the sampling of the posterior distribution of the $(\boldsymbol{\delta}, \boldsymbol{\lambda}, \tau^2, \sigma^2 \mid \mathbf{Z})$ in the second stage, we augment the model in (3.2)

$$
\begin{aligned}
Z_j \mid \delta_j, \sigma_n^2 &\sim \text{Normal}(\delta_j, \sigma_n^2) 1 \leq j \leq p \\
\delta_j | \lambda_j, \tau, \sigma^2 &\sim \text{Normal}(0, \lambda_j^2 \tau^2 \sigma_n^2) \\
\sigma^2 &\sim \text{IG}(a, b) \\
\lambda_j \mid \zeta_j &\sim \text{Inverse-Gamma}(1, 1/\zeta_j) \\
\tau \mid \nu &\sim \text{Inverse-Gamma}(1, 1/\nu) \\
\zeta_j &\sim \text{Inverse-Gamma}(1/2, 1) \\
\nu &\sim \text{Inverse-Gamma}(1/2, 1).
\end{aligned}
\tag{B.1}
$$

In the simulation and real data analysis we use $a = b = 0.1$. The above hierarchy allows direct sampling from the full conditional using Gibbs sampling by sampling iteratively the following full conditionals.

$$
\begin{aligned}
\delta_j | \lambda_j, \tau, \sigma_n^2, Z_j &\sim \text{Normal}\left(\frac{\tau^2 \lambda_j^2}{1 + \tau^2 \lambda_j^2} Z_j, \sigma_n^2 \frac{\tau^2 \lambda_j^2}{1 + \tau^2 \lambda_j^2} \right) \\
\sigma^2 | \mathbf{Z}, \boldsymbol{\delta}, \boldsymbol{\lambda}, \tau &\sim \text{Inverse-Gamma}\left(p + a, \frac{\|\mathbf{Z} - \boldsymbol{\delta}\|^2 + \sum_{i=1}^p \delta_j^2 / (\lambda_j^2 \tau^2)}{2} + b \right) \\
\lambda_j | Z_j, \delta_j, \sigma_n^2, \tau &\sim \text{Inverse-Gamma}\left(1, \frac{1}{\zeta_j} + \frac{\delta_j^2}{2\sigma_n^2 \tau^2} \right) \\
\tau | \boldsymbol{\delta}, \boldsymbol{\lambda}, \sigma_n^2 \nu &\sim \text{Inverse-Gamma}\left(\frac{p+1}{2}, \frac{1}{\nu} + \frac{1}{2\sigma_n^2} \sum_{i=1}^p \frac{\delta_j^2}{\lambda_j^2} \right) \\
\zeta_j | \lambda_j &\sim \text{Inverse-Gamma}\left(1, 1 + \frac{1}{\lambda_j^2} \right) \\
\nu | \tau &\sim \text{Inverse-Gamma}\left(1, 1 + \frac{1}{\tau^2} \right).
\end{aligned}
$$

PCP MCMC sampling algorithm

For the PCP model, the prior on $\tilde{\tau}^2$ is not a conjugate prior, hence we use a Metropolis step to update this parameter. The full MCMC sampling cycles through the following consitional

distributions

$$\boldsymbol{\beta}_2 \mid \hat{\boldsymbol{\beta}}_1, \overline{\mathbf{Y}}_2, \sigma^2, \tilde{\tau}^2 \sim \text{Normal}\left(\overline{\mathbf{Y}}_T + \frac{n_2\tilde{\tau}^2}{n_T + n_2\tilde{\tau}^2}(\overline{\mathbf{Y}}_2 - \overline{\mathbf{Y}}_T), \sigma^2\frac{n_2\tilde{\tau}^2}{n_T + n_2\tilde{\tau}^2}\right)$$

$$\sigma^2 \mid \hat{\boldsymbol{\beta}}_1, \overline{\mathbf{Y}}_2, \tilde{\tau}^2 \sim \text{Inverse-Gamma}\left(2p + a, \frac{n_2\|\overline{\mathbf{Y}}_2 - \boldsymbol{\beta}_2\|^2}{2} + \frac{n_1\|\overline{\mathbf{Y}}_1 - \boldsymbol{\beta}_2\|^2}{2(1+\tilde{\tau}^2)} + b\right)$$

$$\pi(\tilde{\tau}^2 \mid \boldsymbol{\beta}_2, \overline{\mathbf{Y}}_1, \sigma^2) \propto g(\tilde{\tau}^2)(1+\tilde{\tau}^2)^{-p/2}\exp\{-\frac{n_1\|\overline{\mathbf{Y}}_1 - \boldsymbol{\beta}_2\|^2}{2\sigma^2(1+\tilde{\tau}^2)}\}.$$

For the last step an adaptive MH step based on the Hessian of the log posterior is straightforward to implement.

