Transfer learning with Gaussian Processes

Grigorios Skolidis

Doctor of Philosophy
Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh
2012

Abstract

Transfer Learning is an emerging framework for learning from data that aims at intelligently transferring information between tasks. This is achieved by developing algorithms that can perform multiple tasks simultaneously, as well as translating previously acquired knowledge to novel learning problems.

In this thesis, we investigate the application of Gaussian Processes to various forms of transfer learning with a focus on classification problems. This process initiates with a thorough introduction to the framework of Transfer learning, providing a clear taxonomy of the areas of research. Following that, we continue by reviewing the recent advances on Multi-task learning for regression with Gaussian processes, and compare the performance of some of these methods on a real data set. This review gives insights about the strengths and weaknesses of each method, which acts as a point of reference to apply these methods to other forms of transfer learning.

The main contributions of this thesis are reported in the three following chapters. The third chapter investigates the application of Multi-task Gaussian processes to classification problems. We extend a previously proposed model to the classification scenario, providing three inference methods due to the non-Gaussian likelihood the classification paradigm imposes. The forth chapter extends the multi-task scenario to the semi-supervised case. Using labeled and unlabeled data, we construct a novel covariance function that is able to capture the geometry of the distribution of each task. This setup allows unlabeled data to be utilised to infer the level of correlation between the tasks. Moreover, we also discuss the potential use of this model to situations where no labeled data are available for certain tasks. The fifth chapter investigates a novel form of transfer learning called meta-generalising. The question at hand is if, after training on a sufficient number of tasks, it is possible to make predictions on a novel task. In this situation, the predictor is embedded in an environment of multiple tasks but has no information about the origins of the test task. This elevates the concept of generalising from the level of data to the level of tasks. We employ a model based on a hierarchy of Gaussian processes, in a mixtures of expert sense, to make predictions based on the relation between the distributions of the novel and the training tasks. Each chapter is accompanied with a thorough experimental part giving insights about the potentials and the limits of the proposed methods.

Acknowledgements

In the last three and half years of my studies I was fortunate enough to meet and collaborate with several people to whom I owe a great debt and appreciation.

The gratitude I have for Guido Sanguinetti, my supervisor, cannot be described with words. His guidance and peace of mind acted many times as a torch in the dark, offering me unconditional support and encouragement through every step of the Phd journey. His advice and our lengthy conversations have introduced me in a different way of thinking, while his patience during our 'intense' meetings was a lesson itself. This work would not have been possible without his help.

During my studies in the University of Sheffield I had the opportunity to meet and work with many people, of which only a few is possible to acknowledge here. Special thanks go to my MSc thesis advisor Visakan Kadirkamanathan for his supervision during my MSc project and for introducing me to Guido. I would also like to thank Richard Clayton, my second advisor during my PhD study period in Sheffield, for his advice and for introducing me to the problem of Arrhythmia classification. I also had the luck to meet Eleni Vasilaki, a friend whose advice and support are greatly appreciated. Friends matter a lot specially during difficult periods, Maurizio Fillipone has been such an invaluable friend. The countless coffees and drinks we had, 'eased' the Phd tension, while it has given us many funny stories to tell. Shahzad Asif has been my office companion throughout all these three years, I am sure I am going miss him. Our regular breaks outside no matter the weather conditions, have been an important component of the day; exchanging ideas and learning about another's culture so different than mine has been most educative.

Many thanks go to Mark Girolami and Gavin Cawley for the valuable feedback during the regular meetings of the CLIMB project. I would also like to thank all the members of the Machine learning group of the University of Edinburgh who have introduced me to new concepts and improved my understanding during our regular meettings. In particular I would like to thank Chris Williams, Amos Storkey, Botond Cseke, Wolodjia Wentland, and a former member of the group Edwin Bonilla for the useful feedback, suggestions, and help to improve my work.

Just over a year ago I had the luck to meet a wonderful girl called Isabel. Our Skype conferences have been the highlight of my day, while the incidental trips in Spain have been most entertaining and relaxing.

Foremost, I would like to thank my parents, and my brother for never giving up on me, and whose unconditional love and support has helped me in every step of my life.

This work is dedicated to you three.

Last but not least, I would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for the financial support under grand EP/F009461/2.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Grigorios Skolidis*)

To Konstantinos, Athanasia and Christos Skolidi.

Table of Contents

1 Introduction to Transfer Learning **1**

1.1 Taxonomy of Transfer Learning Algorithms 3

1.1.1 Inductive Transfer Learning 5

1.1.2 Transductive Transfer Learning 9

1.1.3 Unsupervised Transfer Learning 12

1.2 Theoretical and Practical issues in Transfer Learning 13

1.2.1 Theoretical analysis 14

1.2.2 Applicability and task relatedness 15

1.2.3 Assumptions in Transfer Learning 17

1.3 Scope and Summary 19

1.4 Thesis Structure 20

1.5 Contributions 21

2 Multi-task Learning with Gaussian Processes for regression **23**

2.1 Non-parametric Bayesian regression 24

2.1.1 Gaussian Process regression 25

2.1.2 Covariance function 26

2.2 Multi-task Gaussian Process Regression 28

2.2.1 Parameter Transfer 30

2.2.2 Inductive Transfer 33

2.2.3 Interlude 41

2.3 Empirical Evaluation of Multi-task learning 43

2.3.1 Predicting pK_a 43

2.3.2 Experimental setup 45

2.3.3 Results 48

2.4 Conclusions 52

3 Multi-task Learning with Gaussian Processes for classification 53
3.1 Multi-task Gaussian Processes . . . 55
3.1.1 Multi-task Regression . . . 55
3.1.2 Multi-task GP Classification . . . 56
3.2 Inference in multi-task GP classification . . . 57
3.2.1 Bayesian Inference: Gibbs Sampling . . . 58
3.2.2 Approximate Inference . . . 63
3.3 Transfer of knowledge & Covariance structure . . . 65
3.3.1 Task covariance matrix . . . 65
3.3.2 Input covariance function . . . 67
3.4 Experimental Results . . . 67
3.4.1 Synthetic data set . . . 69
3.4.2 Arrhythmia data set . . . 72
3.4.3 Landmine data set . . . 74
3.5 Conclusions . . . 76

4 Semi-Supervised Multi-task learning with Gaussian Processes 77
4.1 Introduction . . . 77
4.2 Semi-supervised learning with GPs (SSGP) . . . 79
4.2.1 Frequentist kernel methods for SS learning . . . 80
4.2.2 Bayesian Semi-supervised GP learning . . . 82
4.3 Semi-Supervised Multi-task learning (SS-MTL) with GPs . . . 84
4.3.1 Parameter transfer SS-MTL with GPs . . . 84
4.3.2 Inductive transfer SS-MTL with GPs . . . 86
4.3.3 Graph Laplacian . . . 89
4.3.4 Likelihood . . . 90
4.3.5 Hyperparameter estimation . . . 92
4.4 Experiments . . . 93
4.4.1 Spiral Toy data set . . . 95
4.4.2 Spam Detection . . . 97
4.4.3 Sentiment Analysis . . . 98
4.4.4 Letters Classification . . . 100
4.5 Extensions in Transfer Learning . . . 101
4.6 Conclusions . . . 103

5 Meta-generalisation with Gaussian Processes 105

5.1 Introduction . . . 105
5.2 Meta-generalising . . . 108
5.3 A model for Meta-generalisation . . . 111
5.3.1 Multi-task learning with Gaussian Processes . . . 111
5.3.2 Model . . . 113
5.3.3 Prediction on novel tasks . . . 117
5.4 Experiments . . . 120
5.4.1 Fully observed tasks . . . 122
5.4.2 Partially observed tasks . . . 128
5.5 Conclusions . . . 135

6 Conclusions and Future Research Directions 137

6.1 Conclusions . . . 138
6.2 Future Research Directions . . . 140

A Appendix to Chapter 3 143

A.1 MCMC methods . . . 143
A.1.1 The Metropolis-Hastings algorithm . . . 143
A.1.2 Gibbs Sampling . . . 144
A.2 Gibbs sampling for Multi-task GP Classification . . . 145
A.2.1 Auxiliary Variables . . . 145
A.2.2 Latent function . . . 145
A.2.3 Task covariance . . . 146
A.3 Variational probit model for Multi-task classification . . . 147
A.3.1 Approximate Inference . . . 147
A.3.2 Lower Bound . . . 149
A.3.3 Predictions . . . 152

B Appendix to Chapter 4 154

B.1 Data dependent covariance function . . . 154
B.2 Optimizing the Hyperparameters for the SS-MTL models . . . 155
B.2.1 SS-MTL IND . . . 155
B.2.2 SS-MTL SG . . . 157
B.2.3 SS-MTL FG . . . 158
B.2.4 Additional experimental results . . . 160

B.3 Useful Matrix identities 161

C Appendix to Chapter 5 **163**

C.1 Approximate Inference for CMTMC model 163

C.1.1 $Q(\mathbf{G})$ 163

C.1.2 $Q(\mathbf{H}^t)$ 164

C.2 Lower Bound of the CMTMC model 166

C.2.1 Lower bound on log marginal likelihodd 166

C.2.2 Gradients on lower bound 167

C.3 Predictions with the CMTMC model 168

C.3.1 Point to Point (P2P) predictions 168

C.3.2 Batch predictions 169

D Automatic Classification of Arrhythmic Beats using Gaussian Processes **171**

Bibliography **176**

List of Figures

1.1 Taxonomy of Transfer learning algorithms. 4
1.2 Inductive transfer learning. 5
1.3 Transductive transfer learning. 9
1.4 Unsupervised transfer learning. 12

2.1 Scatter plot of the pK_a data set; red crosses indicate data from the "A" group and blue stars indicate data from the "B" group of tasks, $SEc = SE$ of ionizable center, SE_1 = binned SE of neighbors 1 bond away, and SE_2 = binned SE of neighbors 2 bonds away from center. . 47
2.2 RMSE for the "A" and "B" groups of tasks separately. Figure (a) presents the results for the "A" group, and figure (b) presents the results for the "B" group. 49
2.3 RMSE for all 15 tasks together on the second set of experiments. . . . 50
2.4 Direct comparison between the different constructions of the task covariance matrix and the two sets of experiments; figure (a) presents the results for the "A" group, and figure (b) presents the results for the "B" group of tasks. 51
2.5 Comparison between different standardization procedures, "Assembly" represents results produced by standardizing all task together and "Independent" represents independent standardization. 52

3.1 General Multi-task Probit model. Variables **y** represent the outputs, variables **h** and **f** represent the auxiliary and the latent function variables over the tasks respectively. Variables K^t, α, θ^t and θ^x are used to denote prior distributions of parameters of the task and data covariance function (see section 3.1.2 for more details). 56

3.2 Scatter plot of the two clusters of tasks; figure (a) shows the scatter plot for tasks 1, 2, 3, and figure (b) shows the scatter plot for tasks 4, 5, 6. 69
3.3 Experimental Results on Toy data set. (a) Mean AUC over the 6 tasks, (b) Hinton Diagram of the task covariance matrix, averaged over the 100 independent trials of the results obtained by EP with 50 data points per task. 70
3.4 Decision boundaries with the EP approximation for MTL and STL for 10 training data points; subfigures (a) and (b) show the decision boundaries for task 3 for MTL and STL respectively from the first cluster, and subfigures (c) and (d) show the decision boundaries for task 6 for MTL and STL respectively from the second cluster of tasks. 72
3.5 Experimental Results on the *Arrhythmia* data set. (a) Mean AUC over the 7 tasks, (b) Hinton Diagram of the task covariance matrix, averaged over the 100 independent trials of the results obtained by EP with 50 data points per task 73
3.6 Additional Results on the *Arrhythmia* data set. (a) AUC for task 5 (Recording ID: 221), (c) AUC for task 6 (Recording ID: 223) 74
3.7 Experimental Results on the *Landmine* data set. (a) Mean AUC over the 19 tasks, (b) Hinton Diagram of the task covariance matrix, averaged over the 100 independent trials of the results obtained by EP with 100 data points per task 75

4.1 Sources of information in Semi-supervised multi-task learning (SS-MTL); $\mathbf{X}_j^l$ and $\mathbf{X}_j^u$ denotes the labelled and the unlabelled set of task j, and $\mathbf{Y}_j$ represents the responses for data in $\mathbf{X}_j^l$. 83
4.2 Scatter plot of the Spiral Toy data set: *left* complete data distribution, *center* first task, *right* second task. 94
4.3 Results on the Spiral toy data set. *1st row* : MTL; *2nd row* : SS-STL; *3rd row* : SS-MTL IND only; *4th row* : SS-MTL FG results; Notice that SS-MTL FG is capable of effectively transferring knowledge from the tasks and the unlabelled data. Left column shows results for Task 1, and right column shows results for Task 2. 96
4.4 Performance on spiral toy problem in terms of AUC. 97

5.1 Coupled Multi-Task Multi-Class (CMTMC) model. Variables $\mathbf{f}$ and $\mathbf{g}$ are the two sets of GPs for the multi-task and multi-class classifiers respectively, whereas variables $\mathbf{h}^x$ and $\mathbf{h}^t$ denote the auxiliary variables of the two classifiers; (a) graphical representation of the training phase, (b) graphical representation of Meta-generalising. 113

5.2 Toy data set I distribution; (a) scatter plot and density for the first cluster of tasks (1-3), (b) scatter plot and density for the second cluster of tasks (4-6). 122

5.3 Toy data set I classification Results; (a) Average AUC over the 6 tasks, (b) Hinton Diagram of the task covariance matrix of the CMTMC model computed by averaging over the 25 repetitions with 50 data points per task. 123

5.4 Toy data set II distribution; (a) scatter plot and density for the first cluster of tasks(1-2), (b) scatter plot and density for the second cluster of tasks (3-4). 124

5.5 Toy data set II classification Results; (a) Average AUC over the 4 tasks, (b) Hinton Diagram of the task covariance matrix. 125

5.6 Character Classification Results; (a) Average AUC over the 8 tasks, (b) Hinton Diagram of the task covariance matrix. 126

5.7 Arrhythmia Classification Results; (a) Average AUC over the 7 tasks, (b) Hinton Diagram of the task covariance matrix. 126

5.8 Average AUC on the unseen tasks of Toy data set *I*; (a) training on 2 tasks generalising on 4, (b) training on 4 tasks generalising on 2. . . . 129

5.9 AUC on the 17 unseen tasks of Landmine data set; training on 2 tasks, generalising on 17; (a) AUC over all 17 tasks, (b) AUC over 9 tasks of the first cluster, (c) AUC over 8 tasks of the second cluster. 130

5.10 Average AUC on the 15 unseen tasks of Landmine data set; training on 4 tasks, generalising on 15; (a) Overall AUC over all 15 tasks, (b) Average AUC over 8 tasks of the first cluster, (c) Average AUC over 7 tasks of the second cluster. 131

5.11 Average AUC on the 11 unseen tasks of Landmine data set; training on 8 tasks, generalising on 11; (a) Overall AUC over all 11 tasks, (b) Average AUC over 6 tasks of the first cluster, (c) Average AUC over 5 tasks of the second cluster. 132

5.12 Average AUC on the unseen tasks of Arrhythmia data set on different number of training tasks; (a) training on 2 tasks, generalising on 5, (b) training on 3 tasks, generalising on 4, (c) training on 4 tasks, generalising on 3, (d) training on 5 tasks, generalising on 2 133

5.13 Average AUC on the unseen tasks for the Character classification data set; (a) Training on 4 tasks, (b) Training on 5 tasks, (c) Training on 6 tasks. 135

List of Tables

2.1 Description of the *pKa* dataset; pK_a range is given as min-max values, IHB and NIHB indicates that the group is *capable* and *not capable* of forming internal hydrogen bonds respectively. 45

3.1 Description of the Arrhythmia data set 73
3.2 T-test for the Landmine data set. The second column (No. of tasks) shows the number of tasks with better means, for the method stated in the first column. The third column (Statistically Significant) shows the number of tasks that were statistically significant on the 0.05 significance level (p-value). 75

4.1 Prior distribution and parameterisation for the SS-MTL models. . . . 92
4.2 Gaussian Process Learning frameworks 94
4.3 *AUC of the Spam data set.* Double lines separate the learning frameworks, SS-MTL, MTL, and STL with SS-STL. 99
4.4 *AUC of the Amazon data set.* Double lines separate the learning frameworks, SS-MTL, MTL, and STL with SS-STL 99
4.5 Description of the Letters data set; each column is a task showing the two letters as well as the corresponding number of examples per character (class) 101
4.6 *AUC on the Letters data set.* Double lines separate the learning frameworks, SS-MTL, MTL, and STL with SS-STL 102

6.1 Transfer Learning and Gaussian Process Research. The first column of symbols next to each subsetting characterizes research in the literature, and the second symbol depicts research in this thesis. In the first column, symbol $\square$ stands for settings for which models have been proposed and properly analyzed, symbol $\diamondsuit$ indicates that a model has been proposed but has not been tested, and symbol $\times$ signifies a gap in the literature. In the second column, symbol $\surd$ stands for models proposed and thoroughly examined, symbol $\triangle$ indicates that a model has been proposed but has not been tested, and symbol $?$ points out future work to be proposed 138

B.1 Prior distributions for SS-MTL models. 156
B.2 *AUC and Accuracy on the Spam data set.* 160
B.3 *AUC and Accuracy on the Amazon data set.* 161
B.4 *AUC and Accuracy on the Letters data set.* 161

Chapter 1

Introduction to Transfer Learning

Machine learning algorithms are typically separated into three main categories, *supervised*, *unsupervised* and *reinforcement* learning. This classification depends mostly on the availability of responses y for input data x. Supervised learning deals with problems where for each input x an output y is observed, while in the two latter categories this correspondence does not exist. Unsupervised learning tries to find interesting patterns in the data without explicit supervision, whereas in reinforcement learning the learning algorithm seeks for the optimal sequence of actions that maximizes a reward (Bishop, 2006).

An important aspect of all machine learning methods are the *assumptions* the model is based on. This can be either due to their mathematical convenience or due to expert knowledge. Typically, data are assumed to be *independent* and *identically distributed* (*i.i.d*), or a notion of *smoothness* is assumed about the data generating mechanism (Barber, 2011). Most importantly, it is commonly assumed that the distribution that generated the training and the test data is the same. While this assumption allows to *generalise* on unseen data, there are many situations where it is not valid. This might happen either because the test data comes from a more general population or several conditions have changed since the training data were acquired. The situations where the training and test data distributions are different are usually referred to as *sample selection bias* (Zadrozny, 2004) or *covariate shift* (Shimodaira, 2000).

Until recently, given data for a certain problem learning was mostly performed in isolation of other similar problems. As an example, consider the problem of arrhythmia classification especially in the case where several patients are involved. The learning problem here is to have an automated system, a classifier, that is able to predict the state, normal or arrhythmic, the heart of each patient is in. Arrhythmia is a condition under which the parts of the heart are stimulated in a different order from that of the normal operation of the heart, the normal sinus rhythm. Identifying arrhythmia is performed by experts by ‘reading’ the electrocardiogram, or the ECG signal, that measures the electrical activity of the heart and reflects which parts of the heart are stimulated and in which order. Depending on the condition of the heart, normal or arrhythmic, the order of stimulation of the different parts of the heart of each patient is the same and is reflected in the ECG signal. Thus, the normal and arrhythmic heart beats of different subjects, although not exactly the same because of high variations in the ECG signal, tend to be related. A possible solution to this problem would be to train a classifier for each patient by using annotated data for that specific patient only. On the other hand training each classifier in isolation could potentially miss information and result into a poor model of limited *generalisation* ability if for example not all possible arrhythmia for that specific patient have been identified by the expert, whereas training all patients together by using information from all patients (tasks) could overcome this limitation. In this direction, since the pioneering work of Caruana (1997), *multi-task learning* has emerged as a setting to handle situations where multiple tasks are involved, and training of all tasks is performed in parallel.

Another important aspect is that there are many other real world applications where annotated data are scarce or completely missing, while annotated data from similar problems are plentiful. Returning to the arrhythmia classification example, consider the situation where for some patients annotated data are available while we wish to make predictions on a new patient for which training data are absent. The natural question that rises is how to exploit or combine information from similar tasks to make predictions for the task that limited or no annotated data exist. This type of problems has triggered a lot of research specially in the field of natural language processing (NLP), where annotating text resources can be very expensive. A solution to this type of problems would be to resort to *semi-supervised learning* (Chapelle et al., 2006), that in addition to labeled data it exploits information stemming from unlabeled data to improve the performance of the model. As the semi-supervised approach would not take into account the differences between the distributions of the tasks, another approach to solving this type of problems is known as *Domain Adaptation* (DA) (Daumé III and Marcu, 2006).

Over the last ten to fifteen years there has been a lot controversy over the terminology for this type of settings and only recently a survey paper by Pan and Yang (2010) presented a detailed and justified classification of these problems[1]. Their perspective to these problems is adopted and is reviewed later on with some modifications reflecting our angle of view. The name that was given to this new framework is *Transfer Learning* (TL) and covers all scenarios where the learning problem is embedded in an environment of multiple related tasks. It should be noted that the term "*transfer learning*" has previously been used to refer to scenarios where there the interest lies in making predictions in a task where no labels are provided by exploiting other related annotated tasks (Do and Ng, 2006; Raina et al., 2006, 2007; Yu and Chu, 2008). Other names that have been given are *learning to learn* (Baxter, 1997; Thrun and Pratt, 1998), *inductive-bias learning* (Baxter, 2000), and *meta-learning* (Vilalta and Drissi, 2002) and the interested reader is referred to those for a comprehensive review.

In the following sections of this chapter, we provide an overview of transfer learning algorithms by classifying them into three main categories. We continue by discussing some theoretical and practical issues concerning transfer learning, and finally in the last section of this chapter we give an overview of the subsequent chapters, as well as the contributions of this thesis.

1.1 Taxonomy of Transfer Learning Algorithms

Classifying transfer learning algorithms can vary depending on the criteria used. To proceed further, we now sketch an overall picture of the framework of transfer learning along with some definitions that will be found useful to identify the criteria that will be employed to categorize it. The input data $X = \{x_1, \ldots, x_n\}$ from each learning problem have a feature space $\mathcal{X}$, $x_i \in \mathcal{X}$, and a generating or marginal probability distribution $P(X)$, and the output data $Y = \{y_1, \ldots, y_n\}$ have a label space $\mathcal{Y}$. Given the training set $D = \{X, Y\}$ machine learning algorithms are typically intended for finding a predictive function f, or $p(Y|f,X)$[2] from the probabilistic point of view[3], that maps the inputs to the output space, $\mathcal{X} \to \mathcal{Y}$. Following Pan and Yang (2010), we define the *domain* $\mathcal{D}$ that consists of the feature space and the marginal distribution, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, and the task $\mathcal{T}$ that consists of the label space and the predictive function, $\mathcal{T} = \{\mathcal{Y}, f\}$. For explanatory reasons the description and categorization of transfer learning will

[1]This reflects strictly the author's opinion.

[2]In Pan and Yang (2010) this is written as $p(Y|X)$, it is our opinion that writing $p(Y|f,X)$ to denote the likelihood is better suited.

[3]The predictive function f and conditional distribution of the outputs given the inputs $p(Y|f,X)$ are used interchangeably throughout this chapter.

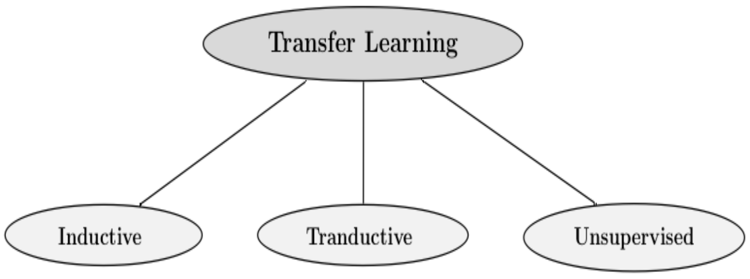


Figure 1.1: Taxonomy of Transfer learning algorithms.

focus on one learning problem that we wish to make predictions, which we will call the target task, and one problem that we wish to exploit, which we will refer to as the source task; note however that the same criteria hold for multiple problems, source or target. To carry on, we now give a formal definition of "*Transfer Learning*" (Pan and Yang, 2010) which will act as the basis for its classification.

Definition 1 *(*Transfer Learning*) Given a source domain $\mathcal{D}_s$ and a learning task $\mathcal{T}_s$, a target domain $\mathcal{D}_t$ and a learning task $\mathcal{T}_t$, transfer learning aims to help improve the learning of the target predictive distribution $f_t(.)$ in $\mathcal{D}_t$ using the knowledge in $\mathcal{D}_s$ and $\mathcal{T}_s$, where $\mathcal{D}_s \neq \mathcal{D}_t$ or $\mathcal{T}_s \neq \mathcal{T}_t$.*

The definition of transfer learning, and the availability of labels Y and input data X during the training of the model are the two basic criteria that we will use to classify transfer learning into three main categories, *inductive*, *transductive*, and *unsupervised* transfer learning.

Notably, this classification is similar to the one employed by Arnold et al. (2007). The difference between them is that in Arnold et al. (2007) transfer learning is classified only into inductive and transductive, while unsupervised transfer learning is mentioned as an alternative name for transductive transfer learning. Transfer Learning in reinforcement learning has also received a great amount of attention but since it falls out of the scope of this work we refer to Taylor and Stone (2009) for a survey on that subject.

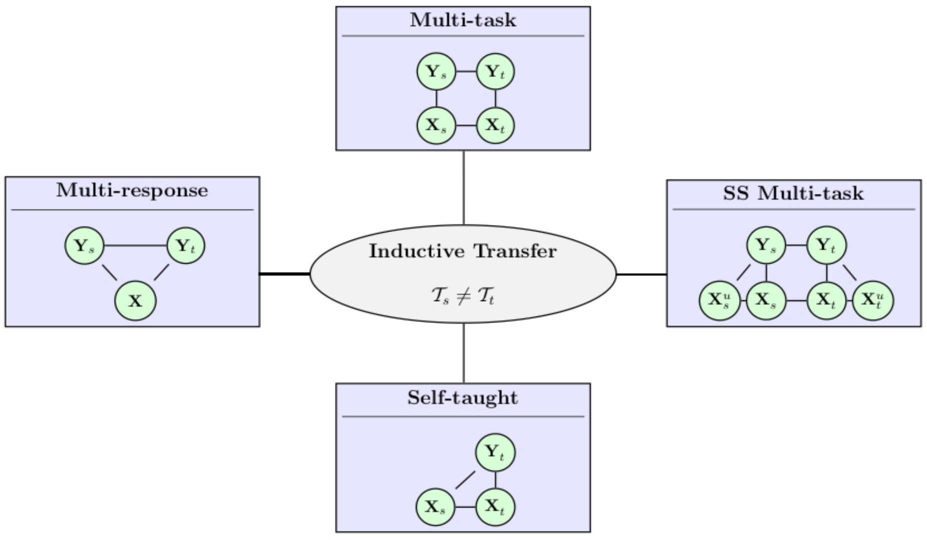


Figure 1.2: Inductive transfer learning.

1.1.1 Inductive Transfer Learning

In *inductive transfer learning*, regardless of the nature of the target and source domain, the target task $\mathcal{T}_t$ is different from the source task $\mathcal{T}_s$, $\mathcal{T}_s \neq \mathcal{T}_t$. Based on the definition of the domain $\mathcal{D}$ and the task $\mathcal{T}$, in inductive transfer learning the tasks can have either different predictive functions, $f_s \neq f_t$, or different label spaces, $\mathcal{Y}_s \neq \mathcal{Y}_t$. However, note that as far as we know the requirement that $\mathcal{T}_s \neq \mathcal{T}_t$, is *not employed* in the modeling process[4], in contrast to Covariate Shift and Sample Selection bias form the transductive category of transfer learning that makes explicit use of these specifications in the training of the model. Antithetically, if the predictive functions and the label space were exactly the same then any form of 'smart' combination of data from different tasks would not be needed, since pooling data from all problems together would be sufficient. Most importantly, a basic requirement for a problem to fall into this category is the presence of annotated data Y_t during training to '*induce*' the predictive function f_t in the target domain. Figure 1.2 shows the individual subsettings of Inductive transfer learning along with the sources of information involved in each one; X_s and X_t are the input data of the source and the target task respectively, Y_s and Y_t are the output data of the source and the target task and X_s^u and X_t^u are the unlabeled 'auxiliary' data of the tasks.

[4]By modeling process is meant the development of the algorithm.

1.1.1.1 Multi-task Learning

The main representative of this category is *multi-task learning* (MTL), and especially the case where the focus is to improve the performance of one specific learning problem. Despite the fact that the majority of the developed multi-task learning algorithms aim at improving the performance of all learning problems simultaneously, unarguably most of them can be employed to improve the performance of the target task only. This issue is specifically addressed in Xue et al. (2007), where multi-task learning is separated into two cases *symmetric* and *asymmetric*. In the first case the goal is to learn jointly multiple tasks, whereas in the latter the goal is to transfer previously acquired knowledge from some source tasks to improve the performance of the target task. From this further separation of multi-task learning it is concluded that the asymmetric case is more in accordance with the definition of transfer learning.

As in the more general case of transfer learning, transferring information between learning problems by training them together has been found to be most appropriate and advantageous in situations where limited annotated data are available for each task. Due to that reason multi-task learning has triggered a great amount of research over the last years, and has been approached by several learning frameworks such as *neural networks* (NN) (Caruana, 1997; Baxter, 1997; Bakker and Heskes, 2003), *deep neural networks* (Collobert and Weston, 2008), *conditional random fields* (Sutton and McCallum, 2005), *support vector machines* (SVMs) (Jebara, 2004), *regularization networks* (Evgeniou et al., 2006), *Gaussian Processes* (GPs) (Schwaighofer et al., 2005; Bonilla et al., 2008), *logistic regression* with *Dirichlet Processes* (DPs) (Xue et al., 2007) and many other.

One of the simplest forms of multi-task learning, which in contrast to the majority of the existing methods does not rely on the development of new algorithms but is based on the idea of *feature augmentation*, is the work of Daumé (2007) which is named EasyAdapt[5] (EA). The ease of implementation of EA is that it can simply act as a preprocessing step and in turn be combined with several standard supervised classifiers such as maximum entropy (MaxEnt) (Nigam et al., 1999), SVMs (Schölkopf et al., 1999), or logistic regression (Bishop, 2006). In essence, if F is the total number of features of both $\mathcal{X}_s$ and $\mathcal{X}_t$[6] ($x \in \mathbb{R}^F$), then EA acts as a mapping where $\Phi(x) \in \mathbb{R}^{F(m+1)}$, where m is the number of learning problems. In the augmented space, the first block of F features is shared between all tasks and is always active, while the other blocks

[5]Note that Pan and Yang (2010) has included EA in the transductive category of TL.

[6]We assume that the tasks have the same feature space.

are task specific and are activated only when the features are extracted by that specific task. For one source and target task the augmented space will be given by,

$$\Phi_s(\mathbf{x}) = <\mathbf{x}, \mathbf{x}, \mathbf{0}>, \qquad \Phi_t(\mathbf{x}) = <\mathbf{x}, \mathbf{0}, \mathbf{x}> \tag{1.1}$$

where $\mathbf{0}$ is the zero vector of length F. It is worth emphasizing the fact that although EA was named "*Frustratingly Easy Domain Adaptation*", it is our opinion that it fits better to the inductive category of TL since it requires labeled data in the target task to work, an opinion that is also shared by Arnold et al. (2007) and Chang et al. (2010).

The term domain adaptation, which was imported from the NLP community, is separated into supervised, semi-supervised, and unsupervised (Daumé III and Marcu, 2006; Daumé et al., 2010); *supervised DA* requires labeled data in the target task in the spirit of asymmetric multi-task learning and will be classified into the inductive setting of TL; *semi-supervised DA* makes use of labeled and unlabeled data from both source and target task and is also classified in the inductive setting as a form of semi-supervised MTL; finally, in *unsupervised DA* which falls into the transductive TL category, no target task labeled examples are available during training.

Last but not least, the interesting scenario where the learning problems have a different label space, $\mathcal{Y}_s \neq \mathcal{Y}_t$, and more specifically different number of classes, has not been left unattended. Parameswaran and Weinberger (2010) extends the large margin nearest neighbor classifiers of Weinberger and Saul (2009) to the multi-task case, where tasks can have different classes; similarly to the SVM formulation the optimization problem is convex providing convergence guarantees, but the use of the *k*NN rule inherently allows it to extent to differing number of classes between the tasks.

1.1.1.2 Multi-response Learning

A setting that is closely related to multi-task learning is that of *multi-response learning* (MRL), which is also known as multi-output learning. In multi-response learning there are still several tasks to be learned but all share the same input space, which means that each input corresponds to several outputs. This is illustrated in figure 1.2 where the source and the target task share the same set of inputs X, in contrast to MTL where each task has a separate set of inputs. Thus, multi-response learning can be considered as a specific instance of inductive transfer learning where the source and target problems have exactly the same domain $\mathcal{D}_s = \mathcal{D}_t$, and the tasks are different $\mathcal{T}_s \neq \mathcal{T}_t$. This situation is most frequent in classification problems where classes are not considered to be mutually exclusive, which is termed as *multi-label classification*. For example, this scenario has been observed in document classification where a document might correspond to several topics or in medical diagnosis where patients are diagnosed with several diseases (Tsoumakas and Katakis, 2007). Multiple response methods in regression have also found useful applications, for example in chemometrics (Breiman and Friedman, 1997), or in modeling Robot arm dynamics (Teh et al., 2005; Chai et al., 2009).

1.1.1.3 Semi-supervised Multi-task Learning

The merits of training multiple tasks together have directed researchers to extend the multi-task setting to exploit unlabeled data, to give the *semi-supervised multi-task learning* framework (SS-MTL). Training multiple tasks together while exploiting information from unlabeled data has been reported to outperform simple multi-task learning (Liu et al., 2009; Zhang and Yeung, 2009); Liu et al. (2009) develop a SS-MTL algorithm for classification problems by employing a variant of the Dirichlet process with an EM algorithm, whereas the work of Zhang and Yeung (2009) in the context of Gaussian Processes applies it in regression problems. Although SS-MTL has not received the attention of MTL, it has successfully been applied in several fields. Examples of that is the work of Ando and Zhang (2005) on NLP problems employing a structural learning algorithm, the work of (Quattoni et al., 2008) on image classification using the (l_1, l_∞) regularization, or in bioinformatics by Qi et al. (2010) where they train a multi-layer perceptron network for predicting protein to protein interactions. The idea of feature augmentation EA (Daumé, 2007), has also been modified in Daumé et al. (2010) to account for unlabeled data giving the semi-supervised analog of EA, as EA++. EA++ being applied in the task of sentiment classification presented superior results than that of EA,verifying the fact that the inclusion of more sources of information results in improved performance.

1.1.1.4 Self-taught Learning

In contrast to multi-task learning where labeled data are available for all tasks, source and target, in *self-taught learning* (SeTL) (Raina et al., 2007) labeled data are available only for the target problem. Self-taught learning, another setting that falls into the inductive category intended mainly for classification problems, uses the unlabeled data from another source task to improve the predictions in the target task for which limited labeled data are available. A condition that needs to be satisfied for a learning problem

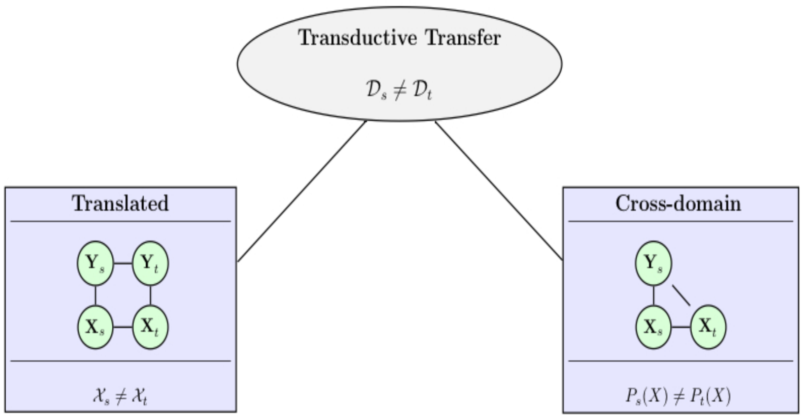


Figure 1.3: Transductive transfer learning.

to be approached by a SeTL algorithm, and simultaneously differentiates it from semi-supervised learning, is that data from the source task cannot be assigned to any of the target tasks classes.

1.1.2 Transductive Transfer Learning

In *transductive transfer learning* the source and target tasks are the same $\mathcal{T}_s = \mathcal{T}_t$, while the domains are different, $\mathcal{D}_s \neq \mathcal{D}_t$. This sub-setting can further be divided to situations where the feature space between the source and target task are different, $\mathcal{X}_s \neq \mathcal{X}_t$, or to situations where the data generating distributions are different, $p_s(X) \neq p_t(X)$.

Originally, the term transductive transfer learning was introduced by Arnold et al. (2007) to refer to scenarios where abundant labeled data for the source task are available, while no labels are provided for the target task. The learning problem in this type of scenarios is how to transform or adapt the predictor learned on the source task to make predictions on the target task. A possible solution to this problem would be to include the target task unlabeled data during the training of the model on the source task labeled data in a transductive sense, whilst accounting for the difference in the distributions. This type of scenarios will be called *cross-domain transfer*, and will refer to situations where labeled data are absent for the target task and $p_s(X) \neq p_t(X)$. Traditionally, the term transductive (Vapnik, 1998; Joachims, 1999) was used to refer to situations where all test data were used during the learning phase constituting the model as test data specific, and in case new data arrived the model had to be retrained. In the transfer learning framework this constraint is relaxed and the term transductive is valid for situations where only a part of the target task data set is used during the training of the model. Differently from transductive learning, in semi-supervised learning the unlabeled data that are used during training, which are usually referred to as *auxiliary* input data, are not considered as part of the test set. The situations where the feature space between the source and the target task are different will be referred to as *translated learning* and will require labeled data from both the source and the target task. Figure 1.3 illustrates the subsettings of Transductive transfer learning along with their specifications and the sources of information that are involved during the training of the model.

1.1.2.1 Translated Learning

Combining tasks with different feature spaces, $\mathcal{X}_s \neq \mathcal{X}_t$, is in general a very difficult problem as a common feature space provides a common metric system to discover similarities between data objects. As a result, learning in transductive TL with different feature spaces requires annotated data in both the source and the target task. The term that will be used to describe this situations is *translated learning* (Dai et al., 2009).

Considering the predicament of this learning problem, the model that was developed in Dai et al. (2009), which was based on the language model of Lafferty and Zhai (2001), was successfully applied in image classification by translating information obtained from text data, and to cross-language classification by using English documents as source data to classify German documents. Another study that addresses this problem is the work by (Arnold, 2009, Ch. 3) who applies it to the problem of name entity recognition (NER).[7] Other related settings that combine datasets with different feature spaces are *co-training* (Blum and Mitchell, 1998), or *multi-view learning* (Minton and Knoblock, 2002), but since they are intended for problems where the same learning problem has more than one feature representations we decide not to include them within the transfer learning framework.

1.1.2.2 Cross-domain transfer

Learning settings that specifically address the issue of different distributions between the source and the target task, $p_s(X) \neq p_t(X)$, with no access to labeled data from the target task but with the same feature space $\mathcal{X}_s = \mathcal{X}_t$, are *sample selection bias*

[7]Note that (Chai, 2010, Ch. 2) also outlines a method to combine tasks of different feature spaces in the context of multi-task Gaussian Processes.

(Zadrozny, 2004; Huang et al., 2007), *covariate shift* (Shimodaira, 2000), and *unsupervised domain adaptation* (DA) (Arnold et al., 2007). Sample selection bias and covariate shift are closely related settings, that handle situations where the training and test data distributions are not the same or the observed (training) data are not a good representative of the whole population. Conceptually, their difference with domain adaptation is that the training and the test data distributions are not the same because of the biased nature of the sampling process, whereas in domain adaptation it is not the same because it comes from a different domain. For a more in depth discussion about the differences of covariate shift and sample selection bias we refer to Storkey and Sugiyama (2007), and to Quiñonero-Candela et al. (2009) about the general problem of "*Dataset shift*".

More formally, given a training/source set X_s originating from $p_s(X)$ for which labels Y_s are observed, and a test/target set X_t from $p_t(X)$ for which labels are not observed where $p_s(X) \neq p_t(X)$ and $\mathcal{X}_s = \mathcal{X}_t$, the goal of *cross-domain transfer* is to construct a predictor that performs well on the target set by utilizing information from the source task.

Unsupervised DA has mostly been applied in NLP problems, some examples of this include the work of: Arnold et al. (2007) who develop a variant of MaxEnt to apply in the problem of NER, Blitzer et al. (2006) that makes use of the idea of structure learning of Ando and Zhang (2005) to perform the PoS[8] task, and Huang and Yates (2009) that improves upon the results of Blitzer et al. (2006) by applying a smoothing technique on top of HMMs. Another approach to unsupervised DA, which tries to match the source and target distributions by finding a shared latent space is a method called *Transfer Component Analysis* by Pan et al. (2009), which compared to other baselines produced superior performance on a text classification problem and on the problem of estimating the location of a mobile device based on the signal values from access points. Finally, concerning the classification of TL it is worth noting that the description of transductive transfer learning and DA presented here is in agreement with the one given in Ben-David et al. (2007), where it is said: "*Unlike in inductive transfer, where the tasks we wish to perform may be related but different, in DA we perform the same task in multiple domains*".

[8]The acronym PoS is used to denote the task of *Part of Speech tagging* that aims at tagging the syntactic role of each word (Collobert and Weston, 2008).

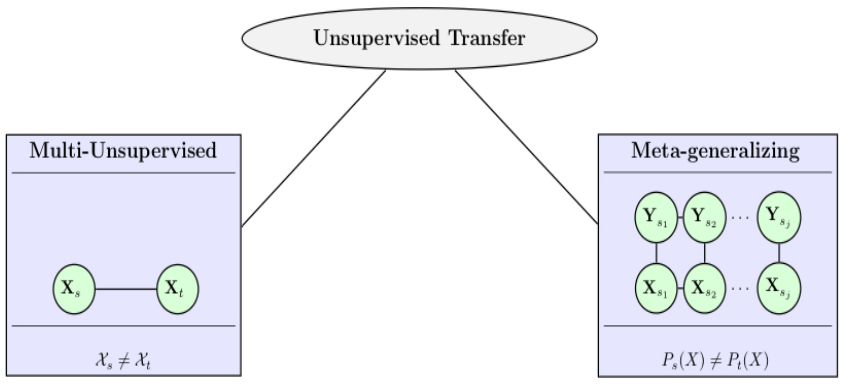


Figure 1.4: Unsupervised transfer learning.

1.1.3 Unsupervised Transfer Learning

Relatively to the first two categories, *unsupervised transfer learning* has not received so much attention. Our perspective of unsupervised transfer learning covers a wider spectrum of problems than the one given in Pan and Yang (2010). In Pan and Yang (2010), the transfer learning analog of unsupervised learning deals with problems like dimensionality reduction, clustering, or density estimation but in scenarios where multiple tasks are involved. This is the exact analog of unsupervised learning in the TL case, as it is intended for problems where there are no annotated data involved either in the source or the target task during training. This type of settings will be referred to as *multi-unsupervised learning* (MUL).

In all of the settings that were discussed so far, the input data of the target task X_T were accessible during training, and one of the major differences between them was the availability of labels either in the source or the target task. A very interesting scenario that is worth investigating, is the case where the learner has no access to any data from the target domain $\mathcal{D}_t$ during training. This feature adds up to the criteria that can be used to classify TL, since in all of the previous cases the observability of the target input data X_t was assured and it was giving exploitable information about the target task. Therefore, this scenario is unsupervised in the transfer learning sense because there is no supervision at all on the side of the target task. The scenario that has just been described is inspired by Baxter (2000), who said "... *a bias learner generalises well if, after seeing sufficiently many training tasks it produces a hypothesis space that with probability contains good solutions to novel tasks*". Additionally, he also commented that a form of *meta-generalisation* is possible, where instead of generalising in the space of data points coming from the same task, the generalisation is performed in the space of tasks. The situation where no training data from the target task are available during training, will be referred to as *Meta-generalisation*.

Therefore, in this work the category of *unsupervised transfer learning* is intended for two types of scenarios (figure 1.4):

1. In *Multi-Unsupervised learning*, where the learning problem is unsupervised like clustering, dimensionality reduction, or density estimation, but when multiple tasks are involved (Pan and Yang, 2010).

2. In *Meta-Generalisation*, where the target input data X_t are *not* accessible during training and the learning problem can be regression or classification.

Some examples of unsupervised transfer learning based on the view of Pan and Yang (2010), is the work of Dai et al. (2008) who proposed a novel method for clustering when data for the target task are scarce, or the model in Wang et al. (2008) that exploits labeled data from a similar source task to perform dimensionality reduction in the target task.

1.2 Theoretical and Practical issues in Transfer Learning

It is clear that the framework of transfer learning raises many issues involving its limits, its applicability, and the assumptions that it is based on. First, it is worth spending a moment to investigate the reasons that have made transfer learning so attractive, triggering so much research around it.

From our angle of view, there are two *motivating reasons* for transfer learning. The first is from the philosophical side of machine learning and the second is from the practical point of view, inspired by real-world scenarios. Although we have to be cautious with any direct parallelism and explanation of how human learning takes place, a widely accepted belief is that humans intelligently apply previously acquired knowledge to tackle problems that they encounter for the first time, acting as a memory based learning and decision making machine. Thus, transfer learning is a step closer to human learning and decision making than conventional machine learning techniques, since its primary goals are to combine multiple sources of information, and to transfer previously acquired knowledge to future tasks. From the practical point of view, as the world becomes more quantitative day by day large amount of data from different tasks are being stored, while labeling requires human effort making annotations for these data expensive and tedious to obtain. Therefore, it would be desireable to have methods that are able to deal with this inadequacy of labeled data, as well as translating a previously encountered problem for which labels are provided to new specifications and learning tasks.

1.2.1 Theoretical analysis

Theoretical work that directly addresses the *limit* of success of transfer learning is concerned with the derivation of error bounds on the generalisation performance for this type of algorithms. This analysis fills the gap for theoretical justification of the empirical benefits that transfer learning has been reported to deliver.

In inductive transfer learning, Baxter (2000) was the first one to provide generalisation error bounds for multi-task learning based on a generalised version of the VC-dimension (Blumer et al., 1989; Vapnik, 1999). Baxter's version of VC-dimension has been used by Ben-David and Borbely (2008), which extends the main results of Ben-David et al. (2002), to provide guarantees that the generalisation error of single-task learning will be larger than multi-task learning in cases of small training sample sizes. Other theoretical analysis that is based on Baxter's results is the work of Ando and Zhang (2005), that derive bounds that the hypothesis space can be estimated more reliably for increasing numbers of tasks. Chai (2009) provides bounds on the generalisation error for the asymmetric case of the multi-task GP model of Bonilla et al. (2008), whose analysis is similar to the one given for the multi-task model with the generalised t-process of Zhang and Yeung (2010). On the feature augmentation framework, Daumé et al. (2010) present theoretical analysis on the generalisation bounds for EA and EA++, showing that EA++ should perform better than EA. EA has previously been theoretically analyzed in the work of Chang et al. (2010) who pointed out the need for combining EA with unlabeled data.

Theoretical analysis for the problem of Domain Adaptation has also received considerable attention. Ben-David et al. (2007) derive bounds on the generalisation performance of the target task in the context of unsupervised domain adaptation. In the DA paradigm, since the goal is to use labeled data from a source task to make predictions in the target task the bound depends on two quantities; one is the generalisation error of the source task and the second is the divergence between the source and target distributions. Other theoretical work on unsupervised DA is the paper by Mansour et al. (2009a), that builds on the work of Ben-David et al. (2007) and define a novel distance measure between distributions, or the work by Blitzer et al. (2008) and Mansour et al. (2009b) that focus on situations of multiple source tasks. Theoretical analysis in other settings of transductive TL is the work by Cortes et al. (2010) in sample selection bias whose analysis is based on a generalised version of the point-based stability of Bousquet and Elisseeff (2002).

1.2.2 Applicability and task relatedness

Concerning the *applicability* of transfer learning, in the introduction of this chapter we gave an example illustrating the necessity for this type of algorithms. This simple example represents only a fraction of the areas it can be applied, since it has been found useful in many areas that involve multiple tasks as in sentiment classification (Blitzer et al., 2007), spam e-mail classification (Bickel and Scheffer, 2007), name entity recognition (NER) (Arnold, 2009) or in other natural language processing problems (Daumé III and Marcu, 2006), bioinformatics (Qi et al., 2010), cancer classification (Zhang et al., 2010), compiler performance prediction (Bonilla et al., 2007), image classification (Raina et al., 2007), HIV therapy screening (Bickel et al., 2008), recommendation systems (Dinuzzo et al., 2008), and many others. However, the numerous examples of the success of transfer learning do not imply that it can applied be in every situation that involves multiple tasks.

Undoubtedly, the applicability and most importantly the success of transfer learning is intimately connected with the notion of *task relatedness*. The concept behind this is that, if two tasks are not related then any transfer of information between them would at the very least not make any difference than training in isolation, while in the worst scenario would even produce an adverse effect in the tasks performance, known as *negative transfer*. Due to that reason several authors have specifically addressed the issue of task relatedness, where for example in Silver and Mercer (2001) task relatedness is defined as the utility of using the samples of the source and target task to develop an effective hypothesis for the target task. Although the definition that was given was rather abstract they also provided and empirically compared several computable measures of task relatedness that could be used in practice. Ben-David and Schuller (2003) take a more theoretical approach to that subject by defining task relatedness in terms of the data generating mechanism, where they say that two task are $\mathcal{F}$-related if the input data of these tasks are generated by applying a function $f \in \mathcal{F}$ on a fixed probability distribution. Another definition that has also been proposed is that of Chai (2010) who says that "*two tasks are related to each other when they benefit mutually under metalearning*", also providing a thorough discussion about the different approaches and concepts about the notion of task relatedness that demonstrates the diversity on that subject.

Clearly, none of the interpretations of task relatedness is right or wrong but it depends on the way you look at the problem. The approach that will be followed here is more from the practical point of view and is based on the available information that can be exploited and on quantities that can be computed. Ideally, it would be desirable to have algorithms that are able to automatically determine which tasks are related and only then transfer knowledge from one task to the other. Without doubt, the process of automatically determining which tasks are related is highly dependent on the available information and more specifically on the level of supervision on the target task.

From the three categories of TL the only one that provides the required level of information to infer which tasks are related without compelling us to make any explicit assumptions is the inductive setting, that is fully supervised in the TL sense. Specifically, in the multi-task and multi-response settings the observability of the inputs and outputs in both the source and target tasks allows to model their predictive functions f_s and f_t, as well as the prior probability distribution of the inputs $p_s(X)$ and $p_t(X)$ independently of the information in the other task. So, a solution to the determination of whether a source and a target task are related, would be to test if the predictive functions f_s and f_t give similar solutions for the same set of points. Therefore, given source and target task training data $X_s = \{x_i^s\}_{i=1}^{n_s}$ and $X_t = \{x_i^t\}_{i=1}^{n_t}$ respectively, with the same feature spaces $\mathcal{X}_s = \mathcal{X}_t$, and denoting the union of these sets as $X = X_s \cup X_t$ with $N = n_s + n_t$, task relatedness can be computed as,

$$\lambda = \sum_{i=1}^{N} |f_t(x_i) - f_s(x_i)|, \tag{1.2}$$

where $x_i \in X$, and λ is the parameter that quantifies the relatedness between those two tasks. Note that the sum over the differences of the two predictive functions in equation 5.1, bears some resemblance with the measure used to compute the error of a classifier that discriminates samples from different distributions in Ben-David et al. (2007). This simple example shows that in the case of multi-task/response learning the available information allows to determine which tasks are related before applying any algorithm that combines them.

In the other two categories of transfer learning the available information is insufficient to perform this test. In the case of transductive and unsupervised TL where the only exploitable information from the side of the target task is the input distribution $p_t(X)$, it has to be assumed (*e.g.* through expert knowledge) that the source tasks contain beneficial information about the learning of the target task.

1.2.3 Assumptions in Transfer Learning

An important aspect of transfer learning are the *assumptions* that it is based on. In contrast with the standard machine learning problems, the TL algorithms require stronger assumptions to be satisfied in order for them to be successful. The reason is that the nature of the problem is more complicated since it involves different sources of information to be combined which in turn imposes more constraints. For example the lack of labels on the target task in the transductive category of cross-domain transfer compels us to assume that the predictive functions between the source and the target task will be same. By assuming that, the predictive function acts as an information bridge between the two tasks allowing to make predictions based on the labels of the source task.

Clearly, depending on the setting and the specifications of the problem these assumptions change. However, in order to allow the different tasks to interact with each other so that they can mutually benefit from the information contained in the other tasks, it is common practice to allow them to share a certain structure. Sharing a structure can take the form of an *implicit assumption* when constructing the model, for example sharing the hidden units of a NN (Caruana, 1997), a common hierarchical prior (Yu et al., 2005) or hyperparameters of a GP (Lawrence and Platt, 2004), similar sparsity patterns (Argyriou et al., 2008; Obozinski et al., 2009) and others. Finally, it is worth saying that if this type of assumptions are enforced in inductive TL then they take a weaker form, because the labels on the target task act as a safeguard to avoid negative transfer if they are not met. On the other hand if they are employed in transductive and unsupervised TL then they are considered stronger since it is usually hard in practice to check their consistency.

An example of an *explicit assumption* is that of the *covariate shift* which presumes that the joint probability distribution of the inputs and outputs of the source task $p_s(Y,X)$ differs from the joint of the target task $p_t(Y,X)$ only in the prior of the inputs

(Shimodaira, 2000; Huang et al., 2007). This means that the conditional probabilities $Y|X$ of the target and the source tasks are the same $p_s(Y|X) = p_t(Y|X) = p(Y|X)$:

$$p_s(Y,X) = p(Y|X)p_s(X)$$
$$p_t(Y,X) = p(Y|X)p_t(X).$$

The importance of the covariate shift assumption, and more specifically that $p_s(X) \neq p_t(X)$, is highlighted by the fact that it acts as one of the basic features of transductive TL, differentiating it from the inductive category. The lack of labels in the target task is the main reason for imposing this assumption, as without it training in the target task would be unsupervised since there is no information about the distribution of the target labels Y. In a more recent work, Ben-David et al. (2010) provide an analysis on the assumptions needed for domain adaptation to be successfull. The tools used for this analysis are based on the PAC framework (Valiant, 1984). Loosely speaking these assumptions are, i) the well studied *covariate shift*, ii) the *similarity of the unlabeled distributions*, which can be a distance measure between the two distributions originally introduced in Kifer et al. (2004), and iii) the *existence of a classifier with low prediction error in both the source and target task.* The last assumption, which was first defined in Ben-David et al. (2007), can be interpreted as the agreement between the labels of the source and target task with the simultaneous existence of a low error predictor in both tasks. In conclusion, it is obvious that the assumptions needed for the transductive and unsupervised TL to succeed are a lot stronger and failure to meet them can have an unpredictable effect on the performance of the target task.

1.3 Scope and Summary

This chapter presented an overview of the framework of Transfer Learning, covering its most important aspects. A precise and thorough classification was presented which extends and partially differs from the survey paper of Pan and Yang (2010). This taxonomy was dictated mostly by the notions of the *domain*, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, and the *task* $\mathcal{T} = \{\mathcal{Y}, f\}$, resulting into three major categories of TL. The first category is the Inductive setting, where $\mathcal{T}_s \neq \mathcal{T}_t$ and includes the subsettings of multi-task, multi-response, semi-supervised multi-task and self-taught learning. The second category is the Transductive setting, where $\mathcal{D}_s \neq \mathcal{D}_t$, consisting of the subsettings of translated learning and cross-domain transfer. The last category is the Unsupervised setting, which does not rely on the notions of the domain and the task. In contrast this category is intended for learning in multiple unsupervised tasks, the exact analog unsupervised learning in the TL case, and the novel setting of meta-generalising, a situation where the learner has no exploitable information during training about the target task. Finally, theoretical issues involving the generalisation performance of TL, the assumptions TL is based on, and the issue of task relatedness have also been discussed.

The framework of Transfer Learning poses many new challenges to the machine learning community. Among them is the development of intelligent ways of multi-source data integration, the estimation of optimal cross-task latent representations, and last but not least the estimation of the level and type of correlation between tasks. In strong connection to these are the situations where there are heterogeneous sources of information with different marginals distributions, but also the most commonly encountered problem of predicting in an undersampled task by using informations extracted from another yet related task.

In this thesis we will take a Bayesian approach to Transfer Learning and examine the application of Gaussian processes, a non-parametric Bayesian method, to three different forms of transfer learning: Multi-task learning, Semi-supervised Multi-task learning and Meta-generalisation. Bayesian methods allow the encoding of expert knowledge in the form of prior probability distributions, as well as to explicitly include assumptions into the model. Moreover, in most of the cases the estimated posterior distributions over model parameters and hyperparameters have an interpretable character, allowing the identification of the model. These characteristics can be beneficial in settings where there are not enough data to adequately model a complex system as in the case of Transfer learning which involves the combination of multiple sources of information and the inclusion of assumptions in the modeling process.

1.4 Thesis Structure

The rest of this thesis in relation to the taxonomy introduced in section 1.1 is organized as follows. Chapters 2 and 3 are concerned with the problem of Multi-task learning from the Inductive category. The subject of Chapter 4 is Semi-supervised Multi-task learning again from the Inductive category, although a model is proposed that can be applied to Self-taught and unsupervised Domain adaptation problems from the Transductive category. Finally, Chapter 5 is concerned with the Unsupervised category of Transfer Learning investigating the scenario of Meta-generalising.

In Chapter 2, we provide a brief introduction to Gaussian Process prediction, and we review the existing approaches to multi-task GP regression. These approaches are classified into two main categories based on the level of transfer of information between the tasks. This classification sheds light on the reasons that have made us choose this model for solving classification problems. We close this chapter by empirically comparing some of the reviewed methods on a real data set.

In Chapter 3, we adapt the multi-task GP model of Bonilla et al. (2008) to the classification scenario. Dealing with non Gaussian likelihoods employed in classification problems, an asymptotically exact inference scheme and two deterministic approximations are developed. Finally, the deterministic methods are evaluated on one toy and two real world applications by providing comparisons with other methods that illustrate their effectiveness.

In Chapter 4, we propose two alternative formulations for Semi-supervised Multi-task learning with Gaussian processes. The proposed model makes use of the model of Bonilla et al. (2008) and the model introduced in Sindhwani et al. (2007) to learn from labeled and unlabeled data for GP classification. One of the key features of the second formulation is that unlabeled data contribute to the learning of the task similarities, a formulation which is more a effective in situations where the task similarities are smaller or there are outlier tasks. Thorough experimental evaluation on two text and one character classification problem gives more insight about the methods.

In Chapter 5, we formally introduce the setting of meta-generalisation, that is making predictions on totally unseen tasks by utilizing the predictors learned from several but related tasks, and we develop a model based on GPs able to tackle this kind of situations. The proposed model tackles the meta-generalisation problem by coupling two GPs, one that learns the task responsibilities through a multi-class classifier, and one that learns the individual prediction tasks as well as the task similarities through a multi-task classifier. Thorough experimentation on several data sets provides more insight to the ambitious problem of meta-generalisation. Finally, in Chapter 6 we make some concluding remarks and we discuss possible avenues that this work could be furthered.

Chapter 2, 3, 4, 5 are all written in a self-contained manner, such that they can be read independently of the other. For that reason, except from Chapter 2, all other chapters give a brief introduction to Gaussian processes and of the multi-task model of Bonilla et al. (2008). Although Chapter 2 presents GPs in a greater extent than the other chapters, for a formal introduction to Gaussian Processes for machine learning we refer to the textbook of Rasmussen and Williams (2005). It should also be noted that throughout the thesis all experimental results from different methods were compared using the *t-test*, when this was found appropriate and necessary.

1.5 Contributions

The contributions of this thesis are reflected in the following papers:

- Skolidis, G., Clayton, R. H., and Sanguinetti, G. (2008). Automatic classication of arrhytmic beats using Gaussian processes. In *IEEE Transactions on Computers in Cardiology*, 921-924, Bologna, Italy, 2008.
  This is the outcome of the work in the early stages of this thesis, which applies Gaussian Processes to the Arrhythmia classification problem and acts as the starting point for this thesis. For reasons of completeness we attach this paper in appendix D

- Skolidis, G., Sanguinetti, G. (2010). Bayesian Multi-task Classication with Gaussian Process Priors, *IEEE Transactions on Neural Networks*, 22(12):2011-2021, Dec. 2011.
  The work of this paper is presented in chapter 3.

- Skolidis, G., Sanguinetti, G. (2011). A case study on Meta-Generalising: a Gaussian Processes approach, *The Journal of Machine Learning Research (JMLR)*, **In Press**.
  This work is presented in Chapter 5.

Papers Under Review:

- Grigorios Skolidis, Katja Hansen, Guido Sanguinetti, Matthias Rupp. Multi-task learning for pK_a prediction, *Journal of Computer-Aided Molecular Design*, **Major revisions**.
  Part of this work is presented in the experimental section of Chapter 2.

- Skolidis, G., Sanguinetti, G. (2011). Semi-Supervised Multi-task learning with Gaussian Processes, submitted to IEEE Transactions on Neural Networks and Learning Systems.
  Chapter 4 is an extended version of this paper.

Chapter 2

Multi-task Learning with Gaussian Processes for regression

Gaussian Processes (GPs) is a Bayesian framework for spatial interpolation; work on GPs dates back in the 1970s in the field of statistics (Matheron, 1973; O'Hagan and Kingman, 1978). Since the work of Williams and Rasmussen (1996) GPs have become extremely popular in the machine learning community and have received a considerable amount of attention. It could be argued that one of the main reasons for the interest in GPs from the machine learning community stems from the work of Neal (1996) who has shown that a GP prior over functions is equivalent to a *neural network* in the limit of infinite hidden nodes. Following that, GPs have successfully been applied to classification problems (Williams and Barber, 1998; Rasmussen and Williams, 2005) or to different types of learning, as semi-supervised learning (Lawrence and Jordan, 2005; Sindhwani et al., 2007; Adams and Ghahramani, 2009), to unsupervised learning tasks such as dimensionality reduction (Lawrence, 2004), density estimation (Adams et al., 2009), or clustering (Kim and Lee, 2007), as well as in reinforcement learning (Rasmussen and Kuss, 2004; Engel et al., 2005). More recently, researchers have started investigating the application of GPs in the framework of transfer learning (Yu and Chu, 2008) and particularly on multi-task learning (Menzefricke, 2000; Schwaighofer et al., 2005; Bonilla et al., 2008).

The aim of this chapter is to provide an overview of the existing multi-task (MT) methods for GP regression. This process initiates with an introduction to Gaussian Processes for regression covering its most important aspects. This introduction intends to provide the reader the necessary tools for understanding the various forms of multi-task learning within the GP framework[1]. At core of this chapter is section 2.2 that systematically reviews and classifies the different approaches to Multi-task GP regression. Based on the level of transfer of information between the tasks this classification results into two categories, the Parameter transfer, and the Inductive transfer. The last section is devoted to the empirical comparison of the different forms of multi-task learning on a real world application, that illustrates the need for these algorithms and the benefits they offer.

2.1 Non-parametric Bayesian regression

In the standard supervised learning problem we have a data set D that consists of N input and output pairs, x_i and y_i respectively, where $x_i \in \mathbb{R}^d$ is a column vector, $X = [x_1, \ldots, x_N] \in \mathbb{R}^{d \times N}$ is the construction matrix, $y_i \in \mathbb{R}$, and $\mathbf{y} = [y_1, \ldots, y_N]$. If the observed outputs y are continuous we have a regression problem, whereas if they are discrete we have a classification problem. This chapter focuses on *regression* problems.

The objective in the regression learning problem is to estimate a function $f(x)$, that maps the inputs to the outputs from the data D, and given that function to make predictions at new (test) points x_*. This situation will be termed as *Single task Learning* (STL), to differentiate it from the scenario of learning multiple tasks in parallel. Additionally, it would be desirable to be able to include prior knowledge in the model as well as to obtain an estimate of the confidence in these predictions. Bayesian statistics have a natural way of quantifying uncertainty at all stages, from uncertainty in prior assumptions to uncertainty in measurements. Moreover, it is possible to use these "prior" distributions to express certain assumptions about the data generating process, and the form of the functions it is expected to observe.

The standard assumption in the regression formalism is that the observed outputs y_i are noisy observations of a function f,

$$y_i = f(x_i) + \varepsilon_i, \tag{2.1}$$

where ε_i is the noise term, which is usually assumed to be a zero mean uncorrelated Gaussian distributed variable with a constant variance, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)\ \forall\ i$, a setting known as *homoscedastic* regression[2] (Silverman, 1985).

[1]The content of section 2.1 can be found with more details in Rasmussen and Williams (2005); it is presented here for completeness.

[2]Situations with input dependent variance noise, termed as *heteroscedastic* regression, require a different treatment than the one presented here and we refer to Goldberg et al. (1998) and Lázaro-Gredilla and Titsias (2011) for approaches based on GPs to this type of problem

2.1.1 Gaussian Process regression

A *non-parametric* approach to the regression problem (equation 2.1) can be pursued by inducing a probability distribution over these functions directly. Gaussian processes are the natural way of defining probability distributions over functions. *A Gaussian process is a collection of random variables, such that any finite number of which have the multivariate Gaussian distribution* (Rasmussen and Williams, 2005).

A GP is completely specified by its first and second order statistics, the mean $m(x)$ and the covariance function $k(x,x')$ (Rasmussen and Williams, 2005),

$$m(x) = \mathbb{E}[f(x)], \qquad k(x,x') = \mathbb{E}[(f(x)-m(x))\left((f(x')-m(x')\right)]. \tag{2.2}$$

To denote that the random variable f is distributed according to a GP we will write,

$$f(x) \sim \mathcal{GP}(m(x),k(x,x')). \tag{2.3}$$

It is assumed that the prior of the latent functions f is given by a GP with zero mean and covariance matrix $\mathbf{K}_{N\times N}$, computed by evaluating any valid covariance function[3] between all N points. The *likelihood* or the *noise model* will be given by, $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f},\sigma^2 I_N)$, where I_N is the $N\times N$ identity matrix. Integrating over the latent functions the *evidence* or the *marginal likelihood* will be given by,

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})d\mathbf{f} = \mathcal{N}(0,\mathbf{K}+\sigma^2 I_N). \tag{2.4}$$

Generalisation to new inputs x_* involves the computation of the predictive density of the latent at the test point given the training set, $p(f_*|\mathbf{y},X,x_*)$. The latent functions at the test point f_* and the training set $\mathbf{f}$ are jointly Gaussian $[\mathbf{f},f_*] \sim \mathcal{N}(0,\mathbf{K}_*)$, with

$$\mathbf{K}_* = \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^T & k_{**} \end{bmatrix}, \tag{2.5}$$

where $\mathbf{k}_* = k(X,x_*)$ is the covariance vector between the training samples X and the test point x_*, and $k_{**} = k(x_*,x_*)$ is the marginal variance at the test point. The *noise free* predictive distribution is given by $p(f_*|\mathbf{y}) = \mathcal{N}(\mu_*,\Sigma_*)$ with,

$$\mu_* = \mathbf{k}_*^T(\mathbf{K}+\sigma^2 I)^{-1}\mathbf{y} \tag{2.6}$$

$$\Sigma_* = \mathbf{k}_{**} - \mathbf{k}_*^T(\mathbf{K}+\sigma^2 I)^{-1}\mathbf{k}_*. \tag{2.7}$$

The noisy version of the predictive distribution $p(f_*|\mathbf{y}) = \mathcal{N}(\mu_*,\Sigma_*)$, which we will write as $p(y_*|\mathbf{y})$ is simply given by adding σ^2 to the variance Σ_*. The predictive mean

[3]More information about covariance functions is given in the following section.

can also be written as

$$\mu_* = \sum_{i=1}^{N} a_i k(x_i, x_*), \tag{2.8}$$

where a_i is the i^{th} component of $\mathbf{a} = (\mathbf{K} + \sigma^2 I)^{-1}\mathbf{y}$, which allows to view GPs as a linear predictor of the outputs $\mathbf{y}$, or a linear combination of N kernel functions, one centered at each training data point.

Equations 2.6 and 2.7 show that making predictions (and optimizing the parameters of the covariance function as shown in the following section) involves the inversion of a $N \times N$ matrix. This matrix operation has complexity $O(N^3)$, and for very large data sets may be infeasible. Due to that reason many approximation schemes have been proposed that reduce the complexity at least to $O(L^2N)$, where L is a user defined parameter and $L \ll N$ (see *e.g.*, Smola and Bartlett (2001); Williams and Seeger (2001); Csató and Opper (2002); Lawrence et al. (2003); Snelson and Ghahramani (2006)). These approximations remove the main limitation of Gaussian processes making them applicable to large data sets; for a review on these methods see Quinonero-Candela et al. (2007). Last notice that the complexity remains the same regardless of the dimensions d of the inputs, making GPs an ideal framework to work with high dimensional data x.

2.1.2 Covariance function

GPs assume that the function $\mathbf{f}$ is distributed according to a multivariate Gaussian distribution whose dependencies are determined by the covariance matrix $\mathbf{K}$. Consequently, at core of the GP prediction is the covariance matrix that captures the information about the correlation of the random variables at the different inputs x. A covariance matrix $\mathbf{K}$ is constructed by evaluating a parametric family of covariance functions $k(x,x')$ between all training data points. A covariance function is valid if it results in a symmetric positive semi-definite matrix $\mathbf{K}$ (Rasmussen and Williams, 2005), $\mathbf{v}^T\mathbf{K}\mathbf{v} \geq 0,\ \forall \mathbf{v}$. Employing a parametric family of covariance functions allows us to infer its parameters from the data. In contrast with the parameters in a linear regression model (Bishop, 2006), these parameters control the distribution of the model and will be called *hyperparameters*, and denoted by θ.

Direct inference of the hyperparameter is not possible, but optimal values can be found in several ways. In a fully Bayesian framework, it is possible to set prior probability distributions over these parameters and infer the posterior distribution by sampling methods (*e.g.*, Gelman et al. (2004)). Another option is again to set a prior over these parameters and optimize the parameters of the prior by maximizing the log of the posterior probability distribution. This is usually referred to as the *Maximum a Posteriori* (MAP) approximation, and the concept is to approximate the posterior of the hyperparameters, $p(\theta|\mathbf{y},X) \propto p(\mathbf{y}|\theta,X)p(\theta)$, with a distribution centered at its mode where $p(\theta|\mathbf{y},X)$ is maximal,

$$\theta_{MAP} = \arg\max_{\theta} p(\theta|\mathbf{y},X) = \arg\max_{\theta} p(\mathbf{y}|\theta,X)p(\theta) \tag{2.9}$$

The simplest way to learn these parameters is make point estimates of them, by maximizing the log of equation 2.9,

$$\log p(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathbf{y}^T(\mathbf{K}+\sigma^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}+\sigma^2 I| - \frac{N}{2}\log 2\pi + \log p(\theta), \tag{2.10}$$

where the gradients over the hyperparameters will be given by:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|X,\theta) = \frac{1}{2}\mathbf{y}^T\left(\mathbf{K}+\sigma^2 I\right)^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j}\left(\mathbf{K}+\sigma^2 I\right)^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(\left(\mathbf{K}+\sigma^2 I\right)^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j}) \\ + \frac{\partial}{\partial\theta_j}\log p(\theta).\end{aligned} \tag{2.11}$$

Note that the standard practice is to remove the dependence on the prior of the hyperparameters and maximize the log of the marginal likelihood given in equation 2.4. Deriving point estimates of the hyperparameters by maximization of the marginal likelihood is usually referred to as type *II Maximum Likelihood* (ML).

The choice of the covariance function varies depending on the application and the nature of the features. Covariance functions can be separated into *stationary* and *non-stationary*. Stationary covariance functions are invariant to translations of the input space and are of the form $k(|x-x'|)$, whereas non-stationary are not (Rasmussen and Williams, 2005). A widely used stationary covariance function is the *Squared exponential* (SE), $k(x_i,x_j) = \theta_0^2\exp\left\{-\sum_{k=1}^{d}\frac{(x_{ik}-x_{jk})^2}{2\theta_1^2}\right\}$, where the parameters θ_0 and θ_1 are the amplitude and the characteristic length scale respectively. A variant of this is the *Automatic Relevance Determination* (ARD) (Neal, 1996) covariance function that can be constructed from the SE by setting a different scale parameter for each input dimension. As a result of this parameterization, the ARD covariance function can be employed for the identification of the relevancy or the removal of features proved to be irrelevant for the problem at hand. Non-stationary covariance function are often of the dot product form, *i.e* $k(x\cdot x')$. An important example is the Linear kernel, $k(x_i,x_j) = \theta_0^2 + \theta_1^2\sum_{k=1}^{d}x_{ik}x_{jk}$, with θ_0 and θ_1 parameters. However, it should be noted that covariance functions can be constructed by adding, multiplying or by convolving other valid covariance functions. For a systematic review of covariance functions and their properties we refer to (Rasmussen and Williams, 2005, Ch. 4).

2.2 Multi-task Gaussian Process Regression

In a multi-task scenario we are interested in learning M functions $\mathbf{f}_j$, from data $X_j = [x_{1j}, \ldots, x_{n_j j}]$, $\mathbf{X} = [X_1, \ldots, X_M]$, and targets $\mathbf{y}_j = [y_{1j}, \ldots, y_{n_j j}]$ with $j = 1, \ldots, M$ and $n_1 + \ldots + n_M = N$. Similarly to STL, we assume the following noise model

$$y_{ij} = f_j(x_{ij}) + \varepsilon_j, \text{ with } \varepsilon_j \sim \mathcal{N}(0, \sigma_j^2), \tag{2.12}$$

where y_{ij} (x_{ij}) denotes the i^{th} output (input) of the j^{th} task, and ε_j is task dependent noise.

The concept behind *multi-task learning* is that the training of all M models is performed simultaneously in order to intelligently transfer information across tasks. This is usually achieved by allowing the different tasks to share a certain structure or parameters. It is expected that in situations of *under-sampled tasks*[4] the inferred common structures or parameters between tasks will be more meaningful than parameters inferred from each task individually as in STL. Intuitively, if annotated data for each task were plentiful then MT learning would not be needed since each task would have sufficient information to infer the model parameters independently. Another important issue is the relatedness of the tasks which many of the approaches take for granted and in practice might not be true. We argue that, since this transfer learning scenario provides adequate information for the identification of the relatedness of the tasks (see section 1.2.2), *the developed methods should accommodate appropriate mechanisms of estimating the cross-task correlations automatically*.

Moreover, multi-task learning can be seen as a multi-response problem (see section 1.1.1.2 and figure 1.2), under which one observes several outputs for the same set of inputs. The difference between these two learning frameworks is that in multi-task learning each task has a different set of inputs, while in multi-response all the tasks share the same set of inputs. In the multi-response (also called multi-output) formulation the output space is a vector valued function, $\mathcal{Y} = \mathbb{R}^M$. Prediction on vector valued functions within the GP framework has long been studied in the geostatistics literature and is known as *co-kriging* (see *e.g.*, Cressie (1993); Wackernagel (2003)). From another point of view, in multi-response learning we are interested in the distribution over

[4]The term *under-sampled task* is used to refer to a task for which limited annotated training data are available.

the matrix variate distribution of the latent functions $\mathbf{F} = [\mathbf{f}_1 \dots \mathbf{f}_M],\ \mathbf{F} \in \mathbb{R}^{N\times M}$ where $\mathbf{f} = \mathrm{vec}(\mathbf{F})$, and subsequently the matrix of the outputs $\mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_M],\ \mathbf{Y} \in \mathbb{R}^{N\times M}$ where $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$[5]. Considering the matrix variate Normal distribution it is possible to derive a well studied model by the geostatistics community known as the "*Intrinsic Model of Coregionalization* (IMC), or in its more general form as the "*Linear Model of Coregionalization* (LMC) (Cressie, 1993; Wackernagel, 2003). The LMC assumes that the different tasks are linear combinations of independent random functions, while the correlations between the tasks are modeled though a separate positive definite matrix. Other approaches to multi-response learning include the use of *Convolution Processes* (CP) to construct multi-output covariance functions (Ver Hoef and Barry, 1998; Higdon, 2002). Under this setting each output is assumed to be a convolution integral between a smoothing kernel and latent function, which is taken to be a GP. In both of these models, the LMC and the CP based approach, the unavailability of outputs for certain tasks can easily be handled in the GP framework by simply removing the likelihood terms of the corresponding unobserved outputs. This characteristic makes the IMC and the CP method applicable to multi-task scenarios.

Other approaches to multi-task learning include the linear mixing of one global and one task dependent latent function, or assigning the same or common prior distributions over the parameters of the different tasks. Although the linear mixing of a global and a task dependent function can be seen as a special case of the LMC model, it is treated independently because of the way the global latent function is estimated and its possible extensions (see section 2.2.2.3). A common feature of the methods with common parameters is that the functions of the different tasks are independent given the parameters.

On this basis, the existing MT approaches to GP regression will be separated into two main categories. The first category will be called *Parameter transfer*, to denote that the transfer of information between the tasks occurs only by sharing some type of parameters. The second category will be called *Inductive transfer*, to cover all other situations where there is some type of correlation between the functions of the individual tasks.

[5]The vec operator concatenates the columns of matrix into a single column

2.2.1 Parameter Transfer

As in most machine learning algorithms the search for the optimal model parameters is of paramount importance. Parameters that can be shared by multiple GPs are:

1. the *hyperparameters* θ^x of the covariance function,

2. the *parameters of the prior of the latent function* $\mathbf{f}$, as the mean of the function and the covariance matrix denoted by θ^f.

The basic idea in the parameter transfer is that the latent functions of the tasks are independent given the parameters,

$$p(\mathbf{f}|\theta,\mathbf{X}) = \prod_{j=1}^{M} p(\mathbf{f}_j|\theta,X_j)p(\theta). \tag{2.13}$$

where we have used θ to refer to both θ^x and θ^f. Dropping the dependence of the prior on the parameters θ this can simply be written as $p(\mathbf{f}|\theta,\mathbf{X}) = \mathcal{N}(0,\mathbf{K})$ where $\mathbf{K}$ is block diagonal to the covariance matrices of each task,

$$\mathbf{K} = \begin{bmatrix} K_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & K_M \end{bmatrix}, \tag{2.14}$$

where each $\mathbf{K}_j$ is the $n_j \times n_j$ covariance matrix between the points in the j^{th} task. This particular setup implies that a test point will be correlated only with data points from the same task. Consequently, the prediction phase remains exactly the same as in STL, *e.g* for a test point from the j^{th} task the mean will be given by $\mu_{*j} = \sum_{i=1}^{n_j} a_{ij}k(x_{ij},x_*)$ with $\mathbf{a}_j = (\mathbf{K}_j + \sigma^2 I)^{-1}\mathbf{y}_j$.

2.2.1.1 Hyperparameter transfer

In the Gaussian process paradigm there are several approaches where the tasks exchange information during training by allowing them to have the same hyperparameters or share a common prior distribution over these hyperparameters (Menzefricke, 2000; Lawrence and Platt, 2004; Zhang and Yeung, 2009).

Menzefricke (2000) and Zhang and Yeung (2009) couple the different tasks within a hierarchical Bayesian framework. The dependencies between the different tasks are captured by imposing a common prior distribution over the hyperparameters of each task θ_j. The common prior over the hyperparameters depends on some common parameters which will be denoted by ω. The posterior distribution of the hyperparameters of all tasks $\theta = [\theta_1^T \dots \theta_M^T]^T$, is given by

$$p(\theta|\mathbf{y}, \omega) \propto \prod_{j=1}^{M} p(\mathbf{y}_j|\theta_j) p(\theta_j|\omega) \tag{2.15}$$

The difference between those two methods lies in the inference method they employ to estimate these hyperparameters. In Menzefricke (2000) the posterior probability distribution of these parameters is estimated by the Hybrid Monte Carlo method (Duane et al., 1987), by adopting a fully Bayesian inference framework. In Zhang and Yeung (2009) the optimal values of the hyperparameters and the parameters of its prior probability distribution ω are estimated by maximizing the log of the un-normalised posterior distribution given in equation 2.15. Additionally, the multi-task setting of Zhang and Yeung (2009) is extended to utilize unlabeled data in a semi-supervised fashion by employing the data-dependent norms introduced in Sindhwani et al. (2005, 2007)[6].

Lawrence and Platt (2004) extend the Informative Vector Machine (IVM) to the multi-task case. The IVM is a method for reducing the complexity of GPs that selects a subset of the training data by minimizing the entropy of the posterior process. In this approach, the different tasks are coupled by sharing the same hyperparamerets which are given point estimates by maximizing the marginal likelihood $p(\mathbf{y}|\mathbf{X}) = \prod_{j=1}^{M} \mathcal{N}\left(\mathbf{y}_j|0, (K_j + \sigma^2 I_{n_j})\right)$, where I_{n_j} is the identity matrix of length n_j, and it is assumed that the noise variance is the same for all tasks.

2.2.1.2 Latent parameter transfer

The Latent parameter transfer category focuses on placing a common prior over the mean and the covariance matrix of a Gaussian distribution (Schwaighofer et al., 2005; Yu et al., 2005; Birlutiu et al., 2010). The conjugate prior distribution for the mean and the covariance matrix of a multivariate Normal distribution $p(\mathbf{u}) = \mathcal{N}(u|\mu, K)$, is the Normal-Inverse Wishart distribution that can be parameterized in terms of the parameters $\theta^f = (\mu_0, \kappa_0, \nu_0, C_0)$(Gelman et al., 2004),

$$p(\mu, K) = \mathcal{N}(\mu|\mu_0, \frac{1}{\kappa_0}K) \mathcal{IW}(K|\nu_0, C_0) \tag{2.16}$$

[6]More information about the data dependent norms and semi-supervised learning can be found in Chapter 4.

where μ_0 is the mean of the prior distribution, $\frac{1}{\kappa_0}K$ is the covariance, and κ_0 is a scalar. The covariance matrix K is distributed according to the Inverse Wishart distribution with $\nu > 2N$ degrees of freedom and a positive definite parameter matrix C_0 (Gupta and Nagar, 2000).

Schwaighofer et al. (2005) assumes that each of the M functions share a common prior distribution. In this case the parameter matrix C_0 of the Inverse-Wishart distribution is set to be any positive definite matrix. Similarly to the Hyperparameter transfer category the functions are conditionally independent given the parameters of the prior. Then the joint distribution factorizes as,

$$p(\mathbf{y},\mathbf{f},\mu,K) = \prod_{j=1}^{M} p(\mathbf{y}_j|\mathbf{f}_j)p(\mathbf{f}_j|\mu,K)p(\mu,K). \tag{2.17}$$

The first problem they consider in this work is that of predictions on a fixed set of inputs, hence transductive learning in a multi-task scenario. If there are n_j data points from each task and N in total then the problem is the estimation of the outputs on the $N - n_j$ data points of each task. Inference of model parameters is performed with an EM algorithm. The *E-step* computes the expectations over the latent functions $\mathbf{f}_j$, and the *M-step* maximizes the parameters of the prior, which are the mean μ, the covariance K, and the noise variance σ^2. Predictions of the unobserved locations is performed in a straightforward manner by utilizing the update equations of the EM-algorithm. Generalisation on completely new points (beyond transduction) is performed with a variant of the Generalised Nyström method, where a kernel smoother is employed to generalise the learned covariance matrix K from the training phase. Yu et al. (2005) assign a common prior distribution over the parameters $\mathbf{a}_j \sim \mathcal{N}(\mathbf{a}_j|\mu,K)$ of each output $\mathbf{y}_{ij} = \sum_{i=1}^{N} a_{ij}k(\mathbf{X},x_i) + \varepsilon$, extending the *Subsets of Regressors* idea of Silverman (1985) (also discussed in Rasmussen and Williams (2005)[Ch. 8.3.1]), to the multi-task case. In this approach the parameter matrix C_0 of the Inverse-Wishart distribution is computed by a covariance function evaluated between data points from all tasks. Parameters of the prior and the estimates of $\mathbf{a}_j$ are learned with an EM algorithm similar to Schwaighofer et al. (2005).

Recently, Birlutiu et al. (2010) adapted the model of Yu et al. (2005) to non-Gaussian likelihoods for preference learning (Fürnkranz and Hüllermeier, 2010). A drawback of the methods presented in Schwaighofer et al. (2005) and Yu et al. (2005) is that, by sharing the same prior over the mean and the covariance matrix it is assumed that all tasks are correlated. This can make the models less effective in situations where some of the tasks are independent or simply do not contain beneficial information for the other tasks. Due to that reason in a follow up paper by the same authors in Yu et al. (2007b), they alleviated this problem by employing a multi-task model based on t-Processes (TP). In the TP formulation the samples of the functions are assumed to have been generated by a multivariate t distribution which is known to be more robust to outlier samples, with "*heavier tails*" than the Gaussian, and thus can accommodate outlier tasks.

2.2.2 Inductive Transfer

The main characteristic of the Inductive transfer category is that it requires some form of correlation between the functions of the different tasks. Other types of transfer, such as parameter transfer, are also allowed as long as the basic requirement is satisfied. It is worth noting that this form of transfer is stronger than the parameter transfer, since by allowing the functions of the tasks to be correlated in essence increases the *region of expertise* of the predictor, or *decreases its uncertainty* or both. By region of expertise is meant the part of the d dimensional space covered by the training data set, and by uncertainty the variance in the predictions. Additionally, it could be argued that if this transfer is combined with the parameter transfer then it is the maximal it can be achieved within the GP framework. On the other hand, the increased flexibility of this category of models should in principle be accompanied with a relatedness checking mechanism to avoid adverse effects, as negative transfer. Under the phenomenon of negative transfer the model's performance is degraded due to the transfer of information between the tasks (see section 1.2.2).

Given these specifications, having knowledge that the tasks are related, it would be beneficial to allow the mean and the variance of a test point from a certain task to be influenced by data from other tasks. As an example consider the following form of the predictive mean of a test point x_* from the j^{th} task,

$$\mu_{*j} = \sum_{i=1}^{n_j} a_{ij} k(x_{ij}, x_*) + \sum_{k \neq j} \zeta_{jk} \sum_{i}^{n_k} a_{ik} k(x_{ik}, x_*). \tag{2.18}$$

In this simple example the predictive mean consists of two terms, the first is due the information contained in the j^{th} task and the second comes from the information in the other tasks weighted by the parameters ζ, that can be used to quantify the level of correlation between the tasks. In the next sections we explore different ways of specifying correlated GPs, which results in making collocated predictions similarly to equation 2.18.

2.2.2.1 Intrinsic transfer

The intrinsic transfer category includes models where the covariance matrix $\mathbf{K}$ of all tasks is represented by the product of two separate matrices. The first matrix is used to model the correlations between the tasks and will be called *task covariance matrix* and denoted by K^t. The second matrix captures the dependencies between the data points, and will be called *data covariance matrix* and denoted by K^x. In the exposition that follows, it is assumed that for each input the outputs for all M tasks are observed, as in multi-response learning.

Assuming that the complete set of responses $\mathbf{y}$ is observed, we can define the matrix $\mathbf{F} = [\mathbf{f}_1 \dots \mathbf{f}_M]$, with $\mathbf{f} = \text{vec}(\mathbf{F})$, where $\mathbf{f}_j \in \mathbb{R}^{N\times 1}$ is the column vector of all function evaluations of task j. The *probability density function* of the matrix variate Normal distribution of $\mathbf{F}$ will be given by (Gupta and Nagar, 2000):

$$(2\pi)^{-\frac{1}{2}NM}|K^t|^{-\frac{1}{2}N}|K^x|^{-\frac{1}{2}M}\exp\left\{-\frac{1}{2}\text{trace}\left(\left(K^t\right)^{-1}\mathbf{F}\left(K^x\right)^{-1}\mathbf{F}^T\right)\right\}, \tag{2.19}$$

where $K^t \in \mathbb{R}^{M\times M}$ and $K^x \in \mathbb{R}^{N\times N}$, and both matrices need to be positive definite. As mentioned before, this configuration implies that the matrix K^t models the correlations between the vectors $\mathbf{f}_j$, *i.e.* the outputs of the tasks in the multi-response/task view, and K^x models the correlations between each element of each vector $\mathbf{f}_j$. Then, by using some matrix algebra involving the Kronecker and the vec operator, equation (2.19) can be written as

$$(2\pi)^{-\frac{1}{2}NM}|K^t\otimes K^x|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}\mathbf{f}^T\left(K^t\otimes K^x\right)^{-1}\mathbf{f}\right\}, \tag{2.20}$$

where $\otimes$ is the Kronecker product. This is the form of the model Bonilla et al. (2008) proposed to the machine learning community for multi-task learning, and has been known to the geostatistics community as the *Intrinsic Model of Coregionalization* (IMC),

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{GP}(0, K^t\otimes K^x). \tag{2.21}$$

Additionally, Bonilla et al. (2008) proposes the use of a *free form* task covariance matrix with hyperparameters θ^t that can be learned from the data to estimate the parameters k^t_{lk} reflecting the level of correlation between the tasks. Positive definite restrictions can be achieved through the *Cholesky decomposition* by parameterizing a lower triangular matrix $L \in \mathbb{R}^{M\times M}$, $K^t = LL^T$. The Cholesky parameterization can be restrictive for large number of tasks since it involves the estimation of $M(M+1)/2$

parameters. This problem can be alleviated by using the *incomplete Cholesky decomposition* as an approximation, $K^t = \tilde{L}\tilde{L}^T$, where $\tilde{L} \in \mathbb{R}^{M\times P}$ with P the rank of K^t. Situations where task descriptor features[7] $\mathbf{t}_j$ are available can be handled in the same setup by employing a parametric task covariance function $k^t_{lk} = k^t(\mathbf{t}_l, \mathbf{t}_k)$ (Bonilla et al., 2007). Moreover, for multi-response problems the size of resulting covariance matrix is $MN \times MN$ which for large number of tasks or data points can be restrictive, and approximation schemes must be employed. For multi-task problems the latent function evaluations for the unobserved outputs can be marginalized out with no further implications which results in a $N \times N$ covariance matrix. Notice also that marginalizing the unobserved outputs destroys the Kronecker structure $K = K^t \otimes K^x$, but the covariance between two latent functions can still be written as

$$\mathrm{cov}[f_l(x) f_k(x')] = k^t_{lk} k^x(x, x') \tag{2.22}$$

which effectively induces correlations between the latent functions through the elements of K^t. Finally, both types of hyperparameters θ^t and θ^x, can be learned by maximizing the marginal likelihood of all tasks via ML *II*.

2.2.2.1.1 Noise model and Predictions The complete likelihood of the model over all tasks becomes $p(\mathbf{y}|\mathbf{f}) \sim \mathcal{N}(\mathbf{f}, D \otimes I)$, with $D_{M\times M}$ diagonal with $D_{jj} = \sigma_j^2$, and $I_{N\times N}$. The predictive distribution for a test point x_* from the j^{th} task $p(y_{*j}|\ \mathbf{y})$ is given by $p(y_{*j}|\ \mathbf{y}) \sim \mathcal{N}(m_{y_{*j}|\mathbf{y}}, \Sigma_{y_{*j}|\mathbf{y}})$ where,

$$\Sigma_{y_{*j}|\mathbf{y}} = \lambda_{**} - \lambda^T \Sigma^{-1} \lambda \tag{2.23}$$

$$m_{y_{*j}|\mathbf{y}} = \lambda^T \Sigma^{-1} \mathbf{y} \tag{2.24}$$

where $\lambda_{**} = k^t_{jj} k^x_{x_*, x_*}$, $\lambda = k^t_j \otimes \mathbf{k}^x_{\mathbf{X}, x_*}$, and

$$\Sigma = K^t \otimes K^x + D \otimes I, \tag{2.25}$$

with k^t_{jj} and k^t_j are the j^{th} diagonal element and the j^{th} column of K^t respectively, $\mathbf{k}^x_{\mathbf{X}, x_*}$ is the vector of covariances between the test point x_* and the training points $\mathbf{X}$, and $\mathbf{k}^x_{x_*, x_*}$ is the variance of the test point. Rewriting the predictive mean in terms of a linear combination of MN basis functions we have that

$$m_{\mathbf{y}_{*j}|\mathbf{y}} = \sum_{i=1}^{MN} \Delta_i (k^t_j \otimes \mathbf{k}^x_{\mathbf{X}, x_*}), \tag{2.26}$$

[7]The term task descriptor feature is used to refer to a separate set of features that are informative about the learning task; task descriptor features can be extracted or computed by the standard input space using expert knowledge as in Bonilla et al. (2007).

where the elements of Δ are computed from $\Delta = \Sigma^{-1}\mathbf{y}$. Equations 2.24 and 2.26 show that this setup explicitly allows the transfer of information in the prediction phase. As an example consider the case where there are two tasks and we wish to make predictions on the first. Here, we will focus on the predictive mean but similar analysis can be performed for the variance. We define the following block matrices,

$$\lambda = \begin{bmatrix} k^t_{11}\mathbf{k}^x_{\mathbf{X},x_*} \\ k^t_{21}\mathbf{k}^x_{\mathbf{X},x_*} \end{bmatrix}, \quad \Sigma^{-1} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \Xi_{11} & \Xi_{12} \\ \Xi_{21} & \Xi_{22} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \tag{2.27}$$

where matrices Ξ_{ij} can be computed from the *Woodbury formula* (see appendix B.3 or Rasmussen and Williams (2005)). Breaking down equation 2.24 and rearranging gives that,

$$m_{\mathbf{y}_{*1}|\mathbf{y}} = k^t_{11}(\mathbf{k}^x_{\mathbf{X},x_*})^T\Xi_{11}\mathbf{y}_1 + k^t_{21}(\mathbf{k}^x_{\mathbf{X},x_*})^T\Xi_{22}\mathbf{y}_2 + k^t_{21}(\mathbf{k}^x_{\mathbf{X},x_*})^T\Xi_{21}\mathbf{y}_1 + k^t_{11}(\mathbf{k}^x_{\mathbf{X},x_*})^T\Xi_{12}\mathbf{y}_2, \tag{2.28}$$

which reveals that the predictive mean is the sum of four terms; the first depends only on the first task multiplied by k^t_{11}, the second depends only on the second task multiplied by the correlation parameter between the two tasks k^t_{21}, and the last two terms account for the correlations between the two tasks. In the general case of M tasks the predictive mean for the j^{th} task can be written as the sum of three terms,

$$m_{\mathbf{y}_{j*}|\mathbf{y}} = k^t_{jj}(\mathbf{k}^x_{\mathbf{X},x_*})^T\Xi_{jj}\mathbf{y}_j + \sum_{l\neq j} k^t_{lj}(\mathbf{k}^x_{\mathbf{X},x_*})^T\Xi_{ll}\mathbf{y}_l + \sum_{\gamma=1}^{M}\sum_{l\neq j} k^t_{lj}(\mathbf{k}^x_{\mathbf{X},x_*})^T\Xi_{\gamma l}\mathbf{y}_\gamma, \tag{2.29}$$

where the first term depends on the information contained on the j^{th}, the second is a weighted sum of the information contained in each of the other tasks, and the third that depends on the cross-task correlations appropriately weighted[8]. Hence, this model can intelligently transfer information of the other tasks when generalising to new points, by weighting appropriately the contribution of the other tasks, and the elements of K^t are equivalent to the ζ parameters in equation 2.18. As Bonilla et al. (2008) pointed out though, the noise-free case when the complete set of responses is observed, *i.e.* $\Sigma = K^t \otimes K^x$, results in cancellation of inter-task transfer. Thus, in the noise-free case, which has been known to the geostatistics literature as autokrigeability, the predictive mean of a task is not affected by information in other tasks and is similar to the Parameter transfer and STL.

[8]Note that each of the terms in equation 2.29 has some sort of dependence with information contained in other tasks since $\mathbf{k}^x_{\mathbf{X},x_*}$ is the covariance vector between all training points and the test point.

2.2.2.1.2 Linear Model of Coregionalization This type of models have previously been studied in the field of geostatistics for multi-output models, and they are known as the *Linear Models of Coregionalization* (LMC)[9]. In its general form the LMC assumes that each of the M outputs is a linear combination of Q independent random functions $u(x)$ or Q groups of R_q independent random functions with the same covariance functions. The exposition of the LMC that is presented here follows Alvarez and Lawrence (2011). The latent function of the j^{th} task will be given by (Journel and Huijbregts, 1978),

$$f_j(x) = \sum_{q=1}^{Q} \zeta_{jq} u_q(x) = \sum_{q=1}^{Q} \sum_{r=1}^{R_q} \zeta_{jq}^r u_q^r(x), \tag{2.30}$$

where we have slightly abused the notation since the ζ^r parameters used here are not the same as in equation 2.18. Without loss of generality, it is assumed that each of the functions for $q = 1, \dots, Q$ and $r = 1, \dots, R_q$ are drawn from a GP with zero mean and covariance $\mathrm{cov}[u_q^r(x), u_{q'}^{r'}(x')] = k^x(x,x')$ if $q = q'$ and $r = r'$ and zero otherwise. It can be shown, that the covariance between two functions $f_l(x)$ and $f_k(x')$ can be written in terms of the covariance of the random functions $u_q^r(x)$ as (Cressie, 1993)

$$\mathrm{cov}[f_l(x), f_k(x')] = \sum_{q=1}^{Q} \sum_{q'=1}^{Q} \sum_{r=1}^{R_q} \sum_{r'=1}^{R_q} \zeta_{lq}^r \zeta_{kq'}^{r'} \mathrm{cov}[u_q^r(x), u_{q'}^{r'}(x')] \tag{2.31}$$

while taking into consideration that the random functions $u_q^r(x)$ are independent the covariance of the vectors $\mathbf{f}_l$ and $\mathbf{f}_k$ reduces to,

$$\mathrm{cov}[\mathbf{f}_l, \mathbf{f}_k] = \sum_{q=1}^{Q} \sum_{r=1}^{R_q} \zeta_{lq}^r \zeta_{kq}^r K_q^x = \sum_{q=1}^{Q} b_{lk}^q K_q^x \tag{2.32}$$

where $b_{lk}^q = \sum_{r=1}^{R_q} \zeta_{lq}^r \zeta_{kq}^r$, and each K_q^x is computed between all N points, thus $K_q^x \in \mathbb{R}^{N \times N}$. The joint distribution of all latent functions $\mathbf{f}_j$ will be given by $p(\mathbf{f}) = \mathcal{N}(0, \mathbf{K}_{LMC})$ with,

$$\mathbf{K}_{LMC} = \sum_{q=1}^{Q} Z_q Z_q^T K_q^x = \sum_{q=1}^{Q} K_q^t \otimes K_q^x, \tag{2.33}$$

where $Z_q \in \mathbb{R}^{M \times R_q}$ with entries ζ_{lk}^r, and the matrix $K_q^t = Z_q Z_q^T$ has elements b_{lk}^q and is also known as the *coregionalization matrix*. The model proposed by Bonilla et al. (2008) is a simplified version of the LMC where $Q = 1$ (equation 2.21)[10]. Note that

[9]In the subsequent analysis it is assumed that the complete set of responses is observed.

[10]Similar analysis of the LMC and IMC can also be found in Chai (2010) and Alvarez and Lawrence (2011).

the LMC is more flexible than the IMC, since it allows to employ multiple covariance functions, but on the other hand estimating their parameters can be computationally restrictive for real world applications. Moreover, as it is conjectured in Alvarez and Lawrence (2011), the rank of each of the matrices K_q^t can be interpreted from the generative point of view. The elements b_{lk}^q of the matrices K_q^t are given by the sum of the R_q weights ζ_{lq}^r of each function in that group. Hence, the rank of each K_q^t is determined by the number of the independent functions that share the same covariance function, that is R_q. Other interpretations of the LMC and the IMC have also been given and we refer the interested reader to Alvarez et al. (2011b) for a review on that subject.

Closely related to the LMC is the *Semiparametric Latent Factor Model* by Teh et al. (2005). This setup also assumes that each of the outputs is a linear combination of Q functions. The overall covariance of all outputs is given by $\tilde{\mathbf{K}} = (\Phi \otimes I)\mathbf{K}^x(\Phi^T \otimes I)$ with $\mathbf{K}^x$ block diagonal on matrices K_q^x and $\Phi \in \mathbb{R}^{M\times Q}$. The covariance matrix $\tilde{\mathbf{K}}$ can also be written as $\tilde{\mathbf{K}} = \sum_{q=1}^{Q} \phi_q \phi_q^T \otimes \mathbf{K}_q^x$ which resolves to the LMC model in equation 2.33, with $\phi_q \in \mathbb{R}^{M\times 1}$ and $\Phi = [\phi_1 \ldots \phi_Q]$. Other applications of the IMC include the work by Osborne et al. (2008) for modeling sensor network data, or for emulating computer codes in Conti and O'Hagan (2010).

2.2.2.2 Convolutional Transfer

As mentioned before in section 2.1.2, it is known that the convolution of a Gaussian Process with a kernel is also a Gaussian Process. This property can be used to construct new covariance functions. For example if $u(z)$ is a GP and $h(x,z)$ is a smoothing kernel, then the function $f(x)$ given by the convolution of $u(z)$ and $h(x,z)$ is also a GP (Barry and Jay, 1996; Rasmussen and Williams, 2005),

$$f(x) = \int h(x,z)u(z)\mathrm{d}z, \tag{2.34}$$

and given that $\mathrm{cov}[u(x),u(x')] = k(x,x')$, the covariance between two data points x and x' can be computed from,

$$\mathrm{cov}[f(x),f(x')] = \int h(x,z)k(z,z')h(x',z')\mathrm{d}z\mathrm{d}z'. \tag{2.35}$$

These results were taken a step further, by using convolutions to construct multiple output covariance functions. Hence, in the Convolutional transfer each function can be represented as the convolution of a kernel $\{h_j(x,z)\}_{j=1}^{M}$ and the latent $u(z)$ (Alvarez and Lawrence, 2011),

$$f_j(x) = \int h_j(x,z)u(z)\mathrm{d}z. \tag{2.36}$$

Similarly to the LMC we can consider a more general form with more than one latent functions $\{u_q(z)\}_{q=1}^{Q}$. Considering the case where $q = 1$, the covariance between two outputs $f_l(x')$ and $f_k(x')$ will be given by (Alvarez et al., 2011a),

$$\mathrm{cov}[f_l(x), f_k(x')] = \int h_l(x,z) \int h_k(x',z')k(z,z')\mathrm{d}z\mathrm{d}z'. \tag{2.37}$$

This covariance defines a prior over the complete set of latent functions $p(\mathbf{f}) = \mathcal{N}(0, K_{\mathbf{f}})$, with the covariance matrix $K_{\mathbf{f}}$ being fully connected between the outputs. As in the Intrinsic transfer for multi-response problems the resulting matrix is of size $MN \times MN$ with block matrices $K_{lk} \in \mathbb{R}^{N\times N}$ denoting the covariance matrix between the l and k outputs. The marginal likelihood in this type of setting will be given by $p(\mathbf{y}|X,\theta) = \mathcal{N}(0, K_{\mathbf{f}} + D\otimes I_N)$ with $D \in \mathbb{R}^{M\times M}$, $D_{jj} = \sigma_j^2$ and I_N is the identity matrix of size N. Predictions are made using the standard formulas for GP prediction given in equations 2.6 and 2.7, whereas optimization of the hyperparameters is performed by maximizing the marginal likelihood. Note that the fully connected covariance matrix $K_{\mathbf{f}}$ allows predictions for a certain task to be influenced by the information in other tasks.

Convolved multi-output GPs were originally proposed in Higdon (2002) and later on by Boyle and Frean (2005) to the machine learning community. A series of papers by Álvarez (Alvarez and Lawrence, 2009; Alvarez et al., 2009, 2011a; Alvarez and Lawrence, 2011) investigate the application of GPs to multi-output models through the convolutional setting as well. The methods proposed in Alvarez and Lawrence (2009, 2011); Alvarez et al. (2011a) present methods for reducing the complexity of the convolutional multi-output GP model that scales as $O(M^3N^3)$, whereas Alvarez et al. (2009) propose a method based on GPs and differential equations to describe a physical system extending the work presented in Lawrence et al. (2007) to more than one latent functions $\{u_q(z)\}_{q=1}^{Q}$.