Appendix B.2: Proofs

Proof of Lemma 3.3.1.

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\beta}_1^o, \boldsymbol{\delta}^o}[(\overline{\mathscr{Y}}_1 - \boldsymbol{\beta}_1^o)^T(\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}^o)] &= \sum_{j=1}^{p}\mathbb{E}_{\beta_{1j}^o, \delta_j^o}[(\overline{Y}_{1j} - \beta_{1j}^o)(\hat{\delta}_j - \delta_j^o)] \\
&= \sum_{j=1}^{p}\mathbb{E}_{\beta_{1j}^o, \delta_j^o}[\mathbb{E}_{\beta_{1j}^o, \delta_j^o}[\overline{Y}_{1j} - \beta_{1j}^o | Z_j](\hat{\delta}_j - \delta_j^o)] \\
&= \sum_{j=1}^{p}\mathbb{E}_{\delta_j^o}[-\frac{1}{1+n_1}(Z_j - \delta_j^o)(\hat{\delta}_j - \delta_j^o)] \\
&= -\frac{1}{1+n_1}\sum_{j=1}^{p}\mathbb{E}_{\delta_j^o}[(Z_j - \delta_j^o)(w_{\tau,\sigma_n^2}(Z_j)Z_j - \delta_j^o)]. \qquad \text{(B.2)}
\end{aligned}$$

If $\delta_j^o = 0$:

$$\mathbb{E}_{\delta_j^o}[(Z_j - \delta_j^o)(w_{\tau,\sigma_n^2}(Z_j)Z_j - \delta_j^o)] = \mathbb{E}_{\delta_j^o}[Z_j^2 w_{\tau,\sigma_n^2}(Z_j)] > 0.$$

If $\delta_j^o \neq 0$:

$$\mathbb{E}_{\delta_j^o}[(Z_j - \delta_j^o)(w_{\tau,\sigma_n^2}(Z_j)Z_j - \delta_j^o)] = \mathbb{E}_{\delta_j^o}[Z_j w_{\tau,\sigma_n^2}(Z_j)(Z_j - \delta_j^o)].$$

Recall that by Stein's Lemma (Stein 1981) if X is a normally distributed random variable with mean μ and variance σ^2, and if h is a function such that $\mathbb{E}[h(X)(X-\mu)]$ and $\mathbb{E}[h'(X)]$ both exist then

$$\mathbb{E}[h(X)(X-\mu)] = \sigma^2\,\mathbb{E}[h'(X)]. \qquad \text{(B.3)}$$

Let $h(z) = z\, w_{\tau,\sigma_n^2}(z)$. Clearly, $\mathbb{E}_{\delta_j^o}[h(Z_j)(Z_j - \delta_j^o)]$ exists since $0 < w_{\tau,\sigma_n^2}(z) < 1$. Furthermore from (3.5) we have:

$$h'(z) = w_{\tau,\sigma_n^2}(z) + \frac{z^2}{\sigma_n^2} \frac{\int_0^1 u^{\frac{3}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du \int_0^1 u^{\frac{-1}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du - \left(\int_0^1 u^{\frac{1}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du\right)^2}{\left(\int_0^1 u^{-\frac{1}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du\right)^2}. \quad \text{(B.4)}$$

from (B.4), we can see that $|h'(z)| \leq 1 + z^2/\sigma_n^2$, hence $\mathbb{E}_{\delta_j^o}[h'(Z)]$ is finite. By the Cauchy-Schwartz inequality

$$\begin{aligned}\int_0^1 u^{\frac{1}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du &= \int_0^1 u^{\frac{3}{4}} g(u)^{1/2} e^{\frac{z^2}{4\sigma_n^2} u}\, u^{\frac{-1}{4}} g(u)^{1/2} e^{\frac{z^2}{4\sigma_n^2} u}\, du \\ &\leq \left(\int_0^1 u^{\frac{3}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du \int_0^1 u^{\frac{-1}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du\right)^{1/2}\end{aligned}$$

therefore

$$\left(\int_0^1 u^{\frac{1}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du\right)^2 \leq \int_0^1 u^{\frac{3}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du \int_0^1 u^{\frac{-1}{2}} g(u) e^{\frac{z^2}{2\sigma_n^2} u}\, du.$$

Hence the second term in the right hand side of (B.4) is non-negative, thus $h'(z) \geq w_{\tau,\sigma_n^2}(z) > 0$. Using Stein's identity (B.3) we get

$$\mathbb{E}_{\delta_j^o}[(Z_j - \delta_j^o)(w_{\tau,\sigma_n^2}(Z_j) Z_j - \delta_j^o)] = \sigma_n^2\, \mathbb{E}_{\delta_j^o}[h'(Z_j)] > 0.$$