2.2.2.3 Mixed effect transfer

Other multi-task models in the Gaussian process literature are what we call the Mixed effect models (Pillonetto et al., 2010; Wang et al., 2010). Under this formulation the latent function of each task is given by the mixture of two functions; a common function across all tasks and a task specific function. The function of the j^{th} task will be written as,

$$f_j(x) = \bar{f}(x) + \tilde{f}_j(x). \tag{2.38}$$

The common or global term $\bar{f}(x)$ is used to describe the similarities between the tasks and the task specific term $\tilde{f}_j(x)$ characterizes the individual differences between the tasks.

Pillonetto et al. (2010) employ the model in equation 2.38 with the addition of two parameters $\bar{\lambda}$ and $\tilde{\lambda}$ that weight the functions $\bar{f}$ and $\tilde{f}$ respectively. Interestingly, the *global* function $\bar{f}$ is evaluated only on the data points from matrix $X_u = [x_{1u}, \dots, x_{n_u u}]$, that is constructed by the distinct elements (*i.e* with no repetitions) of the matrix $\mathbf{X}$, denoted by $X_u \in \mathbb{R}^{d \times n_u}$ where $n_u < N$ is the number of distinct elements. In addition, an online algorithm is developed that is based on the recursive estimation of the conditional distribution of a new task given the others. Of central importance is the predictive mean of the j^{th} task at x_{j*} given by (Pillonetto et al., 2010),

$$\mu_{j*} = \bar{\lambda}^2 \sum_{i=1}^{n_u} \bar{a}_i \bar{k}(x_{iu}, x_{j*}) + \tilde{\lambda}^2 \sum_{i=1}^{n_j} \tilde{a}_{ji} \tilde{k}(x_{ji}, x_{j*}), \tag{2.39}$$

where $\bar{k}(.)$ and $\tilde{k}(.)$ are the covariance functions of the global and task specific function respectively, $\tilde{a}$ depends only on the elements of the j^{th} function $\tilde{\mathbf{f}}_j$, and $\bar{a}$ depends only the global function $\bar{\mathbf{f}}$. In this case there are no correlation parameters as in the IMC but parameters $\tilde{\lambda}$ and $\bar{\lambda}$ quantify the contribution of each function. If $\bar{\lambda}$ is zero all tasks are learned independently, whereas if $\tilde{\lambda}$ is zero all tasks are the same. Parameters $\tilde{\lambda}$ and $\bar{\lambda}$, and the hyperparameters of each covariance function $\tilde{k}(.)$ and $\bar{k}(.)$ can then be estimated by type *II* ML. Hence, this method can automatically estimate the mixing of the tasks, and as a result predictions on a task can be affected by data in other tasks through the contribution of the global term.

The model introduced in (Wang et al., 2010) extends the work of Pillonetto et al. (2010) to allow tasks to group into clusters with the addition of an extra parameter. Equation 2.38 is transformed to,

$$f_j(x) = \bar{f}_{z_j} * \delta_{t_j} + \tilde{f}_j, \tag{2.40}$$

where $*$ denotes the convolution, $z_j \in \{1, \dots, Q\}$, Q is the number of clusters, and δ_{t_j} is the Dirac δ function with support at $t_j \in [0, \mathrm{T}]$ ($x \in [0, \mathrm{T}]$). This formulation allows to group the tasks into clusters which can be essential for situations where not all tasks are correlated or there are outlier tasks, and can be seen as a more flexible version of the model presented in Pillonetto et al. (2010).

We note in passing that an interesting extension of the models proposed in Pillonetto et al. (2010) and Wang et al. (2010) would be the automatic estimation of the input points corresponding to the global latent function evaluations by borrowing ideas from the framework of sparse Gaussian Processes using pseudo-inputs of Snelson and Ghahramani (2006). This extension would allow this type of formulation to be applied in problems with continuous inputs, in contrast with the current formulation of Pillonetto et al. (2010) who applies it in a pharmacokinetic problem with discrete inputs.

2.2.3 Interlude

This section presented an overview of the current approaches to MT-GP regression. These approaches were separated into two major categories, the *Parameter transfer* and the *Inductive transfer*. The main difference of these categories is that in the Parameter transfer the functions of each task are independent whereas in the Inductive transfer there is a form of dependency between the functions. As a result of the correlation of the latent functions, predictions on a task are influenced by the information from other tasks. Additionally, parameter sharing can also be induced in the Inductive category making it a stronger form of transfer learning than the Parameter category, since transfer of information occurs both in the training and the prediction phase.

In the *Intrinsic* and *Convolutional transfer* the covariance matrix of the latent function of all tasks is fully connected allowing the tasks to interact, while in the *Mixed effect transfer* a global function captures the tasks similarities. A characteristic of the Mixed effect transfer is that the estimation of the global function is based on the notion of repeated patterns. This is justified by the fact that these approaches are intended for time-series problems, where the input data are discrete time points. This feature makes the method inappropriate for continuous or high dimensional input spaces. In the Convolutional transfer, each output/task j is modeled with a smoothing kernel $h_j(x,z)$, that can "*be used to capture the degree of smoothness and the length-scale that characterizes each output*" Alvarez and Lawrence (2011). In the Intrinsic transfer, the LMC and IMC can be seen as a weighted sum of independent random processes. This leads to a separable covariance function, with one covariance matrix modeling the correlations of the tasks, and one that models the correlations of the inputs. Most importantly, the ICM does not assume *a priori* that the tasks are correlated but learns the task dependencies from the data. This characteristic makes the method robust against outlier tasks, whereas it can also be used for the identification of the relatedness of the tasks.

Given the characteristics of each method, it is concluded that the CP based transfer and the IMC offer the highest degree of transfer between the tasks. Comparing these two methods, it is obvious that the CP based approach does not accommodate mechanisms for the identification of the relatedness of the tasks. In addition, it is unclear whether the CP approach can efficiently isolate an unrelated task, since each task is a blurred version of the others.

A closely related area of work, which we will not investigate in this thesis is link analysis. Multi-task and link analysis are intimately connected techniques that use data from different tasks either to improve generalisation performance or to model the relations between different entities. The model proposed in Bonilla et al. (2008) is closely related to the work of Yu et al. (2007a), which employs two kernel functions through the tensor product to model the dependencies between different set of entities. The properties of the model proposed in Bonilla et al. (2008), and Yu et al. (2007a) were found attractive enough to motivate work in both directions. In multi-task learning the work in Zhang and Yeung (2010) is inspired by Bonilla et al. (2008), where they employ a t-noise model for the likelihood. In context of stochastic relational models Zhu et al. (2009) propose an algorithm based on MCMC able to handle very large data sets. Moreover, the work for link analysis in Yu and Chu (2008) within the GPs, demonstrates intimate connections with other transfer learning algorithms. The connections of the IMC model of Bonilla et al. (2008) with other MT methods for regression and the setting of link analysis demonstrate its importance and its potentials for application to other transfer learning scenarios.

2.3 Empirical Evaluation of Multi-task learning

The aim of the current section is to demonstrate the effectiveness of the framework of multi-task learning for GP regression. The objective is threefold: one is to illustrate the superiority of multi-task learning upon single-task, secondly to compare the Parameter and Inductive transfer categories, and last but not least to show the benefits it can bring on a real world application[11]. We begin with a short introduction to the application, we then present in detail the experimental setup, and on the last subsection we conclude with a discussion on the results.

[11]This is part of on-going work with collaborators from TU Berlin, Matthias Rupp and Katja Hansen.

2.3.1 Predicting pK_a

The behavior of a compound in solution is influenced by its acidity or basicity which can be measured by its dissociation constant K_a. Strong acids have high K_a values, while strong bases have low K_a values. In this work, we consider the problem of estimating the *dissociation constants* for weakly acidic or basic groups which are expressed as the K_a of that group (Lee and Crippen, 2009). It is known, that when a weak acid dissociates according to the schematic equation $HA \leftrightharpoons A^- + H^+$, then the equilibrium constant is given by,

$$K_a = \frac{a(A^-)a(H_3O^+)}{a(HA)a(H_2O)}, \tag{2.41}$$

where $a(.)$ denotes activities. Additionally, it is known that for low concentrations of $(HA < 1mM)$, the activities can be approximated by concentrations $c(.)$. Then, equation 2.41 can be conveniently rearranged to give the Henderson-Hassellbach equation (Rupp et al., 2010),

$$pK_a \approx pH + \log_{10} \frac{c(HA)}{c(A^-)}, \tag{2.42}$$

where $pK_a = -\log_{10} K_a$.

The prediction of the pK_a value is a very important task, since its value has a dominant role in a plethora of phenomena. For example, in biochemical processes the permeability of the molecule through a membrane, or the stability of a protein and the activity of an enzyme can depend upon the pK_a of the molecules involved. Moreover, in pharmacology, drug formulation is based upon the desired value of pK_a which controls the solubility of the drug molecule. Finally, assessing the hazard or the toxicity associated with an acidic or basic substance that is used in whatever process, whether in a lab or even at home, relates to the pK_a of the substance (whether this substance is an extravagant chemical agent or an everyday cleaning agent). Thus, the ability to estimate the pK_a of a molecule without incorporating experimental approaches that are time consuming or difficult for certain molecules is of vital importance to a wide variety of fields. For a more in depth discussion about this subject we refer the interested reader to Lee and Crippen (2009) and Rupp et al. (2010).

The data set consists of 15 groups of molecular compounds, each one with different number of samples given in table 2.1. Each of these groups has a different pK_a range of values, not only because of their certain atomic composition but also because of the spatial (3D) arrangements of these atoms within the molecule that depend on the substitutions (table 2.1). The data were made available by Matthias Rupp and for preprocessing details we refer to Rupp et al. (2010). The data set contains only molecules with one ionizable center (monoprotic compounds). Molecules were described by the electrophilic superdelocalisability (SE) of the ionizable center atom, its immediate neighbors (one bond away from the center, binned), and their neighbors (two bonds away from the center, binned). Thus, each molecule is represented by a 3-vector (see Rupp et al. (2010), in particular model R' there). The scatter plot of the three-dimensional input vector extracted from each molecule is shown in figure 2.1.

Table 2.1: Description of the pKa dataset; pK_a range is given as min-max values, IHB and NIHB indicates that the group is *capable* and *not capable* of forming internal hydrogen bonds respectively.

No.	**Task**	n	pK_a **range**	**Description**
1	**Aa**	57	5.42 - 10.45	Phenols, *meta/para*-substituted
2	**Ab**	26	3.03 - 9.87	Phenols, *ortho*-substituted, IHB
3	**Ac**	91	0.38 - 12.23	Phenols, *ortho*-substituted, NIHB
4	**Ad**	46	2.82 - 4.85	Benzoic acids, *meta/para*-substituted
5	**Ae**	53	0.65 - 5.09	Benzoic acids, *ortho*-substituted
6	**Af**	143	0.51 - 6.20	Aliphatic carboxylic acids
7	**Ba**	55	-5.00 - 5.48	Anilines
8	**Bb**	23	5.70 - 10.87	Amines, primary
9	**Bc**	23	8.50 - 11.39	Amines, secondary
10	**Bd**	31	6.57 - 11.25	Amines, tertiary
11	**Be**	48	0.67 - 6.47	Pyridines, *meta/para*-substituted
12	**Bf**	34	-2.86 - 7.90	Pyridines, *ortho*-substituted
13	**Bg**	14	-1.63 - 6.81	Pyrimidines
14	**Bh**	26	-0.53 - 7.85	Imidazoles and benzimidazoles
15	**Bi**	28	2.69 - 6.10	Quinolines

Until now, estimation of the acid dissociation constants based on regression models has been done by training a model for each group separately. Regression models that have been considered vary from Linear or Ridge regression, to Neural Networks, tree-based models or Graph kernels and others (Lee and Crippen, 2009; Rupp et al., 2010). Instead of treating each group separately in this work we will investigate the simultaneous training of all groups/tasks together.

2.3.2 Experimental setup

As previously stated, the aim of this section is first to show the benefits of MTL over STL and second to compare the two categories of MTL within the GP framework. Therefore, results will be presented on the following model with GPs:

1. *single-task* learning (STL), that is training a separate model for each task,

2. *multi-task* learning based on the Hyperparameter transfer category (MTL-IND), here all the tasks share the same hyperparameters in a similar fashion to MT-IVM of Lawrence and Platt (2004),

3. *multi-task* learning based on the Intrinsic transfer category (MTL-IMC-COR), with task covariance matrix computed as a correlation matrix[12] (Rebonato and Jäckel, 2000),

4. *multi-task* learning based on the Intrinsic transfer, with task covariance matrix based on the incomplete Cholesky factorization of rank 2 (MTL-IMC-R2),

5. and finally by training a single model by *pooling* all data from all tasks together (Pool).

As shown, we employ two constructions for the task covariance matrix with the IMC to provide more insights about the learning process of the task correlations, and the effects each parameterization has. In all of these models STL or MTL, hyperparameters of the task covariance matrix K^t or the data covariance function $k^x(x,x')$ are estimated by maximizing the marginal likelihood (type *II* ML).

Additionally it should be noted that although both parameterizations of the task covariance matrix allow negative off-diagonal elements which would result in task being anti-correlated, in this application this was not permitted. The reason for this particular choice is more from the practical point of view since preliminary experimentation has revealed that false negative correlation between two tasks can have a disastrous effect on the models performance. This observation with the addition of the known problem of local maxima of type *II* ML optimization, has compelled us to restrict both parameterizations not to allow negative correlations. For example, in the Incomplete Cholesky factorization this was achieved by re-parameterisation, simply by taking the exponential of the parameters and thus not allowing negative parameter values and

[12]Details about the parameterization and computation of the correlation matrix are given in Chapter 3.

consequently negative off diagonal elements in K^t. Also, note that the rank of the Incomplete Cholesky factorization is a model parameter and thus can not be optimized; instead it can be determined in a principled way by model selection, for example by using the Bayesian Information Criterion (Bishop, 2006, Ch. 4.4), an approach that was followed in Chai et al. (2009) for modeling robot inverse dynamics. Here the rank of the Incomplete Cholesky factorization was chosen such that the number of parameters that needs to be estimated is significantly lower than that of the Correlation matrix of Rebonato and Jäckel (2000).

Moreover, there is expert knowledge that there are two clusters of tasks with strong inter-task correlations; the first cluster is comprised by molecules in the "A" class (tasks 1-6), and the second cluster is comprised by molecules on the "B" class (tasks 9-15). Therefore, we consider two types of experiments. In the first round, we separate the tasks into two groups which we term as the "A" and the "B" groups, and employ the multi-task algorithms for each cluster separately. In the second round of experiments all 15 tasks (AB) are employed together. Figure 2.1 shows the scatter plot of the two groups of tasks.

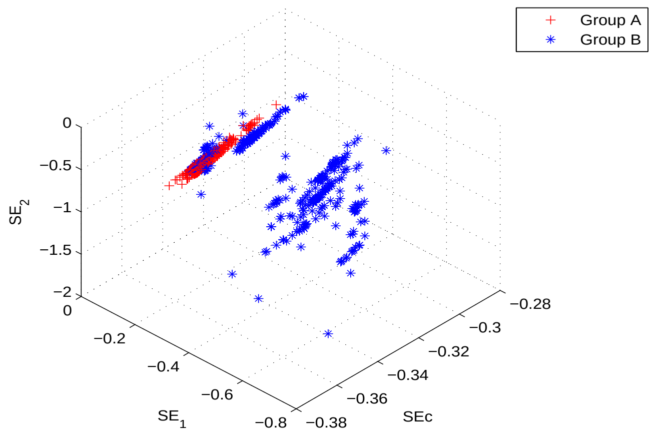


Figure 2.1: Scatter plot of the pK_a data set; red crosses indicate data from the "A" group and blue stars indicate data from the "B" group of tasks, $SEc = SE$ of ionizable center, SE_1 = binned SE of neighbors 1 bond away, and SE_2 = binned SE of neighbors 2 bonds away from center.

As mentioned before, the benefits of multi-task learning are more apparent in situations where limited annotated data are available. For that reason, training of the models is performed for eight different data partitions, starting from 10% up to 80% of the total number of data points in each task. Situations where the percentage of the data set was not an integer were dealt by rounding down to the closest integer. Moreover, for each partitioning experiments are performed 100 times, by randomly selecting the annotated data.

Output data were standardized to zero mean and unit variance according to $y' = (y-\mu_y)/\sigma_y$, where μ_y is the mean and σ_y is the standard deviation (std) of the outputs. The standardization was based on the training data set only, whereas in the prediction phase test points were transformed to the same scale based on the mean and standard deviation of the training set. In the cases of multi-task learning and pooling, the mean and the standard deviation were computed from data from all tasks. Preliminary analysis revealed that standardizing the data for each task separately was resulting in poor performance. We report these results in the end of the following subsection. Performance of the algorithms is assessed in terms of *Root Mean Squared Error* (RMSE) on the unseen data of each task, given by $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_{*i} - f(x_{*i}))^2}$. Another performance measure that could have been used is the negative log probability of the true output y_* under the model $-\log p(y_*|\mathbf{y}, \mathbf{X}, x_*)$ (Rasmussen and Williams, 2005, Ch. 2.5), which would effectively make use of the predictive variance. Additionally, notice that the predictive variance could have also been used to quantify the uncertainty in our predictions. In order to compare the results of the different methods we used the *t-test*.

The covariance function that is chosen is the Linear, $k(x_i, x_j) = \theta_0^2 + \theta_1^2 \sum_{k=1}^{d} x_{ik}x_{jk}$. The choice of the covariance function was suggested by Matthias Rupp so that the results can be compared with other baseline methods. Interestingly, preliminary experiments with the *Squared exponential* (SE) covariance function on the "A" group of tasks revealed that the Linear covariance function was more robust (smaller error bars) and had lower RMSE than the SE.

2.3.3 Results

Figures 2.2 (a) and (b) present the results for the "A" and "B" groups separately, whereas figure 2.3 presents the results for putting all 15 tasks together. For each data partition the mean and the std are computed by first averaging over the tasks and then over the repetitions.

Figure 2.2 (a) shows that the performance of the algorithms on the "A" group of tasks. The results follow a pattern that verifies our classification of multi-task algo-

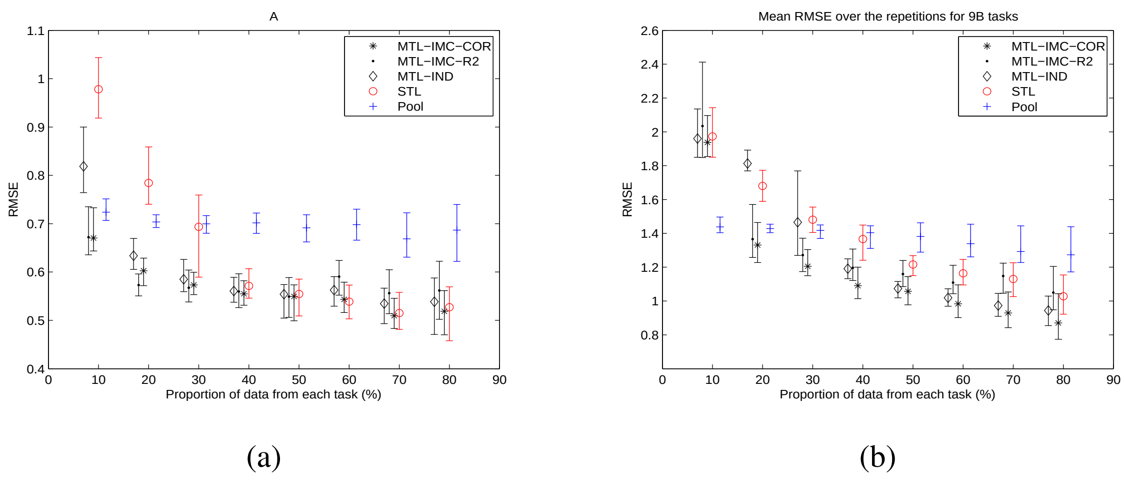


Figure 2.2: RMSE for the “A” and “B” groups of tasks separately. Figure (a) presents the results for the “A” group, and figure (b) presents the results for the “B” group.

rithms. For small data partitions all multi-task algorithms offer a clear advantage over STL, while both IMC models perform better that the IND. Thus, our intuition that the Inductive transfer is stronger than the Parameter transfer is verified through empirical evidence. On the other hand, for small training data sizes (10% and 20 %) the Pool method performs better that the STL, but its error remains almost constant with larger error bars as the training data set increases. This highlights the need for methods having mechanisms of exploiting information from different tasks, and that simple pooling does not work.

Comparing the two IMC models we notice that although the R2 construction of the task covariance matrix produces a higher mean RMSE than the construction based on the correlation matrix for 60%, 70% and 80% of the training set, the differences are not statistical significant (t-test) at 0.05 p-value in any of the data partitions; in contrast, differences in terms of RMSE between the MTL-IMC and the MTL-IND models were significant for 10%, 20%, 30%, and 70% of the available data. More in depth statistical analysis revealed that the differences between MTL-IMC and STL were statistical significant at 0.05 p-value from 10% up to 40% of the available samples, whereas all differences in RMSE between MTL-IMC and the Pool method were significant except for 10% of the samples.

Considering the “B” group separately in figure 2.2 (b), it is shown that for 10% of the training set all MTL and STL perform similarly, while the Pool method exhibits the best performance. Interestingly, for 20% of the data set the STL performs better that the MTL-IND, something that in principle should not happen. As before, for larger training sizes (>40%) the error for the Pool method is higher than all the other methods, with an increasing std highlighting its uncertainty in the predictions. Notably, the IMC-COR method presents a lower mean RMSE over all other methods for training sets higher than 20%. Additionally, the differences in RMSE between the IMC-COR and IMC-R2 were statistical significant for all data partitions except for 20% and 30%. This can be explained, since the rank 2 Incomplete Cholesky factorization can be seen as a poor approximation of the true underlying structure of the tasks. In the case of the "B" group the structure of the tasks is a lot more complex which can easily be inferred from the scatter plot in figure 2.1, where it is shown that data points from group "B" are a lot more dispersed in the space than the data points from the "A" group. Continuing, the reported RMSE values between IMC-COR and MTL-IND were statistical significant at 0.05 p-value for 20%, 30%, 40%, and 70% of the data partitions, whereas differences between IMC-COR and Pool and STL were significant for all data partitions above 20%. This result shows that even for large training sets transfer of information between the task can improve performance.

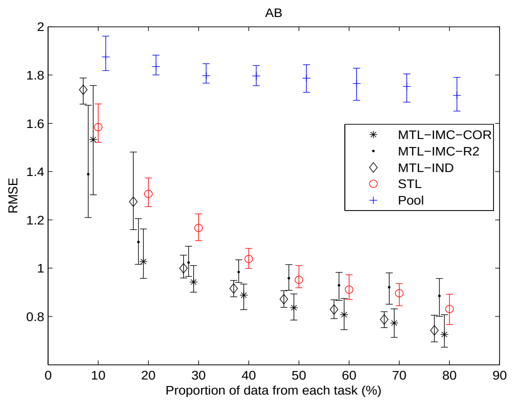


Figure 2.3: RMSE for all 15 tasks together on the second set of experiments.

Putting all tasks together we obtain similar results with the two previous cases (figure 2.3). For 10% of the data set the performance of STL is close to the MTL-IMC models, while it outperforms the MTL-IND. For 20% of the training set STL continues to outperform the MTL-IND model, while the IMC-COR model performs better than all the others. Similarly, to the "B" group the IMC-COR model outperforms the IMC-R2 for all partitions above 20% (statistically significant at 0.05 p-value) which shows that this parameterization is more flexible and that it is able to model the task relationships in a more effective way. It should be noted that parameterizing the task covariance matrix as a correlation matrix based on the work Rebonato and Jäckel (2000) involves the estimation of $(M^2 - M)$ parameters, while the incomplete Cholesky factorization involves only (Mr) parameters where r is the rank and M is the number of tasks. Note also that differences between IMC-COR and MTL-IND were statistically significant at 0.05 p-value for partitions: 20, 30, 40, 50 of the available data, and between IMC-COR and STL for all partitions above 10%.

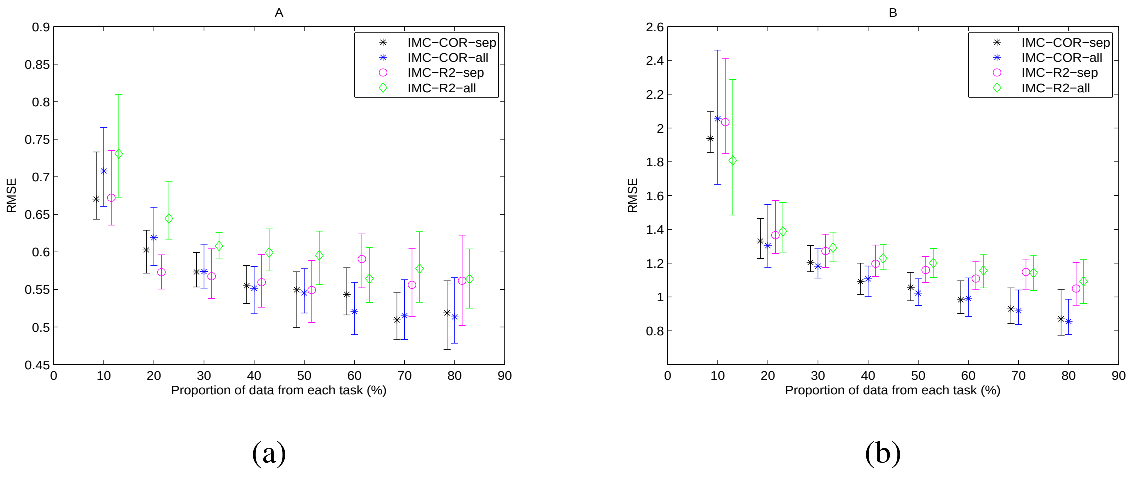


Figure 2.4: Direct comparison between the different constructions of the task covariance matrix and the two sets of experiments; figure (a) presents the results for the “A” group, and figure (b) presents the results for the “B” group of tasks.

Direct comparison of the two constructions of the task covariance function is given in figure 2.4. Additionally, in the same figures we plot the results obtained by considering the groups separately (first round of experiments given in figure 2.2) and both groups of tasks together (second round of experiments given in figure 2.3), which we term as “IMC-...- sep” and “IMC-...-all” respectively. We notice that in the A group of tasks in figure 2.4(a) for 10% of the training set the “R2-sep” construction performs better than the COR methods, and that in the B group in figure 2.4(b) the “R2-all” construction is better than all the other. More in depth comparison between the IMC-COR-sep and the IMC-R2-sep construction revealed that in the 6A group the COR construction produced a lower RMSE for sizes larger than 20% but was statistically significant at 5% p-value only for sizes larger than 40%. In the 9B group the COR construction produced a lower RMSE for all sizes but was statistically significant for 10% and for larger than 30% of the data set size. The fact that for small training sets the R2 construction is better than the COR method can be explained, since the model does not have enough information to produce good estimates of the large number of parameters of the COR method. This is also verified by the fact that for 10% of the training set in both groups “A” and “B”, IMC-COR-sep has lower average error than the IMC-COR-all which has a larger number of parameter to estimate. Moreover, it is observed that in most of the cases the COR-sep and COR-all produce similar results, while concerning the R2 construction the R2-sep performs better than the R2-all in most of the cases. Last, in figure 2.5 we report results for the “A” group, which

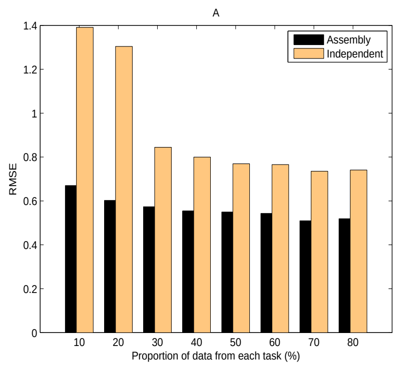


Figure 2.5: Comparison between different standardization procedures, “Assembly” represents results produced by standardizing all task together and “Independent” represents independent standardization.

were obtained by standardizing each task independent of the others (Independent) and by standardizing all tasks together as in all previous experiments (Assembly), which shows a clear advantage of the Assembly method.

2.4 Conclusions

This chapter presented an overview of Multi-task algorithms for GP regression. We discussed the transfer learning mechanisms of each method, which resulted into a weak and a strong form of transfer within the GP framework. We made connections of the framework of multi-task learning with the framework of link analysis, and we have stressed the importance of the IMC model of Bonilla et al. (2008). In the last section, we provided empirical proof that, multi-task learning can significantly improve upon single-task learning for small training sets. On top of that, the experimental section validated empirically our classification of multi-task learning algorithms, showing the gradual improvement in performance from the Parameter to the Inductive transfer. Last but not least, we showed on a real data set the benefits the framework of multi-task learning has to offer and we believe that it can have a significant impact to the problem of predicting pK_a values since it can act as a decision support system for experimental evaluation.

Chapter 3

Multi-task Learning with Gaussian Processes for classification

Having discussed Multi-task GP regression in the previous chapter, we now turn our attention to multi-task classification problems. Multi-task learning (Caruana, 1997) has been a subject of intense research in the machine learning community in recent years. The frequent occurrence of multiple, related learning tasks in real-world problems has motivated the development of machine learning approaches capable of capturing the similarity between tasks and hence leverage any information transfer between them (Bakker and Heskes, 2003; Jebara, 2004; Yu et al., 2005; Bonilla et al., 2007; Argyriou et al., 2008; Yu et al., 2007b). From a motivational point of view, multi-task learning can be particularly useful in situations where a limited amount of data is available in each task, while data from many tasks are readily at hand. For example, predictions in the biomedical field suffer from a massive sample heterogeneity problem: samples from many individuals are often available, but the underlying distribution of the data from each individual may be different, so that simply pooling all data together may be inappropriate. Intuitively, a multi-task approach should be able to avoid this problem by retaining the similarities between the different samples while keeping into account the differences between them. Although it is difficult to specify conditions under which a multi-task approach guarantees an increase in performance (Baxter, 2000; Ben-David and Borbely, 2008; Chai, 2009), empirically it has been shown in many examples that transfer learning does indeed happen, leading to improved results over models trained on the individual tasks.

From a Bayesian perspective, multi-task learning can be simply implemented in a hierarchical fashion, whereby different models trained on separate tasks are joined by placing a common prior distribution over the model parameters (Lawrence and Platt, 2004; Schwaighofer et al., 2005; Yu et al., 2005). While these approaches can effectively capture global similarities between tasks, it limits the influence of the multi-task setting to determining some common hyperparameters, hence providing a relatively inflexible model. An attractive alternative is to model directly the correlations between tasks via a task covariance which can be learned from the data. This approach was recently proposed by Bonilla et al. (2008) in the framework of Gaussian Process (GP) regression, where the overall covariance structure of the data was specified as a Kronecker product of an input-covariance function and a task correlation matrix.

In this chapter, we extend the results of Bonilla et al. (2008) to the classification scenario by using a probit noise model as the output likelihood. The introduction of the non-linear likelihood makes exact inference impossible, leading to the need for approximating techniques. While many of these techniques are by now part of the standard machine learning repertoire, there are a number of important classification problems where a multi-task approach could give significant advantages. We first present a fully Bayesian treatment of the model in the fully observed case, that is equivalent to multi-label classificaiton. In our experience, this approach however presented convergence problems in the most typical multi-task scenario when most responses are missing. We therefore propose two approximate inference approaches obtained by adapting the popular variational Bayes and expectation propagation frameworks to the multi-task Gaussian process classification model, where marginalisation over unobserved latent functions causes no problems. Experiments on a toy data set show that both approaches provide an excellent approximation to the true posterior distribution.

Although, a thorough introduction to MT GP regression and to the IMC model of Bonilla et al. (2008) was given in the previous chapter since every chapter is self contained, in the next section, we briefly review multi-task Gaussian Process regression and introduce our classification model. We then discuss the three inference approaches considered: Gibbs sampling, Expectation-Propagation (EP) and Variational Bayes (VB). We continue by discussing the issue of transfer learning in our model, *i.e.* how information can flow from one task to another during the learning process. Finally in the last section, we present results on three data sets, a synthetic benchmark problem used previously in multi-task classification by Liu et al. (2009) as well as two real data sets. Last note that in relation to the classification of Transfer learning algorithms presented in chapter 1 the content of this chapter as of the previous fall into the Inductive category.

3.1 Multi-task Gaussian Processes

3.1.1 Multi-task Regression

GPs are a popular non-parametric Bayesian tool for regression and classification; at the core of GP prediction is the *covariance function* or kernel, capturing the output covariance at different pairs of input points. In the regression case, observations are generally assumed to be noisy corrupted versions of an underlying stochastic process f depending on the input variable x

$$y_j \sim \mathcal{N}\left(f_j, \sigma^2\right)$$
$$\mathbf{f} \sim \mathcal{N}(\mu, K)$$

where $\mu(x)$ is the mean function and $K_{ij} = k(x_i, x_j)$ is the covariance function, capturing the input dependence of the target statistics.

In the multi-task setting we are interested in learning M related functions f_j for $j = 1, \ldots, M$, from training data x_{ij}, y_{ij} $i = 1, \ldots, n_j$, with $x \in \mathbb{R}^d$, and $n_1 + \ldots + n_M = N$. As usual in GP regression, we assume the following noise model

$$y_{ij} = f_j(x_{ij}) + \varepsilon_j, \text{ with } \varepsilon_j \sim \mathcal{N}(0, \sigma_j^2), \tag{3.1}$$

where y_{ij} (x_{ij}) denotes the i^{th} output (input) of the j^{th} task. Let us consider the vector $\mathbf{y}$ of *complete responses* obtained by stacking the response in *all* tasks to each input point, such that $\mathbf{y} = \text{vec}(\mathbf{Y})$, where $\mathbf{Y} \in \mathbb{R}^{N \times M}$. Of course, in most applications not all entries of this vector will be observed; given the probabilistic nature of GPs, it is straightforward to treat the missing values. Let $\mathbf{f}$ be the latent function values corresponding to the complete responses, again stacked in a single vector ($\mathbf{f} = \text{vec}(\mathbf{F}^T)$) with $\mathbf{F} \in \mathbb{R}^{N \times M}$. Unless specified otherwise, bold letters will be used to denote vectors and plain letters for single variables. Bonilla et al. (2008) encapsulate the multi-task regression problem by selecting the following form for the GP prior over the latent functions

$$p(\mathbf{f}|X) \sim \mathcal{GP}(0, K^t \otimes K^x), \tag{3.2}$$

where $\otimes$ is the Kronecker product, and $K^t \in \mathbb{R}^{M \times M}$ and $K^x \in \mathbb{R}^{N \times N}$ are the task and data covariance matrix respectively. Of central importance is the task covariance matrix which can either be defined by a task covariance function $\mathbf{k}^t(.,.)$ when task-descriptor features x^t are available (Bonilla et al., 2007), or be a free form covariance matrix, specifying inter task correlations (Bonilla et al., 2008). Note that expressing the covariance function of the prior as a Kronecker product has been known in the geostatistics community as the "*Intrinsic Model of Coregionalization*" (IMC) (Cressie, 1993).

The noise model becomes $p(\mathbf{y}|\mathbf{f}) \sim \mathcal{N}(\mathbf{f}, D \otimes I)$, with $D_{M\times M}$ diagonal with $D_{jj} = \sigma_j^2$, and $I_{N\times N}$. Hyperparameters θ^t and θ^x appear both in the task and data covariance functions; they can be estimated within this framework by maximising the evidence or log marginal likelihood. This can be done either by standard gradient descent or using an E-M type algorithm exploiting the Kronecker factorisation of the prior (*cf* Bonilla et al. (2008), section 2.2).

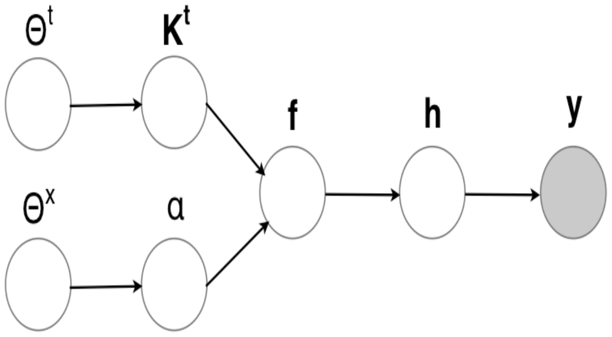


Figure 3.1: General Multi-task Probit model. Variables $\mathbf{y}$ represent the outputs, variables $\mathbf{h}$ and $\mathbf{f}$ represent the auxiliary and the latent function variables over the tasks respectively. Variables K^t, α, θ^t and θ^x are used to denote prior distributions of parameters of the task and data covariance function (see section 3.1.2 for more details).

3.1.2 Multi-task GP Classification

A key asset of GP regression is the conjugacy of the Gaussian noise model (equation 3.1) with the GP prior (equation 3.2), enabling analytical marginalisation of the latent variables. In the classification setting, the target values are discrete, and no such conjugacy is available. In the rest of this chapter, we focus on binary classification employing the probit model (Csató et al., 2000); generalisation to the multi-class probit model or to other binary noise models is in principle straightforward (Girolami and Rogers, 2006)[1].

[1]We observe in passing that Bonilla et al. (2008) also addressed a multi-class classification problem by treating it as a regression problem with Gaussian noise. While this may have been reasonable on the specific data set considered, its general applicability as a classification method is questionable.

The graphical model of figure 3.1 illustrates the dependencies between the variables of a general multi-task probit classification model. In the general case the joint likelihood factorises as:

$$p(\mathbf{y},\mathbf{h},\mathbf{f},K^t,\alpha|\theta^t,\theta^x) = p(\mathbf{y}|\mathbf{h})p(\mathbf{h}|\mathbf{f})p(\mathbf{f}|K^t,\alpha)p(\alpha|\theta^x)p(K^t|\theta^t). \tag{3.3}$$

The relationship between outputs y and the auxiliary variable h is deterministic and given by $p(y_i|h_i) = \theta(y_i h_i)$ for $y_i = \pm 1$, where θ is one if its argument is positive, zero otherwise (*i.e.*, the Heaviside function). The auxiliary variable h_i is given a normal distribution with mean given by f_i and variance 1,

$$p(h_i|f_i) = \mathcal{N}(f_i, 1) \tag{3.4}$$

which leads to the probit model (Albert and Chib, 1993). The latent variable $\mathbf{f}$ integrates the information coming from the data and the tasks through a matrix variate normal distribution with zero mean and covariance given by $K^t \otimes K^x$, as in Bonilla et al. (2008). The matrix K^t is the task covariance matrix encoding information about the tasks, while matrix K^x is the data covariance matrix encoding information coming from the data of each task. Parameters of the prior distribution over the task covariance matrix are denoted by θ^t. Parameters of the data covariance function $K^x(.,.)$ are denoted by α, with θ^x being the associated prior hyperparameters. In the rest of the chapter, we choose a free-form for the task covariance matrix, while the data covariance matrix will be given by an Automatic Relevance Determination (ARD) (Rasmussen and Williams, 2005), *i.e.* a squared exponential covariance with diagonal matrix.

3.2 Inference in multi-task GP classification

Coupling the probit likelihood with a GP prior on the latent functions f, results into a non-Gaussian posterior distribution $p(\mathbf{f}|\mathbf{y})$, making exact inference impossible. In this section, we present three inference approaches to solve this problem. First we present a solution based on Gibbs sampling; this is asymptotically exact but computationally intensive. We then employ two deterministic methods to approximate the non-Gaussian posterior: the EP approximation (Opper and Winther, 2000; Minka, 2001; Rasmussen and Williams, 2005), and a variation of the methods proposed in Csató et al. (2000) and Girolami and Rogers (2006), based on probit regression with a variational EM algorithm. The hyperparameters of the task and data covariance function in the EP and Variational probit regression models were estimated by type II Maximum likelihood (point estimates) for computational efficiency. Incorporating hyper-priors in a variational GP-probit model is possible (Girolami and Rogers, 2006), and extension to the multi-task setting should be straightforward.

3.2.1 Bayesian Inference: Gibbs Sampling

MCMC sampling methods are often employed in Bayesian models as they provably sample from the true posterior distribution in the limit of infinite samples. The Gibbs sampling scheme is a particular type of Markov chain simulation where samples are drawn iteratively from the posterior of a subset of the variables conditioned on all others. A brief introduction to Gibbs sampling and the Metropolis-Hastings (MH) algorithm is provided in appendix A.1, while for a complete treatment on sampling methods the interested reader is referred to Gelman et al. (2004).

To complete the specification of our model in equation 3.3 we need to define prior distributions over the parameters of the task and data covariance matrices. The hyperparameters α of the ARD data covariance are given a log-normal distribution to ensure positivity,

$$p(\alpha|\theta^x) = \log\mathcal{N}(\alpha|\mu_x, \sigma_x^2),$$

where hyper-hyperparameters $\theta^x = \{\mu_x, \sigma_x^2\}$ are fixed to yield reasonably uninformative priors. The covariance matrix over the tasks K^t is given an Inverse Wishart prior probability distribution to ensure conjugacy

$$K^t \sim \mathcal{IW}_M(\beta, \Lambda),$$

where β is the number of degrees of freedom, and Λ is a $M \times M$ positive definite matrix which is known as the parameter matrix (Gupta and Nagar, 2000).

Due to conjugacy, the conditional posterior distribution of $\mathbf{h}, \mathbf{f}, K^t$ can be found analytically. In contrast, no conjugate prior distribution is available for the ARD length-scales α_i and individual MH sub-samplers must be employed to draw samples from their conditional posterior. The update equations of the conditional posteriors are given in the following section 3.2.1.1. It is worth noticing that the MCMC sampling scheme could further be accelerated by following recent advances in sampling for latent Gaussian models (Titsias et al., 2009; Murray et al., 2010).

3.2.1.1 Gibbs Sampler

The posterior of the auxiliary variables **h** conditioned on all other variables is given by

$$p(\mathbf{h}|\mathbf{f},\mathbf{y}) = \prod_{n=1}^{N} \left(f_n + y_n \frac{\mathcal{N}_{h_n}(f_n,1)}{\Phi(y_n f_n)} \right), \tag{3.5}$$

resulting in a product of truncated univariate Gaussians.

Individual MH sub-samplers are employed to sample from the full conditional distribution of the hyper-parameters α as

$$\begin{aligned} p(\alpha|\mathbf{f},\theta^x) &\propto p(\mathbf{f}|\alpha,K^t)p(\alpha|\theta^x) \\ &\propto \mathcal{N}_{\mathbf{f}}(0,K^t \otimes K^x_\alpha)\log\mathcal{N}(\alpha|\mu_x,\sigma_x^2). \end{aligned} \tag{3.6}$$

At each time step a proposed sample α^* is accepted with probability $A(\alpha^*,\alpha)$, where

$$A(\alpha^*,\alpha) = \min\left(1, \frac{\mathcal{N}_{\mathbf{f}}(0,K^t \otimes K^x_{\alpha^*})\log\mathcal{N}(\alpha^*|\mu_x,\sigma_x^2)}{\mathcal{N}_{\mathbf{f}}(0,K^t \otimes K^x_{\alpha})\log\mathcal{N}(\alpha|\mu_x,\sigma_x^2)}\right). \tag{3.7}$$

Continuing the posterior of the latent variables **f** follows as

$$p(\mathbf{f}|\mathbf{h},K^t,\alpha) = \mathcal{N}(\Sigma\mathbf{h},\Sigma), \tag{3.8}$$

where $\Sigma = (K^t \otimes K^x)(I + K^t \otimes K^x)^{-1}$. The posterior of the task covariance matrix K^t is given by

$$p(K^t|\mathbf{f},\theta^t) = \frac{p(\mathbf{f}|K^t)p(K^t|\theta^t)}{Z_{K^t}}, \tag{3.9}$$

where using that $\mathbf{f} = \text{vec}(\mathbf{F})$, the prior of the latent function f can be written as:

$$p(\mathbf{f}|K^t) = \frac{\exp\left\{-\frac{1}{2}\text{trace}\left((K^t)^{-1}\mathbf{F}^T(K^x)^{-1}\mathbf{F}\right)\right\}}{(2\pi)^{\frac{1}{2}NM}|K^t|^{\frac{1}{2}N}|K^x|^{\frac{1}{2}M}}. \tag{3.10}$$

Combining the prior over the latent values (equation 3.10) with the prior of the task covariance matrix given by,

$$p(K^t|\theta^t = \beta,\Lambda) = \frac{2^{-\frac{1}{2}(\beta-M-1)M}|\Lambda|^{\frac{1}{2}(\beta-M-1)}}{\Gamma_M\left[\frac{1}{2}(\beta-M-1)\right]|K^t|^{\frac{1}{2}\beta}}\text{etr}\left\{-\frac{1}{2}K^{t^{-1}}\Lambda\right\}, \tag{3.11}$$

we get the posterior of the task covariance matrix again as an Inverse Wishart distribution:

$$p(K^t|\mathbf{f},\theta^t) = \frac{2^{-\frac{1}{2}(\beta_N-M-1)M}|\Lambda_N|^{\frac{1}{2}(\beta_N-M-1)}}{\Gamma_M\left[\frac{1}{2}(\beta_N-M-1)\right]|K^t|^{\frac{1}{2}(\beta_N)}}\text{etr}\left\{-\frac{1}{2}(K^t)^{-1}(\Lambda_N)\right\}, \tag{3.12}$$

where $\Lambda_N = \mathbf{F}^T(K^x)^{-1}\mathbf{F} + \Lambda$, $\beta_N = \beta + N$, and Γ_M is the multivariate Gamma function and we have used the shorthand "etr" to denote the exponential of the trace, etr $=$ $\exp\{\text{trace}(.)\}$. It is worth taking a moment examining the form of the parameter matrix Λ_N of the posterior of the task covariance matrix. Λ_N is computed from $\mathbf{F}^T(K^x)^{-1}\mathbf{F} + \Lambda$. This representation assumes that the complete set of responses is available, where each column of the $\mathbf{F}$ or $\mathbf{Y}$ matrix represents the latent values or targets for each input at that task. In most real world applications though, it is desirable to combine data from different tasks for which outputs of the other tasks are not observed. Latent values for outputs that are not observed, for which we will refer to as f^u, need special treatment that we discuss in the following section. A possible solution would be to develop an EM algorithm to estimate the missing values, as was done in Bonilla et al. (2008) with Gaussian processes or in Yu et al. (2007b) with t-processes. The implications this situation brings are investigated in the following section, along with a solution to this problem. More information about the conditional posterior distributions of the variables $\mathbf{h}$, $\mathbf{f}$, K^t, can be found in appendix A.2.

The predictive distribution for a new point x_*, for task j, will be given by:

$$p(y_{*j}|x_*, \mathbf{X}, \mathbf{y}, \mathbf{h}) = \Phi\left(\frac{\nu_{*j}}{\sqrt{1+\sigma^2_{*j}}}\right), \tag{3.13}$$

where

$$\nu_{*j} = \left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{X},x_*}\right)^T \left(I + K^t \otimes K^x\right)^{-1}\mathbf{h} \tag{3.14}$$

$$\sigma^2_{*j} = k^t_{jj}k^x_{x_*x_*} - \left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{X},x_*}\right)^T \left(I + K^t \otimes K^x\right)^{-1}\left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{X},x_*}\right), \tag{3.15}$$

with $\mathbf{k}^t_j$ to denote the j^{th} column of K^t, $\mathbf{k}^x_{\mathbf{X},x_*}$ to denote the vector of covariances between the training points $\mathbf{X}$ and the test point x_*, and $k^x_{x_*x_*}$ as the variance of the test point.

3.2.1.2 Treatment of Missing Values

Consistency is an appealing feature of GPs, which enables latent function evaluations corresponding to missing values to be marginalised effortlessly. In the multi-task situation, however, data are more structured, in that the posterior of the task covariance whose parameter matrix, given by $\Lambda_N = \mathbf{F}^T(K^x)^{-1}\mathbf{F} + \Lambda$, requires all latent function evaluations; those that correspond to the observed and to the unobserved outputs. Considering the common situation in a multi-task scenario where for each data point the output of one task is observed we can construct matrix $\mathbf{F} \in \mathbb{R}^{N\times M}$ as:

$$\mathbf{F} = \begin{bmatrix} f^o_{1n_1} & f^u_{2n_1} & \cdots & \cdots & f^u_{Mn_1} \\ f^u_{1n_2} & f^o_{2n_2} & f^u_{3n_2} & \cdots & f^u_{Mn_2} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ f^u_{1n_M} & f^u_{2n_M} & \cdots & \cdots & f^o_{Mn_M} \end{bmatrix}, \tag{3.16}$$

where for example $f^o_{2n_2}$ is the $(n_2 \times 1)$ vector of latent values associated with the observed outputs of task 2 and inputs X_2, while $f^u_{2n_1}$ is the $(n_1 \times 1)$ vector of latent values associated with the unobserved outputs of task 2 for inputs X_1; additionally we have used X_j and n_j to denote the set and number of inputs from each task and $N = \sum_{j=1}^{M} n_j$.

One way of tackling this problem is to modify the sampling scheme described in the previous section to draw samples for all latent functions $f^u_{jn_j}$ that correspond to the unobserved latent functions. In the following we will use $\mathbf{f}^o = [f^o_{1n_1}; f^o_{2n_2}; \dots; f^o_{Mn_M}]$ to denote the vector of all latent function evaluations that correspond to the observed outputs, and by $\mathbf{f}^u = [f^u_{1n_2}; \dots f^u_{1n_M}; \dots; f^u_{Mn_1}; \dots; f^u_{Mn_{M-1}}]$ to denote the vector of all latent function evaluations that correspond to the unobserved outputs; the length of the observed latent functions $\mathbf{f}^o$ will be equal to N, $\mathbf{f}^o \in \mathbb{R}^{N\times 1}$, as that of the auxiliary variables $\mathbf{h} \in \mathbb{R}^{N\times 1}$ and of the outputs $\mathbf{y} \in \mathbb{R}^{N\times 1}$, while the length of the unobserved latent functions will be equal to $N(M-1)$, thus $\mathbf{f}^u \in \mathbb{R}^{N(M-1)\times 1}$. The complete set of latent functions $\mathbf{f}^o$ and $\mathbf{f}^u$ are jointly Gaussian, where by partitioning and reordering the covariance matrix $K^t \otimes K^x \in \mathbb{R}^{MN\times MN}$ into the rows and columns that correspond to the observed and unobserved latent functions we can write,

$$\begin{bmatrix} \mathbf{f}^o \\ \mathbf{f}^u \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{oo} & K_{ou} \\ K_{ou}^T & K_{uu} \end{bmatrix} \right), \tag{3.17}$$

where $K_{oo} \in \mathbb{R}^{N\times N}$ and $K_{uu} \in \mathbb{R}^{(M-1)N\times(M-1)N}$ is the covariance matrix between the observed and the unobserved latents respectively, and $K_{ou} \in \mathbb{R}^{N\times(M-1)N}$ are the cross correlations between the observed and the unobserved latent functions.
The conditional distributions of $\mathbf{f}^o|\mathbf{f}^u$, and $\mathbf{f}^u|\mathbf{f}^o$ will be given by:

$$\begin{aligned} p(\mathbf{f}^o|\mathbf{f}^u, K^t, \alpha) &= \mathcal{N}\left(K_{ou}K_{uu}^{-1}\mathbf{f}^u, K_{oo} - K_{ou}^T K_{uu}^{-1} K_{ou}\right) \\ &= \mathcal{N}(\mu^o_c, \Sigma^o_c), \end{aligned} \tag{3.18}$$

$$\begin{aligned} p(\mathbf{f}^u|\mathbf{f}^o, K^t, \alpha) &= \mathcal{N}\left(K_{ou}^T K_{oo}^{-1}\mathbf{f}^o, K_{uu} - K_{ou}K_{oo}^{-1}K_{ou}^T\right) \\ &= \mathcal{N}(\mu^u_c, \Sigma^u_c), \end{aligned} \tag{3.19}$$

where the dependence of $\mathbf{f}^o$ and $\mathbf{f}^u$ on K^t and α is hidden in the covariance matrices K_{oo}, K_{uu} and K_{ou}; in the following we will drop the dependence on α to keep the notation light. In that situation the posterior of the observed latent functions $\mathbf{f}^o$ conditioned on all other variables will be given by:

$$\begin{aligned}
p(\mathbf{f}^o|\mathbf{h},\mathbf{f}^u,K^t) &= \frac{p(\mathbf{h}|\mathbf{f}^o)p(\mathbf{f}^o,\mathbf{f}^u|K^t)p(K^t|\theta^t)}{\int p(\mathbf{h},\mathbf{f}^o,\mathbf{f}^u,K^t)\mathrm{d}\mathbf{f}^o}, \\
&= \frac{p(\mathbf{h}|\mathbf{f}^o)p(\mathbf{f}^o,\mathbf{f}^u|K^t)p(K^t|\theta^t)}{\int p(\mathbf{h}|\mathbf{f}^o)p(\mathbf{f}^o,\mathbf{f}^u|K^t)p(K^t|\theta^t)\mathrm{d}\mathbf{f}^o}, \\
&= \frac{p(\mathbf{h}|\mathbf{f}^o)p(\mathbf{f}^o|\mathbf{f}^u,K^t)p(\mathbf{f}^u|K^t)}{\int p(\mathbf{h}|\mathbf{f}^o)p(\mathbf{f}^o|\mathbf{f}^u,K^t)p(\mathbf{f}^u|K^t)\mathrm{d}\mathbf{f}^o}, \\
&\propto p(\mathbf{h}|\mathbf{f}^o)p(\mathbf{f}^o|\mathbf{f}^u,K^t), \\
p(\mathbf{f}^o|\mathbf{h},\mathbf{f}^u,K^t) &= \mathcal{N}(\mu_{f^o},\Sigma_{f^o}),
\end{aligned} \tag{3.20}$$

where $\Sigma_{f^o} = (I + (\Sigma_c^o)^{-1})^{-1}$ and $\mu_{f^o} = \Sigma_{f^o}(\mathbf{h} + \mu_c^o)$. Continuing, the posterior of the unobserved latent functions $\mathbf{f}^u$ has no dependence on $\mathbf{h}$, and is simply given by:

$$p(\mathbf{f}^u|\mathbf{f}^o,K^t) = \mathcal{N}(\mu_c^u,\Sigma_c^u). \tag{3.21}$$

The posterior distribution of the auxiliary variables $\mathbf{h}$ is independent of the $\mathbf{f}^u$ variables which similarly to equation 3.5 gives that $p(\mathbf{h}|\mathbf{f}^o,\mathbf{y}) = \prod_{n=1}^N \left(f_n^o + y_n \frac{\mathcal{N}_{h_n}(f_n^o,1)}{\Phi(y_n f_n^o)} \right)$. Using samples from the observed and the unobserved latent functions it is possible to reconstruct the $\mathbf{F}$ matrix and vector $\mathbf{f} = \mathrm{vec}(\mathbf{F})$ in order to draw samples from the posterior of the task covariance matrix and for the hyperparameters α in the Metropolis-Hastings algorithm.

However, the above sampling scheme did not produce satisfactory results in the estimation of the posterior of the hyperparameters α, where it was observed that the MH sampler was not converging. It should be noted that in the Multi-task setting this is a rather difficult inference problem since it requires the estimation of the posterior of $(M-1)N$ latent function evaluations that correspond to unobserved outputs from the N latent functions of the observed outputs. Due to these problems this setting will not be tested in any real or simulated data in the experimental part of this chapter.

3.2.2 Approximate Inference

3.2.2.1 EP approximation

In this section we outline the EP approximation for multi-task GP classification, for a complete treatment of EP for the conventional GP classifier, see Rasmussen and Williams (2005). In this case, we will marginalise the auxiliary variable h and work directly with the likelihood given by $p(y|f) = \Phi(yf) = \int_{-\infty}^{yf} \mathcal{N}(u|0,1)\,\mathrm{d}u$. Hyperparameters of the input and task covariance matrix are given point estimates via type II maximum likelihood, so that nodes K^t and α are treated as parameters and not as random variables. To harmonise the notation we will now denote these parameters as θ^x and θ^t.

The posterior over the latent variables, $p(\mathbf{f}|\mathbf{y})$, is proportional to a product of the prior and the factorised non-Gaussian likelihoods. EP approximates these non-Gaussian likelihoods by a product of un-normalised Gaussians $t_i(f_i)$,

$$\prod_{i=1}^{N} p(y_i|f_i) \simeq \prod_{i=1}^{N} t_i(f_i|\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) = \mathcal{N}(\tilde{\mu}, \tilde{\Sigma}) \prod_{i=1}^{N} \tilde{Z}_i \tag{3.22}$$

where $\tilde{\mu}$ is a vector of $\tilde{\mu}_i$'s, and $\tilde{\Sigma}$ is diagonal with $\tilde{\Sigma}_{ii} = \tilde{\sigma}_i^2$. Which leads to the approximated distribution of the latent variables $q(\mathbf{f}|\mathbf{y})$

$$q(\mathbf{f}|\mathbf{y}) = \frac{1}{Z_{EP}} p(\mathbf{f}) \prod_{i=1}^{n} t_i\left(f_i|\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2\right) = \mathcal{N}(\mu, \Sigma), \tag{3.23}$$

where $\mu = \Sigma\tilde{\Sigma}^{-1}\tilde{\mu}$, $\Sigma = \left[\left(K^t \otimes K^x\right)^{-1} + \tilde{\Sigma}^{-1}\right]^{-1}$, and Z_{EP} is approximated by EP marginal likelihood.

EP updates the individual scaled functions t_i sequentially by termwise refinement. At each step the current t_i is left out, giving rise to the cavity distribution $q_{-i}(f_i)$ which is then combined with the true likelihood $p(y_i|f_i)$, to give a non-Gaussian distribution. A Gaussian approximation is chosen to approximate the non-Gaussian distribution from the previous step, which is then used to compute the parameters of t_i by moment matching between the two distributions.

Hyperparameters can be estimated by approximate type II maximum likelihood using the EP log marginal likelihood. The gradients with respect to hyperparameters θ_x and θ_t are given by:

$$\frac{\partial log Z_{EP}}{\partial \theta_j} = \frac{1}{2}\tilde{\mu}^T (K^t \otimes K^x + \tilde{\Sigma})^{-1} \Omega (K^t \otimes K^x + \tilde{\Sigma})^{-1} \tilde{\mu} - \frac{1}{2}\mathrm{tr}\left((K^t \otimes K^x + \tilde{\Sigma})^{-1}\Omega\right), \tag{3.24}$$

and

$$\Omega = \begin{cases} K^t \otimes \frac{\partial K^x}{\partial \theta^j} & \text{if } \theta^j = \theta^x \\ \\ \frac{\partial K^t}{\partial \theta^j} \otimes K^x & \text{if } \theta^j = \theta^t \end{cases} \tag{3.25}$$

The predictive distribution of a new point is computed by equation 3.13 as in section 3.2.1.1, but with ν_{*j}, σ^2_{*j} given by,

$$\nu_{*j} = \left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{X},x_*}\right)^T \left(K^t \otimes K^x + \tilde{\Sigma}\right)^{-1} \tilde{\mu} \tag{3.26}$$

$$\sigma^2_{*j} = k^t_{jj} k^x_{x_* x_*} - \left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{x},x_*}\right)^T \left(K^t \otimes K^x + \tilde{\Sigma}\right)^{-1} \left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{x},x_*}\right), \tag{3.27}$$

3.2.2.2 Variational Probit Regression

We now employ a variational approximation in the spirit of Girolami and Rogers (2006). We refer to the model in figure 3.1 where again hyperparameters are estimated via type II maximum likelihood. A variational treatment of this problem involves approximating posteriors over latent variables in an ensemble of factored posteriors of the form

$$p(\Theta|\mathbf{y}, X, t, \theta^t, \theta^x) \approx \prod_{i=1} Q(\Theta_i) = Q(\mathbf{f})Q(\mathbf{h}). \tag{3.28}$$

It can be shown that the lower bound on the log marginal likelihood $\log p(\mathbf{y}|X, \theta^t, \theta^x) \geq \mathbb{E}_{Q(\Theta}\{\log\ p(\mathbf{y}, \mathbf{f}, \mathbf{h}|X)\} - \mathbb{E}_{Q(\Theta}\{\log\ Q(\Theta)\}$ is maximised by distributions of an unnormalized form

$$Q(\Theta_i) \propto \exp(\mathbb{E}_{Q(\Theta\setminus\Theta_i)}\{\log p(y, \Theta|X, \theta^t, \theta^x)\}), \tag{3.29}$$

where $Q(\Theta \setminus \Theta_i)$ denotes the factorised distribution with the i^{th} component removed.

A variational EM algorithm is derived, where in the E-step the expectations of the variational parameters are computed, while in the M-step the hyperparameters θ^t, θ^x are optimised given the expectations computed in the previous step. The lower bound on the log marginal likelihood is given by (see appendix A.3.2 for details),

$$\mathcal{L}(Q) = \mathbb{E}_{Q(\Theta)}[\log p(\mathbf{y}, \mathbf{h}, \mathbf{f}|X, \theta^t, \theta^x)] - \mathbb{E}_{Q(\Theta)}[\log Q(\mathbf{h})Q(\mathbf{f})] \tag{3.30}$$

$$= \sum_{i=1}^{N} \log z_i - \frac{1}{2}\log|I + K^t \otimes K^x| - \frac{1}{2}\tilde{\mathbf{f}}^T (K^t \otimes K^x)^{-1}\tilde{\mathbf{f}}, \tag{3.31}$$

where Z_i is the normalisation constant of approximating distribution $Q(\mathbf{h})$, given by $Z_i = \Phi(y_i \tilde{f}_i)$.

3.2.2.2.1 E-step : Compute the sufficient statistics for each variational parameter $Q(\Theta_i)$ given by equation (3.29), based on current θ^t, θ^x .

$$Q(\mathbf{f}) \propto \exp\{\mathbb{E}\{\log p(\mathbf{h}|\mathbf{f}) + \log p(\mathbf{f}|X)\}\} = \mathcal{N}_{\mathbf{f}}(\tilde{\mathbf{f}}, \Sigma), \tag{3.32}$$

$$Q(\mathbf{h}) \propto \exp\{\mathbb{E}\{\log p(\mathbf{y}|\mathbf{h}) + \log p(\mathbf{h}|\mathbf{f})\}\} = \prod_{i=1}^{N} \frac{\mathcal{N}_{h_i}(\tilde{f}_i, 1)}{\Phi(y_i \tilde{f}_i)} \tag{3.33}$$

where $\tilde{\mathbf{f}} = \Sigma\tilde{\mathbf{h}}$, and $\Sigma = K^t \otimes K^x(I + K^t \otimes K^x)^{-1}$.

3.2.2.2.2 M-step : Optimise θ^t, θ^x based on the last E-step.

$$\frac{\partial \mathcal{L}(Q)}{\partial \theta^j} = -\frac{1}{2}\text{trace}((I + K^t \otimes K^x)^{-1}\Omega) + \frac{1}{2}\tilde{\mathbf{f}}^T(K^t \otimes K^x)^{-1}\Omega(K^t \otimes K^x)^{-1}\tilde{\mathbf{f}}, \tag{3.34}$$

where Ω is given in equation (3.25), as in the EP approximation.

The predictive equation for the Variational probit model is the same as for the Gibbs sampling scheme and is given in equation 3.13. More details about the Variational probit model of this section can be found in appendix A.3.1.

3.3 Transfer of knowledge & Covariance structure

3.3.1 Task covariance matrix

Learning the covariance structure of the tasks is performed either by drawing samples as in section 3.2.1 from the posterior of the task covariance matrix or through optimisation of the hyperparameters as in sections 3.2.2.1,3.2.2.2, and is an important component of the method proposed here.

The task covariance matrix can either come from a covariance function, when task descriptor features are available, a case investigated by Bonilla et al. (2007) and Yu et al. (2007a) in different contexts, or it can have a free form as in Bonilla et al. (2008). In the case when a free form covariance matrix is used, K^t acts as a correlation matrix capturing the dependencies between the tasks. Despite the “free form” structure of K^t, positive definiteness restrictions must be retained. Bonilla et al. (2008) achieved positive semidefinite guarantees by parameterising the lower triangular matrix L of the Cholesky decomposition $K^t = LL^T$.

If we want to restrict the task covariance matrix to be a *correlation* matrix of the tasks, except from the symmetric and positive definite restrictions, it has to have a unit diagonal. This restriction, easy as it may seem, poses certain difficulties and many attempts have been made to solve this constrained optimisation problem. A solution to this problem is presented in Rebonato and Jäckel (2000), where the correlation matrix is still decomposed as $K^t = BB^T$, with B $(M \times M)$, and each row vector of B can be viewed as coordinates lying in the unit hypersphere. If we denote by b_{ij} the elements of the matrix B, then these $M \times M$ coordinates are obtained from $M \times (M-1)$ angular coordinates θ^t_{ij} by:

$$b_{ij} = \begin{cases} \cos\theta_{ij} \prod_{k=1}^{j-1} \sin\theta_{ik} & \text{for } \ j = 1,\dots,M-1 \\ \\ \prod_{k=1}^{j-1} \sin\theta_{ik} & \text{for } \ j = M \end{cases}$$

A possible drawback of this setup would be that the number of correlation parameters θ_t that need to be estimated is given by, $M^2 - M$, which grows quadratically with the number of tasks M. It is interesting to note that, if the resulting off-diagonal elements of K^t are different from zero, then samples from one task will affect predictions of the other tasks, as the mean of the predictive distribution of the j^{th} task, given by

$$\mathbb{E}\left[p(y_{j*}|\mathbf{y})\right] = (\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{X},x_*})^T \Sigma^{-1} \mathbf{y}, \tag{3.35}$$

is computed by weighting observations from task i, where $i \neq j$, by the i^{th} element of k^t_j. Similar regularisation is performed in the computation of the variance of the predictive distribution. Therefore, the task covariance matrix acts as a transfer of knowledge between tasks.

3.3.2 Input covariance function

The choice of the covariance function of the inputs in single task training depends on the task of interest, and even similar tasks can be trained with different covariance functions. In this specific multi task scenario that we are adopting, the covariance function hence the hyperparameters are shared amongst the tasks. While this reduces somewhat the flexibility of the model, performing the optimisation of the hyperparameters using data from all tasks results in a regularised optimal solution, which can be seen as bias selection (Baxter, 2000). Furthermore, if the ARD covariance function (Neal, 1996) is used, then an optimal subset of features is estimated, which is shared across the tasks.

Thus the transfer of knowledge between tasks in this method is the result of two sources of information. The first is through the task covariance matrix, whose operation on the input covariance matrix results in samples from one task affecting predictions of other tasks. The second stems from the optimisation of the shared hyperparameters of the input covariance function, as a task-regularised optimal solution.

3.4 Experimental Results

Evaluation of the proposed multi-task framework is performed on a synthetic data set and two real data sets. Throughout all of the experiments, we use the Automatic Relevance Determination (ARD) covariance function (Rasmussen and Williams, 2005) for the input covariance K^x, and a free form correlation matrix (Rebonato and Jäckel, 2000) for the task covariance matrix K^t. We report results obtained both with a VB and EP[2] approximation to the posterior distribution, except in one of the real data sets, where the VB algorithm proved very slow due to the size of the data set. Numerical inaccuracies in the EP approximation were corrected by using Rasmussen's solve_chol.m[3] function, which solves linear equations from the Cholesky factorisation, while Rasmussen's excellent optimisation routine minimize.m[4] was used in some of the simulations for the optimization of the hyperparameters.

In order to check that there is transfer of knowledge, so that performance is improved when tasks are trained together compared to in isolation, the multi-task GP is compared with the standard single task GP probit classifier. For the single task GP classifier we use the EP approximation with an ARD covariance function.

In addition, on all data sets we compare with two alternative multi-task learning approaches. One is an alternative GP-based approach where the various tasks are coupled only through sharing the hyperparameters of the data covariance function k^x. This will be termed as IND-MTL since in this form of multi-task learning there are no correlations between the latent functions of each task. This is in some sense a weaker form of multi-task learning, and is essentially the one used in Lawrence and Platt (2004), inspired by well established hierarchical Bayesian modelling paradigms. The second is a state of the art multi-task classifier based on support vector machines[5] (SVM) (Jebara, 2004), which tackles the problem of multi-task SVM feature and kernel selection, based on the maximum entropy discrimination framework. This performs non-linear

[2]Matlab code to allow replication of the experiments available at http://homepages.inf.ed.ac.uk/gsanguin/software.html.

[3]Code available at http://www.gaussianprocess.org/gpml/code/matlab/doc/

[4]Code available at http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/

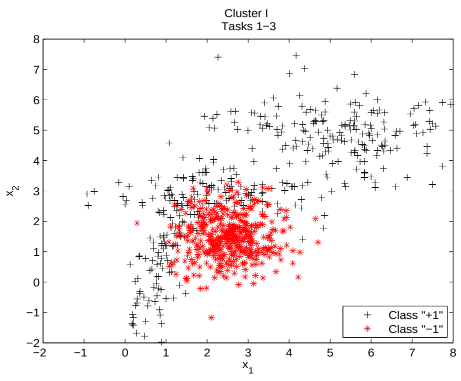
[5]Code available at http://www.cs.columbia.edu/ jebara/code/multisparse/

classification by mapping input data x implicitly into Hilbert space through a feature map ϕ. In this case there are still M discriminant functions, one for each task, which however share a kernel selection vector s, which controls the weight of each base kernel. In all experiments a large number of kernels (10 on the toy data, 18 on the two real data sets) was inserted into the algorithm, which would then automatically decide which kernels are more relevant for classification. Throughout all experiments the regularisation parameters α (kernel selection variable) and C (threshold) (see Jebara (2004) for details) were determined through cross-validation.

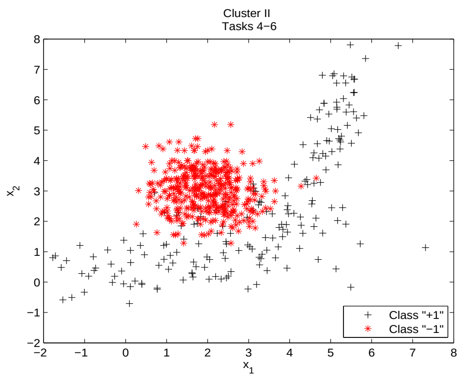
The experimental setup is identical for all data sets, and follows a similar approach to Liu et al. (2009). It is known that the merits of multi-task compared to single task learning are more apparent when few data per task are available. For this reason, in all experiments we show results starting with few data points per task and increasing up to a number where single-task learning approaches the performance of multi-task. In order to estimate the dependence of the results on the training set, each experiment was repeated 100 times on independently sampled training sets of the same size. It is important to note that in the case of single task learning each data set from each task is learnt in isolation, which results in more models being trained.

To assess the performance of the various algorithms, we considered the Receiver Operator Characteristic (ROC) curves, which plot *sensitivity* versus 1-*specificity* of the classifier for varying values of the threshold posterior probabilities (bias in the SVM case). The performance measure used to assess the different algorithms is the Area Under the Curve (AUC) (Hanley and Mcneil, 1982), as it provides a measure that depends on the positive and negative predictive value (PPV and NPV), and not just the overall accuracy. Assessment of the different algorithms is then performed by plotting the median of the AUC over the 100 runs of the experiments and the resulting confidence intervals for varying sizes of the training set. Statistical significance of the results was assessed with standard t-test.

While the purpose of this chapter is to propose a model, rather than optimise its running time, it is still important, from the practitioner's perspective, to assess the relative speed of the various algorithms proposed. Naturally, single-task learning, as well as the multi-task induced by hyperparameter sharing, are faster as they consider a simpler covariance structure. Among the approximate inference approaches to multi-task GP, in our experiments EP was consistently faster than the variational approximation (this depends on the number of VB iterations of course). As a yardstick, EP took approximately 1 hour to run on the arrhythmia data set with seven tasks and 50 training



(a)



(b)

Figure 3.2: Scatter plot of the two clusters of tasks; figure (a) shows the scatter plot for tasks 1, 2, 3, and figure (b) shows the scatter plot for tasks 4, 5, 6.

points per task. As a comparison, the SVM-based method took roughly 40 minutes considering the cross-validation for the determination of the model parameters.

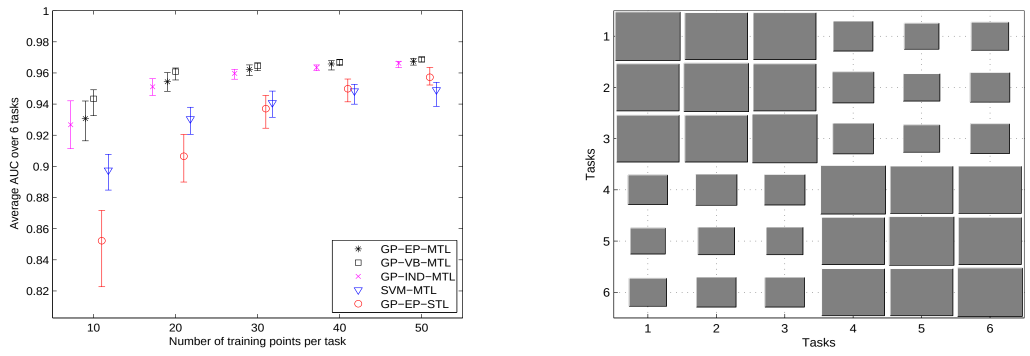
3.4.1 Synthetic data set

The synthetic data set is a 2-dimensional data set previously used in Liu et al. (2009). It is comprised of six binary classification tasks. Data for the first three tasks are generated from a mixture of two partially overlapping Gaussian distributions, and similarly for the remaining three tasks, thus tasks 1-3 and 4-6 are identical and vary only due to the noise induced.

Data from the first cluster of tasks (tasks 1, 2 and 3) for class "+1" were generated by a mixture of Gaussian distributions defined by $p^{c=1}_{f=1}(x) = \sum_{k=1}^{3} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$; the mixing coefficients are given by $\pi_1 = 0.3$, $\pi_2 = 0.3$, $\pi_3 = 0.4$, with respective means and covariances as $\mu_1 = (1,1)$, $\mu_2 = (3,3)$ and $\mu_3 = (5,5)$, and covariances

$$\Sigma_1 = \begin{pmatrix} 0.3 & 0.7 \\ 0.7 & 0.3 \end{pmatrix}, \ \Sigma_2 = \begin{pmatrix} 3.0 & 0.0 \\ 0.0 & 0.3 \end{pmatrix}, \ \text{and} \ \Sigma_3 = \begin{pmatrix} 3.0 & -0.5 \\ -0.5 & 0.3 \end{pmatrix}. \tag{3.36}$$

Data from the second cluster of tasks (tasks 4, 5 and 6) for class "+1 were also generated again by a mixture of Gaussian distributions, $p^{c=1}_{f=2}(x) = \sum_{k=4}^{6} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$ with $\pi_4 = 0.3$, $\pi_5 = 0.3$, $\pi_3 = 0.4$, $\mu_1 = (0.5, 0.5)$, $\mu_2 = (3,2)$ and $\mu_3 = (5,5)$, and covariances $\Sigma_4 = \Sigma_2$, $\Sigma_5 = \Sigma_3$, and $\Sigma_3 = \Sigma_1$. Data for class "-1" for both clusters of tasks are generated by single Gaussians. The generating distribution for the first cluster of



(a) (b)

Figure 3.3: Experimental Results on Toy data set. (a) Mean AUC over the 6 tasks, (b) Hinton Diagram of the task covariance matrix, averaged over the 100 independent trials of the results obtained by EP with 50 data points per task.

tasks is given by

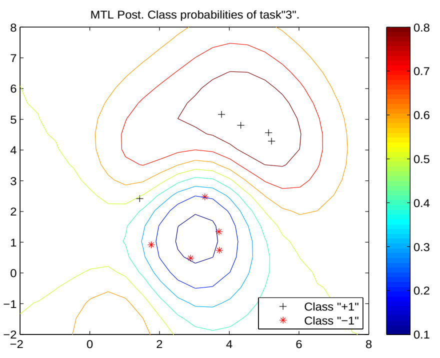
$$p_{f=1}^{c=2}(x) \sim \mathcal{N}\left(\begin{pmatrix} 2.5 \\ 1.5 \end{pmatrix}, \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}\right), \tag{3.37}$$

and for the second cluster by

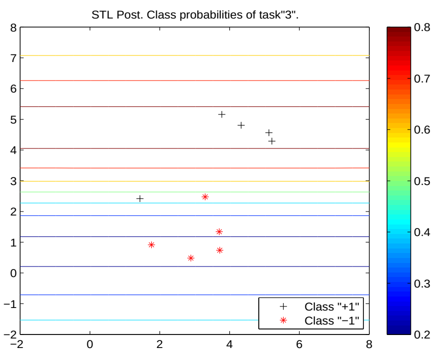
$$p_{f=2}^{c=2}(x) \sim \mathcal{N}\left(\begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}\right). \tag{3.38}$$

The scatter plot of the toy data set is shown in figure 3.2, which illustrates the overlap between the classes of each cluster of tasks as well as the differences of the two clusters.

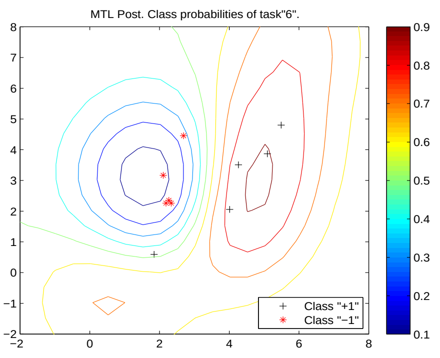
Figure 3.3.a shows the mean and the error bars of the AUC over the six tasks for both approximations for multi task GP classification (EP & VB), multi-task with independent GPs (IND-MTL), single-task GP learning, and for the SVM-based method. It is clear that all multi-task learning algorithms significantly outperform single-task learning, reflecting the fact that indeed in this case there are important correlations between the tasks. Figure 3.4 highlights the differences between the boundaries produced from MTL and STL for 10 training points (5 from each class) for tasks 3 and 6. As shown in the left panel of figure 3.4 the MTL boundaries are a lot more broad than the STL boundaries in the right panel. In addition, simple inspection reveals that the MTL classifier gives high probabilities to regions where no training data from that task



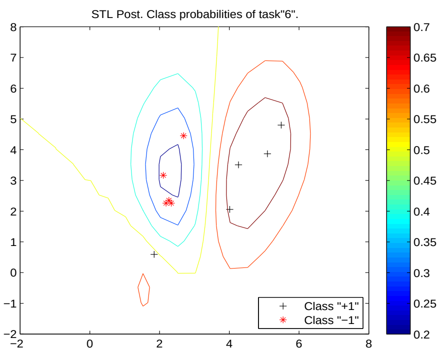
(a)



(b)



(c)



(d)

Figure 3.4: Decision boundaries with the EP approximation for MTL and STL for 10 training data points; subfigures (a) and (b) show the decision boundaries for task 3 for MTL and STL respectively from the first cluster, and subfigures (c) and (d) show the decision boundaries for task 6 for MTL and STL respectively from the second cluster of tasks.

are present due to the task similarities shown in the Hinton diagram (Hinton, 1989) in figure 3.3.b.

It is also worth noticing that all multi-task GP methods improve upon the SVM-based method (with a statistically significant improvement in 82% of the cases at 5% p-value). Larger differences in performance are noticed when few data points are used for training. As expected those differences decay as the number of training points increases. Comparing the multi-task GP methods we see that VB performs slightly better than EP and IND in the case when few data points are available, although the differences are minimal when more than 10 points per task are available. It is worthwhile comparing the results to those reported in Liu et al. (2009), which used a semi-supervised multi-task logistic regression model to perform classification. The reported AUC in that paper (average only) was of approximately 90% when 30 labelled data points per task were used, roughly 5% less than the performance achieved by our method. This is probably a result of the greater flexibility of the GP as a classifier, compared to the logistic regression employed in Liu et al. (2009). Figure 3.3.b shows the Hinton diagram of the optimised task covariance matrix (larger blocks denote values closer to 1), showing that the algorithm is able to correctly learn the similarity between the tasks.

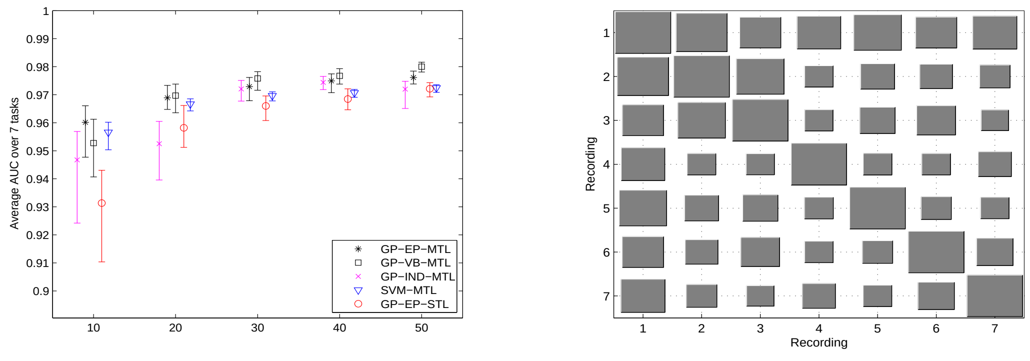
3.4.2 Arrhythmia data set

In this data set, we are provided with seven recordings of ECG signals from seven different patients. Each recording corresponds to a large number of heart beats; the goal is to classify each heart beat into two classes, either normal or premature ventricular contraction (PVC) arrhythmic beats. This problem was already considered using single task GP classifiers in Skolidis et al. (2008). Data for this set of experiments were taken from the MIT-BIH Arrhythmia database (Goldberger et al., 2000); each recording was sampled at $360Hz$. Annotation provided by the database was used to separate the beats before any preprocessing. Each beat segment, consisting of 360 data points (one minute), was transformed into the frequency domain using a Fast Fourier Transform with a Hanning window. Only the first ten harmonics are used as features for classifying heart beats, as most of the information of the signal is contained in these harmonics. In this case, each recording/patient is treated as a separate task. A detailed description of the data set is given in table 3.1.

Table 3.1: Description of the Arrhythmia data set

Task ID	1	2	3	4	5	6	7
Recording ID	106	200	203	217	221	223	233
Total number of data	2021	2567	2970	406	2349	2417	3053
Number of Normal heart beats	1503	1740	2526	244	1954	1955	2224
Number of PVC heart beats	518	827	444	162	395	462	829

The mean AUC over the *100 repetitions* is shown in figure 3.5.a for all five methods



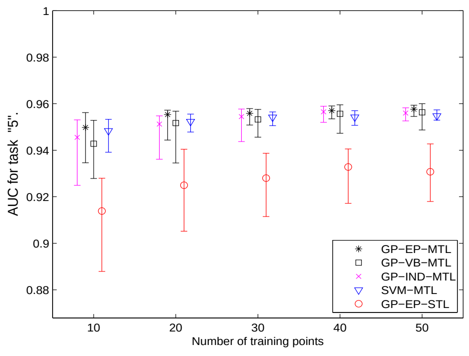
(a) (b)

Figure 3.5: Experimental Results on the *Arrhythmia* data set. (a) Mean AUC over the 7 tasks, (b) Hinton Diagram of the task covariance matrix, averaged over the 100 independent trials of the results obtained by EP with 50 data points per task

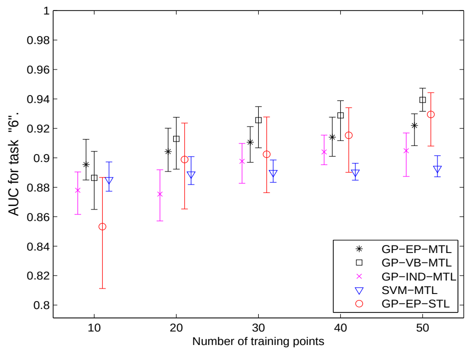
considered. It is noticed that the EP, VB, and the SVM methods significantly outperform the IND and the STL method with small error bars. It is interesting to note that for 20 training data points per task STL performs better than the MTL with independent latent functions. Further investigation however reveals that there are two types of tasks in terms of performance. Some tasks (1,2,3,4,7) appear to be reasonably straightforward, achieving more than 95% of AUC for all multi-task and single task learning methods applied, and for all training set sizes considered. In contrast, tasks 5 and 6 are harder classification tasks, and in these cases the multi task approaches yield a considerable advantage, both in terms of performance (higher median AUROC) and in terms of robustness (smaller error bars). These results are given in figure 3.6, (a), and (b) respectively. As in the synthetic example this difference degrades as the number of training points per task increases. The Hinton diagram of the task covariance matrix, in figure 3.5.b, shows the similarity between tasks. Not surprisingly, the results are not as clear cut as in the toy data set, and no firm conclusions can be made.

3.4.3 Landmine data set

The final set is a landmine detection problem, consisting of 19 tasks where each point is represented by nine features, previously investigated in Liu et al. (2009), and Xue et al. (2007). This data set is tested only with the EP approximation and the Independent multi-task GP setting, as the large number of tasks made the VB approach slower.



(a)

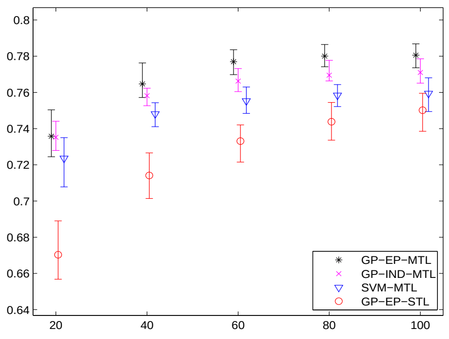


(b)

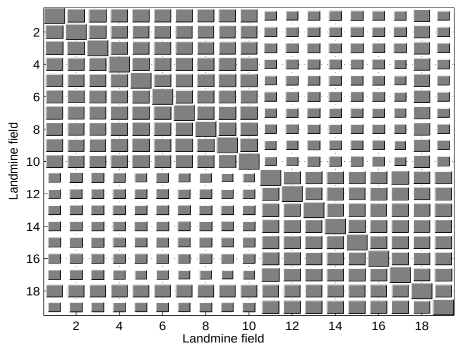
Figure 3.6: Additional Results on the *Arrhythmia* data set. (a) AUC for task 5 (Recording ID: 221), (c) AUC for task 6 (Recording ID: 223)

Also, we consider at least 20 training points per task, given the larger number of parameters to be estimated in the task covariance matrix. Results[6] shown in figure 3.7.a, demonstrate a significant difference, in terms of performance, between the multi-task learning algorithms with both GPs and SVMs over the single-task learning with GPs, while comparing the performance of the multi-task algorithms it is noticed that the EP approximation outperforms the other two. A more accurate analysis of the results shows that the GP method significantly (at 5% p-value) outperforms the SVM method in the overwhelming majority of tasks (table 3.2). Liu et al. (2009) reported a mean AUC of approximately 78% on this problem (irrespective of the training size, since their semi-supervised approach benefited from using unlabelled data). Our approach achieves a slightly better performance when only 20 data points are available, and a significantly better performance as the size of the training set increases. Figure 3.7.b shows the Hinton diagram of the task covariance matrix, clearly showing the presence of two clusters containing the first ten and the remaining nine tasks. This is in good agreement with previously reported results on this data set (*cf* Liu et al. (2009), figure 3.c).

[6]As in the previous two examples the reported median is over the 100 repetitions and not over the tasks.



(a)



(b)

Figure 3.7: Experimental Results on the *Landmine* data set. (a) Mean AUC over the 19 tasks, (b) Hinton Diagram of the task covariance matrix, averaged over the 100 independent trials of the results obtained by EP with 100 data points per task

Table 3.2: T-test for the Landmine data set. The second column (No. of tasks) shows the number of tasks with better means, for the method stated in the first column. The third column (Statistically Significant) shows the number of tasks that were statistically significant on the 0.05 significance level (p-value).

	No. of tasks					**Statistically significant**				
Data points per task	**20**	**40**	**60**	**80**	**100**	**20**	**40**	**60**	**80**	**100**
MTL-EP > STL-EP	19	19	19	19	18	19	19	19	19	18
MTL-EP > MTL-SVM	15	16	15	16	16	15	16	15	16	16

3.5 Conclusions

In this chapter, we presented a Bayesian multi-task classification model based on Gaussian process priors. The work extends the regression model of Bonilla et al. (2008) to the classification scenario, introducing a number of novel aspects. First of all, we consider non-Gaussian likelihoods in order to address classification problems. Secondly, we present a number of inference strategies, including an asymptotically exact sampling scheme and two efficient deterministic approximations. Finally, we demonstrated on a number of important real-world tasks the benefits of this approach, both over single-task learning and competing multi-task approaches.

Chapter 4

Semi-Supervised Multi-task learning with Gaussian Processes

We present a probabilistic framework for transferring learning across tasks and between labelled and unlabelled data. The approach is based on Gaussian Process prediction and incorporates both the geometry of the data and the similarity between tasks within a Gaussian process covariance, allowing Bayesian prediction in a natural way. We discuss the transfer of learning in a multi-task scenario in the two cases where the underlying geometry is assumed to be the same across tasks and where different tasks are assumed to have independent geometric structures, and we discuss the way similar ideas can be employed in different transfer learning scenarios such as domain adaptation and self-taught learning. We demonstrate the method on a number of real data sets, indicating that the semi-supervised multi-task approach can result in very significant improvements in performance when very few labelled training examples are available.

4.1 Introduction

Advances in data gathering technologies have led to what many dub *the age of data*. This deluge of data is not matched by a corresponding increase in the quality and availability of annotation; devising methodologies to improving learning from sparse and often poor supervision is one of the major challenges for machine learning and pattern recognition. Besides its obvious importance in applications, the problem also has a fundamental scientific relevance: unsupervised or weakly supervised learning has to be one of the fundamental building blocks behind human cognition.

Broadly speaking, two related scenarios are often encountered: unannotated data for a prediction task is plentiful, but we are interested in prediction based on a relatively small set of labelled data. In this scenario, we are interested in leveraging information contained in the abundant unlabelled pool to improve the performance of the supervised predictor. This line of enquiry is typical of *Semi-supervised learning* (SSL) methods, and has received much attention within the machine learning community in recent years (Chapelle et al., 2006; Seeger, 2001). Alternatively, one may have data for many related prediction tasks, and may wish to construct a more powerful predictor by exploiting task relatedness to transfer learning across tasks. Within this transfer learning scenario, one may distinguish several important subcases (Pan and Yang, 2010): in this chapter, we will mostly focus on the *multi-task learning* (MTL) scenario, whereby each of the tasks we are interested in has at least some labelled data.

Naturally, leveraging extra information comes at the cost of enforcing further modelling assumptions within the learning paradigm. SSL usually relies on assumptions about the global structure of the data (Chapelle et al., 2006):

- Nearby points in high density regions are assumed to share the same label with high probability, the "*smoothness assumption*".
- Decision boundaries are assumed to lie in low data density regions, the "*cluster assumption*".
- Data are assumed to lie on a low-dimensional manifold, the "*manifold assumption*".

Similarly to SSL, MTL assumes correlations between tasks to achieve better predictions by training in parallel across tasks (Pan and Yang, 2010). Naturally, if these assumptions do not hold, then the inclusion of unlabelled data or the parallel training across tasks might even degrade performance.

While both SSL and MTL are notable success stories, relatively little work has investigated the situation where information is transferred both from unlabelled data and across tasks. The problem is particularly complicated when correlations between tasks are to be inferred from data, as one needs to solve the non-trivial task of determining how to use unlabelled data in determining task relatedness. In this chapter, we present a probabilistic model to transfer learning from unlabelled data across different tasks, which will be referred to as *Semi-supervised Multi-task learning* (SS-MTL). We work within the Gaussian Process (GP) framework, using the "*Intrinsic Model of Coregionalization*" (IMC) (Cressie, 1993; Bonilla et al., 2008), which encodes task correlations in a structured covariance matrix. To enable semi-supervised learning within this model, we couple the LCM with the data-dependent prior of Sindhwani et al. (2005, 2007) which allows us to incorporate the global geometric structure of the data within the learning framework. We demonstrate how this coupled covariance does indeed lead to significant advantages on real tasks where labelled data is extremely sparse.

The rest of this chapter is organised as follows. In section 4.2 we review the construction of data dependent conditional priors from the frequentist and the Bayesian point of view. In section 4.3.1 we discuss a weak form of transfer learning for SS-MTL with GPs that is based on the parameter transfer approach presented in section 2.2.1. In section 4.3.2 we present our method for SS-MTL which falls in the Inductive transfer GP type of approach. We then propose two different models, one that exploits the geometry of the unlabelled data of each task independently, and one in which the geometry of a task influences the prior of all other tasks. Additionally, in section 4.3.3 we present how the graph Laplacian is constructed, and we propose a novel method for estimating automatically the graph structure. In section 4.4 we present results of the two SS-MTL methods on one artificially generated data set and three real data sets. Finally, in section 4.5 we discuss possible extensions of the proposed settings to other forms of transfer learning. With respect to the classification of *Transfer learning* (TL) algorithms presented in chapter 1, the content of this chapter falls primarily into the *Inductive* category of TL. However note that one of the extensions of the models that we discuss in section 4.5 is intended for *Domain Adaptation* (DA) which would naturally fall into the *Transductive* category of TL.

4.2 Semi-supervised learning with GPs (SSGP)

GPs are a flexible non-parametric model for regression and classification which have become increasingly popular in recent years (Rasmussen and Williams, 2005). A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution; if f is a sample from a GP, we write

$$f \sim \mathcal{GP}(\mu, K), \tag{4.1}$$

where $\mu(x)$ is the mean function and $K_{ij} = k(x_i, x_j)$ is the covariance function, capturing the input dependence of the target statistics. By definition, the vector $\mathbf{f}$ obtained by evaluating f at a finite set of input points $x_1, \ldots, x_N$ is normally distributed with mean μ given by evaluating the mean function μ at the input points, and covariance Σ obtained by evaluating the covariance function k at every pair of input points. Given target variables $y_1 \ldots, y_N$ and a noise model connecting the targets with $\mathbf{f}$, standard methods allow to obtain a posterior process (Rasmussen and Williams, 2005). In addition, GPs are connected to frequentist kernel methods (Schölkopf et al., 1999) through the covariance function: both a valid *covariance* function for a GP and the *kernel* function of a Reproducing Kernel Hilbert Space (RKHS) must obey the same conditions (Mercer's theorem).

4.2.1 Frequentist kernel methods for SS learning

In this section we provide a brief introduction to the ideas the data-dependent norms of Sindhwani et al. (2005) are based on; for proofs and theoretical issues concerning data dependent norms on RKHS the interested reader is referred to Sindhwani et al. (2005) and Belkin et al. (2006).

In a semi-supervised learning setting, the learner is provided with a set of labelled examples $L = \{(x_{l1}, y_1), \ldots, (x_{ln_l}, y_{n_l})\}$, and an additional set of unlabelled examples $U = \{x_{u1}, \ldots, x_{un_u}\}$, where n_l and n_u is the number of labelled and unlabelled data respectively. The union of the labelled and unlabelled sets will be denoted by $D = L \cup U$, while other samples unseen to the learner will be denoted by T. Moreover, it is usually assumed that the labelled examples are generated according to a probability distribution P on $X \times \mathbb{R}$ and that unlabelled examples are drawn according to the marginal distribution $\mathcal{P}_X$ of P. The concept behind the semi-supervised learning framework is that information about the marginal distribution $\mathcal{P}_X$ can help for better function learning. In order to make use of the marginal distribution certain assumptions have to be taken into account, as the smoothness assumption stated in the previous section (Belkin et al., 2006).

A Mercer kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, has a RKHS $\mathcal{H}_K$ of functions $f \to \mathbb{R}$ with the corresponding norm, *i.e.*, $\| f \|_K = \sqrt{\langle f, f \rangle_K}$, where we have used the kernel K to denote inner products and norms in the corresponding Hilbert space $\mathcal{H}_K$, that is $\langle ., . \rangle_K$, and $\| \, . \, \|_K$ instead of $\langle ., . \rangle_{\mathcal{H}_K}$, and $\| \, . \, \|_{\mathcal{H}_K}$, respectively (Belkin et al., 2006). In the standard supervised regularisation framework one wishes to estimate a function from $\mathcal{H}_K$ by minimising $R_{\text{reg}}[f]$ whose solution $f(x)$ is given by the representer theorem (Smola and Schölkopf, 2002, Ch. 4.2),

$$R_{\text{reg}}[f] = \arg\min_{f \in \mathcal{H}} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \| f \|_K^2, \quad \text{with} \quad f(x) = \sum_{i=1}^{L} \alpha_i k(x_i, x), \tag{4.2}$$

where $V(x_i, y_i, f)$ is an appropriate loss function, as the squared loss $(y_i - f(x_i)))^2$

giving rise to *Regularized least squares*, or the hinge loss $[1 - y_i f(x_i)]_+$, giving rise to *Support Vector Machines* (SVMs), $[.]_+$ denotes the positive part, and $\| f \|_K^2$ is the squared norm in $\mathcal{H}_K$ and *can be thought of as a generalisation to functions of the n_l-dimensional quadratic form* $\mathbf{f}^T K^{-1} \mathbf{f}$ (Rasmussen and Williams, 2005, Ch. 6.1).

Belkin et al. (2006) transformed this minimisation problem to account for the unlabelled data by adding an extra regularisation term as,

$$f^* = \arg \min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \| f \|_K^2 + \gamma_I \| f \|_I^2, \tag{4.3}$$

where the term $\| f \|_I^2$ reflects the *intrinsic* structure of the marginal $\mathcal{P}_X$, and parameters γ_A and γ_I are the regularisation parameters for the ambient and the intrinsic structure of the data respectively. As an empirical substitute of the structure of the marginal distribution Belkin et al. (2006) used the graph Laplacian (Belkin and Niyogi, 2003),

$$\gamma_I \| f \|_I^2 = \frac{\gamma_I}{(u+l)^2} \sum_{i,j=1}^{l+u} \left(f(x_i) - f(x_j) \right)^2 W_{ij} = \frac{\gamma_I}{(u+l)^2} \mathbf{f}^T L \mathbf{f}, \tag{4.4}$$

where W_{ij} are the edge weights in a graph, $L = D - W$ is the graph Laplacian, $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$, and $\mathbf{f} = [f(x_1), \ldots, f(x_{l+u})]$.[1]

In a follow up paper this kernel based SS framework has been reinterpreted in terms of a family of data-dependent norms on RKHS (Sindhwani, 2007). It can be shown (Sindhwani et al., 2005), that the new kernel $\tilde{K}$ associated with the new space of functions $\tilde{\mathcal{H}}$ that takes into account the intrinsic geometry of the marginal distribution $\mathcal{P}_X$ will be given by,

$$\tilde{K}(x,z) = K(x,z) - \Sigma_{D,x}^T (I + L\Sigma_{D,D})^{-1} L \Sigma_{D,z} \tag{4.5}$$

where $\Sigma_{D,D}$ is the kernel matrix between all points in set D, $\Sigma_{D,x} = [k(x_1,x), \ldots, k(x_{l+u},x)]$ is the kernel column vector between points in D and x and similarly for $\Sigma_{D,x}$. The data-dependent kernel of Sindhwani et al. (2005) given in the above equation states that the kernel function evaluated at two points is influenced by all labelled and unlabelled data through the kernel matrix Σ_{DD} and the graph Laplacian L.

4.2.2 Bayesian Semi-supervised GP learning

In order to extend the GP framework to the semi-supervised case, we will follow Sindhwani et al. (2007) and use the manifold approach to semi-supervised learning. This

[1] We have slightly abused the notation by using D to denote both the union of the labelled and unlabelled data, and in the construction of the graph Laplacian, but the use of this letter can easily be inferred from the context.

leverages the geometric structure implied by the data distribution to inform the prediction process. Explicitly, the prior distribution over the latent function values $\mathbf{f}_D$ conditioned on the geometry of the problem, which we refer to as $\mathcal{G}$, is now composed of two parts:

$$p(\mathbf{f}_D|\mathcal{G}) \propto \exp(-\frac{1}{2}\mathbf{f}_D^T Q \mathbf{f}_D) p(\mathbf{f}_D), \tag{4.6}$$

where the first term in the r.h.s. of equation 4.6 contains the dependency of the latent variables on the geometry of the problem, while $p(\mathbf{f}_D) \sim \mathcal{N}(0, \Sigma_{DD})$ is a standard GP prior with zero mean and Σ_{DD} covariance matrix. The matrix Q is a graph regulariser reflecting the geometric structure of the data; a common choice is the graph Laplacian or a power thereof. For more information about the construction of data dependent priors we refer to Sindhwani et al. (2007).

Additionally, as can be seen in equation 4.6, the data-dependent prior creates a conceptual difficulty in dealing with new data in the set T, as the graph Laplacian depends on all data (including potentially novel samples in the set T). To extend the approach from the transductive to the inductive setting, we assume that the geometry of the problem ($\mathcal{G}$) is independent of all latent function evaluations at points in set T. Given that $\mathbf{f}_X = \{\mathbf{f}_D, \mathbf{f}_T\}$ are jointly Gaussian, the joint prior of $\mathbf{f}_X$ given $\mathcal{G}$ will be given by: $p(\mathbf{f}_X|\mathcal{G}) \propto \exp\left(-\frac{1}{2}\mathbf{f}_X^T \tilde{\Sigma}_{XX}^{-1} \mathbf{f}_X\right)$, where

$$\tilde{\Sigma}_{XX}^{-1} = \begin{bmatrix} \Sigma_{DD} & \Sigma_{DT} \\ \Sigma_{DT}^T & \Sigma_{TT} \end{bmatrix}^{-1} + \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}.$$

Using the partitioned inverse formula, we obtain the following proposition (for details see appendix B.1) (Sindhwani et al., 2007):

Proposition: Given equation 4.6, for any finite collection of data points X, the random variables $\mathbf{f}_X = \{f(x)\}_{x \in X}$ conditioned on $\mathcal{G}$ have a multivariate normal distribution $\mathcal{N}(0, \tilde{\Sigma}_{XX})$. The elements of the covariance matrix $\tilde{\Sigma}_{XX}$ are given by evaluating the following kernel function $\tilde{K} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$

$$\tilde{K}(x,z) = K(x,z) - \Sigma_{D,x}^T (I + Q\Sigma_{D,D})^{-1} Q \Sigma_{D,z}, \tag{4.7}$$

for $x, z \in \mathcal{X}$, where $\Sigma_{D,x}$, $\Sigma_{D,z}$ denote the covariance vectors between points x, z respectively, and all data points in set D. This defines the *Semi-Supervised Gaussian Process* (SSGP) as a GP with a modified kernel function given in equation 4.7. It can be observed that if Q is invertible, *i.e* positive definite, then $(I + Q\Sigma_{D,D})^{-1} Q$ is equal to $\left(Q^{-1} + \Sigma_{D,D}\right)^{-1}$. If matrix Q is singular, which is the case in the graph Laplacian since the smallest eigenvalue is equal to zero (Belkin and Niyogi, 2002), then we can add a small ridge term to alleviate this problem. Combining the data-dependent norm in equation 4.7 with the Laplacian matrix Q similarly to the previous section 4.2.1, has been shown to be an appropriate norm on a function space where the *cluster* and *manifold* semi-supervised assumptions hold (Sindhwani et al., 2005). Note also that the Laplacian matrix of a graph can be seen as empirical substitute of the Laplace-Beltrami operator and defines a measure of smoothness with respect to a manifold when the underlying space is a Riemannian manifold. More details about the construction of the graph Laplacian are given in section 4.3.3.

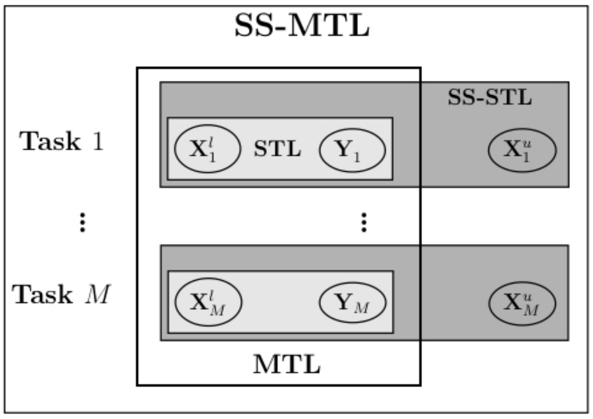


Figure 4.1: Sources of information in Semi-supervised multi-task learning (SS-MTL); $\mathbf{X}_j^l$ and $\mathbf{X}_j^u$ denotes the labelled and the unlabelled set of task j, and $\mathbf{Y}_j$ represents the responses for data in $\mathbf{X}_j^l$.

4.3 Semi-Supervised Multi-task learning (SS-MTL) with GPs

In a Semi-Supervised Multi-Task Learning (SS-MTL) scenario we are interested in learning M related functions $\mathbf{f}_j$ by leveraging information from labelled and unlabelled data (figure 4.1). The multi-task machinery enables us to learn together these M functions, while ideas from semi-supervised learning can help to exploit the overall geometric structure of the data in each task.