□

Proof of Lemma 3.3.2. Define the integral:

$$I_k(Z) = \int_0^1 u^k g(u) e^{\frac{Z^2}{2\sigma_n^2} u}\, du. \quad \text{(B.5)}$$

From (B.3) and (B.4) we have

$$\mathbb{E}_{\delta_j^o}[(Z_j - \delta_j^o)(w_{\tau,\sigma_n^2}(Z_j) Z_j - \delta_j^o)] = \mathbb{E}_{\delta_j^o}\left[\sigma_n^2\, w_{\tau,\sigma_n^2}(Z_j) + Z_j^2 \left(\frac{I_{\frac{3}{2}}(Z_j)}{I_{-\frac{1}{2}}(Z_j)} - \left(\frac{I_{\frac{1}{2}}(Z_j)}{I_{-\frac{1}{2}}(Z_j)}\right)^2\right)\right] \quad \text{(B.6)}$$

If $\delta_j^o = 0$ then we can bound the expectation in (B.6) using only the last term:

$$\mathbb{E}_{\delta_j^o}[(Z_j - \delta_j^o)(w_{\tau,\sigma_n^2}(Z_j)Z_j - \delta_j^o)] \geq \mathbb{E}_{\delta_j^o}\left[Z_j^2\left(\frac{I_{\frac{3}{2}}(Z_j)}{I_{-\frac{1}{2}}(Z_j)} - \left(\frac{I_{\frac{1}{2}}(Z_j)}{I_{-\frac{1}{2}}(Z_j)}\right)^2\right)\right].$$

Now using Theorem 3.4 in van der Pas et al. (2014) and noticing the same lower bound used for the posterior variance we get:

$$\mathbb{E}_{\delta_j^o}[(Z_j - \delta_j^o)(w_{\tau,\sigma_n^2}(Z_j)Z_j - \delta_j^o)] \approx \sigma_n^2 \tau \sqrt{\log \frac{1}{\tau}}.$$

If $\delta_j^o \neq 0$, we first assume that $|\delta_j^o| \geq \sqrt{2\sigma_n^2 \log(p/q)}$. In this case the non-zero entries are all greater than a minimum threshold necessary for signal recovery (van der Pas et al. 2014). In this case, we have $\mathbb{E}_{\delta_j^o}[(w_{\tau,\sigma_n^2}(Z_j))] = 1 + o(1)$, hence the result. □

Proof of Theorem 3.3.3. Under the assumptions of Lemma 3.3.2

$$\mathbb{E}_{\boldsymbol{\beta}_1^o}[||\overline{\mathscr{Y}}_1 - \boldsymbol{\beta}_1^o||^2] = \sigma^2 \frac{p}{n_1}$$
$$\mathbb{E}_{\boldsymbol{\delta}^o}[||\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}^o||^2] \leq \sigma_n^2 q \log(\frac{p}{q})(1 + o(1))$$

and using (B.2) and Lemma 3.3.2 we have

$$\mathbb{E}_{\boldsymbol{\beta}_1^o, \boldsymbol{\delta}^o}[(\overline{\mathscr{Y}}_1 - \boldsymbol{\beta}_1^o)^T(\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}^o)] \approx -\frac{\sigma^2}{n_1}(q + (p-q)\tau \sqrt{\log \frac{1}{\tau}}.$$

Finally, using (3.15) and combining the three terms above proves the first part of the theorem. For the second part, under Assumption (1) the first term *i.e.* $\sigma^2 \frac{p}{n_1}$ is $o(1)$ and hence the result is clear. □

APPENDIX

C

SUPPLEMENTARY MATERIAL FOR CHAPTER 4

Appendix C.1: Computational details for SGMCMC

Here we give the detailed algorithms of the SG methods with adaptive drifts. The RMSprop (Root Mean Square Propagation) algorithm is an optimization algorithm originally developed for training neural networks models. It adapts the learning rates of each parameter based on the historical gradient information. This can be seen as adaptive preconditioning method.

Algorithm 1: RMSprop Algorithm

Input: Initial parameter values θ_0, learning rate h_0, decay rate ρ, small constant ϵ

Output: Optimized parameter values θ

Initialize square gradient accumulator $r_0 = 0$;

while *not converged* **do**

Sample minibatch without repetition; Compute gradient $\bar{g}$ on mini-batch;

Accumulate squared gradient: $r_t \leftarrow \rho\, r_{t-1} + (1-\rho)\bar{g} \odot \bar{g}$;

Update parameters: $\theta_{t+1} \leftarrow \theta_t - h_t \bar{g} \oslash \sqrt{r_t + \epsilon}$;

Momentum SGD is an optimization algorithm that uses a Neseterov momentum term to accelerate the convergence in the presence of high curvature or noisy gradients. Momentum SGD proceeds as follows

Algorithm 2: Momentum SGD Algorithm

Input: Initial parameter values θ_0, learning rate h_0, momentum term α

Output: Optimized parameter values θ

Initialize velocity $v_0 = 0$;

while *not converged* **do**

Sample minibatch without repetition; Compute gradient $\bar{g}_t$ on mini-batch;

Update velocity: $v_t \leftarrow \alpha v_{t-1} - h_t \bar{g}$;

Update parameters: $\theta_{t+1} \leftarrow \theta_t + v_t$;

The Adam algorithm combines ideas from RMSprop and momentum to adaptively adjust learning rates.