Let each task j have labelled data $L_j = \{\mathbf{x}_{kj}^l, \mathbf{y}_{kj}\}_{k=1}^{n_j^l}$ of size n_j^l for which outputs are available, and unlabelled data $U_j = \{\mathbf{x}_{kj}^u\}_{k=1}^{n_j^u}$ of size n_j^u for which outputs are not observed, where $\mathbf{x}_{kj} \in \mathbb{R}^{d\times 1}$ where d is the number of features for each sample. Let

$D_j = L_j \cup U_j$ of size $N_j = n^l_j + n^u_j$ be the item set of task j, and $D = D_1 \cup D_2 \ldots \cup D_M$ be the total item set of all tasks, with $N = \sum_{j=1}^{M} N_j$. Additionally, we will define the set of all labelled points from all tasks as $L = L_1 \cup \ldots \cup L_M$ with $N^l = \sum_{j=1}^{M} n^l_j$. As in the previous section all other data points unseen to the learner will be denoted by T.

The classification of MTL algorithms that was employed in chapter 2 can be extended in the Semi-supervised case by making use of the data-dependent priors of Sindhwani et al. (2007), to give the Parameter and the Inductive SS transfer analogs. In the Parameter transfer SS-MTL the latent functions $\mathbf{f}_j$ are learned by using information from labelled and unlabelled data and are coupled by sharing a common hyper-prior over some parameters. As in the MTL case this formulation results in latent functions of different tasks being independent given these parameters and the unlabelled data. In the Inductive SS transfer the latent functions are also influenced by unlabelled data but there is some form of correlation between these functions which results in predictions for a task being affected by the learned functions in other tasks.

4.3.1 Parameter transfer SS-MTL with GPs

Parameter transfer semi-supervised multi-task learning in the context of Gaussian Processes has been investigated in Zhang and Yeung (2009). In this work, the transfer of information between the tasks takes place by inducing a common hyper-prior on the hyperparameters θ^x_j of the data covariance function. The prior of the latent function of each task is defined as $\mathbf{f}_j|D_j \sim \mathcal{N}(0, \tilde{K}_{jj})$, where $\mathbf{f}_j \in \mathbb{R}^{n^l_j \times 1}$ is the latent function evaluated *only* at the observed locations of the j^{th} task and $\tilde{K}_{jj} \in \mathbb{R}^{n^l_j \times n^l_j}$. Similarly to SS-GP the covariance matrix $\tilde{K}_{jj}$ is found by evaluating the data-dependent kernel of Sindhwani et al. (2007) defined in equation 4.7. Thus, the covariance between two points $x,\ z$ from task j will given by,

$$\tilde{k}^x(x,z) = k^x(x,z) - \left(\mathbf{k}^x_{D_j,x}\right)^T (\alpha^{-1} I + Q_j K^x_{D_j,D_j})^{-1} Q_j \left(\mathbf{k}^x_{D_j,z}\right), \tag{4.8}$$

where $\mathbf{k}^x_{D_j x}$ and $\mathbf{k}^x_{D_j z}$ are the covariance vectors between all labelled and unlabelled data points in D_j, $K^x_{D_j,D_j} \in \mathbb{R}^{N_j \times N_j}$ is the covariance matrix between all labelled and unlabelled points in the j^{th} task which is constructed by evaluating the covariance function parameterised by θ^x_j. As in SS-GP the geometry of the tasks is encoded through the graph Laplacian $Q_j \in \mathbb{R}^{N_j \times N_j}$ between the points in D_j, and the parameter α controls the contribution of the unlabelled data points. Finally, estimating the optimal hyperparameters is performed by maximising the log marginal likelihood. Assuming that the prior of each set of hyperparameters is given by $\theta_j^x \sim \mathcal{N}(\mathbf{m}_\theta, \Sigma_\theta)$[2], and that the tasks are corrupted with a common noise term with variance σ, the log-likelihood is computed from (Zhang and Yeung, 2009),

$$\log p(\mathbf{y}|D) = -\frac{1}{2}\sum_{j=1}^{M}\left[\mathbf{y}_j^T\left(\tilde{K}_{L_j,L_j}^x + \sigma^2 I\right)^{-1}\mathbf{y}_j + \log\left|\tilde{K}_{L_j,L_j}^x + \sigma^2 I\right|\right]$$
$$-\frac{1}{2}\sum_{j=1}^{M}\left[\left(\mathbf{m}_\theta - \theta_j\right)^T\Sigma_\theta^{-1}\left(\mathbf{m}_\theta - \theta_j\right) + \log|\Sigma_\theta|\right] + \text{constant}, \qquad (4.9)$$

where we have used that $\mathbf{y}_j = \{y_{1j}, \ldots, y_{n_j^l j}\}$, with $\mathbf{y} = \text{vec}(\mathbf{Y})$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_M]$. Computing the gradients for parameters θ_j, $\sigma, \mathbf{m}_\theta$, and Σ_θ is straightforward and we refer to Zhang and Yeung (2009) for details.

Another possible setting for performing SS-MTL similar to Zhang and Yeung (2009) would be to employ the data dependent norms of Sindhwani et al. (2005) but to constrain the different GPs to share the same hyperparameters. This is the semi-supervised analog of the method proposed for MTL in Lawrence and Platt (2004) and it will be referred to as the SS-MTL IND model or simply SS-IND. The data-dependent prior of the latent functions will be independent given the hyperparameters,

$$p(\mathbf{f}|D) = \prod_{j=1}^{M}\mathcal{N}_{\mathbf{f}_j}(0, \tilde{K}_{L_j,L_j}^x), \qquad (4.10)$$

where we have used that $\mathbf{f} = \text{vec}(\mathbf{F})$ with $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_M]$. Finally, the optimal hyperparameters of the covariance function and the graph Laplacian will be estimated type *II* ML (see section 4.3.5). This approach will be employed in the experimental part and will be compared with the semi-supervised methods from the Inductive category.

4.3.2 Inductive transfer SS-MTL with GPs

In the Inductive transfer category we will use the IMC model (Cressie, 1993; Bonilla et al., 2008) as a framework for MTL; this introduces $N^l \times M$ latent variables $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_M]$, with $\mathbf{f} = \text{vec}(\mathbf{F})$, corresponding to the *complete set of responses*, *i.e.* the set of outputs in all tasks corresponding to each input point. The model then induces correlations between tasks by enforcing the covariance of the joint latent variables to factorise as the Kronecker product of a task correlation matrix with a standard GP

[2]Note that the Normal distribution does not ensure positivity, and the covariance function that is employed is given by $k(x,x') = \theta_1 x^T x' + \theta_2 \exp\left\{-\frac{||x-x'||^2}{2\theta_3^2}\right\}$, and parameters θ_1 and θ_2 should be positive. Another possible choice could be the Gamma distribution.

covariance

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{GP}(0, K^t \otimes K^x).$$

Naturally, the complete set of responses is rarely available; we can use the consistency property of GPs to marginalise the latent variables corresponding to unobserved responses.

In the SS-MTL setting we consider two different formulations to use information from unlabelled data. In the first one, we assume *independence* between the geometry of different tasks. This formulation will be termed *Static Geometry* (SG), since transfer of geometry can be achieved only through the task covariance matrix. The second formulation allows a more flexible relationship since the multi-task setting is induced directly on the geometry of the data. Given that the geometric structure of one task can affect the prior of other tasks, this formulation will be termed *Flexible Geometry* (FG).

4.3.2.1 Static Geometry

In the SG scenario, the latent functions evaluations on the labelled and unlabelled points of each task $\mathbf{f}_{s_j} \in \mathbb{R}^{N_j \times 1}$, are associated with the geometry ($\mathcal{G}_{s_j}$) of that specific task only. In this case the prior of each of the latent functions $\mathbf{f}_{s_j}$ conditioned on the geometry of the task $\mathcal{G}_{s_j}$ will be given by:

$$p(\mathbf{f}_{s_j}|\mathcal{G}_{s_j}) \propto \exp(-\frac{1}{2}\mathbf{f}_{s_j}^T Q_j \mathbf{f}_{s_j}) p(\mathbf{f}_{s_j}), \tag{4.11}$$

where Q_j is the graph Laplacian of the j^{th} task, and $p(\mathbf{f}_{sj}) \sim \mathcal{N}(0, K^x_{D_j,D_j})$ is the GP prior for that specific task only, with covariance $K^x_{D_j,D_j} \in \mathbb{R}^{N_j \times N_j}$ evaluated between all labelled and unlabelled points in task j. If we denote by $\mathbf{f}_S$ the collection of the latent functions from all tasks $\mathbf{f}_S = [\mathbf{f}_{s_1}^T, \ldots, \mathbf{f}_{s_M}^T]^T$, then their joint distribution will be a zero mean multivariate Gaussian with covariance matrix $K^x_{D,D} \in \mathbb{R}^{N \times N}$, with block matrices $K^x_{D_j,D_j}$ in the diagonal, and $K^x_{D_i,D_j}$ as the covariance matrices between the different tasks. The prior of $\mathbf{f}_S$ conditioned on the geometry of all tasks $\mathcal{G}_S$ will be given by a zero mean Gaussian distribution with covariance $\Delta = \left((K^x_{D,D})^{-1} + Q_S\right)^{-1}$, as

$$p(\mathbf{f}_S|\mathcal{G}_S) \propto \exp\left(-\frac{1}{2}\mathbf{f}_S^T \Delta^{-1} \mathbf{f}_S\right), \tag{4.12}$$

where Q_S is block diagonal on Q_j's,

$$Q_S = \begin{bmatrix} Q_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Q_M \end{bmatrix},$$

which reflects the independence assumption between the geometry of different tasks.

Finally, conditioning on $\mathcal{G}_S$ and inducing correlations between the tasks, the prior over the latent variables associated with the complete set of responses will be given by:

$$p(\mathbf{f}|D) \sim \mathcal{GP}(0, K^t \otimes \tilde{K}^x), \tag{4.13}$$

where $\otimes$ is the Kronecker product, K^t is a positive semi-definite matrix specifying inter-task correlations, and $\tilde{K}^x \in \mathbb{R}^{N^l \times N^l}$ is the data dependent covariance function over the *labelled inputs* defined in equation 4.7, where $\Sigma_{D,D}$ and Q are replaced by $K^x_{D,D}$ and Q_S, which are constructed by the *labelled and unlabelled data from all tasks*. The last equation completes the semi-supervised multi-task Gaussian process prior, which as shown in equation 4.13 is completely specified by this special structure in the covariance function. Notice that the vector of latent variables associated with the complete set of responses is of size $N^l M$, as is the Kronecker product of the task correlation matrix ($M \times M$) with the SSGP prior ($N^l \times N^l$).

In the special but common case when only one response is observed per input, we notice that unlabelled data from each task cannot affect the covariance between other tasks *a priori*. The only way in which unlabelled data from one task can affect predictions on other tasks is through the task covariance matrix if the two tasks are found to be correlated.

4.3.2.2 Flexible Geometry

In FG, as in the standard IMC, we start by defining the vector of latent function $\mathbf{f}_F = \{\mathbf{f}_{f_1}, \ldots, \mathbf{f}_{f_M}\}$, corresponding to the complete set of responses, where $\mathbf{f}_{f_j} \in \mathbb{R}^{N\times 1}$, and $\mathbf{f}_F \in \mathbb{R}^{MN\times 1}$. We then associate the latent functions $\mathbf{f}_F$ with the geometry of the data written as $\mathcal{G}_F = \{\mathcal{G}_{f_1}, \ldots, \mathcal{G}_{f_M}\}$. In this setting the multi-task hypothesis is induced directly on the prior over the latent functions $\mathbf{f}_F$ as:

$$p(\mathbf{f}_F) \sim \mathcal{GP}(0, K^t \otimes K^x_{D,D}), \tag{4.14}$$

while the prior of $\mathbf{f}_F$ conditioned on $\mathcal{G}_F$ will be proportional to :

$$p(\mathbf{f}_F|\mathcal{G}_F) \propto \exp(-\frac{1}{2}\mathbf{f}_F^T(K^t \otimes Q_D)\mathbf{f}_F)p(\mathbf{f}_F). \tag{4.15}$$

Notice that now the task covariance matrix K^t appears both in the distribution reflecting the geometry as $K^t \otimes Q_D$, and in the prior $p(\mathbf{f}_F)$. In this case the matrix Q_D is the graph

Laplacian between data in all tasks, in contrast to SG which is block diagonal on the tasks. We denote the joint distribution of $\mathbf{f}_F$ and the latent function f_{T_j} of an unseen data point x_{T_j} from task j by $\mathbf{f}_\mathbf{x} = \{\mathbf{f}_F, f_{T_j}\}$, whose prior is given by:

$$\begin{bmatrix} \mathbf{f}_F \\ f_{T_j} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K^t \otimes K^x_{D,D} & \mathbf{k}^t_j \otimes \mathbf{k}^x_{D,T_j} \\ \left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{D,T_j}\right)^T & k^t_{jj} k^x_{T_j,T_j} \end{bmatrix}\right), \tag{4.16}$$

where $\mathbf{k}^t_j$ is the j^{th} column of the task covariance matrix K^t, and k^t_{jj} its jj^{th} element, $\mathbf{k}^x_{D,T_j}$ is the covariance vector between points in set D and x_{T_j}, and $k^x_{T_jT_j}$ is the marginal variance of x_{T_j}, and finally we will use Q_D to denote the graph Laplacian between all points in set D. As in SSGP, conditioning on $\mathcal{G}_F$ the prior of $\mathbf{f}_\mathbf{x}$ will be proportional to:

$$p(\mathbf{f}_\mathbf{x}|\mathcal{G}_F) \propto \exp\left(-\frac{1}{2}\mathbf{f}_\mathbf{x}^T \tilde{\Sigma}^{-1}_{\mathbf{xx}} \mathbf{f}_\mathbf{x}\right), \tag{4.17}$$

where

$$\tilde{\Sigma}^{-1}_{\mathbf{xx}} = \begin{bmatrix} K^t \otimes K^x_{D,D} & \mathbf{k}^t_j \otimes \mathbf{k}^x_{D,T_j} \\ \left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{D,T_j}\right)^T & k^t_{jj} k^x_{T_jT_j} \end{bmatrix}^{-1} + \begin{bmatrix} K^t \otimes Q_D & 0 \\ 0 & 0 \end{bmatrix}.$$

It can be proved that for any finite collection of data points $\mathbf{x}$, the random variables $\mathbf{f}_\mathbf{x} = \{f(x)\}_{x\in\mathbf{x}}$ conditioned on $\mathcal{G}_F$, have a multivariate normal distribution $\mathcal{N}(0, \tilde{K}^x_F)$, where $\tilde{K}^x_F$ is the covariance matrix whose elements are given by evaluating the following covariance function $\tilde{k}_F : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, thus[3]

$$\tilde{k}_F(x,z) = k^t_{ij} k^x(x,z) - \left(\mathbf{k}^t_i \otimes \mathbf{k}^x_{D,x}\right)^T \left(I + (K^t \otimes Q_D)(K^t \otimes K^x_{D,D})\right)^{-1} (K^t \otimes Q_D)\left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{Dz}\right). \tag{4.18}$$

Here x and z are points from task i and j respectively, k^t_{ij} is the ij^{th} element of K^t, and $\mathbf{k}^x_{D,x}$ & $\mathbf{k}^x_{D,z}$ is the covariance vector between points in set D and x & z respectively. Similarly to the SG formulation, the FG model is completely determined by the special structure in the covariance function in equation 4.18. The prior of the latent functions $\mathbf{f}$ on the *labelled* points from all tasks in the *flexible geometry* formulation will be given by,

$$p(\mathbf{f}|D) \sim \mathcal{GP}(0, \tilde{K}_F). \tag{4.19}$$

The main difference between the FG and SG methods is that in FG the multi-task setting is induced directly on the geometry of the tasks, through the Kronecker

[3]The derivation of the covariance function follows a similar treatment to the SS-GP and can be found in appendix B.1.

factorisation of the task and the graph Laplacian matrix. As a consequence, in FG unlabelled data contribute to the learning of the task covariance matrix, in contrast to SG were only labelled data contribute.

4.3.3 Graph Laplacian

Of central importance to all of these semi-supervised settings is the matrix Q, which encodes the assumptions that the model is based on. The construction of the matrix Q is based on the fact that a data adjacency matrix can act as an empirical substitute of the geometry of the data distribution (Belkin and Niyogi, 2003). The usual choice for Q is the graph Laplacian, $Q = L^p$ which derives from the fact that the Laplace Beltrami operator can be approximated by a weighted Laplacian of an adjacency graph (Belkin and Niyogi, 2002). While in this work we focus on the Laplacian matrix, we observe in passing the possible use of novel graph regularisers as in Chu et al. (2007). Constructing the adjacency matrix of the graph Laplacian is performed in three steps (Belkin and Niyogi, 2002).

1. In the first step we construct the adjacency graph, that is determining which nodes (data points) are connected. This can be done either from a geometric or nearest neighbour perspective. In the geometric approach, two nodes x_i, x_j are connected by an edge if the Euclidean distance of the two points is smaller than a parameter ε, $||x_i - x_j||^2 < \varepsilon$. In the nearest neighbour approach, two nodes are connected if they are among the k nearest neighbours.

2. In the second step the entries of the adjacency matrix are weighted. This can can be done either with the Heat kernel $W_{ij} = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma_l^2}\right)$, or in the simpler approach where $W_{ij} = 1$ if two nodes are connected and zero otherwise.

3. In the third step we compute the graph Laplacian as $L = D - W$, and $D_{ii} = \sum_j W_{ji}$.

Another option would be to use the *normalised* graph Laplacian $\tilde{L}$ computed as $\tilde{L} = D^{-1/2} L D^{-1/2}$, which has been shown to present better or competitive results with the simple Laplacian (see Von Luxburg et al. (2008) for theoretical guarantees of the normalised Laplacian). A drawback of both variations of the first step is that the parameter ε, and the number of nearest neighbours k have to be tuned by cross-validation and are not amenable to gradient based optimisation methods. To overcome this problem in the geometric setting, we first notice that selecting the points within a certain sphere of radius ε can be seen as multiplying the Euclidean norm by a step function. We propose to approximate this step function by a sigmoid *edge function c*

$$c_{ij}(\varepsilon) = \frac{1}{1 + e^{\left(-\varepsilon + ||x_i - x_j||^2\right)}}, \tag{4.20}$$

making $\varepsilon > 0$ a parameter which can be estimated using gradient optimisation methods. Thus, including the edge function in the geometric approach of constructing the graph adjacency matrix we can write the first and the second step as $W_{ij} = c_{ij} \exp\left(-\frac{||x_i - x_j||^2}{2\sigma_l^2}\right)$ with the Heat kernel or $W_{ij} = c_{ij}$ in the simple approach. In the experimental part, we present results for both the nearest neighbour and the geometric approach, selecting the threshold in the geometric approach by optimising the parameter ε.

4.3.4 Likelihood

While the previous sections described how to incorporate unlabelled data and task similarity within a unified prior distribution, the choice of the likelihood (noise model) depends on the specific learning task at hand. In the following, we will focus on regression and binary classification tasks, although extension to the multi-class scenario is in principle straightforward (see for example Girolami and Rogers (2006) for a variational treatment on multi-class GP classification). The prior distribution of each model, the covariance structure, and the parameterisation employed are summarised in table 4.1.

In *regression tasks* it is usually assumed that the output are noisy corrupted versions of the stochastic process, thus the noise model will be $p(y_{ij}|f_{ij}) = \mathcal{N}(f_{ij}, \sigma_{n_j}^2)$, where $\sigma_{n_j}^2$ is the variance of the noise of the j^{th} task. In the SS-MTL scenario the noise model becomes $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, D \otimes I)$, where $D_{jj} = \sigma_{n_j}^2$ is the noise of the j^{th} task, and I is the identity matrix of appropriate size.

In *classification tasks* we will use the *probit* regression noise model, where the likelihood is given by the cumulative distribution of a Gaussian (Rasmussen and Williams, 2005)

$$p(y|f) = \int_{-\infty}^{yf} \mathcal{N}(u|0,1)\,\mathrm{d}u.$$

The SS-MTL model applied to probit classification problems is intractable due to the non-conjugacy of the probit likelihood with the GP prior. This can be handled by employing stochastic or deterministic approximations to the non-Gaussian posterior (see chapter 3). In this chapter we employ the EP approximation (Opper and Winther, 2000;

Table 4.1: Prior distribution and parameterisation for the SS-MTL models.

Model	*Prior*	*Covariance*	*Laplacian*
SS-MTL IND	$p(\mathbf{f}\|D) = \prod_{j=1}^{M} \mathcal{N}_{\mathbf{f}_j}(0, K_{jj}^{i})$	$K_{jj}^{i} = \frac{1}{\gamma_A}\tilde{K}_{jj}^{x}$	$Q_j^i = \frac{\gamma_I}{\gamma_A} Q_j$
SS-MTL SG	$p(\mathbf{f}\|D) = \mathcal{N}(0, K^s)$	$K^s = K^t \otimes \left(\frac{1}{\gamma_A}\tilde{K}^x\right)$	$Q^s = \frac{\gamma_I}{\gamma_A} Q_S$
SS-MTL FG	$p(\mathbf{f}\|D) = \mathcal{N}(0, K^f)$	$K^f = \frac{1}{\gamma_A}\tilde{K}_F$	$Q^f = \frac{\gamma_I}{\gamma_A} Q_D$

Minka, 2001), as it produces accurate approximations to the posterior with lower computational requirements than stochastic methods such as Markov Chain Monte Carlo. A detailed description of the EP approximation for binary GP classification which naturally extends to the SS-MTL paradigm proposed in this work can be found in Rasmussen and Williams (2005). Here, we only provide a sketch of the EP algorithm while for a full description we refer to Rasmussen and Williams (2005). The EP algorithm approximates the likelihood $p(y_i|f_i)$ by an un-normalised Gaussian distribution over the latent variables f_i, where the product of these *local likelihood approximations* is computed as,

$$\prod_{i=1}^{N^l} p(y_i|f_i) \simeq \prod_{i=1}^{N^l} t_i(f_i|\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) = \mathcal{N}(\tilde{\mu}, \tilde{\Sigma}) \prod_{i=1}^{N^l} \tilde{Z}_i, \tag{4.21}$$

where $\tilde{\mu}$ is a vector with μ_i elements and $\tilde{\Sigma}$ is diagonal with $\tilde{\Sigma}_{ii} = \tilde{\sigma}_i$, and we have used N^l to emphasise that the latent function is evaluated only at the labelled data points. The approximated posterior over the latent functions is given by (Rasmussen and Williams, 2005)

$$q(\mathbf{f}|\mathbf{y}) = \mathcal{N}(\mu, \Sigma), \tag{4.22}$$

where $\mu = \Sigma\tilde{\Sigma}^{-1}\tilde{\mu}$ and $\Sigma = \left(K^{-1} + \tilde{\Sigma}^{-1}\right)^{-1}$, and we have used the general K to refer to all covariance matrices for each scenario we have described in sections 4.3.1 and 4.3.2. EP then proceeds by estimating the individual t_i functions sequentially each time leaving out the current t_i and computing the *cavity distribution* $q_{-i}(f_i)$ which is combined with true likelihood term $p(y_i|f_i)$ to form a non-Gaussian distribution. On the last steps this non-Gaussian distribution is approximated by a Gaussian whose moments are matched with the parameters of t_i to have the same marginal distribution with the product of $q_{-i}(f_i)$ and $p(y_i|f_i)$.

4.3.5 Hyperparameter estimation

The *Inductive transfer SS-MTL* GP models, SG and FG, of section 4.3.2 have four groups of hyper-parameters related to the structure of the covariance function. The *first group* of hyper-parameters is the one related to the task covariance matrix, which is denoted by θ^t. The *second group* are the hyper-parameters θ^x of the data covariance function $K^x(x,z)$ that depends on the choice of the covariance function. The *third group* of hyper-parameters is related to the computation of the graph regulariser matrix Q, and will be denoted by θ^l. Moreover, we add two more parameters γ_I and γ_A as $Q = \frac{\gamma_I}{\gamma_A} L^p$ and $\frac{1}{\gamma_A} K$ (see table 4.1) to balance the ambient and intrinsic covariance. This additional parameterisation has its roots in the kernel based construction on the data-dependent norms (equation 4.3) and it has also been employed in Sindhwani et al. (2007). We include γ_I and γ_A in a separate group of hyperparameters $\theta^g = \{\gamma_I, \gamma_A,\}$ as the *fourth group*. Hence, for the geometric construction we will have that $\theta^l = \{\varepsilon, \sigma_l\}$, while in the NN construction the number k of nearest neighbours will be set by preliminary analysis, and we will have that $\theta^l = \{\sigma_l\}$.

The *SS-MTL IND* GP model from the parameter transfer category of SS-MTL of section 4.3.1 has only three groups of hyperparameters, since there is no task covariance matrix to be computed. Hence, the SS-MTL IND model will have only θ^x, θ^l, and θ^g.

Moreover, we remove the scale redundancy inherent in the Kronecker product by fixing the task covariance matrix to be a correlation matrix according to Rebonato and Jäckel (2000)[4]. In regression problems the variance of the noise σ_n needs to be estimated which adds an extra parameter. All four groups of hyper-parameters θ^x, θ^t, θ^l and θ^g are estimated by approximate type II maximum likelihood using the log marginal likelihood (Rasmussen and Williams, 2005). The gradients for each set of hyperparameters and for each model can be found in appendix B.2.

4.4 Experiments

We test the behaviour of the proposed approach on a toy data set and on three multi-task classification problems by comparing the performance of the proposed methods in terms of the Area Under the Curve (AUC) they achieve. To demonstrate the advantages of leveraging both unlabelled data and task relatedness, we report the results that

[4]For details about the construction and the hyperparameters of the task correlation matrix see Chapter 3.

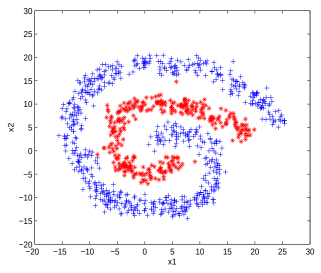
Table 4.2: Gaussian Process Learning frameworks

Learning framework	**Single task**	**Multi-task**	
	No transfer	*Parameter transfer*	*Inductive transfer*
Supervised	STL	MTL-IND	MTL-IMC
Semi-supervised	SS-STL	SS-MTL IND	SG , FG

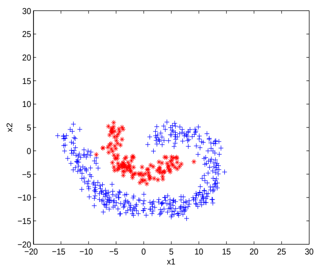
were obtained on a number of baselines and competing methods. In detail, we compare with the SSGP model of Sindhwani et al. (2007)(SS-STL), the supervised IMC model of Bonilla et al. (2008) adapted for classification (MTL-IMC) (chapter 3), multi-task GPs with shared hyperparameters of the covariance functions (MTL-IND) similarly to Lawrence and Platt (2004), and single-task learning with GPs (STL-GP) (Rasmussen and Williams, 2005). To provide comparisons with another SS-MTL method based on GPs, we re-implemented and adapted to the classification scenario the method of Zhang and Yeung (2009) discussed in section 4.3.1. This approach, which was originally proposed for regression problems, couples different tasks by sharing a common hyper-prior over the hyperparameters of the covariance function. The only difference between the algorithm proposed in Zhang and Yeung (2009) and the implementation in this work is that we couple the different GPs by constraining them to share the same hyperparameters which we estimate by type *II* ML, and it will be referred to as the SS-MTL IND method in general and as SS-IND in the tables of the experiments. Also note that in order to compare the results of the different methods we used the t-test.

Additionally, in the semi-supervised methods of SS-STL, FG and SG we present results with the geometric as well as with the nearest neighbour construction of the Laplacian, which we refer to as 'G' and 'NN' respectively; for the SS-MTL IND model we present results only with the geometric construction. See table 6.1 for a summary of acronyms used for Gaussian Process methods.

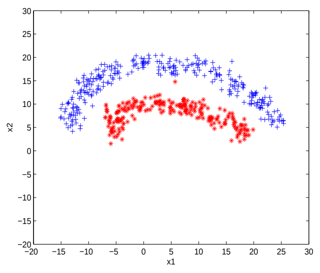
The experimental setup is identical for all three classification problems and the toy data set. In all semi-supervised methods 50 unlabelled points per task were used in the semi-supervised learning. The number of unlabelled data was determined by preliminary analysis, which revealed that in most of the cases by inclusion of a higher number unlabelled data did not result in any significant improvement and due to the large number of experiments it was fixed to 50. All types of parameters are estimated through evidence maximisation, except for the number of nearest neighbours (NN), which were



(a)



(b)



(c)

Figure 4.2: Scatter plot of the Spiral Toy data set: *left* complete data distribution, *center* first task, *right* second task.

determined by preliminary analysis, and are reported in the last rows of tables 4.3,4.4, and 4.6. Additionally, in the construction of the graph Laplacian we adopt the simple approach where $W_{ij} = 1$ if two nodes are connected and zero otherwise. For each size of training (data points per task), the experiments were repeated 25 times. Note that the mean and standard deviations reported in tables 4.3, 4.4, and 4.6 are computed by first averaging over the tasks for each data partition and then over the repetitions.

In the spam detection and the sentiment analysis problems as the input covariance function K^x we employ the linear kernel, which is usually applied in text applications of high input dimensionality. For both of these text classification applications we use the *bag of words* model (Lewis, 1998). This model simply uses as features the frequency of appearance of all words without taking into account the order of the words or the grammar and the syntax of the document. Additionally, it is known that this model compared with other more sophisticated methods of the Natural Language processing literature produces similar results (Lewis and Jones, 1996), which justifies our choice. For the toy data set and the letters classification problem we employ the *RBF* covariance function. Another distance measure that has been tried in the letters data set, where each letter is represented by 8×16 binary pixel images, was the Hamming distance between the two binary vectors; however, the RBF covariance function that computes the squared Euclidean distance seemed to produce similar or better results.

4.4.1 Spiral Toy data set

To investigate the behaviour of the proposed approach, we first tested it on the toy data set shown in figure 4.2. The toy data set consists of two tasks, shown in figures 4.2.b and 4.2.c respectively. To demonstrate the functionality of each method, in figure 4.3

we show the decision boundaries we obtained by training the MTL, the SS-STL, the SS-MTL IND and the SS-MTL FG models with 16 labelled data points from each task and 100 and 68 unlabelled data points from task 1 and 2 respectively. Subfigures 4.3 (a) and (b) show the decision boundaries obtained from the simple MTL classifier: this does not find task correlations, and since labels are too few it produces a globally poor classifier. In the following row, subfigures 4.3 (c) and (d) show the decision boundaries obtained by training the SSGP classifier on each task separately, which accurately leverages information from unlabelled data but cannot account for information learned in the other tasks. In the third row, subfigures 4.3 (e) and (f) show the boundaries for the SS-MTL IND model, which highlights the fact that this form of SS-MTL learning results in a weaker form of transfer since data from other tasks only affect the parameter estimation process and not the prediction. Note, that although the SS-MTL IND does not estimate as effective as the SS-STL model the high density areas of each class, it does find correctly the boundaries at 0.5 threshold, except in the case of class “-1” for task 1 where it misses some unlabelled points. In the last row, subfigures 4.3 (g) and (h) show the decision boundaries created by the FG-G model. The contour plot of the decision boundaries show that the SS-MTL classifier from the Inductive category takes into account the geometry of the unlabelled data while detecting that the two tasks are correlated. Note that both variations SG&FG as well as with the geometric or the nearest-neighbour construction of the Laplacian produced similar results, for brevity we present only FG-G. To compare the performance of these models on the spiral toy data set we generated 300 data points from each task equally separated between the classes and we trained them with 2, 4, 8, and 10 labelled data.The experiments were repeated 25 times, each time by randomly selecting the labelled data and 50 unlabelled data. The obtained results for this set of experiments are shown in figure 4.4. Comparing the SS-MTL methods it is noted that the parameter transfer approach SS-MTL IND produces a lower average AUC of approximately $25-30\%$ than that of the Inductive transfer approaches SG and FG. It is observed that the SG method achieves a higher AUC than the FG model for 2 and 4 labelled data points, whereas both of these models outperform all the others. For 8 and 10 labelled data all methods achieve an AUC higher than 90%. Note that the differences between the SG and FG models were not statistically significant at 0.05 p-value, whereas differences between the SG and the FG model and all others were statistically significant at 0.05 p-value.

In addition, it is worth noticing that the SS-STL method performs better than MTL-IMC and most importantly than the SS-IND method for 2 labelled points, which was

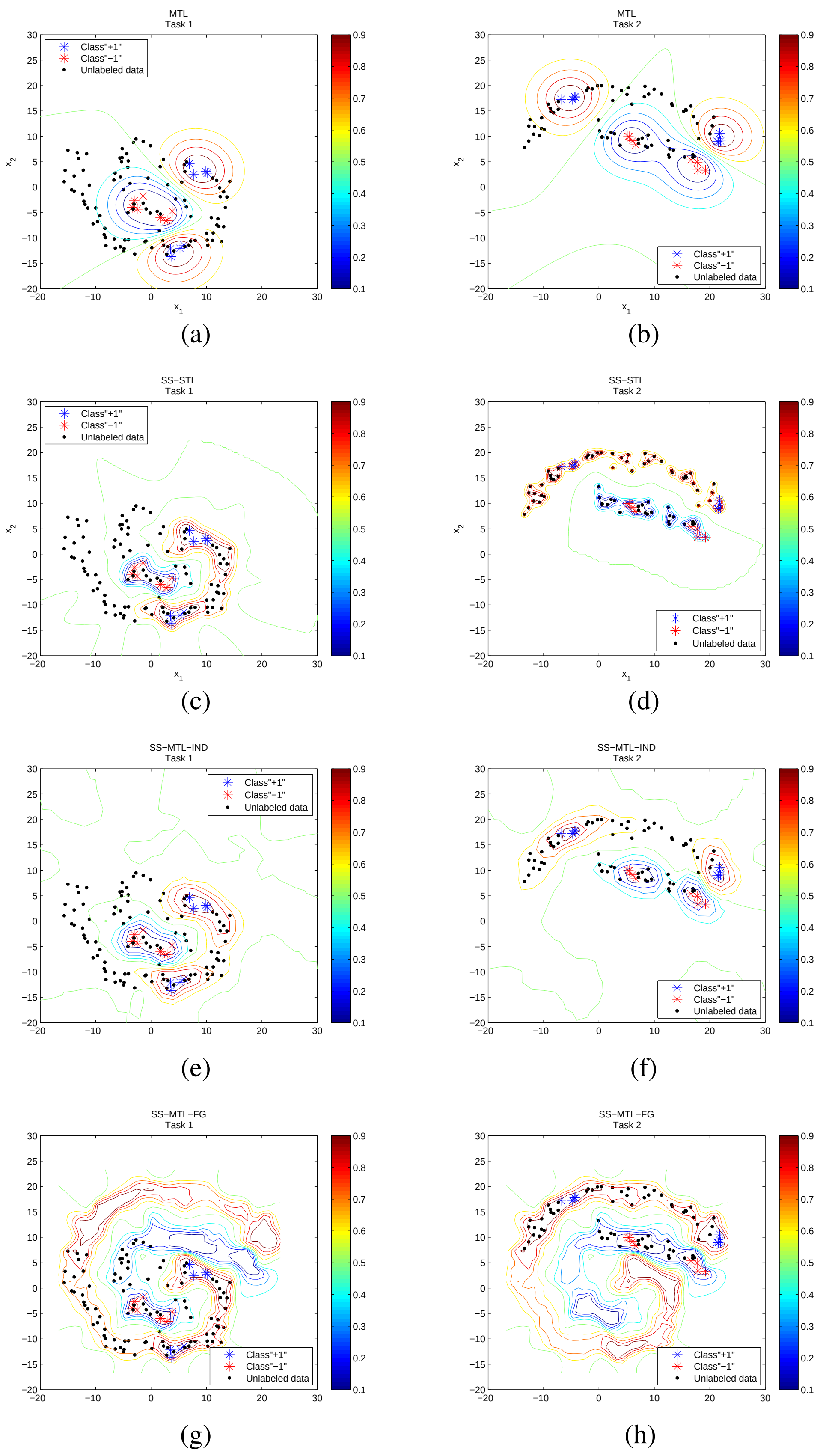


Figure 4.3: Results on the Spiral toy data set. *1st row* : MTL; *2nd row* : SS-STL; *3rd row* : SS-MTL IND only; *4th row* : SS-MTL FG results; Notice that SS-MTL FG is capable of effectively transferring knowledge from the tasks and the unlabelled data. Left column shows results for Task 1, and right column shows results for Task 2.

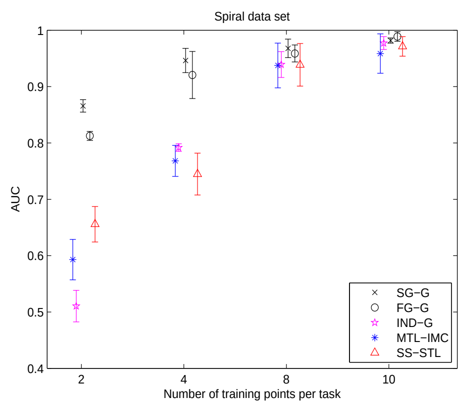


Figure 4.4: Performance on spiral toy problem in terms of AUC.

not expected; although note that for 4 labelled points the SS-MTL IND method performs better that the SS-STL method satisfying our expectations.

4.4.2 Spam Detection

The spam detection data set comes from the ECML/PKDD 2006 Discovery Challenge[5]. Data are collected from three different users, with 2500 mails available for each one which are split into 50% spam, and 50% non-spam.

The results of the experiments for this data set are summarised in table 4.3. In general, for small training sizes the SS-MTL methods SG and FG, both with the geometric (G) or the nearest neighbour (NN) Laplacian, produce a higher mean AUC than the SS-MTL IND, both MTL, the STL and the SS-STL methods. We note that in 4 out of 6 training data partitions the best AUC is produced by the SG formulation with the geometric construction. Note that the differences between the SG-G method and the SS-IND-G, MTL-IMC, MTL-IND, and all STL semi-supervised or not, were statistically signicant at 0.05 p-value for 10, 20, 30, and 50 labelled data points. Additionally, comparing the means between SG and FG we found that they were not statistically signicant at 0.05 p-value, which indicates that all SS-MTL methods perform equally well. For few training points per task we observe a very significant improvement of the SS-MTL algorithms over MTL and SS-STL. As expected, this difference decreases as the number of labelled samples increases, whereas for 100 labelled data point per task the best performance is achieved by the MTL method.

Comparing the methods of the parameter transfer approach, the MTL-IND and

[5]http://www.ecmlpkdd2006.org/challenge.html.

Table 4.3: *AUC of the Spam data set*. Double lines separate the learning frameworks, SS-MTL, MTL, and STL with SS-STL.

METHOD	*No. of training data points per task* 10	20	30	40	50	100
			AUC			
SG-G	**87.81 ± 5.62**	**92.64 ± 2.80**	**94.04 ± 2.14**	94.99 ± 1.82	96.28 ± 1.00	96.60 ± 0.47
FG-G	83.60 ± 9.25	92.32 ± 3.18	93.30 ± 3.89	**95.18 ± 1.24**	94.44 ± 1.40	96.09 ± 0.65
SG-NN	85.42 ± 5.42	91.57 ± 3.36	93.24 ± 1.92	94.61 ± 1.51	**95.79 ± 0.82**	96.33 ± 1.23
FG-NN	85.56 ± 5.07	91.38 ± 3.09	93.65 ± 2.10	94.39 ± 1.66	95.77 ± 0.93	96.69 ± 0.74
SS-IND-G	81.47 ± 7.74	87.38 ± 3.33	90.71 ± 1.82	92.72 ± 1.17	93.03 ± 1.75	95.43 ± 0.54
MTL	82.21 ± 7.96	90.97 ± 2.13	93.39 ± 1.70	94.14 ± 1.45	95.34 ± 0.75	**96.94 ± 0.44**
MTL-IND	80.89 ± 10.49	86.63 ± 0.80	91.82 ±1.87	93.02 ± 1.57	94.63 ± 0.91	96.68 ± 0.55
SS-STL-NN	77.01 ± 6.43	85.94 ± 4.53	90.23 ± 2.07	91.75 ± 1.87	93.81 ± 1.02	96.16 ± 0.66
SS-STL-G	76.14 ± 6.36	84.96 ± 3.87	89.67 ± 2.17	91.75 ± 2.07	92.92 ± 1.63	95.02 ± 1.15
STL	64.62 ± 6.17	71.45 ± 8.44	81.94 ± 6.17	85.13 ± 6.69	91.99 ± 4.80	96.36 ± 1.73
NN	3	6	9	12	15	20

its SS analog of SS-MTL IND, we see that for few labelled points (10, and 20) the SS-MTL IND produces a higher AUC than the MTL-IND, while for labelled points higher than 20 we notice that although there are high error bars MTL-IND produces a higher mean. We are not able to find a justified explanation for this type of behaviour, although comparing the Inductive transfer approaches we notice that there also are some cases where the MTL method performs better than the SS-MTL methods SG and FG (*e.g.* for 30 labelled data where MTL does better than FG-G and SG-NN, for 50 labelled points MTL outperforms FG-G, and for 100 points where MTL produces the highest average mean).

4.4.3 Sentiment Analysis

The sentiment classification data set[6], previously used in Blitzer et al. (2007), is comprised of 1000 positive, and 1000 negative product reviews about four types of products, taken from *amazon.com*. The products/tasks for which reviews are available are: *books, dvds, electronics*, and *kitchen appliances*.

The results obtained for this data set are presented in table 4.4. Semi-supervised multi-task methods and particularly SG produce the best performance for all training sizes. Comparing the SS-MTL methods we see that in 5 out of 6 sizes of labelled training data the SG formulation outperforms the FG and the SS-MTL IND formulations. Further analysis revealed that the differences in performance between the SG-NN method and all MTL, SS-STL and STL were statistically signicant at 0.05 p-value. We note that the STL and SS-STL models perform extremely poorly, while all other

[6]http://www.cs.jhu.edu/∼mdredze/datasets/sentiment/.

Table 4.4: *AUC of the Amazon data set.* Double lines separate the learning frameworks, SS-MTL, MTL, and STL with SS-STL

METHOD	*No. of training data points per task*					
	20	30	40	50	60	90
AUC						
SG-G	70.45 ± 3.78	$\mathbf{73.61 \pm 2.08}$	76.00 ± 1.68	77.58 ± 1.59	78.84 ± 1.42	81.53 ± 1.21
FG-G	69.82 ± 3.86	72.99 ± 2.14	75.82 ± 1.79	77.32 ± 1.57	78.55 ± 1.52	81.32 ± 1.13
SG-NN	$\mathbf{70.55 \pm 4.14}$	73.55 ± 2.11	76.03 ± 1.95	$\mathbf{78.12 \pm 1.55}$	$\mathbf{79.04 \pm 1.36}$	$\mathbf{81.83 \pm 1.10}$
FG-NN	68.96 ± 4.68	73.08 ± 2.30	$\mathbf{76.19 \pm 1.67}$	76.99 ± 2.71	78.57 ± 2.79	81.77 ± 1.07
SS-IND-G	61.39 ± 4.35	66.55 ± 2.48	69.03 ± 2.07	71.03 ± 1.46	72.69 ± 1.18	75.19 ± 1.30
MTL	61.56 ± 2.85	66.18 ± 3.93	72.13 ± 4.35	75.59 ± 3.68	77.50 ± 2.84	81.33 ± 1.24
MTL-IND	61.60 ± 2.98	66.10 ± 3.30	69.44 ± 2.85	71.50 ± 2.51	73.55 ± 1.56	76.20 ± 1.09
SS-STL-NN	63.35 ± 2.29	66.63 ± 2.14	69.71 ± 2.00	71.77 ± 2.23	73.62 ± 1.55	76.34 ± 1.16
SS-STL-G	59.57 ± 2.87	62.84 ± 3.15	66.12 ± 2.68	69.69 ± 3.41	72.32 ± 2.50	75.59 ± 1.76
STL	53.75 ± 1.85	65.93 ± 2.24	68.79 ± 2.08	71.05 ± 2.28	73.18 ± 1.77	76.10 ± 1.18
NN	3	4	5	6	7	10

semi-supervised methods and MTL offer a significant improvement compared to them for few labelled data points. Comparing the SS-MTL IND with the MTL-IND we note that the inclusion of unlabelled data points does not lead to any improvements. On the other hand, comparing the SS-MTL IND with the SS-STL and the MTL-IND with the STL methods we observe that both the SS-MTL IND and MTL-IND methods are always better than the SS-STL and STL respectively; this observation satisfies the assumption of Lawrence and Platt (2004) in the semi-supervised and the standard supervised case that the search for a common across tasks set of hyperparameters can be beneficial.

4.4.4 Letters Classification

The letters classification problem[7] consists of eight binary classification tasks between different handwritten letters. The number of samples from each letter of every task are reported in table 4.5. Differently from the spam classification and the sentiment analysis problems, the letters data set has highly imbalanced classes and there are tasks that are anti-correlated, for example task *g/y* with task *a/g* are anti-correlated since letter "*g*" is assigned to different classes. On the other hand, there are tasks that are highly correlated because the same letter appears in the same class of another task, for example tasks 3 and 8 due to letter "*n*", and tasks 4 and 6 because of letter "*a*", while there are other tasks which by simple inspection we can argue that there is no reason to assume that are correlated, for example task *c/e* with *f/t*.

The performance of the methods on the letter data set is reported in table 4.6. In

[7]http://multitask.cs.berkeley.edu/

this data set where correlations between some of the tasks are non-existent or even antithetic, it is noticed that the best performance in terms of AUC is achieved by MTL-IND. Note that the differences in performance in terms of AUC between MTL-IND and all other methods except STL, for 30, 40 and 50 data points, and SS-STL-NN for 40 data points, were statistically significant at 0.05 p-value. This is one of the simplest forms of transfer learning within the GP framework and taking into account that there are substantial differences between the tasks, it is our opinion that MTL-IND produced the best average AUC because the models from the Inductive category were not able to learn correctly the task correlations. Another fact that complements our argument is that among the SS-MTL methods the best performance is achieved by the FG-G formulation, under which unlabelled data assist in the learning of the task covariance matrix. It is interesting to note that in terms of accuracy (see table B.4 in appendix B.2.4) the SS-MTL algorithms offer a significant improvement over the conventional MTL, SSGP and STL algorithms for all training set sizes except for 30 labelled data points per task where STL performs better. In addition, related to the parameters transfer approach we observe that the inclusion of the unlabelled data during the training in the SS-MTL IND model in essence hurts its performance, since for all training sizes the average AUC of SS-MTL IND is lower than that of MTL-IND by $2-1\%$; however it should be noted that as in the two previous data sets the results for the SS-MTL IND model were obtained by the geometric construction of the Laplacian and it could be argued that the NN construction could improve these results. In contrast, in the Inductive transfer type of approach this behaviour is not observed and the SS analogs SG and FG produce higher AUC than the MTL-IMC for all training sets.

Table 4.5: Description of the Letters data set; each column is a task showing the two letters as well as the corresponding number of examples per character (class)

Task	1	2	3	4	5	6	7	8
Letter	**c**	**g**	**m**	**a**	**i**	**a**	**f**	**h**
No. of data	2017	2460	1596	4016	4895	4016	918	858
Letter	**e**	**y**	**n**	**g**	**j**	**o**	**t**	**n**
No. of data	4928	1218	5004	2460	188	3880	2131	5004

Table 4.6: *AUC on the Letters data set*. Double lines separate the learning frameworks, SS-MTL, MTL, and STL with SS-STL

Method	No. of training data points per task				
	10	20	30	40	50
AUC					
SG-G	82.71 ± 6.12	88.47 ± 1.01	90.98 ± 0.72	91.46 ± 0.81	92.51 ± 0.70
FG-G	84.13 ± 2.10	89.30 ± 1.05	91.61 ± 0.89	92.52 ± 0.62	93.39 ± 0.48
SG-NN	78.04 ± 0.23	86.91 ± 2.22	89.57 ± 6.76	91.15 ± 2.57	92.36 ± 1.89
FG-NN	82.03 ± 4.77	88.28 ± 1.88	90.32 ± 1.35	91.71 ± 1.31	92.73 ± 0.70
SS-IND-G	84.32 ± 1.86	88.36 ± 1.28	90.85 ± 0.91	91.75 ± 0.59	92.75 ± 0.51
MTL	81.33 ± 2.40	85.64 ± 1.89	88.76 ± 1.46	90.46 ± 1.05	91.69 ± 0.90
MTL-IND	**86.09 ± 1.92**	**90.15 ± 0.92**	**92.15 ± 0.77**	**92.90 ± 0.56**	93.70 ± 0.50
SS-STL-NN	80.26 ± 3.38	86.85 ± 3.27	91.07 ± 2.03	92.48 ± 1.04	92.89 ± 1.52
SS-STL-G	79.93 ± 2.71	87.12 ± 2.29	90.77 ± 1.25	91.99 ± 0.97	93.27 ± 0.74
STL	83.85 ± 2.93	88.70 ± 2.77	91.89 ± 1.40	92.88 ± 0.75	**93.77 ± 0.54**
NN	3	4	5	6	7

4.5 Extensions in Transfer Learning

We describe here how the FG formulation can be adapted to handle two different scenarios that fall inside the general transfer learning framework (Pan and Yang, 2010), potentially providing a promising unifying framework for transfer learning with GPs. The first scenario that we consider is *Domain Adaptation* (DA) (Daumé III and Marcu, 2006), which aims at making predictions on a target task for which no labelled training data are available by transferring learning from a different task, usually referred to as the source task, that contains labelled data. In *Self-taught Learning* (SeTL) (Raina et al., 2007), the second scenario that we are interested in, unlabelled data from an auxiliary task are used to improve predictions on a target task for which limited annotated data are accessible. Both scenarios involve a task $\mathcal{T}_l$ that has labelled and unlabelled data $L_l = \{X_{L_l}, Y_{L_l}\}$ and $U_l = \{X_{U_l}\}$ with $D_l = L_l \cup U_l$, and a task $\mathcal{T}_u$ for which only unlabelled data are available $D_u = U_u = \{X_{U_u}\}$, and $A = \{X_{L_l}, X_{U_l}, X_{U_u}\}$. Both approaches involve transfer of learning between tasks and between labelled and unlabelled data, which bears a close resemblance with the scenario of semi-supervised multi-task learning that was the main core of this chapter. The difference between SS-MTL and DA, SeTL is that in DA and SeTL there are no available labels for some tasks and that those two methods aim at improving performance on one target task and not on all tasks simultaneously as in SS-MTL. The main difference between DA and SeTL is that in DA labels are missing on the target task, whereas in SeTL labels are missing in the source task. Hence, in DA the labelled task is the source task and the unlabelled task is the target, while in SeTL is the opposite. In the following, for explanatory reasons we concentrate on one labelled and one unlabelled task, while extension to multiple source or target tasks using the formulation that we propose is straightforward.

In a similar fashion to the FG formulation, the vector of latent functions $\mathbf{f}_A = \{\mathbf{f}_{L_l}, \mathbf{f}_{L_u}, \mathbf{f}_{U_u}\}$, conditioned on $\mathcal{G}_A = \{\mathcal{G}_{L_l}, \mathcal{G}_{L_u}, \mathcal{G}_{Uu}\}$ will be proportional to $p(\mathbf{f}_A|\mathcal{G}_A) \propto \exp(-\frac{1}{2}\mathbf{f}_A^T(K_A^t \otimes Q_A)\mathbf{f}_A)p(\mathbf{f}_A)$ where $K_A^t = \begin{bmatrix} k_{ll}^t & k_{lu}^t \\ k_{lu}^t & k_{uu}^t \end{bmatrix}$, and K_A^x and Q_A are constructed using all data points in D_l and D_u. The difference with FG is that during the learning phase the latent function can be evaluated only at the visible locations, that is the input points of the labelled task $\mathcal{T}_l$. Thus, the *data-dependent* prior of the labelled task will be given by a zero-mean Gaussian distribution with covariance matrix given by,

$$\tilde{K}_{L_lL_l} = k_{ll}^t K_{L_lL_l}^x - (\mathbf{k}_l^t \otimes \mathbf{k}_{L_lA}^x)^T \left(I + (K_A^t \otimes Q_A)(K_A^t \otimes K_A^x)\right)^{-1} (K_A^t \otimes Q_A)(\mathbf{k}_l^t \otimes \mathbf{k}_{L_lA}^x), \quad (4.23)$$

where $K_{L_lL_l}^x$ is the covariance matrix between the points in X_{L_l}, $\mathbf{k}_l^t$ is task correlation vector between the labelled and unlabelled task, and $\mathbf{k}_{L_lA}^x$ is covariance matrix between X_{L_l} and A. This formulation would allow to first inform the learning process about the distribution of the unlabelled task through the graph Laplacian and secondly to learn the level of correlation between the tasks, k_{lu}^t.

Preliminary experimental testing of this approach in the DA scenario for the Spam classification and the Sentiment analysis problems revealed that our approach was not performing better than the baseline of training only with labelled data from the source task and predicting on the target task. Some of the problems that have been identified so far is that the information contained in the graph Laplacian was not strong enough to drive the gradients of the task hyperparameters related to the unlabelled task k_{lu}^t and k_{uu}^t to meaningful results. We believe that further constraints will have to be imposed to regularise the model in the absence of labels in certain tasks. A possible constraint that we wish to experiment with in the future is to force the model to use unlabelled data, rather that estimating the contribution of them through the graph Laplacian, which would simply mean dropping the dependence of the model to hyperparameters θ^g.

4.6 Conclusions

We have presented a novel method for transfer learning using GP priors. We have implemented and extensively tested the approach in a MTL scenario. However we also indicate how the same approach could be used in different transfer learning scenarios such as DA and SeTL, potentially providing a powerful unifying framework for transfer learning. Within the MTL scenario, the crucial issue is deciding how to incorporate unlabelled information in the transfer of learning. Similarly to the multi-task scenario, semi-supervised learning can also be separated into two forms of transfer, the *parameter* SS-MTL, and the *Inductive* SS-MTL. In the parameter transfer multiple independent GPs with data-dependent priors are coupled by sharing the same hyperparameters. The Inductive transfer approach that we employ is based on the IMC approach of Cressie (1993) and Bonilla et al. (2008). We presented two possible approaches: in the static geometry case, tasks are assumed to have independent geometric structures, so that transfer of geometry can only happen through the correlations between tasks. Alternatively, in the flexible geometry case, labelled and unlabelled data from all tasks are used to determine the geometry.

Thorough experiments provided more in depth understanding about the workings of the proposed methods, which are summarized into the following observations:

- In situations where tasks are strongly correlated (spam detection and sentiment analysis), the Inductive transfer SS-MTL and particularly the SG formulation outperformed the FG, the SS-MTL IND, the MTL-IMC, the MTL-IND, SS-STL and the STL approaches.

- Weak forms of transfer as the parameter transfer approach are more appropriate settings for situations where task correlations are weak. This has been observed in the Letters data set, where in terms of AUC the MTL-IND model outperformed all other methods.

- The static geometry proved to be more appropriate for data sets of highly correlated tasks, while the flexible approach seems to be more effective when the task correlations are smaller, as in the letters classification problem. However, the importance of the FG formulation lies in the ease it can be extended to other forms transfer learning as DA and SeTL.

- The Nearest Neighbour construction of the Laplacian compared to the Geometric produced comparable results, nevertheless its value persists in that parameter ε can be estimated using gradient based optimisation methods.

In general, the complexity of the algorithms within the GP framework rises as data points are included due to the computation and subsequently the inversion of the covariance matrix K. In the semi-supervised learning framework exploiting unlabelled data adds an additional cost of computing and inverting the graph Laplacian and the covariance matrix at all labelled and unlabelled data points, as well as the cost of estimating extra hyperparameters related to the SS framework. In the SS parameter transfer the covariance matrix and the graph Laplacian depends only on data from each task. On the other side, in the SS Inductive transfer the covariance and the Laplacian matrix depends on all data from all tasks making it more computationally expensive and not directly applicable for a large number of tasks, and labelled and unlabelled data points. However, it is our opinion that the SS-MTL framework that we propose in this work can easily be combined with other methods that reduce the complexity of GP prediction (Quinonero-Candela et al., 2007; Alvarez and Lawrence, 2009), which we intent to pursue in the future .

We would like to bring to your attention that the SG and the FG formulations have been employed in the Pka data regression data set (section 2.3), but since it did not offer any improvements compared to the standard MTL-IMC method of Bonilla et al. (2008) results are omitted. Last, while we believe the theoretical and experimental results in SS-MTL we present demonstrate the value of our approach, one of the most interesting features is the ease with which it can be extended to other transfer learning scenarios.

Chapter 5

Meta-generalisation with Gaussian Processes

In the three previous chapter we have dealt mostly with the Inductive category of *Transfer Learning* (TL) and partially with the Transductive category. We now turn our attention to the Unsupervised category of TL and propose a novel model for meta-generalisation, *i.e.* performing prediction on novel tasks based on information from multiple different but related tasks. The model is based on two coupled Gaussian processes with structured covariance function; one model performs predictions by learning a constrained covariance function encapsulating the relations between the various training tasks, while the second model determines the similarity of new tasks to previously seen tasks. We demonstrate empirically on several real and synthetic data sets both the strengths of the approach and its limitations due to the distributional assumptions underpinning it.

5.1 Introduction

The central problem of supervised learning is *generalisation*, learning input/ output relations from training data that, when applied to unseen test data, will give good performance (in terms of an appropriate loss function). A common assumption underlying many supervised learning algorithms is that the training and testing data distribution are the same, which allows them to make predictions of future instances of the problem at hand. On the other hand, in the complex world that we live in we are usually faced with unseen but similar problems, situations which human intelligence handles by adaptively taking decisions on the new tasks using knowledge from similar tasks.

In this direction, *Transfer learning* has emerged as a framework to handle situations where there are multiple but related problems to be solved. The term TL is used here in its broader sense, to cover more specific areas of research such as domain adaptation, co-variate shift, sample selection bias, self-taught learning, and multi-task learning. The differences between these subfields of TL lies mostly in the availability of outputs (labels) for input data in the various tasks, no matter if it is a regression or classification problem (Arnold et al., 2007). For example, the situation where labels are available for all tasks is tackled by multi-task learning, which synergistically solves the learning problem in all tasks simultaneously (Caruana, 1997; Bakker and Heskes, 2003; Ando and Zhang, 2005). Domain adaptation (Daumé III and Marcu, 2006; Daumé, 2007; Crammer et al., 2008; Mansour et al., 2009b; Pan et al., 2009), co-variate shift (Sugiyama et al., 2007; Storkey and Sugiyama, 2007; Bickel et al., 2009), and sample selection bias (Huang et al., 2007) are settings appropriate for problems where labels are only available for a task that is similar to the task that we wish to make predictions in (target task). Contrary to domain adaptation, and sample selection bias, self-taught learning (Raina et al., 2007) is a setting where labelled data are available for the target task, but the learning algorithm wishes to also use unlabelled data from a source task to improve performance. In its own right, self-taught learning is distinguishable from semi-supervised learning (Chapelle et al., 2006), where labelled and unlabelled data are assumed to come from the same task. The purpose of all these TL approaches is to enhance the generalisation power of a specific algorithm by leveraging related (but different) knowledge from multiple tasks. In particular, it is generally assumed that at least the input data for the target task will be available *during the learning*, so that a measure of the similarity between the training and target tasks can be estimated.

The question that we wish to raise in this chapter is whether the notion of generalisation can be extended to the level of tasks as a form of *meta-generalisation*. Meta-generalisation is a concept introduced in Baxter (2000), where the author argues whether a transfer learning algorithm can generalise well on totally *unseen* tasks after seeing sufficiently many *source* (or training) tasks. We emphasise that this is more than a theoretically interesting question. Our motivating example is a strongly applied one: we wish to create an automated diagnosis tool that can accommodate variability among patients, so that, once trained on a sufficient number of patients, it can generalise to new patients. In his work Baxter derives bounds on the generalisation error of this problem in terms of a generalised VC-dimension parameter, as well as comments that the number of source tasks and examples per task required to ensure good performance on novel tasks has to be sufficiently large. While Baxter (2000) derives an algorithm to select a subset of features to perform multi-task learning based on Neural Networks (NN), his work is more on the theoretical side as no experimental results are presented. Besides that, the model proposed in this work needs to be retrained in case a new target task arrives in order to learn a small number of task dependent parameters.

One way to approach meta-generalising is through domain adaptation, by training a model on the data of the source and the target set of tasks (Ben-David et al., 2007). This type of approach, as well as the model proposed in Baxter (2000), are essentially trained in a transductive way, as the algorithm is able to make predictions only on the test tasks that is trained on, or needs to be retrained in case a new task arrives. Obviously, the performance and the success of domain adaptation algorithms depends strongly on certain assumptions, with most important the similarity between the target and the source distribution. Clearly, if these assumptions are violated then the success of these algorithms is doubtful (Ben-David et al., 2010).

The problem of sampling the space of tasks to make predictions on totally unseen tasks in the inductive setting, which is the exact analog of generalising in the level of tasks, has to the best of our knowledge not been specifically addressed. As we mentioned before, TL is separated into different sub-categories based on the level of supervision on the target task. Multi-task learning can be seen as an *Inductive* TL learning algorithm since input data and labels are available for all the tasks that we wish to make predictions. On the other end, settings like to Domain adaptation, Covariate shift or Sample selection bias, can be viewed as a form of *Transductive* TL since the algorithm can exploit only the input distribution of the target task they want to make predictions (Arnold et al., 2007). On this basis, meta-generalising can be considered as a form of *Unsupervised* TL, since the learning algorithm does not have any exploitable information about the target tasks during training. Note, that this classification of TL algorithms is different from the one employed in Pan and Yang (2010), where unsupervised TL encapsulates problems like dimensionality reduction, density estimation, or clustering but in situations where multiple tasks are involved.

In this chapter we investigate the use of coupled Gaussian process (GP) models to address this problem. The model uses a multi-class Gaussian process for assigning probabilistically unseen tasks to source tasks (determining task responsibilities), and then uses a multi-task Gaussian process Bonilla et al. (2008) to perform prediction in individual tasks. Extensive testing on real and simulated data shows the promise of the model, as well as giving insight on the underlying assumptions.

The rest of the chapter is organised as follows: in section 5.2 we formally define the meta-generalizing problem, emphasising the main assumptions and highlighting the important special case of *fully observed tasks*. In section 5.3.2 and 5.3.3 we present our model and the inference methodology used. Our empirical findings are presented in section 5.4, and we conclude in section 5.5 by discussing the merits of our model in the context of the wider literature in transfer learning and meta-generalisation.

5.2 Meta-generalising

In this section, we introduce some notations, and we formally state the problem of meta-generalising. For simplicity, we concentrate on binary classification problems within each task, while we note that the same formalism applies to regression and multi-class classification problems.

In a meta-generalising scenario the learner is provided with a set of *source* tasks $\mathcal{T}_S = \{\mathcal{T}_{s1}, \dots, \mathcal{T}_{sM}\}$ which are used for training the model; testing is then performed on a set of *target* tasks $\mathcal{T}_T = \{\mathcal{T}_{t1}, \dots, \mathcal{T}_{tH}\}$. Each of the M source tasks will contain a training set of input/ output pairs (x, y), while data from any of the H target tasks are hidden. For later convenience, we will define the whole training set across tasks as a set of triples $T^s = \{x^s, y^{st}, y^{sx}\}$, where $x^s \in \mathbb{R}^d$ is the input feature vector, $y^{sx} \in \{-1, +1\}$ are the class labels, and $y^{st} \in \{1, \dots, M\}$ is the source task label indicating to which task the input/output pair pertains. Moreover, we will write $X^s_j = \{x^s_{ij}\}_{i=1}^{n^s_j}$ to denote the total item set of of the j^{th} source task. The total number of training pairs available will be denoted by $N^s = \sum_{j=1}^M n^s_j$, where n^s_j is number of data points from the j^{th} source task.

Each of the H target tasks $\mathcal{T}^t_j$ will consist of a set $X^t_j = \{x^t_{ij}\}_{i=1}^{n^t_j}$ of input points, where n^t_j is number of data points from the j^{th} target task and both types of labels are missing. Likewise, the total number of points from the target tasks will be denoted by $N^t = \sum_{j=1}^M n^t_j$. For reasons that will become clear later on, it is further assumed that for each target task data point x^t_j there is information that it comes from the j^{th} target task, but there is no knowledge with which of the source tasks is more similar. Note that each source task training input x^s_i is assigned two types of labels. This implies supervision in both the levels of the tasks and the data, through y^{st} and y^{sx} respectively; task labels y^{st} indicate from which of the source task a specific data point comes from, as a form of *meta-level information*, and class labels y^{sx} indicate to which class inside the task the data point belongs to, as a form of *inter-task information*.

Meta-generalisation, as all machine learning methods, relies on certain assumptions. We concentrate on two basic assumptions; the first one is the *similarity of the distribution* of the target task with at least one of the source tasks, while the second one is the agreement between the labels of the distributions termed as *low-error joint prediction* (Ben-David et al., 2010). Differently from (Ben-David et al., 2010), we will define the *low-error joint prediction* as the error between the predictive functions f_s and f_t of a source and a target task respectively, evaluated at the union of the source and the target sets $X = X^s \cup X^t$. Hence, the error λ_e of the joint prediction between a source and a target task can be computed from,

$$\lambda_e = \sum_{i=1}^{N} |f_t(x_i) - f_s(x_i)|, \tag{5.1}$$

where $N = N^s + N^s$, and $x_i \in X$. Intuitively, if the error λ_e is large then there is a disagreement between the labels of the source and target tasks distribution. Also note that, in a multi-task scenario where labelled data are available for both the source and target task the error λ_e can be computed by training two separate models under the same learning framework (*e.g.* NN, GPs, etc). Thus, the predictive functions of the source and target tasks can be estimated separately and λ_e can then used to quantify the relatedness of the two tasks. Conversely, in the scenarios of meta-generalising and domain adaptation one has to *assume* that the error λ_e will be low, since labels are available only for the source tasks. If one of these assumptions is not valid, then meta-generalisation can not be expected to guarantee success. We now give a formal definition of meta-generalising.

Definition 2 *Given a set of source tasks $\mathcal{T}_S$ and a set of target tasks $\mathcal{T}_T$, meta-generalising is an inductive inference method that aims at making predictions on the set of target tasks by sampling the space of source tasks .*

We further define two possible scenarios: in the *fully observed tasks* case, we assume that the similarity of the distribution assumption is perfectly met, so that the data generating distribution of the target task is the same as that of one of the source tasks (but we do not know which one). This assumption is relaxed in the *partially observed tasks* scenario, where we still assume similarity of the distribution but we do not necessarily have identity.

The meta-generalising setting implies that there is hierarchical structure in the problem. The data of each task are on the base level and the distribution of the tasks is on the meta level. Hence, it is intuitive that mechanisms are required to

1. Model the distribution of the data of each task, and the distribution of source tasks (correlation between tasks).

2. Infer how much correlated a target task is with the source tasks.

The first prerequisite leads us to multi-task learning, as many approaches offer mechanisms to model both the data and the task distribution (Bakker and Heskes, 2003; Yu et al., 2005; Ando and Zhang, 2005; Xue et al., 2007; Argyriou et al., 2008; Bonilla et al., 2008; Daumé III, 2009). Following the multi-task route, informally speaking, the second prerequisite can be translated as the problem of which of the M outputs of the multi-task classifier to select to make predictions for the target task. In some cases, task-descriptor features may be available, giving a direct measure of task similarity. In this work, we are interested in the general case where no reliable task descriptor features are available; we will then learn similarities between tasks through a *distribution matching* pursuit.

Another way of approaching the problem of meta-generalisation is through the framework of *mixtures of experts* (ME) (Jacobs et al., 1991; Waterhouse, 1997), under which a bigger learning problem is broken down to smaller subproblems that are handled by individual experts. The underlying assumption of this framework is that the data are generated by different processes (Waterhouse, 1997, Ch. 2), an assumption that can also be made in the multi-task setting about the data generating mechanism of each task. Under the ME framework each expert is used to model the data generating process of each subproblem. These experts are then combined through a gating network that models the responsibilities of the experts on each data partition. Hence, attacking the meta-generalisation problem through the ME framework can be seen as an unsupervised alternative method to that problem, that does not use the information about the origins of each task (the source task labels) but instead allows the algorithm to automatically infer the data partitions and the regions of expertise of each expert. Therefore the ME approach is in direct connection to multi-task learning and meta-generalisation in which case the experts are equivalent to the tasks, and this framework could be used as a rough lower bound on the performance of a multi-task classifier. Note though that in principle, it would be desirable to be able to automatically infer the number of experts as in Rasmussen and Ghahramani (2001), instead of presetting them as in Tresp (2000), which would be a similar mechanism of finding cluster of tasks.

5.3 A model for Meta-generalisation

Having identified the nature of the problem, we now propose a model for meta-generalising. The model builds upon the multi-task learning framework of Bonilla et al. (2008) which is able to capture the dependencies between the data and the tasks. In addition, we employ a classifier over the tasks to learn the task labels (from which task each data point comes from). Both of those two learning mechanisms, multi-task setting and classification of the tasks, are modelled by Gaussian Processes (GPs), which are coupled by sharing a common hyper-prior. In the rest of this section, we first give a short introduction to GPs and we review multi-task learning with GPs (Bonilla et al., 2008), we then present the model for meta-generalising, and finally we describe how to make predictions on new tasks.

5.3.1 Multi-task learning with Gaussian Processes

Gaussian processes (Rasmussen and Williams, 2005) provide a flexible modelling framework for supervised learning which has become increasingly popular in recent years. A Gaussian Process is a probability distribution over functions f, where the joint distribution of function evaluations over a finite set of inputs is a multivariate Gaussian distribution. At core of the GP prediction is the *covariance function* or *kernel*, that models the output covariance at different pairs of input points, and in essence acts as a measure of similarity between different input locations. In order for a covariance function to be valid it has to be positive semidefinite, and has to satisfy Mercer's theorem.

In this and the following subsection (subsections 5.3.1 and 5.3.2) we will use x, y^t, and y^x to refer to x^s, y^{st}, and y^{sx} to keep the notation light, since in the learning phase only source tasks are involved. In a multi-task scenario the interest lies in learning M related functions $\mathbf{f}_j,\ j=1,\ldots,M$, from training data $x_{ij},\ y_{ij},\ \ i=1,\ldots,n_j$, with $x \in\ \mathbb{R}^d$, and $n_1+\ldots+n_M=N$. Data points from task j will be denoted by $X_j=[x_{1j},\ldots,x_{n_jj}]$ and $\mathbf{X}=[X_1,\ldots,X_M]$ will be used to denote the set of all data points. Focussing on a regression problem for simplicity, the noise model will be given by

$$y_{ij}=f_j(x_{ij})+\varepsilon_j,\ \text{with}\ \varepsilon_j\sim\mathcal{N}(0,\sigma_j^2), \tag{5.2}$$

where y_{ij} (x_{ij}) denotes the i^{th} output (input) of the j^{th} task. We note that each input point has M function values associated with it (one per task); this *complete set of* *responses* will rarely be observed in practice, but function values corresponding to unobserved values can easily be marginalised using the consistency of GPs

The multi-task model of Bonilla et al. (2008), which has been known in the geostatistics community as the "*Intrinsic Coregionalization Model*" (ICM) (Cressie, 1993), can be elegantly recovered from the theory of matrix variate distributions (Gupta and Nagar, 2000). Define the vector $\mathbf{f}$ by stacking the columns of $\mathbf{F} = [\mathbf{f}_1 \ldots \mathbf{f}_M]$ into a single vector, $\mathbf{f} = \text{vec}(\mathbf{F})$, where $\mathbf{f}_j \in \mathbb{R}^{N\times 1}$ is the column vector of all latent functions evaluations of task j, and $\mathbf{F} \in \mathbb{R}^{N\times M}$. Then the *probability density function* of matrix F will be given by:

$$(2\pi)^{-\frac{1}{2}NM}|K^t|^{-\frac{1}{2}N}|K^x|^{-\frac{1}{2}M}\exp\left\{-\frac{1}{2}\text{trace}\left((K^t)^{-1}\mathbf{F}^T(K^x)^{-1}\mathbf{F}\right)\right\}, \tag{5.3}$$

where $K^t \in \mathbb{R}^{M\times M}$ and $K^x \in \mathbb{R}^{N\times N}$ (Gupta and Nagar, 2000). This configuration implies that the matrix K^t models the correlations between the vectors $\mathbf{f}_j$, *i.e.* the tasks in the multi-task view, and K^x models the correlations between each element of vectors $\mathbf{f}_j$. In the GP framework, this correlation between function evaluations at different input points is captured by the covariance function. Then, by using some matrix algebra involving the vec and Kronecker operator, equation 5.3 can be written in the form Bonilla et al. (2008) proposed

$$p(\mathbf{f}|X) = \mathcal{GP}(0, K^t \otimes K^x). \tag{5.4}$$

Employing this type of prior for the latent functions $\mathbf{f}$ the noise model for the regression problem stated in equation 5.2 becomes, $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, D\otimes I)$, where $D_{M\times M}$ is diagonal with $D_{jj} = \sigma_j^2$ and $I_{N\times N}$ is the identity matrix.

The key element of this formulation is the task covariance matrix K^t which reflects the task correlations. For example, if K^t was fixed to the identity matrix, then all tasks would be independent but they would still share the same hyperparameters of the covariance function. Of course, one of the main goals of multi-task learning is to learn these task dependencies. Bonilla et al. (2008) approached this problem by parameterising the task covariance matrix, always retaining positive definite restrictions, and treating these parameters as hyperparameters to be learned. Positive definite guarantees were achieved, by parameterising a lower triangular matrix L to employ the Cholesky factorisation $K^t = LL^T$. Most importantly, parameters related to the data covariance function or the task covariance matrix can be learned in the standard GP formulation, by maximising the marginal likelihood $p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})d\mathbf{f}$.

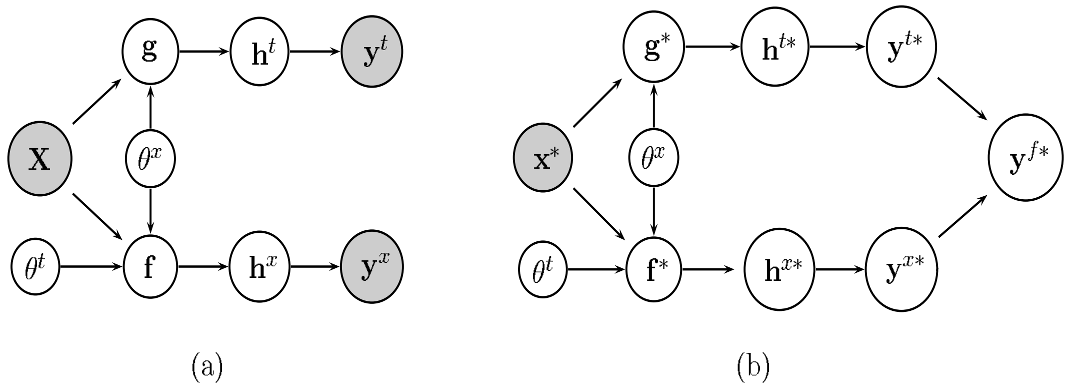


Figure 5.1: Coupled Multi-Task Multi-Class (CMTMC) model. Variables $\mathbf{f}$ and $\mathbf{g}$ are the two sets of GPs for the multi-task and multi-class classifiers respectively, whereas variables $\mathbf{h}^x$ and $\mathbf{h}^t$ denote the auxiliary variables of the two classifiers; (a) graphical representation of the training phase, (b) graphical representation of Meta-generalising.

5.3.2 Model

In this section we describe the Coupled Multi-Task Multi-Class (CMTMC) model we propose for meta-generalisation. The objectives of the model are first to model the dependencies between the tasks, and second to assign unseen tasks to source tasks by finding task similarities. The first objective is met through the Multi-task part of the model, while the second is achieved through the Multi-class classifier. Figure 5.1 shows the graphical model of the CMTMC classifier. Notation introduced in section 5.3.1 applies here. Moreover, from section 5.2 we have that $y^t \in \{1,\ldots,M\}$ and $y^x \in \{-1,+1\}$ as the task and class labels respectively. Since both class and task prediction are effectively classification models, we choose the probit and multinomial probit models as noise models respectively. Following Albert and Chib (1993), we define two sets of auxiliary variables $\mathbf{h}^t = \mathrm{vec}(\mathbf{H}^t)$, and $\mathbf{h}^x = \mathrm{vec}(\mathbf{H}^x)$, where $\mathbf{H}^t \in \mathbb{R}^{N\times M}$ and $\mathbf{H}^x \in \mathbb{R}^{N\times M}$. As shown later on, the use of these auxiliary variables enables the multinomial and the binary probit model respectively. For later convenience, we will be using $\mathbf{h}^t_j$ and $\mathbf{h}^t_n$ to denote the j^{th} column and n^{th} row of matrix $\mathbf{H}^t$, and similarly for matrix $\mathbf{G}$.

Figure 5.1 shows that there are two directed channels of variables. The upper channel, with variables $C^t = \{\mathbf{g}, \mathbf{h}^t, \mathbf{y}^t\}$, is responsible for learning the task labels, thus from which task each data point comes from, while the lower channel, with variables $C^x = \{\mathbf{f}, \mathbf{h}^x, \mathbf{y}^x\}$, learns to classify the data points inside every task and to find task correlations, through the standard multi-task classifier.

Thus, there are two sets of Gaussian Processes. The first set is responsible for the multi-task classification $\mathbf{f}|\mathbf{X},\theta^x,\theta^t \sim \mathcal{GP}(0,K^t \otimes K^x)$, where variables θ^x and θ^t are used to denote the hyperparameters of the data covariance function and task matrix respectively. The second set of GPs is responsible for the classification over the tasks $\mathbf{g}|\mathbf{X},\theta^x \sim \mathcal{GP}(0,I \otimes K^x)$, where $\mathbf{g} = \text{vec}(\mathbf{G})$, $\mathbf{G} \in \mathbb{R}^{N\times M}$ as $\mathbf{G} = [\mathbf{g}_1,\ldots,\mathbf{g}_M]$, and $\mathbf{g}_j \in \mathbb{R}^{N\times 1}$. As in the multi-class case we will have that $\mathbf{f} = \text{vec}(\mathbf{F})$, where $\mathbf{F} = [\mathbf{f}_1,\ldots,\mathbf{f}_M]$ and $\mathbf{f}_j \in \mathbb{R}^{N\times 1}$. In the rest of the chapter we will write K^x to denote the covariance matrix between all data points $\mathbf{X}$, unless specified otherwise. Moreover, I and K^t will be $M \times M$, where the identity matrix in the multi-class case implies independence between the classes, thus $\mathbf{g}_j|\mathbf{X},\theta^x \sim \mathcal{GP}(0,K^x)$. The key objective is to learn M functions $\mathbf{g}_j$ for the multi-class classifier and M related functions $\mathbf{f}_j$ for the multi-task classifier.

Note that the data covariance matrix K^x is shared by both sets of processes $\mathbf{g}$ and $\mathbf{f}$. This is graphically illustrated by the fact that the node of hyperparameters θ^x is connected to both latent functions; thus, the multi-class and the multi-task classifier share the same hyperparameter space for θ^x. The multi-class classifier is restricted to have the same covariance function across the classes in contrast with the standard model for multi-class classification with GPs, which in principle allows you to use different covariance functions across classes. In fact, the CMTMC model could be decoupled into two separate classifiers with different sets of hyperparameters θ^x between the two processes $\mathbf{f}$ and $\mathbf{g}$. Seemingly, this decoupling would result in a more flexible model, but preliminary experiments with both models, the CMTMC and the decoupled model, has shown that this restriction does not affect the performance. In contrast, it reduces dramatically the computational cost since the hyperparameters of the data covariance function need to be estimated only once.

The probit model is enabled in both channels by a standardised normal noise model over the auxiliary variables, $h^t_{ij}|g_{ij} \sim \mathcal{N}(g_{ij},1)$, and $h^x_i|f_i \sim \mathcal{N}(f_i,1)$ (Albert and Chib, 1993; Csató et al., 2000; Girolami and Rogers, 2006). The relationship between outputs y^t and y^x and auxiliary variables h^t, and h^x is deterministic and will be given by:

$$y^t_i = j \;\text{ if }\; h^t_{ji} = \max_{1\le k\le M}\{h^t_{ki}\}, \tag{5.5}$$

$$p(y^x_i|h^x_i) = \theta(y^x_i h^x_i) \;\text{ for }\; y^x_i = \pm 1, \tag{5.6}$$

where θ is one if its argument is positive and zero otherwise, i.e., the Heaviside function, which completes the specification of the model.

5.3.2.1 Inference

Classification problems imply non-Gaussian noise models, which make inference intractable. To address this intractability, we adopt a variational approximate treatment to the problem, as it is computationally more efficient than sampling-based methods while retaining a reasonable accuracy in empirically approximating posterior marginals. For a comprehensive comparison between these approximations for GP multi-class classification, and on the multinomial probit model the interested reader in referred to Girolami and Rogers (2006). The dependencies of the random variables $\Theta = \{\mathbf{G}, \mathbf{H}^t, \mathbf{f}, \mathbf{h}^x\}$ are depicted graphically in Figure 5.1.a and are summarised in the joint likelihood of the CMTMC model as:

$$p(\mathbf{y}^t, \mathbf{y}^x, \Theta|\theta^x, \theta^t, \mathbf{X}) = p(\mathbf{y}^t|\mathbf{H}^t)p(\mathbf{H}^t|\mathbf{G})p(\mathbf{G}|\theta^x, \mathbf{X})p(\mathbf{y}^x|\mathbf{h}^x)p(\mathbf{h}^x|\mathbf{f})p(\mathbf{f}|\theta^x, \theta^t, \mathbf{X}). \tag{5.7}$$

Variational methods approach this problem by approximating the joint posterior of the latent variables Θ within a family of tractable distributions; in our case, we will approximate the joint posterior as a factored distribution $p(\Theta|\mathbf{y}^t, \mathbf{y}^x, \mathbf{X}, \theta^t, \theta^x) \approx Q(\Theta) = \prod_{i=1} Q(\Theta_i) = Q(\mathbf{g})Q(\mathbf{h}^t)Q(\mathbf{f})Q(\mathbf{h}^x)$. Minimising the Kullback-Leibler divergence between the approximating and the true distribution is equivalent to maximising the following lower bound on the marginal likelihood

$$\log p(\mathbf{y}^t, \mathbf{y}^x|\mathbf{X}, \theta^x, \theta^t) \geq \int Q(\Theta) \log \frac{p(\mathbf{y}^t, \mathbf{y}^x, \Theta|\mathbf{X}, \theta^x, \theta^t)}{Q(\Theta)} \mathrm{d}\Theta, \tag{5.8}$$

which is found by applying Jensen's inequality(MacKay, 2003). Standard results show that the distributions that maximise the lower bound are given by

$$Q(\Theta_i) = \frac{\exp(\mathbb{E}_{Q(\Theta \backslash \Theta_i)}\{\log p(\mathbf{y}^t, \mathbf{y}^x, \Theta|\mathbf{X}, \theta^t, \theta^x)\})}{\int \exp(\mathbb{E}_{Q(\Theta \backslash \Theta_i)}\{\log p(\mathbf{y}^t, \mathbf{y}^x, \Theta|\mathbf{X}, \theta^t, \theta^x)\})\mathrm{d}\Theta_i} \tag{5.9}$$

where $Q(\Theta \setminus \Theta_i)$ denotes the factorized distribution with the i^{th} component removed. Inference and learning are performed in a variational EM algorithm: the E-step computes the variational posteriors on the variables Θ, and the M-step optimises the hyperparameters θ^t, θ^x given the expectations computed in the previous step. At each (E or M) iteration the variational lower bound provably increases (or at worst remains unchanged), and these two steps are repeated until convergence. We now briefly summarise the calculations needed to perform the E and M steps. We omit any details and emphasise only the occurrence of the special form covariance function we employ; fuller details can be found in appendices C.1, and C.2 for the approximate posteriors and the lower bound respectively.

5.3.2.1.1 E-step The approximate posteriors for the multi-class classifier will be given by,

$$Q(\mathbf{G}) = \prod_{j=1}^{M} \mathcal{N}_{\mathbf{g}_j}(\tilde{\mathbf{g}}_j, \Sigma^g), \tag{5.10}$$

$$Q(\mathbf{H}^t) = \prod_{n=1}^{N} \mathcal{N}_{\mathbf{h}_n^t}^{y_n^t}(\tilde{\mathbf{g}}_n, I), \tag{5.11}$$

where the tilde sign is used to denote the posterior expectation of that variable, for example $\tilde{g}(h^t) = \mathbb{E}_{Q(h^t)}[g(h^t)]$, and we have used $\Sigma^g = K^x(I+K^x)^{-1}$, and $\tilde{\mathbf{g}}_j = \Sigma^g \tilde{\mathbf{h}}_j^t$. The multivariate Gaussian $\mathcal{N}_{\mathbf{h}_n^t}^{y_n^t}(\tilde{\mathbf{g}}_n, I)$ is truncated such that if $y_n^t = i$ then the i^{th} dimension has the largest value. A full derivation can be found in appendix C.1. In the lower channel, the approximate posteriors for the multi-task classifier will be given by,

$$Q(\mathbf{f}) = \mathcal{N}_{\mathbf{f}}(\tilde{\mathbf{f}}, \Sigma^f), \tag{5.12}$$

$$Q(\mathbf{h}^x) = \prod_{i=1}^{N} \tilde{f}_i + y_i^x \frac{\mathcal{N}_{\tilde{f}_i}(0,1)}{\Phi(y_i^x \tilde{f}_i)}, \tag{5.13}$$

where $\tilde{\mathbf{f}} = \Sigma^f \tilde{\mathbf{h}}^x$, and $\Sigma^f = K^t \otimes K^x (I + K^t \otimes K^x)^{-1}$;

5.3.2.1.2 M-step The M-step optimises the lower bound with respect to the hyperparameters θ^x and θ^t. This is performed by gradient descent; computation of the gradients of the lower bound given in equation 5.8 are somewhat intricate and are given in appendix C.2.

5.3.3 Prediction on novel tasks

While in the previous section we described how to train the model on training data from the source tasks, we now describe how to perform predictions on unseen target tasks. We adopt a mixture of experts type approach; in these networks, multiple outputs are combined and weighted according to the responsibilities they have on a certain prediction task. In a similar manner, the multi-task classifier of the CMTMC model can be seen as a multi-output predictor, and the classifier over the task labels (multi-class) can be used to infer the responsibilities of the outputs of the multi-task classifier, since it produces posterior probabilities of task memberships. Then predictions on novel tasks are computed according to

$$p(y^f_* = +1|x^t, \mathbf{X}, \mathbf{y}^t, \mathbf{y}^x) = \sum_{j=1}^{M} p(y^x_{*j} = +1|x^t, y^t_{*j}, \mathbf{X}, \mathbf{y}^x) p(y^t_{*j}|x^t, \mathbf{X}, \mathbf{y}^t), \qquad (5.14)$$

where $p(y^x_{*j} = +1|x^t, y^t_{*j}, \mathbf{X}, \mathbf{y}^x) = p(y^x_{*j} = +1|x^t, \mathbf{X}, \mathbf{y}^x)$ is the posterior of the j^{th} task belonging to class "+1" from the multi-task classifier, and $p(y^t_{*j}|x^t, \mathbf{X}, \mathbf{y}^t)$ is the posterior of x^t coming from the j^{th} source task, or the test point task responsibility from the multi-class classifier. A graphical representation of this process is given in figure 5.1.b, where it is shown that nodes y^t_*, and y^x_* are combined to give the final predictions y^f_*.

However, the meta-generalisation scenario presents some additional challenges which are not found in classical mixture of experts models. In many cases, a target task consists of a *batch* of input points, and the simple fact that they all come from the same task contains valuable information about the correlations between the associated outputs. Another closely related issue is that of the correlation between the target task and the source tasks. In many multi-task problems it is a usual phenomenon to observe groups of highly correlated tasks (e.g. figure 5.3.b), while other times tasks are correlated but in a more random fashion (e.g. figure 5.6.b, 5.7.b). As we will see in the experimental sections, this can have important consequences in terms of predictive accuracy, and in terms of choosing an appropriate prediction model.

In the following, we present two distinct scenarios for inferring the task responsibilities. Given a target task with n^t data points $X^t = \{x^t_1, x^t_2, \ldots, x^t_{n^t}\}$, in the first scenario we treat each data point from the target task individually to infer its task responsibilities, which we will refer to as *Point to Point Gating* (P2PGat). This approach neglects the information that all target points come from the same task, and as we will see in the experimental section, is more appropriate when inter-task correlations are weaker. In the second scenario we wish to combine the information from all n^t test points from the target task to infer the overall task responsibilities for the target task, which we will refer to as *Batch* predictions.

5.3.3.1 Point to Point Gating

Given a new input point x^t which lacks both class and target labels, the CMTMC model combines the predictions of a multi-task classifier using task responsibilities obtained from the multi-class classifier channel. Thus, two sets of quantities need to be computed. The first set are the posterior probabilities of the M outputs $p(y^x_{*j} = +1|x^t, \mathbf{X}, \mathbf{y}^x)$ of the multi-task classifier, as

$$\begin{aligned} p(y^x_{*j} = +1|x^t, \mathbf{X}, \mathbf{y}^x) &= \int p(y^x_{*j} = 1|h^x_*) p(h^x_*|x^t, \mathbf{X}, \mathbf{y}^x) \mathrm{d}h^x_* \\ &= \int_0^{+\infty} \mathcal{N}_{h^x_{*j}}(\nu_{*j}, \upsilon_{*j}{}^2) \mathrm{d}h^x_* = \Phi\left(\frac{\nu_{*j}}{\sqrt{1+\sigma^2_{*j}}}\right) \end{aligned} \tag{5.15}$$

where

$$\nu_{*j} = (\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{X},x^t})^T \left(I + K^t \otimes K^x\right)^{-1} \tilde{\mathbf{h}}^x, \tag{5.16}$$

$$\sigma^2_{*j} = k^t_{jj} k^x_{x^t,x^t} - \left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{X},x^t}\right)^T \left(I + K^t \otimes K^x\right)^{-1} \left(\mathbf{k}^t_j \otimes \mathbf{k}^x_{\mathbf{X},x^t}\right), \tag{5.17}$$

and we have used $\mathbf{k}^t_j$, k^t_{jj} to denote the j^{th} column and the jj^{th} element of K^t respectively, and $\mathbf{k}^x_{\mathbf{X},x^t}$ and $k^x_{x^t,x^t}$ to denote the covariance vector between $\mathbf{X}$ and x^t, and the marginal variance of the test point respectively.

The second set of quantities are the task responsibilities which are computed from (Girolami and Rogers, 2006),

$$\begin{aligned} p(y^t_* = k|x^t, \mathbf{X}, \mathbf{y}^t) &= \int p(y^t_* = k|h^t_*) p(h^t_*|x^t, \mathbf{X}, \mathbf{y}^t) \mathrm{d}h^t_* \\ &= \int_{-\infty}^{+\infty} \mathcal{N}_{h^t_{*k}}(\mu^g_{*k}, \upsilon^2_{*k}) \prod_{m \neq k} \int_{-\infty}^{h^t_{*k}} \mathcal{N}_{h^t_{*m}}(\mu^g_{*m}, \upsilon^2_{*m}) \, \mathrm{d}h^t_{*m} \, \mathrm{d}h^t_{*k}, \end{aligned} \tag{5.18}$$

which can be evaluated using numerical integration as:

$$p(y^t_* = k|x^t, \mathbf{X}, \mathbf{y}^t) = \mathbb{E}_{p(u)} \left\{ \prod_{j \neq k} \Phi\left(\frac{1}{\upsilon_{*j}} \left[u\upsilon_{*k} + \mu^g_{*k} - \mu^g_{*j}\right]\right) \right\}, \tag{5.19}$$

where $u \sim \mathcal{N}_u(0,1)$, and

$$\mu^g_{*m} = (\mathbf{k}^x_{\mathbf{X},x^t})^T (I + K^x)^{-1} \tilde{\mathbf{h}}^t_m, \tag{5.20}$$

$$\upsilon^2_{*m} = 1 + k^x_{x^t,x^t} - (\mathbf{k}^x_{\mathbf{X},x^t})^T (I + K^x)^{-1} (\mathbf{k}^x_{\mathbf{X},x^t}). \tag{5.21}$$

In the P2P scenario, the novel input points are not assumed to share a common task label. Therefore, class prediction is performed straightforwardly on every new input by inserting the posterior probabilities obtained in equations 5.15, and 5.19 in the gating network given by equation 5.14. More details about the posterior class probabilities of the multi-class classifier are given in appendix C.3.

5.3.3.2 Batch

In a Bayesian way using all test points X^t to infer the overall task responsibility is performed by replacing the univariate distributions from equation 5.18 with the appropriate multivariate. As a result the second integral of equation 5.18 becomes the multivariate cumulative distribution function$\int_{-\infty}^{\mathbf{h}^t_{*k}} \mathcal{N}_{\mathbf{h}^t_{*m}}(M^g_{*m}, \Upsilon_*)\, \mathrm{d}\mathbf{h}^t_{*m}$. Specifically the mean and the variance of the auxiliary variables $\mathbf{h}^t_{*m}$ on the batch of test points X^t will be given by:

$$M^g_{*m} = \mathbb{E}[\mathbf{h}^t_{*m}|X^t] = (\mathbf{K}^x_{\mathbf{X},X^t})^T\,(I+K^x)^{-1}\,\tilde{\mathbf{h}}^t_m, \tag{5.22}$$

$$\Upsilon_* = \mathrm{cov}[\mathbf{h}^t_{*m}|X^t] = I + K^x_{X^t,X^t} - (\mathbf{K}^x_{\mathbf{X},X^t})^T\,(I+K^x)^{-1}\,(\mathbf{K}^x_{\mathbf{X},X^t}), \tag{5.23}$$

where $\mathbf{K}^x_{\mathbf{X},X^t}$ is the $N \times n^t$ covariance matrix of all training points $\mathbf{X}$, and all target task data points X^t, and $K^x_{X^t,X^t}$ is the $n^t \times n^t$ full covariance matrix of X^t. Equations 5.22 and 5.23, indicate that inferring the tasks responsibilities on a set of points depends not only on the correlations between the test points and the train points but also on the correlations between the test points themselves.

On the other hand, truncated multivariate Gaussian distributions are hard to deal with, and usually approximations are applied (Deak, 1980; Genz, 1992; Gassmann et al., 2002). The dimensions of the multivariate distribution function in the batch prediction problem depend on the number of data points n^t of the target task, which can be several thousands depending the application. To the best of our knowledge no method can tackle very high dimensional c.d.f. , and even approximations can become extremely computationally intensive when n^t is more than a few dozens (these estimations would be carried out within the inner loop of a VBEM algorithm, which would obviously further aggravate the problem). A solution to this problem is to assume that data points from the test task are i.i.d. from the unknown data generating distribution, and approximate it by:

$$p(\mathbf{y}^t_* = k|X_*, \mathbf{X}, \mathbf{y}^t) \approx \frac{\prod_{i=1}^{n^t} p(y^t_{*i} = k|x^t_i, \mathbf{X}, \mathbf{y}^t)}{\sum_{m=1}^{M} \prod_{j=1}^{n^t} p(y^t_{*j} = m|x^t_j, \mathbf{X}, \mathbf{y}^t)}, \tag{5.24}$$

where $p(y^t_{*i} = k|x_{*i}, \mathbf{X}, \mathbf{y}^t)$ are the task responsibilities computed individually for each test point. Despite the fact that this approximation assumes that the covariance matrix between the test points $K^x_{X^t,X^t}$ is diagonal, thus considers each point independently, effectively task responsibilities are computed by using information from all data from the target task. We will adopt this approximation in the experimental section; we also experimented with using a reduced rank approximation for Υ_*, but this did not appear to yield any empirical advantages while retaining a computationally feasible rank.

5.4 Experiments

This section aims at providing insights into the workings of our meta-generalising model through empirical evidence. Experiments are presented for both the fully observed and partially observed task scenarios described in section 5.2, and in both cases we investigate both the P2P gating and the Batch mode of predictions on new tasks. The fully observed tasks case, considered in section 5.4.1, investigates the situation where data generating distribution of the target task is actually the same as that of one of the source tasks. In this case all available tasks are used in the training phase, but in the testing phase the model has no information from which of the source task the target task comes from. The second set of experiments, described in section 5.4.2, considers the case of the partially observed tasks. In this case the data generating distribution of the target task does not match the distribution of one of the source tasks, so that the set of source tasks is strictly a subset of the set of all tasks. Training is performed on the source tasks, and testing on the totally unseen target tasks. While both scenarios are plausible applications of meta-generalising, section 5.4.2 gives more insight into the connections between the correlation structure of the tasks and the task prediction mechanism on totally unseen tasks.

Five different data sets are considered in the experiments. The first two data sets are artificially generated to demonstrate the strengths and the limitations of the method; the first one satisfies the assumptions of the model, and the second one, which is only considered in section 5.4.1, is in conflict with them. The third data set is a character classification problem between commonly confused handwritten letters. The fourth data set is an automated diagnosis problem: annotated heartbeats from ECG recordings are used to discriminate normal from arrhythmic beats, and each patient is considered as a task. The last data set, which is considered only in the second set of experiments, is a landmine detection problem. More details are given in each section separately. We present results for different training set sizes, and for each training size experiments are repeated 25 times by randomly selecting the data points used for training from each task. Furthermore, in both scenarios three types of outputs are considered from the CMTMC model; the batch written as "BatchMCAppr", the P2P gating written as "P2PMCGat", and the "MAP" estimate which simply selects the output of the multi-task classifier that has the highest posterior, something that is usually considered in classifier fusion techniques (Kuncheva, 2002). When found necessary the results of the different methods of making predictions are compared using the t-test. As our method essentially relies on the covariance structure between tasks, two types of baseline comparisons are possible: in the worst case, results should not be worse than completely ignoring the task structure and pooling together all training data. We refer to this baseline as Pool. In the best case, our method should not be statistically better than a method which leverages the same covariance structure and has access to all the task label information, e.g. a standard multi-task learning approach. We refer to this best-case scenario as MTL; we compare with this only in the fully observed task scenario, as in the partially observed case the meta-generalising results are generally quite far from this best case.

All methods are compared in terms of Area Under the Curve (AUC) (Hanley and Mcneil, 1982), as one of the most appropriate measure of performance for imbalanced data sets[1]. In all experiments the task covariance matrix K^t was parameterised as a correlation matrix (Rebonato and Jäckel, 2000), with unit diagonal, while the data covariance function K^x is set specifically for each data set depending the application.

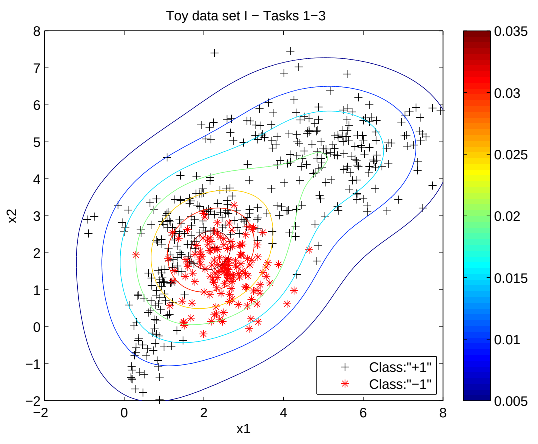
5.4.1 Fully observed tasks

In this scenario, the data distribution of the target task is the same as that of (at least) one of the source tasks. This guarantees that the similarity of distribution assumption is met, however, as we'll see in the case of Toy data *II*, the low joint prediction error assumption is not automatically satisfied. Obviously, the actual input data will be different, due to the stochasticity of the data generating process. Intuitively, the success of the model depends strongly on whether the model will be able to infer correctly from which of the source tasks the target task actually comes from.

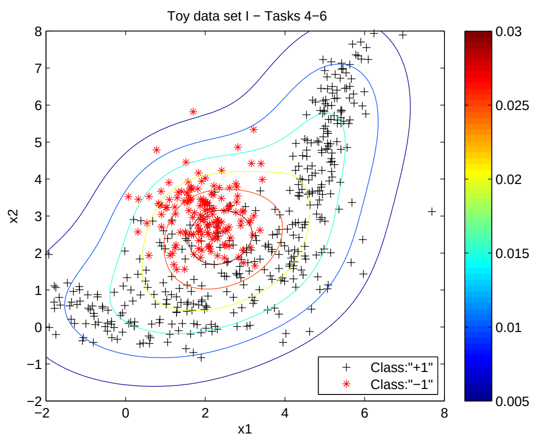
5.4.1.1 Toy data set *I*

The first toy data set is comprised of six binary classification tasks. This toy problem was previously used in Liu et al. (2009) in the context of semi-supervised multi-task learning. Data for the first three tasks are generated from a mixture of two partially overlapping Gaussian distributions, and similarly for the remaining three tasks. Hence, the six tasks cluster in two groups; for each task 600 data points were generated, which were equally divided between the two classes. The scatter plots of the two clusters are shown in figures 5.2.a and 5.2.b.

[1]Another measure for imbalanced data sets is known as the Precision-Recall curve, or the average precision (Davis and Goadrich, 2006; Brodersen et al., 2010).

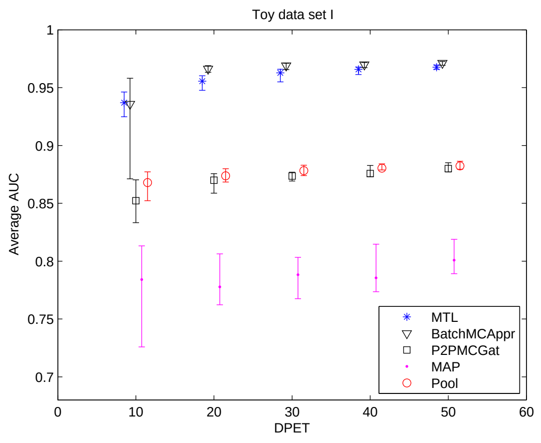


(a)

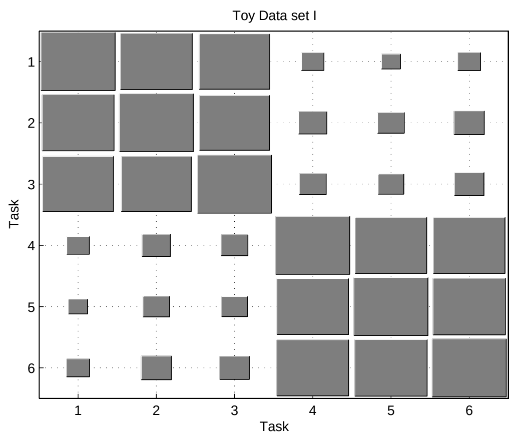


(b)

Figure 5.2: Toy data set I distribution; (a) scatter plot and density for the first cluster of tasks (1-3), (b) scatter plot and density for the second cluster of tasks (4-6).



(a)



(b)

Figure 5.3: Toy data set I classification Results; (a) Average AUC over the 6 tasks, (b) Hinton Diagram of the task covariance matrix of the CMTMC model computed by averaging over the 25 repetitions with 50 data points per task.

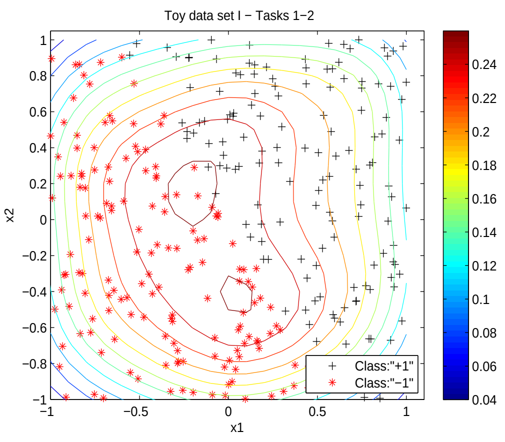
This data set is ideal for demonstrating the concept of the meta-generalising for three reasons. First of all the assumptions of the model are satisfied. Secondly, the tasks group in two clusters. The third reason is that the densities of the clusters though similar are not exactly the same; this is illustrated in figures 5.2.a and 5.2.b, which shows the contour plot of the densities of the two clusters. We use an Automatic Relevance Determination (ARD) data covariance function, which employs a different characteristic length scale for each feature, and is able to identify which features are more relevant for classification (Rasmussen and Williams, 2005).

Classification results are presented in figure 5.3.a; the Y axis is the AUC, and the X axis is the number of data points from each task (DPET) used for training.