Algorithm 3: Adam Algorithm

Input: Initial parameter values θ_0, learning rate h_0, exponential decay rates for moments α_1, α_2, small constant ϵ

Output: Optimized parameter values θ

Initialize moment estimates $m_0 = 0$, $v_0 = 0$, time step $t = 0$;

while *not converged* **do**

Sample minibatch without repetition; Compute gradient $\bar{g}$ on mini-batch;

Update biased first moment estimate: $m_{t+1} \leftarrow \alpha_1 m_t + (1-\alpha_1)\bar{g}$;

Update biased second raw moment estimate: $v_{t+1} \leftarrow \beta_2 v + (1-\alpha_2)\bar{g} \odot \bar{g}$;

Correct bias in moment estimates: $\hat{m}_t \leftarrow m_t/(1-\alpha_1^t)$, $\hat{v}_t \leftarrow v_t/(1-\alpha_2^t)$;

Update parameters: $\theta_{t+1} \leftarrow \theta_t - \alpha \hat{m}_t \oslash (\sqrt{\hat{v}_t} + \epsilon)$;

Appendix C.2: Additional results for SGFS

Maximum likelihood estimation

As discussed in the simulation study section, a small modification to the SGRLD algorithm yields a stochastic Fisher scoring method for maximum likelihood estimation using the Vecchia approximation. To perform Fisher scoring, we only need to remove the injected noise and drift terms from the updates in (4.13). Let $\boldsymbol{\beta}_t, \boldsymbol{\theta}_t$ and h_t, the current values of the parameters and

the step size respectively. The SGFS scoring updates are

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + h_t p^4_{\mathscr{B}}(\boldsymbol{\theta}_t)^{-1}\left(p^3_{\mathscr{B}}(\boldsymbol{\theta}_t) - p^4_{\mathscr{B}}(\boldsymbol{\theta}_t)\boldsymbol{\beta}_t\right) \tag{C.1}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + h_t \mathscr{I}_{\mathscr{B}}(\boldsymbol{\theta})^{-1}\nabla_{\boldsymbol{\theta}_t}\bar{\ell}_{\mathscr{B}}(\boldsymbol{\beta}_t, \boldsymbol{\theta}_t), \tag{C.2}$$

where $\mathscr{I}_{\mathscr{B}}(\boldsymbol{\theta})$ is given in (4.31) and $\nabla_{\boldsymbol{\theta}_t}\bar{\ell}_{\mathscr{B}}(\boldsymbol{\beta}_t, \boldsymbol{\theta}_t)$ is the vector with elements defined in (4.30).

We compare the SGFS method to the full data Fisher scoring method in Guinness (2019) and the most widely used SGD variants. We limit the setting to $N = 10^4$ locations, the learning rate scheduling and batch size dimensions are kept the same, and the number of epochs is set to 10. To avoid overfitting in the SGD methods we use the stopping rule proposed by Chee and Toulis (2018). This method keeps a running average of the inner product of successive gradients and detects when this quantity changes sign. The results are summarized in Table C.1.

Table C.1: Mean squared error of covariance parameters from 10^4 locations. Mean (Monte Carlo standard errors) over 100 simulations. The proposed SGFS method is compared with other SG methods (RMSPROP, ADAM, MSGD) and the full likelihood GpGp method.

Algorithm	Variance	Range	Smoothness	Nugget	Time (in seconds)
RMSPROP	0.12(0.03)	0.39(0.069)	0.72(0.12)	$0.02(3\cdot10^{-4})$	9.26
ADAM	0.52(0.07)	0.69(0.02)	0.93(0.06)	$0.11(3\cdot10^{-3})$	11.71
MSGD	0.19(0.06)	0.48(0.10)	1.08(0.09)	$0.17(8\cdot10^{-3})$	10.05
GPGP	0.92(0.16)	0.09(0.02)	1.04(0.25)	$0.21(6\cdot10^{-3})$	38.72
SGFS	0.06(0.014)	0.10(0.04)	0.16(0.01)	$0.11(2\cdot10^{-2})$	14.03

The results in Table C.1 show that the SGFS outperforms the other methods in terms of estimation error. Compared to GPGP, the stochastic methods take at most half the time while performing twenty times more iterations.