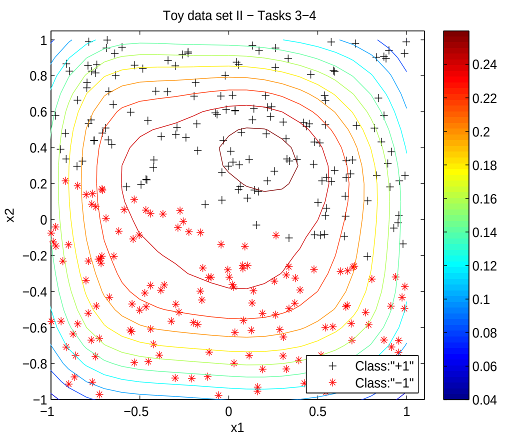
The results show that, in this toy problem, the Batch mode performs similarly to the ideal MTL case (differences in terms of AUC were not statistically significant at 0.05 p-value), although it has a high variance for the case of 10 DPET. The P2PGat and Pooling method perform approximately 10% worse that the Batch, while the MAP estimate gives roughly 20% less than the Batch. Moreover, figure 5.3.b shown the Hinton diagram (Hinton, 1989) of the task covariance matrix of the CMTMC model which accurately recovers the structure of the tasks.

5.4.1.2 Toy data set *II*

The second toy data set consists of four tasks which group into two clusters. The scatter plot as well as the density of the two clusters are shown in figures 5.4.a and 5.4.b, for the first and second cluster respectively. The main feature of this data set, evident visually from Figure 5.4, is the similarity of the data generating distribution for the two tasks. While the densities are peaked in different locations, without class labels the tasks are almost identical, meaning that the multi-class classifier cannot learn to discriminate between the two tasks. As in the previous example, each task consisted of 600 data points equally divided between the two classes, and we used the ARD covariance function. Figure 5.5.a shows the results the different methods produced. As expected, the Batch mode fails to correctly identify the task responsibilities; as a result, it gives a lower average AUC than the MTL, a difference which does not decrease with the number of DPET, indicating statistical inconsistency. This is reinforced by the Hinton diagram of K^t in figure 5.5.b, where it fails to identify the clusters of the tasks. Even though this difference is small it is significant for this easy problem, where the MTL algorithm performs close to 100%. Additionally, the P2PGat, the Pooling, and the MAP estimates perform better that the Batch, but they also fail to reach the



(a)



(b)

Figure 5.4: Toy data set II distribution; (a) scatter plot and density for the first cluster of tasks(1-2), (b) scatter plot and density for the second cluster of tasks (3-4).

performance of MTL. Note that differences P2PGat and Pooling were not statistically significant at 0.05 p-value, whereas differences between Batch and both P2PGat and Pooling were statistically significant at 0.05 p-value for 30 and 50 data points.

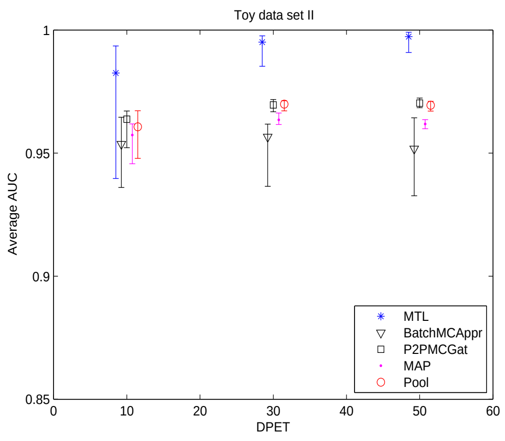
5.4.1.3 Character Classification

In this data set the task is to learn to classify between commonly confused handwritten letters, which is included in the ”Transfer learning Toolkit” of Berkeley University available at http://multitask.cs.berkeley.edu/. This data set is comprised of eight binary classification tasks. The characters that are used and the number of samples are given in table 4.5. Each sample is a 16×8 image, which results into a binary 128 feature vector. The covariance function that is employed for this data set is the *Radial Basis Function* (RBF).

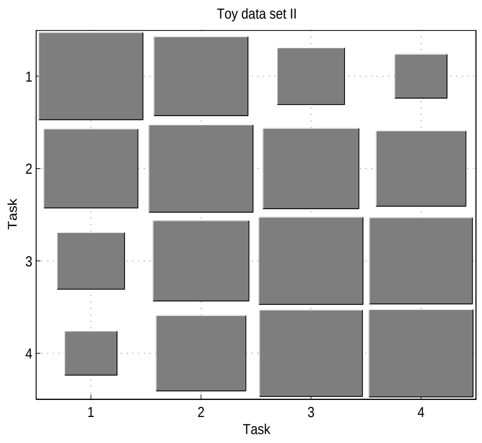
The classification results for this data set are presented in figure 5.6.a. The Batch method approaches the ideal MTL performance, and outperforms the P2PGat, Pooling, and the MAP methods. Differences in terms of AUC between the MTL and the Batch methods were statistically significant at 0.05 p-value, whereas more in depth comparison of the performances of the Pool and P2PGat methods revealed that the Pool method performed better that the P2PMC at significance level of 0.05 p-value.

Figure 5.6.b shows the Hinton diagram of the task covariance matrix[2]. This Hinton

[2]In the Hinton diagram of the Character classification problem (figure 5.6.b) task correlations smaller

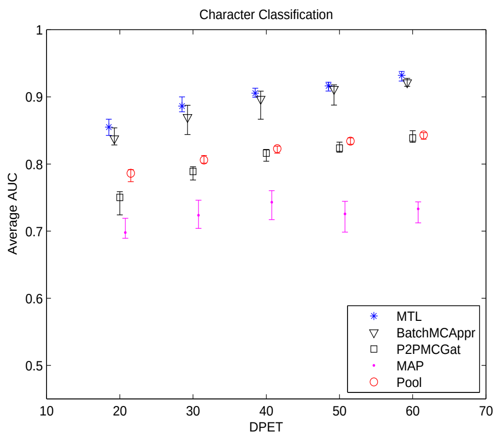


(a)

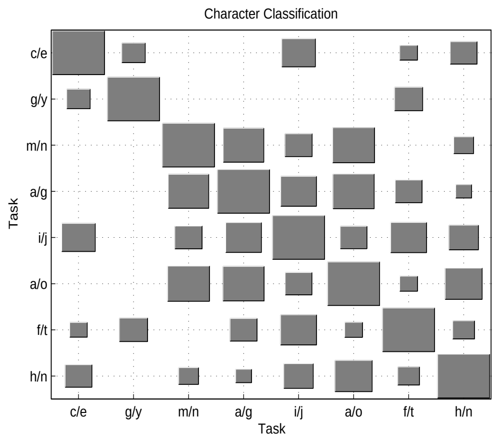


(b)

Figure 5.5: Toy data set II classification Results; (a) Average AUC over the 4 tasks, (b) Hinton Diagram of the task covariance matrix.



(a)



(b)

Figure 5.6: Character Classification Results; (a) Average AUC over the 8 tasks, (b) Hinton Diagram of the task covariance matrix.

diagram indicates a more random structure between the tasks, but finds that some tasks are more correlated than others, for example ‘a/g’ with ‘a/o’, and ‘i/j’ with ‘f/t’. It should be noted though, that in this data set the “low-error joint prediction” assumption is partially violated since there is label disagreement between tasks ‘a/g’ and ‘g/y’, where the ‘g’ letter belongs to class “+1” in task ‘a/g’ and to “-1” in task ‘g/y’. As it is shown in the fully observed case type of experiments, this does not seem to have any adverse effect on the performance of the model, presumably as the difference between a and y is sufficient to unambiguously assign the target task to the correct source task.

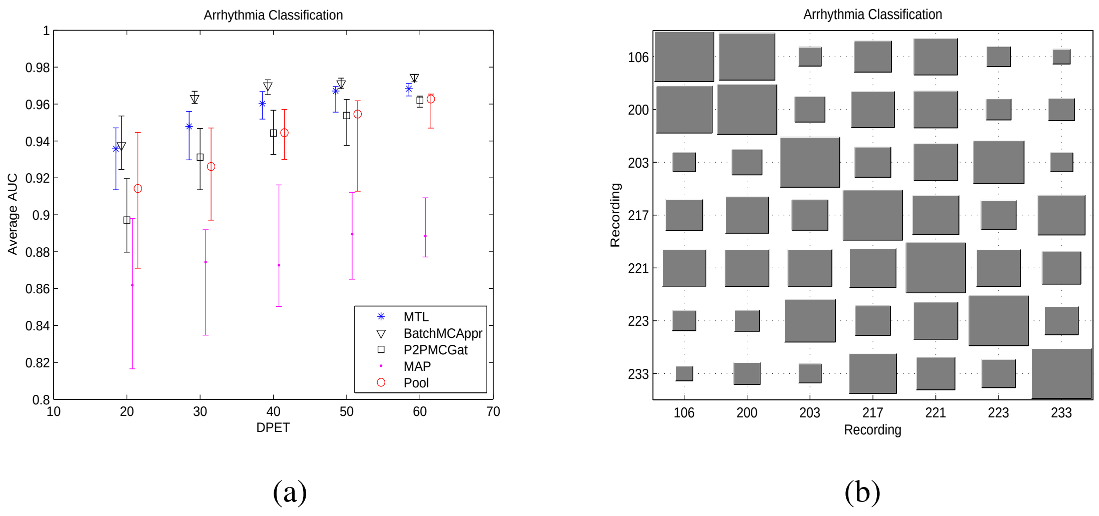


Figure 5.7: Arrhythmia Classification Results; (a) Average AUC over the 7 tasks, (b) Hinton Diagram of the task covariance matrix.

5.4.1.4 Arrhythmia Classification

The arrhythmia data set consists of seven ECG recordings from different patients, which were acquired from the MIT-BIH Arrhythmia database (Goldberger et al., 2000). Each recording corresponds to a large number of heart beats, which is summarised in table 3.1. Each patient is treated as a separate task, and the goal is to classify each heart beat into two classes, normal or premature ventricular contraction (PVC) arrhythmic beats. The same problem was considered in Skolidis et al. (2008) using single task GP classifiers. Each recording was sampled at 360Hz, and annotation provided by the database was used to separate the beats before any preprocessing. Each beat segment, consisting of 360 data points (one minute), was transformed into the frequency domain

than 0.05 were set to zero because of problems in producing the figure.

using a Fast Fourier Transform with a Hanning window. Only the first ten harmonics are used as features for classifying heart beats, as most of the information of the signal is contained in these harmonics.

Figure 5.7.a shows the average AUC over the seven tasks. On average, the Batch method performs better than other approaches. Interestingly, the MAP approach is consistently worse than other methods, a situation that will be reversed in the partially observed tasks scenario. Differences between Pool and P2PGat were not statistically significant at 0.05 p-value at any training data size, while the MTL method outperformed the Batch method for all training sizes larger than 20 at a significance level of 0.05 p-value. As in the character classification problem the task covariance matrix K^t, shown in figure 5.7.b , demonstrates that there are correlations between the tasks but in more random way.

5.4.1.5 Observations

This set of experiments has demonstrated the effectiveness of the CMTMC model in situations were the data distribution of the target task comes from one of the source tasks. Several observations are made:

1. In the fully observed tasks scenario, the space of tasks has been sampled sufficiently (by definition). In this case the Batch mode should theoretically be the best method, since all data points are needed to produce an accurate estimate of the density of the target task. This is empirically confirmed in our investigation, as Batch closely approaches the MTL results in all cases.

2. If the ”low error joint prediction” assumption is violated, then meta-generalising becomes a very hard problem, possible unsolvable. The performance on the second toy example was not particularly bad, since all methods achieved higher that 90% in terms of AUC, but none of methods reached the performance of the MTL algorithm. Moreover, the performance did not appreciably improve when more training data was provided, indicating statistical inconsistency. This effect could be dramatically increased if for example the classes between the clusters were anticorrelated, so that similar data generating distributions could be potentially associated with opposite predictions. Note though that if discriminative task descriptor features are available then this problem can be overcome, because augmenting the feature space would result in a different mapping of the latent function f.

3. If the model assumptions are met, the correlation structure of the tasks does not have a strong influence on the predictions, since the Batch mode outperformed the P2P gating and MAP estimate in all experiments. As we will see, this will be a crucial difference between the fully and partially observed tasks scenario.

5.4.2 Partially observed tasks

We now consider the harder problem of making predictions on completely unseen tasks. In this case, a priori we have no guarantee that any of the underlying modelling assumptions (similarity of distribution and low-error joint prediction) may hold. However, in some situations it is not unrealistic to assume that inter-task correlations will be structured, for example by the presence of *clusters* of similar tasks. These clusters may be evident from the experimental design of the problem (as in the case of the landmine data set discussed below), or may become evident from the training phase on the source tasks, if the learned task covariance matrix exhibits a strong block structure.

We are not aware of other methods that has distribution matching mechanism to perform predictions on totally unseen tasks. Therefore, in this section we will only compare the different inference mechanisms of the CMTMC model (Batch and P2P) with a GP model trained by pooling all data together and with the MAP combination of classifiers.

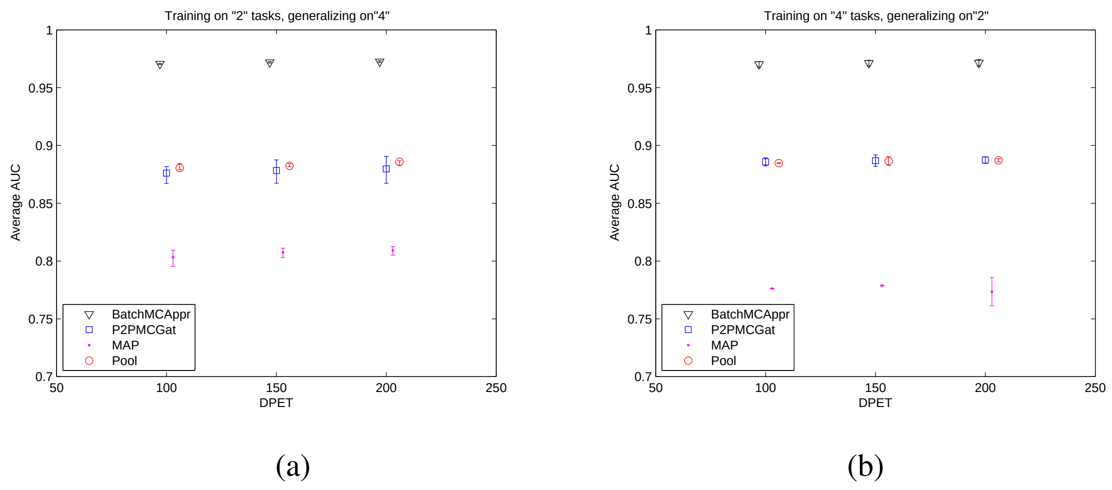


Figure 5.8: Average AUC on the unseen tasks of Toy data set I; (a) training on 2 tasks generalising on 4, (b) training on 4 tasks generalising on 2.

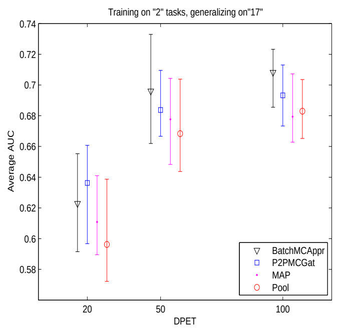
5.4.2.1 Toy data set *I*

We consider the toy data set of section 5.4.1.1, that consists of two clusters of tasks; in this section, training tasks are selected by randomly selecting equal number of tasks from each cluster. The challenge for the model is to correctly classify the task, given the similarity of the task distributions between the two clusters (see Figure 5.2). Experimental results are presented for two and four training tasks in figures 5.8.a and 5.8.b respectively. Naturally, as this data set is designed to match our modelling assumptions, the Batch method outperforms all other methods; it is interesting however that the method successfully detects from which cluster of tasks the unseen target task comes from even for relatively small training set sizes. Comparing the performance of the Toy data set *I* in the fully and partially observed cases, in figures 5.3 and 5.8 respectively, reveals that the same levels of AUC are achieved in both experimental setups, indicating that the task classification GP is highly confident of the correct result.

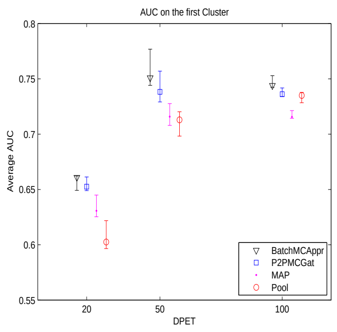
5.4.2.2 Landmine Detection

The landmine detection data set consists of images measured with airborne radar systems, and the goal is to predict landmines or clutter (Xue et al., 2007). Data are collected from 19 landmine fields, which are considered as subtasks, and each point is represented by a nine-dimensional feature vector. Tasks 1-10 correspond to regions that are relatively highly foliated while tasks 11-19 correspond to regions that are bare earth or desert. The experimental setup suggests the presence of two clusters of tasks corresponding to the geomorphology of the region the observations come from; this is confirmed by our preliminary investigation (not shown), as well as results on this data set by Xue et al. (2007); Liu et al. (2009). Thus, in this data set training tasks are set by randomly selecting equal number of tasks from the first cluster, tasks 1-10, and from the second cluster, tasks 11-19. Experiments are presented for two, four, and eight training tasks. The data covariance function that is used for this data set is the ARD.

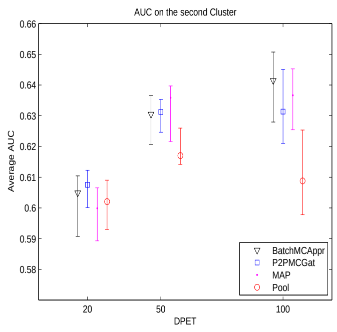
Figures 5.9.a, 5.10.a, and 5.11.a shows the average AUC on the 17, 15, and 11 unseen target tasks for each partition respectively. It is noticeable that, despite the slightly higher mean of the Batch method, there are large overlapping error bars between the methods. Large error bars give evidence that there might be two levels of performance. Therefore, for each partition we provide the average AUC for each cluster separately; subfigures (b) from figures 5.9, 5.10, and 5.11 show the average AUC for the first cluster, and subfigures (c) for the second cluster. Measuring the AUC in each cluster separately gives significantly smaller error bars, and reveals interesting structures in the problem. Specifically, the performance on the first cluster is always better than on the second cluster by a considerable margin (of approximately 10 percentage points), achieving AUCs mostly over 70% in the first cluster (figure 5.9.b).



(a)



(b)



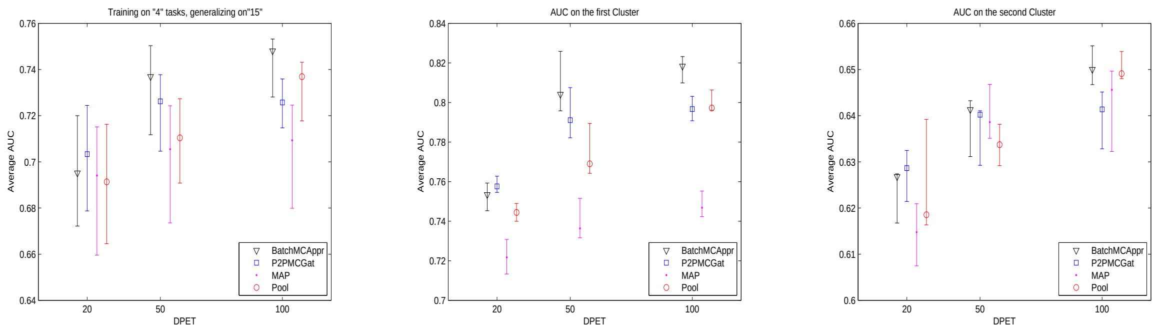
(c)

Figure 5.9: AUC on the 17 unseen tasks of Landmine data set; training on 2 tasks, generalising on 17; (a) AUC over all 17 tasks, (b) AUC over 9 tasks of the first cluster, (c) AUC over 8 tasks of the second cluster.

Moreover, comparing the methods on each cluster separately we see that the Batch method outperformed the pooling and the P2PGat in most of the cases, particularly in the first cluster where the advantages become very significant as we increase the number of tasks/ DPETs. The correlation structure within the second cluster is looser, implying a weaker applicability of our modelling assumptions. This is reflected in the experimental results, where lower AUCs are achieved and the differences between the various methods are not significant.

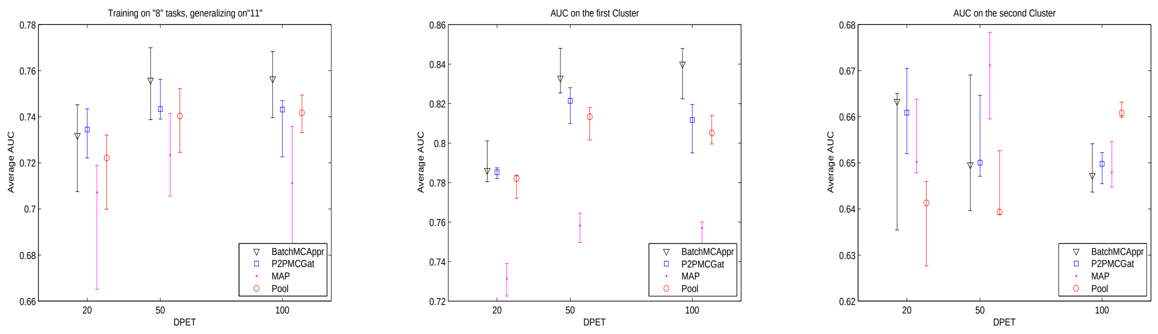
More in depth statistical analysis revealed that for 2 training tasks the Batch and the P2PGat method performed equally well with the exception of training with 50 data points on the first Cluster where the differences were statistically significant at 0.05 p-value; for 4 and 8 training tasks the differences between the Batch and the P2PGat were statistically significant for 50 and 100 data points on the first Cluster of tasks.

It should be pointed out, however, that this is a substantially harder pattern recognition task compared to the toy data set considered above. For example, Liu et al. (2009), which investigated semi-supervised MTL on this data set, achieved a best performance of 78% AUC; the CMTMC (which relies on the more flexible GP framework for MTL) achieves an average AUC close to 76% on totally unseen tasks having trained on 8 source tasks with 100 DPET.



(a) (b) (c)

Figure 5.10: Average AUC on the 15 unseen tasks of Landmine data set; training on 4 tasks, generalising on 15; (a) Overall AUC over all 15 tasks, (b) Average AUC over 8 tasks of the first cluster, (c) Average AUC over 7 tasks of the second cluster.



(a) (b) (c)

Figure 5.11: Average AUC on the 11 unseen tasks of Landmine data set; training on 8 tasks, generalising on 11; (a) Overall AUC over all 11 tasks, (b) Average AUC over 6 tasks of the first cluster, (c) Average AUC over 5 tasks of the second cluster.

5.4.2.3 Arrhythmia Classification

As a second real data set, we return to the arrhythmia classification problem introduced in subsection 5.4.1.4. The results from the fully observed tasks scenario indicate an unclear pattern of correlations between the tasks, as summarised in the task covariance matrix figure 5.7.b, which calls into question the validity of the similarity of distribution assumption. Fortunately, in this application the classes have a physical interpretation. For example normal heart beats between different patients, although not exactly the same, can be expected to be similar, and a PVC arrhythmic heart beat of one patient can not have the wave form of a normal heart beat from another patient. This allows us to assume that the classes between the tasks will not be anti-correlated, so that at least the low-error joint prediction assumption should approximately hold.

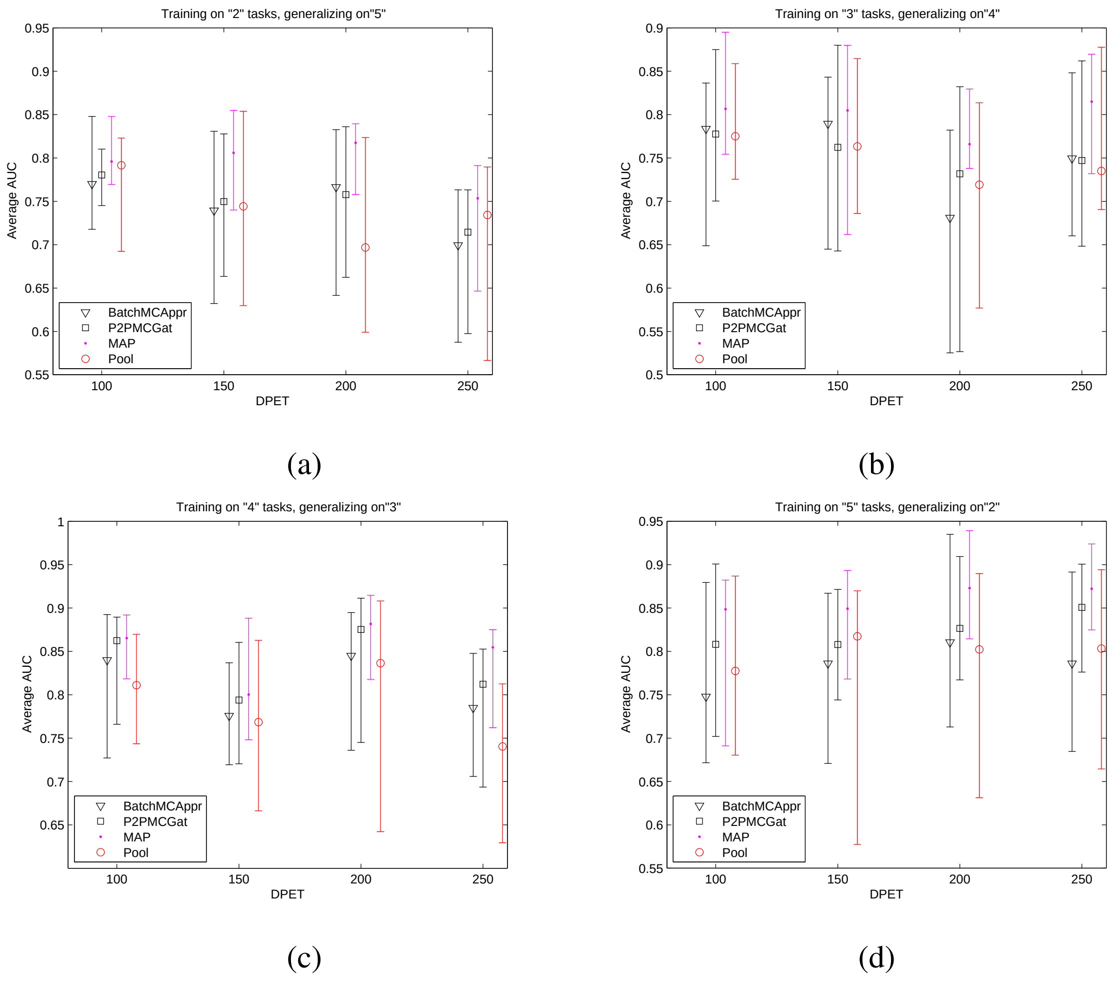


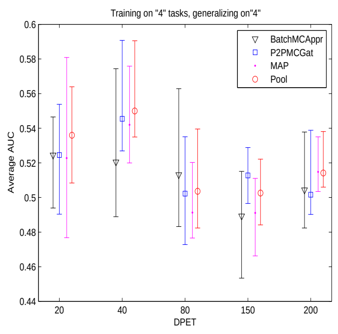
Figure 5.12: Average AUC on the unseen tasks of Arrhythmia data set on different number of training tasks; (a) training on 2 tasks, generalising on 5, (b) training on 3 tasks, generalising on 4, (c) training on 4 tasks, generalising on 3, (d) training on 5 tasks, generalising on 2

Since there are no obvious clusters among tasks, in this set of experiments the training tasks are chosen by randomly selecting some for training and keeping the rest as test tasks. Figure 5.12 presents the results on the unseen tasks that were obtained by training the CMCMT model with 2, 3, 4, and 5 tasks. First of all, we observe that the average AUC in the partially observed case is a lot lower than in the fully observed case, something perhaps to be expected since, contrary to the previous two examples, the model assumptions are not fully met in this data set. Surprisingly, the method that achieved the best performance was the MAP, and no principled justification can be given for that. Secondly, we observe that the performance in this set of experiments exhibits some interesting patterns as the number of training tasks increases. Specifically, for two, three or four training tasks the performance of all methods does not significantly improve as we increase the number of DPETs, and in some cases it even deteriorates. This indicates that if the space of tasks has not been sampled sufficiently, the model can not yield good generalisation performance to new tasks, even if the number of training data increases. In contrast, for five training tasks the MAP and P2PGat methods yield a significant improvement of performance as the number of data points increases (levelling off after 200 DPETs).

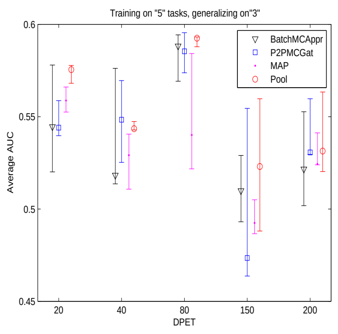
Empirically, it would appear that the P2PGat method is preferable to the Batch method when the model assumptions are violated. Intuitively, one could argue that the Batch method is less flexible, as the relative contribution of the different single-class predictors is fixed across all points in the target task. Therefore, if the model assumptions are violated, leading to an incorrect task labelling, the propagated error could have a worse effect in Batch than in P2Gat. This is partly confirmed by the analysis of Toy data set *II* in section 5.4.1.2, where the model assumptions were violated and P2PGat gave significantly higher AUC than the Batch method.

5.4.2.4 Character Classification

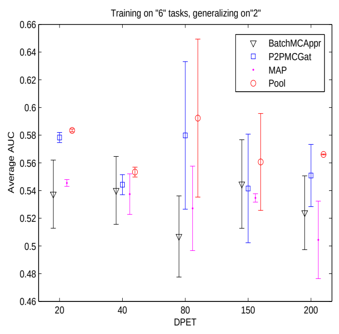
For reasons of completeness, we present an analysis of the character classification problem in the partially observed tasks scenario. Here the validity of the model assumptions is dubious; nevertheless, we believe that interesting lessons can be learned from model failure. The fully observed tasks analysis of the character classification problem did not reveal any clusters of tasks. Furthermore, there is no reason to believe that the low-error joint prediction assumption may hold: some tasks might even be anticorrelated, as in tasks 'a/g' and 'g/y', where letter 'g' belongs to the negative class for task 'a/g', and to the positive class for task 'g/y'. Therefore, the character classification problem is ill-suited for this type of experiments. This is borne out by experimental evidence: we present results with 4, 5, and 6 training tasks, which are shown in subfigures a, b, and c of figure 5.13 respectively. As is shown, increasing the number of tasks and the number of training points per task does not improve the performance in any of the methods, indicating statistical inconsistency of the model assumptions with the data.



(a)



(b)



(c)

Figure 5.13: Average AUC on the unseen tasks for the Character classification data set; (a) Training on 4 tasks, (b) Training on 5 tasks, (c) Training on 6 tasks.

5.4.2.5 Observations

Meta-generalising in a partially observed tasks scenario is an extremely hard problem; nevertheless, we believe there are some interesting points that can be made from the previous experimental analysis. Below we summarise the most important observations for this scenario.

1. In situations where there are clusters of tasks, even though the model hasn't seen all tasks, the Batch method can still make accurate predictions that reaches the performance of the fully observed tasks case. Pragmatically, one could consider whether the training phase of the model has revealed clusters of tasks when deciding which prediction method to apply.

2. In multi-task problems where the correlations between the tasks are less pronounced, but where the low-error joint prediction is satisfied and where a sufficient number of training tasks is available, the method that is most appropriate is the P2PGat, since it provides a more flexible task assignment mechanism than the Batch mode. The validity of the low-error joint prediction assumption can sometimes be assessed from the nature of the problem (as in the arrhythmia case).

3. Sufficient exploration of the task space is essential for the success of the method. While we have not tested our model for very large numbers of training tasks,

the results suggest that often a significant improvement in performance can be achieved when the number of training tasks crosses a critical number, indicating a sufficient coverage of the task space.

5.5 Conclusions

In this chapter we presented an investigation on the use of Gaussian Processes for meta-generalisation, i.e. predicting on unseen learning tasks by leveraging the information of several, related tasks. Our model attacks the meta-generalisation problem by coupling two GPs, a multi-class classifier that learns task responsibilities, and a multi-task classifier that learns prediction models on individual tasks as well as learning the global correlation structure between training tasks. While it should be emphasised that this is an initial attempt to address what is certainly a very ambitious problem, we believe the model will prove useful to understand meta-generalisation. First of all, it provides a constructive approach to meta-generalisation: most previous studies (Baxter, 2000) have been mainly theoretical investigations attempting to establish the necessary conditions for meta-generalisation to work, or have focused on the domain adaptation scenario (Ben-David et al., 2007, 2010). Our model is an attempt to translate these conditions into a model, and to investigate how well such a model may perform on real meta-generalisation problems.

It is important to remark that our method crucially relies on the ability to learn the covariance matrix of a GP: the fundamental ingredient in the work is the task correlation matrix which captures the correlations between source tasks. This not only has a significant impact on the prediction results, but can reveal the presence of clusters of tasks within the data, hence guiding the choice of the appropriate prediction method (Batch or P2P). Many multi-task learning approaches do not explicitly model the correlations, but transfer learning solely through some shared prior over parameters (Yu et al., 2005, e.g.). While this could have computational advantages, we would argue that the implicit modelling of task correlations would make them less suitable for meta-generalisation.

Other possible ways of modeling the task covariance matrix would be by extracting task descriptor features from the training and test data sets and using those as inputs for the *task covariance function* (Bonilla et al., 2007). In this setting the task covariance function would act as the gating network in place of the multi-class classifier in the CMTMC model we proposed in this work.

While we believe that our results are encouraging and help clarify the importance of the various assumptions underlying meta-generalisation, it remains undeniable that in many practical situations it is impossible to assess the validity of these assumptions, making meta-generalisation an extremely challenging problem. Possible avenues to extend the applicability of the approach could be to consider task descriptor features, or to introduce a semi-supervised element in the model in the spirit of domain adaptation approaches.

Chapter 6

Conclusions and Future Research Directions

This thesis examined the application of Gaussian Processes to the framework of Transfer Learning (TL). We provided the reader the necessary background information about the framework of transfer learning, which was divided into three major categories: the *Inductive*, the *Transductive*, and the *Unsupervised* categories of TL. Until now, research conducted in the TL framework with GPs was mostly oriented towards learning problems that are classified in the Inductive category of TL, and particularly for Multi-task learning (MTL) and the closely related setting of Multi-responce learning.

Table 6.1 shows the individual learning settings of each category of TL in relation to research conducted with GPs in the past years and in this thesis. The first column of symbols next to each subsetting characterizes research in the literature, and the second symbol depicts research in this thesis. Symbol $\square$ stands for settings for which models have been proposed and properly analyzed, symbol $\diamond$ indicates that a model has been proposed but has not been tested, and symbol $\times$ signifies a gap in the literature until now. As can easily be seen from the first column of symbols, from the eight subsettings of TL only two, Multi-task/response and SS Multi-task have properly been addressed, while from the subsetting of Cross-domain transfer only the problem of covariate shift has been approached in Storkey and Sugiyama (2007); on top of that an approach for translated learning has been proposed by Chai (2010). Related to research conducted in this thesis, symbol $\surd$ stands for models proposed and thoroughly examined, symbol $\triangle$ indicates that a model has been proposed but has not been tested, and symbol $?$ points out future work to be proposed. Note that a more general model that has been proposed for covariate shift by Storkey and Sugiyama (2007)

Table 6.1: Transfer Learning and Gaussian Process Research. The first column of symbols next to each subsetting characterizes research in the literature, and the second symbol depicts research in this thesis. In the first column, symbol $\square$ stands for settings for which models have been proposed and properly analyzed, symbol $\diamond$ indicates that a model has been proposed but has not been tested, and symbol $\times$ signifies a gap in the literature. In the second column, symbol $\surd$ stands for models proposed and thoroughly examined, symbol $\triangle$ indicates that a model has been proposed but has not been tested, and symbol ? points out future work to be proposed

Inductive			**Transductive**			**Unsupervised**		
Multi-task Multi-response	$\square$	$\surd$	*Tranlated*	$\diamond$		*MT Unsupervised*	$\times$	?
SS Multi-task	$\square$	$\surd$	*Cross-domain*	$\square$	$\triangle$	*Meta-generelazing*	$\times$	$\surd$
Self-taught	$\times$	$\triangle$						

6.1 Conclusions

This thesis examined the application of Gaussian Processes to three different forms of Transfer Learning (TL), by making extensive use of the IMC model of Cressie (1993) and Bonilla et al. (2008). We placed emphasis on how the Kronecker factorization of the GP covariance prior of Bonilla et al. (2008) can be exploited to be employed in various forms of TL with a focus on classification problems. The original contributions of this thesis are presented in chapters 3, 4, and 5, with the addition of chapter 2 whose contribution is considerable from the application point of view and chapter 1 that provides an overview of the framework of Transfer Learning. Chapters 2 and 3 examine the multi-task scenario and chapter 4 extends the previous two chapters to the semi-supervised case, where a model is described that is able to tackle Self-taught and Domain adaptation problems. Hence, chapters 2-4 investigate the Inductive and the Transductive category of TL, whereas chapter 5 is concerned with the problem of Meta-generalising from the Unsupervised category of TL.

Chapter 2 provided the reader a detailed and justified classification of existing multi-task approaches for GP regression which resulted into a weak form of transfer as the Parameter Transfer, and a stronger form as the Inductive transfer. In the experimental part of chapter 2 we applied MT GP regression to the prediction of pK_a values for different molecules, where MTL was found to offer a significant improvement upon STL for small training sizes. Additionally, although we have not tested all available methods from each category, empirical evidence from the methods tested supports our argument that the Parameter transfer is weaker than the IMC model of Bonilla et al. (2008) from the Inductive transfer, which is also supported by experimental results in chapters 3 and 4 on classification problems.

In chapter 3 we applied the multi-task model of Bonilla et al. (2008) to classification problems; by employing one stochastic and two deterministic approximations we deal with the non-Gaussian likelihood classification problems impose. In the approximate approach we employed one algorithm based on the factorized distributions of Variational inference and one algorithm based on the EP approximation, to approximate the non-Gaussian posterior of the latent function. Experimental results on three data sets showed that MTL with GPs greatly improves upon STL for few training data and produces competitive or better results compared to methods from the Parameter transfer and with other competing methods such as SVMs.

In chapter 4 we address the problem of semi-supervised multi-task learning within the GP framework. Combining the SS model of Sindhwani et al. (2007) and the MTL model of Bonilla et al. (2008), we found that the taxonomy of multi-task learning introduced in chapter 2 can also be extended in the SS case, to give the SS analogs of the Parameter and the Inductive transfer. In the SS Parameter transfer the latent functions of each task were learned from labeled and unlabeled data but were independent given the parameters, whereas in the SS Inductive transfer inter-task dependencies were introduced through the IMC model giving rise to two possible formulations to exploit unlabeled data. In the first model, the Static Geometry (SG), unlabeled data of each task are treated independently whereas in the second approach, the Flexible Geometry (FG), the unlabeled data of each task communicate through the task covariance matrix. Experimental results on one artificial and three real data sets give more insights about the methods; the Spam classification and the Sentiment analysis problems, manifest the benefits SS-MTL has to offer for few labeled training data; for the Letters classification problem whose tasks do not have a high degree of correlation, we observed that the best performance was achieved by the weakest and simplest form of MTL the Parameter transfer. It is concluded that the weak form of transfer of the Parameter category should be preferred when task correlations vary significantly. Additionally, we observed that in general the SG performed better than the FG formulation except in the case of the Letters data set which could be justified as that the FG could learn better the task correlations. However, the importance of the FG formulation is on that it can act as the cornerstone for attacking more complex problems as the Self-taught learning and Domain adaptation. Although until now our preliminary results on DA and SeTL were not satisfactory compared to other baseline methods, we believe that it is a promising route to follow.

In chapter 5 we move on to the Unsupervised category of TL and we address the problem of Meta-generalising with GPs through the setting of multi-task learning. Although this setting has been proposed by Baxter (2000), to our knowledge we are the first to put it into a formal context, conjecture the assumptions that it should be based on, develop a model for it and finally thoroughly examine under what circumstances it can work. We show that Meta-generalising can be applied with performance reaching that of MTL(MTL is considered the supervised analog of Meta-generalising) in situations where either we have knowledge that all possible tasks have been sampled (Fully observed case) or in situations where there are clusters of tasks. In those two situations we have empirically shown that the optimal way of making predictions is by using information by all data points from the test target task (Batch). In situations where the tasks are correlated but in a more random way we observed that Meta-generalising can achieve better levels of performance than baselines using the mixture of experts way of predictions (P2P) that was proposed. We believe that these two observations in this extreme form of Transfer learning highlight the importance the structure of the space of the tasks has and the need for methods that can efficiently identify it.

6.2 Future Research Directions

This section suggests some possible directions for future work that we believe are worth examining. Clearly, table 6.1 shows a gap in the literature in the categories of Transductive and Unsupervised transfer Learning, and partially in the Inductive category for Self-taught learning. Although this thesis made a first attempt to solve problems like Domain adaptation, Self-taught learning and Meta-gereralizing, we believe that these forms of Transfer learning are extremely challenging and that other methods based on Gaussian Process models could be developed to solve them.

Semi-supervised multi-task learning compared to multi-task learning is more recent and in general has not received as much attention. As we saw the importance of this learning setting goes beyond its main objective being able to leverage information from different tasks and unlabeled data because of its close relation to the problems of DA and SeTL. Due to that reason we believe that there should more focus on that setting from the theoretical and the practical point of view. Although there has been a considerable amount of work on theoretically characterizing the benefits single SS learning has reported to deliver (Philippe, 2007; Lafferty and Wasserman, 2008; Singh et al., 2009), SS-MTL has not received that much attention with the exception of the work of Daumé et al. (2010). In MT-GPs Chai (2009) has theoretically analyzed the IMC, whereas this type of analysis (generalisation error bounds for SS-MTL) is missing and we believe would be of paramount importance. In relation to models proposed in this thesis for Domain adaptation and Self-taught learning, in the future we intend to experiment with different constructions of the Laplacian or with variations of the models proposed in section 4.5 on more data sets to improve performance.

Related to the Unsupervised category, there is very limited work addressing problems like dimensionality reduction (DR) (Wang et al., 2008; Pan et al., 2009) or clustering (Yang et al., 2009; Dai et al., 2008) for multiple tasks. Dimensionality reduction with GPs has been approached with the Gaussian Process Latent Variable Model (GP-LVM) of Lawrence (2004). It is our opinion that the GP-LVM can act as the basis for constructing a model that will be able to leverage information from multiple tasks to perform dimensionality reduction, exactly as the SSGP model of Sindhwani et al. (2007) acted in the Semi-supervised multi-task case. Similarly to the MTL and to the SS-MTL scenarios the GP-LVM can be modified in such a way to give the Parameter transfer and the Inductive transfer analogs for dimensionality reduction. In the Parameter transfer the prior distribution of some parameters can be shared while in the Inductive transfer we can allow the latent functions of the different task that map the high dimensional data to a low dimensional manifold to interact through the Kronecker product factorization of the prior. The successful combination of the GP-LVM with the IMC model would provide a unifying treatment of Transfer learning with GPs through the IMC model, excluding the problem of translated learning.

Last but not least we would like to stress that although Meta-generalising is an extremely hard problem and its applicability to a real situation at first sight might be questionable, we believe that there are real applications where it can actually be beneficial. In addition to the arrhythmia classification problem, the estimation of pK_a values of different molecules is an ideal problem that can be recast as a Meta-generalisation problem since the ability to predict the pK_a value of a molecule by finding with which task is correlated can be used as a decision support system. Therefore, in the future we intend to adapt the CMTMC model for multi -task regression problems and apply it in the pK_a data set.

Appendix A

Appendix to Chapter 3

This appendix provides additional information for the Gibbs sampling scheme in section 3.2.1.1, and the Variational probit model in section 3.2.2.2. A complete and detailed exposition of the EP approximation that naturally extends to the multi-task scenario can be found in (Rasmussen and Williams, 2005, Ch. 3.6) and will not be reproduced here.

A.1 MCMC methods

Markov Chain Monte Carlo (MCMC) is a general family of methods for approximating the posterior distribution of random variables that cannot be computed in closed form. This is achieved by drawing samples in a sequential manner such that each sample depends on its previous value, and thus the samples form a Markov chain.

A.1.1 The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is a generalisation of the Metropolis algorithm that generates samples from a proposal distribution $q(\Theta^t|\Theta^{t-1})$ and accepts them based on an *acceptance ratio*. The difference between the Metropolis and the Metropolis-Hastings algorithm is that in the latter the proposal distribution is *assymetric* $q(\Theta^t|\Theta^{t-1}) \neq q(\Theta^{t-1}|\Theta^t)$. A new sample Θ^* is accepted with probability,

$$A(\Theta^*, \Theta^{t-1}) = \min\left(1, \frac{\tilde{p}(\Theta^*)q(\Theta^{t-1}|\Theta^*)}{\tilde{p}(\Theta^{t-1}q(\Theta^*|\Theta^{t-1})}\right) \tag{A.1}$$

where $A(\alpha_*, \alpha)$ is the acceptance ratio, and $\tilde{p}(\Theta^*)$ is the unnormalised posterior distribution of parameter Θ. The Metropolis-Hastings procedure is to draw a number from a uniform distribution on the unit interval and accept the new sample if the number is smaller or equal to $A(\Theta^*, \Theta^{t-1})$.

A.1.2 Gibbs Sampling

The Gibbs sampling scheme is a particular type of Markov chain simulation that updates groups of variables sequentially. Consider the case where a model has v groups of random variables Θ_j whose posterior distribution needs to be estimated. The joint distribution of all v groups of variables $\Theta = \{\Theta_1, \dots, \Theta_v\}$ can be written as,

$$p(\Theta) = p(\Theta_j|\Theta_{-j})p(\Theta_{-j}) \tag{A.2}$$

where $p(\Theta_j|\Theta_{-j})$ is the distribution of the j^{th} group conditioned on all other groups, and $p(\Theta_{-j})$ denotes the joint probability distribution of all other groups of variables except the j^{th}. It is assumed that while $p(\Theta)$ is too complex to draw samples from $p(\Theta_j|\Theta_{-j})$ is easier to work with (MacKay, 2003). Using Baye's rule the conditional distribution of a group of variables can be written as,

$$p(\Theta_j|\Theta_{-j}) = \frac{p(\Theta_{-j}|\Theta_j)p(\Theta_j)}{Z_\Theta}, \tag{A.3}$$

where $p(\Theta_j)$ is the prior distribution of that group and Z_Θ is the normalization constant. The prior distribution $p(\Theta_j)$ is usually taken to be conjugate to the likelihood so that the posterior has the same functional form as the prior, and the normalization term Z_Θ is computed from,

$$Z_\Theta = \int p(\Theta_{-j}|\Theta_j)p(\Theta_j)\mathrm{d}\Theta_j. \tag{A.4}$$

In a nutshell, Gibbs sampling updates each group of variables Θ_j at a time conditioned on all the others. This operation is performed in a cyclic manner such that each parameter Θ_j^t at iteration t, is updated conditioned on the most recent updated parameters Θ_{-j}^t. There are two issues with the Gibbs sampling scheme that need attention. The first is that the initial state is not a good representative of the whole distribution and the second is that consecutive samples have a high degree of correlation (Barber, 2011). The first issue is resolved by discarding the first 1/3 samples drawn, something that is usually refereed to as *burn-in* stage, and the second is tackled by subsampling the sequence of samples by retaining only one sample every ten produced for the final computation. Last note that if not all of the parameters have tractable(conjugate) posteriors, then Gibbs sampling can be combined with individual *Metropolis-Hastings*

(MH) algorithms to draw samples from these posteriors. A complete treatment of the subject of sampling methods can be found in many textbooks as MacKay (2003); Gelman et al. (2004); Bishop (2006); Barber (2011).

A.2 Gibbs sampling for Multi-task GP Classification

It is straightforward to derive the update equations for variables **h**, and **f** which were given in equations 3.5, 3.8. These updates can easily be computed by conditioning on the variables that they depend and normalizing appropriately. For explanatory reasons we show in detail how to compute the update equation for the task covariance matrix given in equation 3.9, which acts as a concrete example that the posterior has the same functional form as the prior when the prior is conjugate to the likelihood.

A.2.1 Auxiliary Variables

The posterior of the auxiliary variables conditioned on all other variables is given by Bayes rule as,

$$\begin{aligned} p(\mathbf{h}|\mathbf{f},\mathbf{y}) &= \frac{p(\mathbf{f}|\mathbf{h})p(\mathbf{y}|\mathbf{h})}{p(\mathbf{f},\mathbf{y})} \\ &= \prod_{n=1}^{N} \frac{1}{Z_n} \mathcal{N}_{h_n}(f_n, 1)\delta(h_n) \\ &= \prod_{n=1}^{N} \left(f_n + y_n \frac{\mathcal{N}_{h_n}(f_n, 1)}{\Phi(y_n f_n)} \right), \end{aligned} \tag{A.5}$$

which is the product of truncated univariate gaussians.

A.2.2 Latent function

The posterior of the latent variables **f** is found by:

$$\begin{aligned} p(\mathbf{f}|\mathbf{h},K^t,\alpha) &= \frac{p(\mathbf{h}|\mathbf{f})p(\mathbf{f}|K^t,\alpha)}{p(\mathbf{h})} \\ &= \mathcal{N}_{\mathbf{f}}(\Sigma\mathbf{h},\Sigma), \end{aligned} \tag{A.6}$$

where $\Sigma = \left(I + (K^t \otimes K^x)^{-1}\right)^{-1} = (K^t \otimes K^x)\,(I + K^t \otimes K^x)^{-1}$.

A.2.3 Task covariance

The posterior of the task covariance matrix K^t is given by,

$$p(K^t|\mathbf{f},\theta^t) = \frac{p(\mathbf{f}|K^t)p(K^t|\theta^t)}{Z_{K^t}}, \tag{A.7}$$

This involves the computation of three terms. The first is the distribution of the latent function f conditioned on K^t, where by exploiting some properties of the Kronecker and the "vec" operator can be written as,

$$p(\mathbf{f}|K^t) = \frac{\exp\left\{-\frac{1}{2}\mathrm{trace}\left(\left(K^t\right)^{-1}\mathbf{F}^T\left(K^x\right)^{-1}\mathbf{F}\right)\right\}}{(2\pi)^{\frac{1}{2}NM}|K^t|^{\frac{1}{2}N}|K^x|^{\frac{1}{2}M}}, \tag{A.8}$$

where as mentioned previously on Chapter 3, Γ_M is the multivariate Gamma function of M dimensions, and "etr" is used to denote the the exponential of the trace, etr $=$ $\exp\{\mathrm{trace}(.)\}$. The second is the prior of the task covariance matrix,

$$p(K^t|\theta^t = \beta, \Lambda) = \frac{2^{-\frac{1}{2}(\beta-M-1)M}|\Lambda|^{\frac{1}{2}(\beta-M-1)}}{\Gamma_M\left[\frac{1}{2}(\beta-M-1)\right]|K^t|^{\frac{1}{2}\beta}}\mathrm{etr}\left\{-\frac{1}{2}{K^t}^{-1}\Lambda\right\}, \tag{A.9}$$

and the third is the normalization constant given by,

$$Z_{K^t} = \int p(\mathbf{f}|K^t)p(K^t|\theta^t)\mathrm{d}K^t. \tag{A.10}$$

The product of the prior of K^t and the distribution of $\mathbf{f}$ conditioned on K^t $(p(\mathbf{f}|K^t)p(K^t|\theta^t))$, which appears also inside the integral of the normalizing constant Z_{K^t} is given by,

$$\frac{1}{(2\pi)^{\frac{1}{2}NM}|K^x|^{\frac{1}{2}M}}\frac{2^{-\frac{1}{2}(\beta-M-1)M}|\Lambda|^{\frac{1}{2}(\beta-M-1)}}{\Gamma_M\left[\frac{1}{2}(\beta-M-1)\right]|K^t|^{\frac{1}{2}(N+\beta)}}\mathrm{etr}\left\{-\frac{1}{2}\left(\left(K^t\right)^{-1}\left(\mathbf{F}^T\left(K^x\right)^{-1}\mathbf{F}+\Lambda\right)\right)\right\}, \tag{A.11}$$

where by inspection it is noticed that this is an unnormalised $\mathcal{IW}$ distribution with a new scale matrix and degrees of freedom given by $\Lambda_N = \left(\mathbf{F}^T\left(K^x\right)^{-1}\mathbf{F}+\Lambda\right)$, and $\beta_N = N+\beta$, respectively. Normalizing appropriately inside the integral of Z_{K^t} gives that,

$$Z_{K^t} = \frac{(2\pi)^{-\frac{1}{2}NM}|K^x|^{-\frac{1}{2}M}2^{-\frac{1}{2}(\beta-M-1)M}|\Lambda|^{\frac{1}{2}(\beta-M-1)}}{\Gamma_M\left[\frac{1}{2}(\beta-M-1)\right]}\overbrace{\int\mathcal{IW}_M(K^t|\beta_N,\Lambda_N)\mathrm{d}K^t}^{=1}$$

$$= \frac{(2\pi)^{-\frac{1}{2}NM}|K^x|^{-\frac{1}{2}M}2^{-\frac{1}{2}(\beta-M-1)M}|\Lambda|^{\frac{1}{2}(\beta-M-1)}}{\Gamma_M\left[\frac{1}{2}(\beta-M-1)\right]} \tag{A.12}$$

Substituting equations A.12 and A.11 to the equation A.7 gives that the posterior of the task covariance matrix K^t will be given by,

$$p(K^t|\mathbf{f},\theta^t) = \frac{\frac{2^{-\frac{1}{2}(\beta-M-1)M}|\Lambda|^{\frac{1}{2}(\beta-M-1)}}{(2\pi)^{\frac{1}{2}NM}|K^x|^{\frac{1}{2}M}\Gamma_M\left[\frac{1}{2}(\beta-M-1)\right]|K^t|^{\frac{1}{2}(\beta+N)}}\text{etr}\left\{-\frac{1}{2}(K^t)^{-1}\left(F(K^x)^{-1}F^T+\Lambda\right)\right\}}{\frac{(2\pi)^{-\frac{1}{2}NM}|K^x|^{-\frac{1}{2}M}2^{\frac{1}{2}(NM)}|\Lambda|^{\frac{1}{2}(\beta-M-1)}\Gamma_M\left[\frac{1}{2}(\beta+N-M-1)\right]}{\Gamma_M\left[\frac{1}{2}(\beta-M-1)\right]\left|F(K^x)^{-1}F^T+\Lambda\right|^{\frac{1}{2}(\beta+N-M-1)}}}$$

$$= \frac{2^{-\frac{1}{2}(\beta_N-M-1)M}|\Lambda_N|^{\frac{1}{2}(\beta_N-M-1)}}{\Gamma_M\left[\frac{1}{2}(\beta_N-M-1)\right]|K^t|^{\frac{1}{2}(\beta_N)}}\text{etr}\left\{-\frac{1}{2}(K^t)^{-1}(\Lambda_N)\right\} \tag{A.13}$$

From the last equation A.13, it is shown that the posterior of K^t is an Inverse Wishart distribution with scale matrix $\Lambda_N = F(K^x)^{-1}F^T + \Lambda$, and degrees of freedom $\beta_N = \beta + N$.

A.3 Variational probit model for Multi-task classification

A.3.1 Approximate Inference

The treatment that follows for $Q(\mathbf{f})$ and $Q(\mathbf{h})$ assumes that the complete set of responses is *not* available. This implies that the values of the latent function and the auxiliary variable have to be computed only at the observed N locations. The case where for each input the outputs for all M are observed can be handled by substituting the sums over N with the sum over NM observations. Moreover, we will write K^x without any subscripts to denote the covariance matrix between all training points from all tasks $\mathbf{X}$, unless stated otherwise.

A.3.1.1 $Q(\mathbf{f})$

$$\begin{aligned}
Q(\mathbf{f}) &\propto \exp\left\{\mathbb{E}_{Q(\mathbf{h})}\left\{\sum_{n=1}^{N}\log p(h_n|f_n) + \log p(\mathbf{f}|\mathbf{X})\right\}\right\}, \\
&\propto \exp\left\{\mathbb{E}_{Q(\mathbf{h})}\left\{-\frac{1}{2}\mathbf{h}^T\mathbf{h} + \mathbf{f}^T\mathbf{h} - \frac{1}{2}\mathbf{f}^T\mathbf{f} - \frac{1}{2}\mathbf{f}^T\left(K^t\otimes K^x\right)^{-1}\mathbf{f} + const.\right\}\right\}, \\
&\propto \exp\left\{-\frac{1}{2}\mathbf{f}^T(I + \left(K^t\otimes K^x\right)^{-1})\mathbf{f} + \mathbf{f}^T\tilde{\mathbf{h}} + const.\right\},
\end{aligned} \tag{A.14}$$

which gives that $Q(\mathbf{f}) = \mathcal{N}_{\mathbf{f}}(\tilde{\mathbf{f}}, \Sigma)$ where $\tilde{\mathbf{f}} = \Sigma\tilde{\mathbf{h}}^x$, and

$$\Sigma = (I + (K^t \otimes K^x)^{-1})^{-1} = K^t \otimes K^x(I + K^t \otimes K^x)^{-1}. \tag{A.15}$$

A.3.1.2 $Q(\mathbf{h})$

$$\begin{aligned} Q(\mathbf{h}) &\propto \exp\left\{\mathbb{E}_{Q(\mathbf{f})}\left\{\sum_{n=1}^{N} \log p(y_n|h_n) + \log p(h_n|f_n)\right\}\right\}, \\ &\propto \exp\left\{\log\left(\prod_{n=1}^{N} p(y_n|h_n)\right) + \log\left(\prod_{n=1}^{N} \mathcal{N}_{h_n}(\tilde{f}_n, 1)\right)\right\}, \\ &\propto \prod_{n=1}^{N} \mathcal{N}_{h_n}(\tilde{f}_n, 1)\delta(h_n), \end{aligned} \tag{A.16}$$

which gives that $Q(h_n) = \frac{1}{z_n}\mathcal{N}_{h_n}(\tilde{f}_n, 1)\delta(h_n)$, and we have that

$$Q(h_n) = \left\{ \begin{array}{c} \frac{1}{Z_n}\int_0^{+\infty} h_n\mathcal{N}_{h_n}(\tilde{f}_n, 1)\mathrm{d}h_n \ \text{ for } \ y_n = +1 \\ \\ \frac{1}{Z_n}\int_{-\infty}^{0} h_n\mathcal{N}_{h_n}(\tilde{f}_n, 1)\mathrm{d}h_n \ \text{ for } \ y_n = -1 \end{array} \right\}, \tag{A.17}$$

For $y_n = +1$ we have:

$$\begin{aligned} \int_0^{\infty} h_n\mathcal{N}_{h_n}(\tilde{f}_n, 1)\mathrm{d}h_n &= \int_{-f_n}^{\infty} (z + \tilde{f}_n)\mathcal{N}_z(0, 1)\mathrm{d}z, \text{(where we have used : } z = h_n - \tilde{f}_n) \\ &= \int_{-f_n}^{\infty} z\mathcal{N}_z(0, 1)\mathrm{d}z + \tilde{f}_n\int_{-f_n}^{\infty} \mathcal{N}_z(0, 1)\mathrm{d}z \\ &= \int_{-f_n}^{\infty} z\mathcal{N}_z(0, 1)\mathrm{d}z + \tilde{f}_n\Phi(f_n). \end{aligned} \tag{A.18}$$

Continuing, $\int_{-f_n}^{\infty} z\mathcal{N}_z(0, 1)\mathrm{d}z$ is computed as

$$\begin{aligned} \int_{-f_n}^{\infty} z\mathcal{N}_z(0, 1)\mathrm{d}z &= \frac{1}{(2\pi)^{\frac{1}{2}}}\int_{-\tilde{f}_n}^{\infty} ze^{-\frac{z^2}{2}}\mathrm{d}z, \ \ \text{(using } t = \frac{z^2}{2} \text{ and } \mathrm{d}t = z\mathrm{d}z) \\ &= \frac{1}{(2\pi)^{\frac{1}{2}}}\int_{\frac{\tilde{f}_n{}^2}{2}}^{\infty} e^{-t}\mathrm{d}t = -\frac{1}{(2\pi)^{\frac{1}{2}}}\left[e^{-t}\right]_{\frac{\tilde{f}_n{}^2}{2}}^{\infty} = \frac{1}{(2\pi)^{\frac{1}{2}}}e^{-\frac{\tilde{f}_n{}^2}{2}} \\ &= \mathcal{N}_{\tilde{f}_n}(0, 1) \end{aligned} \tag{A.19}$$

The normalization constant for $y = +1$ is found by,

$$\begin{aligned} Z_n &= \int_0^{\infty} \mathcal{N}_{h_n}(\tilde{f}_n, 1)\mathrm{d}h_n \qquad \text{(setting again } z = h_n - \tilde{f}_n) \\ &= \int_{-\tilde{f}_n}^{\infty} \mathcal{N}_z(0, 1)\mathrm{d}h_n = 1 - \Phi(-\tilde{f}_n) = \Phi(\tilde{f}_n). \end{aligned} \tag{A.20}$$

Putting together equations A.18, and A.20 gives that

$$\tilde{h}_n = \tilde{f}_n + \frac{\mathcal{N}_{\tilde{f}_n}(0,1)}{\Phi(\tilde{f}_n)} \tag{A.21}$$

A similar treatment follows for $y_n = -1$; here we only show the derivation of the normalization constant, which is given by ,

$$\begin{aligned} Z_n &= \int_{-\infty}^{0} \mathcal{N}_{h_n}(\tilde{f}_n, 1)\mathrm{d}h_n \qquad (z = h_n - \tilde{f}_n) \\ &= \int_{-\infty}^{-\tilde{f}_n} \mathcal{N}_z(0,1)\mathrm{d}h_n \\ &= \Phi(-\tilde{f}_n). \end{aligned} \tag{A.22}$$

Thus, in a compact form for both $y_n = \pm 1$ we will have:

$$Q(h_n) = \tilde{f}_n + y_n \frac{\mathcal{N}_{\tilde{f}_n}(0,1)}{\Phi(y_n \tilde{f}_n)}. \tag{A.23}$$

A.3.2 Lower Bound

This section provides a step by step derivation of the lower bound $\mathcal{L}(Q)$. In order to keep the notation light we will be using $\mathbf{K}$ to refer to the Kronecker product of the task covariance matrix K^t with the data covariance matrix K^x, thus $\mathbf{K} = K^t \otimes K^x$.

$$\begin{aligned} \mathcal{L}(Q) &= \mathbb{E}_{Q(\mathbf{h})Q(\mathbf{f})}\{\log p(\mathbf{y},\mathbf{h},\mathbf{f}|\mathbf{X})\} - \mathbb{E}_{Q(\mathbf{h})Q(\mathbf{f})}\{\log Q_{\mathbf{h}}(\mathbf{h})Q_{\mathbf{f}}(\mathbf{f})\} \\ &= \mathbb{E}_{Q(\mathbf{h})Q(\mathbf{f})}\{\log p(\mathbf{h}|\mathbf{f})\} && \text{(A.24)} \\ &\quad + \mathbb{E}_{Q(\mathbf{f})}\{\log p(\mathbf{f}|\mathbf{X})\} && \text{(A.25)} \\ &\quad - \mathbb{E}_{Q(\mathbf{h})}\{\log Q(\mathbf{h})\} && \text{(A.26)} \\ &\quad - \mathbb{E}_{Q(\mathbf{f})}\{\log Q(\mathbf{f})\}. && \text{(A.27)} \end{aligned}$$

To simplify the derivation each term of the lower bound is computed separately. For term A.24 we will have:

$$\begin{aligned} \mathbb{E}_{Q(\mathbf{h})Q(\mathbf{f})}\{\log p(\mathbf{h}|\mathbf{f})\} &= \mathbb{E}_{Q(\mathbf{h})Q(\mathbf{h})}\left\{-\frac{N}{2}\log 2\pi - \frac{1}{2}\log I - \frac{1}{2}(\mathbf{h}-\mathbf{f})^T(\mathbf{h}-\mathbf{f})\right\} \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2} < \mathbf{h}^T\mathbf{h} > + < \mathbf{h}^T >< \mathbf{f} > - \frac{1}{2} < \mathbf{f}^T\mathbf{f} > \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2} < \mathbf{h}^T\mathbf{h} > + \tilde{\mathbf{h}}^T\tilde{\mathbf{f}} - \frac{1}{2}\mathrm{trace}(< \mathbf{f}\mathbf{f}^T >) \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2} < \mathbf{h}^T\mathbf{h} > + \tilde{\mathbf{h}}^T\tilde{\mathbf{f}} - \frac{1}{2}\mathrm{trace}(\Sigma + \tilde{\mathbf{f}}\tilde{\mathbf{f}}^T) \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2} < \mathbf{h}^T\mathbf{h} > + \tilde{\mathbf{h}}^T\tilde{\mathbf{f}} - \frac{1}{2}\mathrm{trace}(\Sigma) - \frac{1}{2}\tilde{\mathbf{f}}^T\tilde{\mathbf{f}} \end{aligned} \tag{A.28}$$

Term A.25 is computed as:

$$\begin{aligned}
\mathbb{E}_{Q(\mathbf{f})}\left\{\log p(\mathbf{f}|\mathbf{X})\right\} &= \mathbb{E}_{Q(\mathbf{f})}\left\{-\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}\right\}\\
&= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\text{trace}\left(\mathbf{K}^{-1} < \mathbf{f}\mathbf{f}^T >\right)\\
&= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\text{trace}\left(\mathbf{K}^{-1}(\Sigma + \tilde{\mathbf{f}}\tilde{\mathbf{f}}^T)\right)\\
&= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\text{trace}\left(\mathbf{K}^{-1}\left(\mathbf{K}(I+\mathbf{K})^{-1} + \tilde{\mathbf{f}}\tilde{\mathbf{f}}^T\right)\right)\\
&= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\text{trace}\left((I+\mathbf{K})^{-1}\right) - \frac{1}{2}\tilde{\mathbf{f}}^T\mathbf{K}^{-1}\tilde{\mathbf{f}}
\end{aligned} \tag{A.29}$$

Term A.26 is computed as:

$$\begin{aligned}
-\mathbb{E}_{Q(\mathbf{h})Q(\mathbf{f})}\left\{\log Q_{\mathbf{h}}(\mathbf{h})\right\} &= \mathbb{E}_{Q(\mathbf{h})Q(\mathbf{f}}\left\{-\sum_{n=1}^{N}\log\frac{1}{Z_n}\mathcal{N}_{h_n}(\tilde{\mathbf{f}}_n, 1)\right\}\\
&= \sum_{n=1}^{N}\log Z_n + \frac{N}{2}\log 2\pi + \frac{1}{2}\mathbb{E}\left\{(\mathbf{h}-\tilde{\mathbf{f}})^T(\mathbf{h}-\tilde{\mathbf{f}})\right\}\\
&= \sum_{n=1}^{N}\log Z_n + \frac{N}{2}\log 2\pi + \frac{1}{2}\mathbb{E}\left\{\mathbf{h}^T\mathbf{h} - 2\mathbf{h}^T\tilde{\mathbf{f}} + \tilde{\mathbf{f}}^T\tilde{\mathbf{f}}\right\}\\
&= \sum_{n=1}^{N}\log Z_n + \frac{N}{2}\log 2\pi + \frac{1}{2} < \mathbf{h}^T\mathbf{h} > - < \mathbf{h}^T > \tilde{\mathbf{f}} + \frac{1}{2}\tilde{\mathbf{f}}^T\tilde{\mathbf{f}}\\
&= \sum_{n=1}^{N}\log Z_n + \frac{N}{2}\log 2\pi + \frac{1}{2} < \mathbf{h}^T\mathbf{h} > - \tilde{\mathbf{h}}^T\tilde{\mathbf{f}} + \frac{1}{2}\tilde{\mathbf{f}}^T\tilde{\mathbf{f}}
\end{aligned} \tag{A.30}$$

The last term A.27 is computed as:

$$\begin{aligned}
-\mathbb{E}_{Q(\mathbf{f})}\left\{\log Q_{\mathbf{f}}(\mathbf{h})\right\} &= \mathbb{E}_{Q(\mathbf{f})}\left\{-\log\mathcal{N}_{\mathbf{f}}(\tilde{\mathbf{f}}, \Sigma)\right\}\\
&= \frac{N}{2}\log 2\pi + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\mathbb{E}_{Q(\mathbf{f})}\left\{(\mathbf{f}-\tilde{\mathbf{f}})^T\Sigma^{-1}(\mathbf{f}-\tilde{\mathbf{f}})\right\}\\
&= \frac{N}{2}\log 2\pi + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\mathbb{E}_{Q(\mathbf{f})}\left\{\mathbf{f}^T\Sigma^{-1}\mathbf{h} - 2\mathbf{f}^T\Sigma^{-1}\tilde{\mathbf{f}} + \tilde{\mathbf{f}}^T\Sigma^{-1}\tilde{\mathbf{f}}\right\}\\
&= \frac{N}{2}\log 2\pi + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\text{trace}\left(\Sigma^{-1} < \mathbf{f}\mathbf{f}^T >\right) - < \mathbf{f}^T > \Sigma^{-1}\tilde{\mathbf{f}} + \frac{1}{2}\tilde{\mathbf{f}}^T\Sigma^{-1}\tilde{\mathbf{f}}\\
&= \frac{N}{2}\log 2\pi + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\text{trace}\left(\Sigma^{-1}\left(\Sigma + \tilde{\mathbf{f}}\tilde{\mathbf{f}}^T\right)\right) - \tilde{\mathbf{f}}^T\Sigma^{-1}\tilde{\mathbf{f}} + \frac{1}{2}\tilde{\mathbf{f}}^T\Sigma^{-1}\tilde{\mathbf{f}}\\
&= \frac{N}{2}\log 2\pi + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\text{trace}(I) + \frac{1}{2}\text{trace}\left(\Sigma^{-1}\tilde{\mathbf{f}}\tilde{\mathbf{f}}^T\right) - \tilde{\mathbf{f}}^T\Sigma^{-1}\tilde{\mathbf{f}} + \frac{1}{2}\tilde{\mathbf{f}}^T\Sigma^{-1}\tilde{\mathbf{f}}\\
&= \frac{N}{2}\log 2\pi + \frac{1}{2}\log|\Sigma| + \frac{N}{2} + \frac{1}{2}\tilde{\mathbf{f}}^T\Sigma^{-1}\tilde{\mathbf{f}} - \tilde{\mathbf{f}}^T\Sigma^{-1}\tilde{\mathbf{f}} + \frac{1}{2}\tilde{\mathbf{f}}^T\Sigma^{-1}\tilde{\mathbf{f}}\\
&= \frac{N}{2}\log 2\pi + \frac{1}{2}\log|\Sigma| + \frac{N}{2}
\end{aligned} \tag{A.31}$$

Finally, putting together equations A.28, A.29, A.30, and A.31 the lower bound $\mathcal{L}(Q)$ will be given by:

$$\begin{aligned}
\mathcal{L}(Q) &= -\frac{1}{2}\text{trace}(\Sigma) + \sum_{n=1}^{N} \log Z_n - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\text{trace}\left((I+\mathbf{K})^{-1}\right) - \frac{1}{2}\tilde{\mathbf{f}}^T\mathbf{K}^{-1}\tilde{\mathbf{f}} \\
&\quad + \frac{1}{2}\log|\Sigma| + \frac{N}{2} \\
&= \sum_{n=1}^{N} \log Z_n - \frac{1}{2}\tilde{\mathbf{f}}^T\mathbf{K}^{-1}\tilde{\mathbf{f}} - \frac{1}{2}\log|(I+\mathbf{K})| \\
&= \sum_{n=1}^{N} \log Z_n - \frac{1}{2}\tilde{\mathbf{f}}^T(K^t \otimes K^x)^{-1}\tilde{\mathbf{f}} - \frac{1}{2}\log|(I+K^t \otimes K^x)|. \qquad \text{(A.32)}
\end{aligned}$$

A.3.3 Predictions

Inferring the posterior probability of a test point x_* from task j, belonging to class "$+1$" involves the computation of several integrals given below,

$$p(y_{*j} = +1|x_*, \mathbf{X}, \mathbf{y}) = \int p(y_{*j} = 1|h_{*j})p(h_{*j}|x_*, \mathbf{X}, \mathbf{y})\mathrm{d}h_{*j}, \qquad \text{(A.33)}$$

$$p(h_{*j}|x_*, \mathbf{X}, \mathbf{y}) = \int p(h_{*j}|f_{*j})p(f_{*j}|x_*, \mathbf{X}, \mathbf{y})\mathrm{d}f_{*j}, \qquad \text{(A.34)}$$

$$p(f_{*j}|x_*, \mathbf{X}, \mathbf{y}) = \int p(f_{*j}|\mathbf{f})Q(\mathbf{f})\mathrm{d}\mathbf{f} \qquad \text{(A.35)}$$

Where $Q(\mathbf{F}) \sim \mathcal{N}_{\mathbf{f}}(\tilde{\mathbf{f}}, \Sigma)$ and from standard GP results we know that, the mean and the variance of $p(f_{*j}|\mathbf{f})$ will be given by,

$$\mathbb{E}[f_{*j}|\mathbf{f}] = (\mathbf{k}_j^t \otimes \mathbf{k}_{\mathbf{X},x_*}^x)^T(K^t \otimes K^x)^{-1}\mathbf{f}, \qquad \text{(A.36)}$$

$$\text{cov}[f_{*j}|\mathbf{f}] = k_{jj}^t k_{x_*,x_*} - (\mathbf{k}_j^t \otimes \mathbf{k}_{\mathbf{X},x_*}^x)^T(K^t \otimes K^x)^{-1}(\mathbf{k}_j^t \otimes \mathbf{k}_{\mathbf{X},x_*}). \qquad \text{(A.37)}$$

Then the mean and the variance of $p(f_{*j}|x_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\mu_*, \sigma_*^2)$ are given by:

$$\mu_{*j} = (\mathbf{k}_j^t \otimes \mathbf{k}_{\mathbf{X},x_*})^T\left(I + K^t \otimes K^x\right)^{-1}\tilde{\mathbf{h}}, \qquad \text{(A.38)}$$

$$\sigma_{*j}^2 = k_{jj}^t k_{x_*,x_*} - (\mathbf{k}_j^t \otimes \mathbf{k}_{\mathbf{X},x_*})^T\left(I + K^t \otimes K^x\right)^{-1}(k^t \otimes \mathbf{k}_{\mathbf{X},x_*}), \qquad \text{(A.39)}$$

where k_{jj}^t and $\mathbf{k}_j^t$ are the j^{th} diagonal element and the j^{th} column of K^t, $\mathbf{k}_{\mathbf{X},x_*}$ is the covariance vector evaluated between all data points $\mathbf{X}$ and the test point x_*, k_{x_*,x_*} is the marginal variance of the test point, and we write K^x without any subscripts to denote the covariance matrix between all training points from all tasks.

Returning to the computation of $p(h_{*j}|x_*, \mathbf{X}, \mathbf{y})$ in equation(A.34), we will have that,

$$p(h_{j*}|x_*, \mathbf{X}, \mathbf{y}) = \int \mathcal{N}_{h_{*j}}(f_{*j}, 1)\mathcal{N}_{f_{*j}}(\mu_{*j}, \sigma_{*j}^2)\mathrm{d}f_{*j}. \qquad \text{(A.40)}$$

Marginalizing out f_{*j}, the mean ν_{*j} and the variance υ_{*j}^2 of $h_{*j}|x_{*j}$ will be given by, $\nu_{*j} = \mu_{*j}$, and $\upsilon_{*j}^2 = 1 + \sigma_{*j}^2$. Finally, the posterior probability of the test point belonging to the "+1" class in equation A.33 can be computed from,

$$\begin{aligned}
p(y_{*j} = 1|x_*, \mathbf{X}, \mathbf{y}) &= \int p(y_{*j} = 1|h_{*j}) p(h_{*j}|x_*, \mathbf{X}, \mathbf{y}) \mathrm{d}h_{*j} \\
&= \int \delta(h_{*j} > 0) p(h_{*j}|x_*, \mathbf{X}, \mathbf{y}) \mathrm{d}h_{*j} = \int_0^{+\infty} p(h_{*j}|x_*, \mathbf{X}, \mathbf{y}) \mathrm{d}h_{*j} \\
&= \int_0^{+\infty} \frac{1}{(2\pi)^{\frac{1}{2}}} \frac{1}{\upsilon_{*j}} \exp\left\{ -\frac{1}{2} \frac{(h_{*j} - \nu_{*j})^2}{\upsilon_{*j}^2} \right\} \mathrm{d}h_{*j} \\
&\text{(by setting } t = \frac{h_{*j} - \nu_{*j}}{\upsilon_{*j}}, \text{ and } \mathrm{d}t = \frac{1}{\upsilon_{*j}} \mathrm{d}h_{*j}) \\
&= \int_{-\frac{\nu_{*j}}{\upsilon_{*j}}}^{+\infty} \frac{1}{(2\pi)^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2} t^2 \right\} \mathrm{d}t = \int_{-\frac{\nu_{*j}}{\upsilon_{*j}}}^{+\infty} \mathcal{N}_t(0,1) \mathrm{d}t \\
&= \Phi\left(\frac{\nu_{*j}}{\upsilon_{*j}} \right) = \Phi\left(\frac{\nu_{*j}}{\sqrt{1 + \sigma_{*j}^2}} \right) \qquad \text{(A.41)}
\end{aligned}$$

Appendix B

Appendix to Chapter 4

B.1 Data dependent covariance function

This section shows how the data-dependent norm can be recovered by employing two times the matrix inversion lemma. The prior of $\mathbf{f}_X = \{\mathbf{f}_D, \mathbf{f}_T\}$ conditioned on the geometric variables $\mathcal{G}$ will be given by,

$$p(\mathbf{f}_X|\mathcal{G}) \propto \exp\left(-\frac{1}{2}\mathbf{f}_X^T \tilde{\Sigma}_{XX}^{-1}\mathbf{f}_X\right), \quad \text{where} \quad \tilde{\Sigma}_{XX}^{-1} = \begin{bmatrix} \Sigma_{DD} & \Sigma_{DT} \\ \Sigma_{DT}^T & \Sigma_{TT} \end{bmatrix}^{-1} + \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}.$$

To continue we will rewrite $\tilde{\Sigma}_{XX}^{-1}$ as

$$\tilde{\Sigma}_{XX}^{-1} = \begin{bmatrix} \Lambda_{DD} + Q & \Lambda_{DT} \\ \Lambda_{DT}^T & \Lambda_{TT} \end{bmatrix},$$

where we have used that

$$\begin{bmatrix} \Sigma_{DD} & \Sigma_{DT} \\ \Sigma_{DT}^T & \Sigma_{TT} \end{bmatrix}^{-1} = \begin{bmatrix} \Lambda_{DD} & \Lambda_{DT} \\ \Lambda_{DT}^T & \Lambda_{TT} \end{bmatrix}. \tag{B.1}$$

Finally, by employing one more time the matrix inversion lemma we have that,

$$\tilde{\Sigma}_{XX} = \begin{bmatrix} \Phi_{DD} & \Phi_{DT} \\ \Phi_{DT}^T & \Phi_{TT} \end{bmatrix}.$$

The quantity that is of interest is Φ_{TT}, where using equation B.41 will be written as,

$$\Phi_{TT} = \Lambda_{TT}^{-1} + \Lambda_{TT}^{-1}\Lambda_{DT}^T\Phi_{DD}\Lambda_{DT}\Lambda_{TT}^{-1}, \tag{B.2}$$

which involves the computation of two sets of quantities. The procedure is listed below in three steps:

1. First, computing the block matrices of equation B.1, using equations B.37, B.34 and B.35 gives that

$$\Lambda_{TT} = \left(\Sigma_{TT} - \Sigma_{DT}^T \Sigma_{DD}^{-1} \Sigma_{DT}\right)^{-1}, \tag{B.3}$$

$$\Lambda_{DD} = \Sigma_{DD}^{-1} + \Sigma_{DD}^{-1} \Sigma_{DT} \Lambda_{TT} \Sigma_{DT}^T \Sigma_{DD}^{-1}, \tag{B.4}$$

$$\Lambda_{DT} = -\Sigma_{DD}^{-1} \Sigma_{DT} \Lambda_{TT}, \tag{B.5}$$

2. Second, using equation B.38 we have that,

$$\Phi_{DD} = \left(\Lambda_{DD} + Q - \Lambda_{DT} \Lambda_{TT}^{-1} \Lambda_{DT}^T\right)^{-1},$$

where substituting equations B.4, and B.5 in the previous equation we have that

$$\Phi_{DD} = \left(\Sigma_{DD}^{-1} + Q\right)^{-1} = \Sigma_{DD} - \Sigma_{DD}(Q^{-1} - \Sigma_{DD})^{-1} \Sigma_{DD}. \tag{B.6}$$

3. Finally, substituting equations B.3, B.5 and B.6 to Φ_{TT} with some simple matrix manipulations gives that,

$$\Phi_{TT} = \Sigma_{TT} - \Sigma_{DT}^T \left(\Sigma_{DD} + Q^{-1}\right)^{-1} \Sigma_{DT}, \tag{B.7}$$

$$= \Sigma_{TT} - \Sigma_{DT}^T \left(I + Q\Sigma_{DD}\right)^{-1} Q \Sigma_{DT} \tag{B.8}$$

This three step procedure completes the derivation of the data-dependent covariance function given in equations B.7, and B.8.

B.2 Optimizing the Hyperparameters for the SS-MTL models

This section computes the gradients with respect to the hyperparameters of the SS-MTL models presented in sections 4.3.1 and 4.3.2. We give again the table showing the priors and the parameterization of each model in table B.1 to avoid any confusion. Concerning hyperparameters γ_A and γ_I we take the exponential to ensure positivity.

B.2.1 SS-MTL IND

In the SS-MTL IND model both groups of hyperparameters θ^x and θ^l appear inside the covariance matrix $\tilde{K}_{jj}^i$ of each task. Initially we will denote both types of hyperparameters by θ and then we will show the derivatives for each group explicitly. Terms of the

Table B.1: Prior distributions for SS-MTL models.

Model	*Prior*	*Covariance*	*Laplacian*
SS-MTL IND	$p(\mathbf{f}\|D) = \prod_{j=1}^{M} \mathcal{N}_{\mathbf{f}_j}(0, K^i_{jj})$	$K^i_{jj} = \frac{1}{\gamma_A}\tilde{K}^x_{jj}$	$Q^i_j = \frac{\gamma_I}{\gamma_A} Q_j$
SS-MTL SG	$p(\mathbf{f}\|D) = \mathcal{N}(0, K^s)$	$K^s = K^t \otimes \left(\frac{1}{\gamma_A}\tilde{K}^x\right)$	$Q^s = \frac{\gamma_I}{\gamma_A} Q_S$
SS-MTL FG	$p(\mathbf{f}\|D) = \mathcal{N}(0, K^f)$	$K^f = \frac{1}{\gamma_A}\tilde{K}_F$	$Q^f = \frac{\gamma_I}{\gamma_A} Q_D$

log marginal likelihood that depend on the hyperparameters θ of the data covariance function and the graph Laplacian for *regression* problems are,

$$\log p(\mathbf{y}|A,\theta) = -\frac{1}{2}\sum_{j=1}^{M}\left[\mathbf{y}_j^T\left(\tilde{K}^i_{jj} + \sigma_n I\right)^{-1}\mathbf{y}_j + \log\left|\tilde{K}^i_{jj} + \sigma_N I\right|\right] + \text{constant}. \quad \text{(B.9)}$$

Its derivatives with respect to the hyperparameters will be given by,

$$\frac{\partial p(\mathbf{y}|A,\theta)}{\partial\theta} = \frac{1}{2}\sum_{j=1}^{M}\left[\mathbf{y}_j^T\left(\tilde{K}^i_{jj} + \sigma_n I\right)^{-1}\frac{\partial \tilde{K}^i_{jj}}{\partial\theta}\left(\tilde{K}^i_{jj} + \sigma_n I\right)^{-1}\mathbf{y}_j - \operatorname{tr}\left(\left(\tilde{K}^i_{jj} + \sigma_n I\right)^{-1}\frac{\partial \tilde{K}^i_{jj}}{\partial\theta}\right)\right]. \quad \text{(B.10)}$$

For *classification* problems the log marginal likelihood that we approximate by the EP algorithm will be given by,

$$\log p(\mathbf{y}|A,\theta) = -\frac{1}{2}\sum_{j=1}^{M}\left[\tilde{\mu}_j^T\left(\tilde{K}^i_{jj} + \tilde{\Sigma}_j\right)^{-1}\tilde{\mu}_j + \log\left|\tilde{K}^i_{jj} + \tilde{\Sigma}_j\right|\right] + \text{constant}, \quad \text{(B.11)}$$

where $\tilde{\Sigma}_j$ and $\tilde{\mu}_j$ are parameters for the j^{th} task estimated by the EP approximation. Similarly to the regression case the derivatives for classification tasks will be computed by,

$$\frac{\partial p(\mathbf{y}|A,\theta)}{\partial\theta} = \frac{1}{2}\sum_{j=1}^{M}\left[\tilde{\mu}_j^T\left(\tilde{K}^i_{jj} + \tilde{\Sigma}_j\right)^{-1}\frac{\partial \tilde{K}^i_{jj}}{\partial\theta}\left(\tilde{K}^i_{jj} + \tilde{\Sigma}_j\right)^{-1}\tilde{\mu}_j - \operatorname{tr}\left(\left(\tilde{K}^i_{jj} + \tilde{\Sigma}_j\right)^{-1}\frac{\partial \tilde{K}^i_{jj}}{\partial\theta}\right)\right]. \quad \text{(B.12)}$$

Inspecting equations B.10 and B.12, it is observed that it is only required to compute the exact derivatives for each group of hyperparameters of the covariance matrix $\tilde{K}^i_{jj}$ of each task. Including all parameters and taking the exponential of γ_A and γ_I the data dependent covariance matrix for the j^{th} task $\tilde{K}^i_{jj}$ will be given by,

$$\tilde{K}^i_{jj} = \frac{1}{e^{\gamma_A}}\left(K^x_{L_j,L_j} - \left(K^x_{L_j,D_j}\right)^T\left(I + \frac{e^{\gamma_I}}{e^{\gamma_A}}Q_j K^x_{D_j,D_j}\right)^{-1}\frac{e^{\gamma_I}}{e^{\gamma_A}}Q_j\left(K^x_{L_j,D_j}\right)\right), \quad \text{(B.13)}$$

Moreover, we will define as

$$\Psi_j = \left(I + \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_j K^x_{D_j,D_j} \right)^{-1}, \tag{B.14}$$

$$B_j = \Psi_j \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_j. \tag{B.15}$$

The *gradients* for θ^x are given by:

$$\frac{\partial \tilde{K}^i_{jj}}{\partial \theta^x} = \frac{1}{e^{\gamma_A}} \left(\frac{\partial K^x_{L_j,L_j}}{\partial \theta^x} - 2 \left(\frac{\partial K^x_{L_j,D_j}}{\partial \theta^x} \right)^T B_j K^x_{L_j,D_j} + \left(K^x_{L_j,D_j} \right)^T B_j \frac{\partial K^x_{D_j,D_j}}{\partial \theta^x} B_j \left(K^x_{L_j,D_j} \right) \right). \tag{B.16}$$

The *gradients* with respect to θ^l are computed as,

$$\frac{\partial \tilde{K}^i_{jj}}{\partial \theta^l} = \frac{1}{e^{\gamma_A}} \left(K^x_{L_j,D_j} \right)^T \Psi_j \left(\frac{e^{\gamma_I}}{e^{\gamma_A}} \frac{\partial Q_j}{\partial \theta^l} \right) \left[K^x_{D_j,D_j} \Psi_j \left(\frac{e^{\gamma_I}}{e^{\gamma_A}} Q_j \right) - I \right] \left(K^x_{L_j,D_j} \right). \tag{B.17}$$

The *gradients* with respect to γ_A *and* γ_I are computed as,

$$\frac{\partial \tilde{K}^i_{jj}}{\partial \gamma_A} = -\tilde{K}^i_{jj} - \frac{1}{e^{\gamma_A}} \left(K^x_{L_j,D_j} \right)^T \Psi_j \left(\frac{e^{\gamma_I}}{e^{\gamma_A}} Q_j \right) \left[K^x_{D_j,D_j} \Psi_j \left(\frac{e^{\gamma_I}}{e^{\gamma_A}} Q_j \right) - I \right] K^x_{L_j,D_j}, \tag{B.18}$$

$$\frac{\partial \tilde{K}^i_{jj}}{\partial \gamma_I} = \frac{1}{e^{\gamma_A}} \left(K^x_{L_j,D_j} \right)^T \Psi_j \left(\frac{e^{\gamma_I}}{e^{\gamma_A}} Q_j \right) \left[K^x_{D_j,D_j} \Psi_j \left(\frac{e^{\gamma_I}}{e^{\gamma_A}} Q_j \right) - I \right] \left(K^x_{L_j,D_j} \right). \tag{B.19}$$

B.2.2 SS-MTL SG

The marginal likelihood in the SS-MTL SG model is similar to the SS-MTL IND model by removing the summation over the tasks and substituting the covariance matrix $\tilde{K}^i_{jj}$ by $K^s = K^t \otimes \left(\frac{1}{\gamma_A} \tilde{K}^x \right)$. Note if we were exploiting the property of the Kronecker factorization which gives that $\alpha_a A_a \otimes \alpha_b A_b = \alpha_a \alpha_b (A_a \otimes A_b)$, it would simply change the appearance of the gradients.

The derivative of the log marginal likelihood for *regression* tasks with respect to all groups of hyperparameters will be given by,

$$\frac{\partial p(\mathbf{y}|A,\theta)}{\partial \theta} = \frac{1}{2} \left[\mathbf{y}^T \left(K^s + \sigma_n I \right)^{-1} \frac{\partial K^s}{\partial \theta} \left(K^s + \sigma_n I \right)^{-1} \mathbf{y}_j - \operatorname{tr} \left(\left(K^s + \sigma_n I \right)^{-1} \frac{\partial K^s}{\partial \theta} \right) \right]. \tag{B.20}$$

The derivative of the approximated log marginal likelihood for *classification* tasks will be computed by,

$$\frac{\partial p(\mathbf{y}|A,\theta)}{\partial\theta} = \frac{1}{2}\left[\tilde{\mu}^T\left(K^s+\tilde{\Sigma}_j\right)^{-1}\frac{\partial K^s}{\partial\theta}\left(K^s+\tilde{\Sigma}_j\right)^{-1}\tilde{\mu} - \operatorname{tr}\left(\left(K^s+\tilde{\Sigma}_j\right)^{-1}\frac{\partial K^s}{\partial\theta}\right)\right]. \tag{B.21}$$

The data-dependent covariance matrix $\tilde{K}^x$ for the SS-MTL SG model will be given by,

$$\frac{1}{e^{\gamma_A}}\tilde{K}^x = \frac{1}{e^{\gamma_A}}\left(K^x_{D,D} - \left(K^x_{L,D}\right)^T\left(I+\left(\frac{e^{\gamma_I}}{e^{\gamma_A}}Q_S\right)K^x_{D,D}\right)^{-1}\left(\frac{e^{\gamma_I}}{e^{\gamma_A}}Q_S\right)\left(K^x_{L,D}\right)\right). \tag{B.22}$$

The *gradients* of K^s with respect to the hyperparameters of the task covariance matrix θ^t will be given by,

$$\frac{\partial K^s}{\partial\theta^t} = \frac{\partial K^t}{\partial\theta^t}\otimes\frac{1}{e^{\gamma_A}}\tilde{K}^x. \tag{B.23}$$

The *gradients* of K^s with respect to all other hyperparameters except of θ^t , which we write as $\theta^{-t} = \{\theta^x,\theta^l,\theta^g\}$ are computed in a similar way to the SS-MTL IND model,

$$\frac{\partial K^s}{\partial\theta^{-t}} = K^t\otimes\frac{\partial\frac{1}{e^{\gamma_A}}\tilde{K}^x}{\partial\theta^{-t}}. \tag{B.24}$$

The computation the second term of the Kronecker factorization of the above equation $\left(\frac{\partial\frac{1}{e^{\gamma_A}}\tilde{K}^x}{\partial\theta^{-t}}\right)$ is performed in an identical manner to the one used for the computation of the gradients in the SS-MTL IND model in the previous section. This is achieved by replacing the data covariance matrices of each task with the the appropriate matrices constructed by all tasks, and similarly for the graph Laplacian. Then we will substitute for $K^x_{L_j,D_j}\rightarrow K^x_{L,D}$, and $K^x_{D_j,D_j}\rightarrow K^x_{D,D}$, and $Q_j\rightarrow Q_S$. As in the SS-MTL IND model we will define the following matrices $\Psi = \left(I+\frac{e^{\gamma_I}}{e^{\gamma_A}}Q_SK^x_{D,D}\right)^{-1}$, and $B = \Psi\frac{e^{\gamma_I}}{e^{\gamma_A}}Q$. For example, the gradients with respect to θ^x are computed as,

$$\frac{\partial\frac{1}{e^{\gamma_A}}\tilde{K}^x}{\partial\theta^x} = \frac{1}{e^{\gamma_A}}\left(\frac{\partial K^x_{L,L}}{\partial\theta^x} - 2\left(\frac{\partial K^x_{L,D}}{\partial\theta^x}\right)^T B\left(K^x_{L,D}\right) + \left(K^x_{L,D}\right)^T B\frac{\partial K^x_{D,D}}{\partial\theta^x}B\left(K^x_{L,D}\right)\right) \tag{B.25}$$

B.2.3 SS-MTL FG

The derivative of the log marginal likelihood for the SS-MTL FG model is obtained by replacing K^s with K^f in equations B.20 and B.21, for regression and classification tasks respectively. The covariance matrix for the SS-MTL FG model is computed from,

$$\begin{aligned} K^f &= \frac{1}{e^{\gamma_A}} K^t \otimes K^x_{L,L} \\ &- \frac{1}{e^{\gamma_A}} \left(K^t \otimes K^x_{L,D}\right)^T \left(I + (K^t \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_D)(K^t \otimes K^x_{D,D})\right)^{-1} \left(K^t \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_D\right) \left(K^t \otimes K^x_{L,D}\right). \end{aligned} \tag{B.26}$$

Similarly, to the two previous models we will set as,

$$\begin{aligned} \Psi_f &= \left(I + (K^t \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_D)(K^t \otimes K^x_{D,D})\right)^{-1}, \\ B_f &= \Psi_f \left(K^t \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_D\right) \end{aligned}$$

The *gradients* with respect to the hyperparameters of the task covariance matrix θ^t are computed from,

$$\begin{aligned} \frac{\partial K^f}{\partial \theta^t} &= \frac{1}{e^{\gamma_A}} \left(\frac{\partial K^t}{\partial \theta^t} \otimes K^x_{L,L}\right) - \frac{2}{e^{\gamma_A}} \left(\frac{\partial K^t}{\partial \theta^t} \otimes K^x_{L,D}\right)^T B_f \left(K^t \otimes K^x_{L,D}\right) \\ &+ \frac{1}{e^{\gamma_A}} \left(K^t \otimes K^x_{L,D}\right)^T \Psi_f \left[\left(\frac{\partial K^t}{\partial \theta^t} \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_D\right) \left(K^t \otimes K^x_{D,D}\right)\right] B_f \left(K^t \otimes K^x_{L,D}\right) \\ &+ \frac{1}{e^{\gamma_A}} \left(K^t \otimes K^x_{L,D}\right)^T \Psi_f \left[\left(K^t \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_D\right) \left(\frac{\partial K^t}{\partial \theta^t} \otimes K^x_{D,D}\right)\right] B_f \left(K^t \otimes K^x_{L,D}\right) \\ &- \frac{1}{e^{\gamma_A}} \left(K^t \otimes K^x_{L,D}\right)^T \Psi_f \left(\frac{\partial K^t}{\partial \theta^t} \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_D\right) \left(K^t \otimes K^x_{L,D}\right) \end{aligned} \tag{B.27}$$

The *gradients* with respect to the hyperparameters of the data covariance function θ^x are given by,

$$\begin{aligned} \frac{\partial K^f}{\partial \theta^x} &= \frac{1}{e^{\gamma_A}} \left(K^t \otimes \frac{\partial K^x_{L,L}}{\partial \theta^x}\right) - \frac{2}{e^{\gamma_A}} \left(K^t \otimes \frac{\partial K^x_{L,D}}{\partial \theta^x}\right) \\ &+ \frac{1}{e^{\gamma_A}} \left(K^t \otimes K^x_{L,D}\right)^T B_f \left(K^t \otimes \frac{\partial K^x_{D,D}}{\partial \theta^x}\right) B_f \left(K^t \otimes K^x_{L,D}\right) \end{aligned} \tag{B.28}$$

The *gradients* with respect to the hyperparameters of the graph Laplacian θ^l are computed as,

$$\frac{\partial K^f}{\partial \theta^l} = \frac{1}{e^{\gamma_A}} \left(K^t \otimes K^x_{L,D}\right)^T \Psi_f \left(K^t \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} \frac{\partial Q_d}{\partial \theta^l}\right) \left[\left(K^t \otimes K^x_{D,D}\right) B_f - I\right] \left(K^t \otimes K^x_{L,D}\right). \tag{B.29}$$

The *gradients* with respect to γ_A are computed from,

$$\frac{\partial K^f}{\partial \gamma_A} = -K^f - \frac{1}{e^{\gamma_A}} \left(K^t \otimes K^x_{L,D}\right)^T \Psi_f \left(K^t \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_d\right) \left[\left(K^t \otimes K^x_{D,D}\right) B_f - I\right] \left(K^t \otimes K^x_{L,D}\right). \tag{B.30}$$

The *gradients* with respect to γ_I are computed from,

$$\frac{\partial K^f}{\partial \gamma_I} = \frac{1}{e^{\gamma_A}} \left(K^t \otimes K^x_{L,D}\right)^T \Psi_f \left(K^t \otimes \frac{e^{\gamma_I}}{e^{\gamma_A}} Q_d\right) \left[\left(K^t \otimes K^x_{D,D}\right) B_f - I\right] \left(K^t \otimes K^x_{L,D}\right). \tag{B.31}$$

B.2.4 Additional experimental results

In this section, we present the accuracy the SS-MTL, MTL, SS-STL and STL methods achieved on the Spam, Sentiment and Letters classification problem. For ease of comparison between the performance measures, AUC and Accuracy, we report on the same the table the AUC and the Accuracy of each problem. The classes of the Spam and the Sentiment problems are balanced and hence the accuracy of the models follows the same pattern as for the AUC (see tables B.2 and B.3). In the letters classification problem, which is highly imbalanced, the AUC and the Accuracy of the models follow a different pattern, table B.4: in terms of AUC the best performance is achieved by MTL-IND, whereas in terms of Accuracy the best performance is achieved by FG in 3 out of 5 training sizes and by STL in 2 out of 5.

Table B.2: *AUC and Accuracy on the Spam data set*.

METHOD	*No. of training data points per task*					
	10	20	30	40	50	100
AUC						
SG-G	**87.81 ± 5.62**	**92.64 ± 2.80**	**94.04 ± 2.14**	94.99 ± 1.82	96.28 ± 1.00	96.60 ± 0.47
FG-G	83.60 ± 9.25	92.32 ± 3.18	93.30 ± 3.89	**95.18 ± 1.24**	94.44 ± 1.40	96.09 ± 0.65
SG-NN	85.42 ± 5.42	91.57 ± 3.36	93.24 ± 1.92	94.61 ± 1.51	**95.79 ± 0.82**	96.33 ± 1.23
FG-NN	85.56 ± 5.07	91.38 ± 3.09	93.65 ± 2.10	94.39 ± 1.66	95.77 ± 0.93	96.69 ± 0.74
SS-IND-G	81.47 ± 7.74	87.38 ± 3.33	90.71 ± 1.82	92.72 ± 1.17	93.03 ± 1.75	95.43 ± 0.54
MTL	82.21 ± 7.96	90.97 ± 2.13	93.39 ± 1.70	94.14 ± 1.45	95.34 ± 0.75	**96.94 ± 0.44**
MTL-IND	80.89 ± 10.49	86.63 ± 0.80	91.82 ±1.87	93.02 ± 1.57	94.63 ± 0.91	96.68 ± 0.55
SSGP-NN	77.01 ± 6.43	85.94 ± 4.53	90.23 ± 2.07	91.75 ± 1.87	93.81 ± 1.02	96.16 ± 0.66
SSGP-G	76.14 ± 6.36	84.96 ± 3.87	89.67 ± 2.17	91.75 ± 2.07	92.92 ± 1.63	95.02 ± 1.15
STL-GP	64.62 ± 6.17	71.45 ± 8.44	81.94 ± 6.17	85.13 ± 6.69	91.99 ± 4.80	96.36 ± 1.73
ACCURACY						
SG-G	**78.73 ± 3.95**	**84.57 ± 3.73**	**85.92 ± 5.10**	87.12 ± 4.35	**89.87 ± 1.53**	90.06 ± 0.96
FG-G	75.44 ± 8.39	84.45 ± 3.39	85.87 ± 4.45	**87.99 ± 1.89**	86.08 ± 1.77	88.35 ± 1.09
SG-NN	76.21 ± 4.14	83.00 ± 5.15	85.08 ± 3.11	87.20 ± 2.12	89.18 ± 1.60	**90.32 ± 1.67**
FG-NN	75.60 ± 3.90	82.35 ± 4.37	86.03 ± 2.68	86.91 ± 2.38	89.20 ± 1.44	90.48 ± 1.05
SS-IND-G	72.71 ± 6.92	77.90 ± 3.87	81.64 ± 2.44	83.71 ± 2.46	84.49 ± 2.73	87.43 ± 0.96
MTL-GP	69.47 ± 13.54	81.75 ± 2.49	84.97 ± 2.68	85.53 ± 2.40	87.80 ± 1.38	90.29 ± 1.05
MTL-IND	71.81 ± 9.72	75.79 ± 11.51	82.78 ± 3.12	83.87 ± 2.77	86.49 ± 1.24	89.87 ± 1.13
SSGP-NN	64.76 ± 5.97	75.61 ± 4.05	80.79 ± 3.20	82.43 ± 2.47	85.62 ± 1.26	89.63 ± 1.06
SSGP-G	65.78 ± 4.52	74.69 ± 4.24	80.24 ± 2.56	82.36 ± 2.68	84.04 ± 1.76	87.58 ± 1.53
STL-GP	51.24 ± 7.45	53.51 ± 8.16	66.21 ± 8.68	71.31 ± 8.16	81.58 ± 7.24	88.44 ± 2.68
NN	3	6	9	12	15	20

Table B.3: *AUC and Accuracy on the Amazon data set.*

METHOD	*No. of training data points per task* 10	20	30	40	50	100
AUC						
SG-G	70.45 ± 3.78	**73.61 ± 2.08**	76.00 ± 1.68	77.58 ± 1.59	78.84 ± 1.42	81.53 ± 1.21
FG-G	69.82 ± 3.86	72.99 ± 2.14	75.82 ± 1.79	77.32 ± 1.57	78.55 ± 1.52	81.32 ± 1.13
SG-NN	**70.55 ± 4.14**	73.55 ± 2.11	76.03 ± 1.95	**78.12 ± 1.55**	**79.04 ± 1.36**	**81.83 ± 1.10**
FG-NN	68.96 ± 4.68	73.08 ± 2.30	**76.19 ± 1.67**	76.99 ± 2.71	78.57 ± 2.79	81.77 ± 1.07
SS-IND-G	61.39 ± 4.35	66.55 ± 2.48	69.03 ± 2.07	71.03 ± 1.46	72.69 ± 1.18	75.19 ± 1.30
MTL-GP	61.56 ± 2.85	66.18 ± 3.93	72.13 ± 4.35	75.59 ± 3.68	77.50 ± 2.84	81.33 ±1.24
MTL-IND	61.60 ± 2.98	66.10 ± 3.30	69.44 ± 2.85	71.50 ± 2.51	73.55 ± 1.56	76.20 ± 1.09
SS-STL-NN	63.35 ± 2.29	66.63 ± 2.14	69.71 ± 2.00	71.77 ± 2.23	73.62 ± 1.55	76.34 ± 1.16
SS-STL-G	59.57 ± 2.87	62.84 ± 3.15	66.12 ± 2.68	69.69 ± 3.41	72.32 ± 2.50	75.59 ± 1.76
STL-GP	53.75 ± 1.85	55.13 ± 1.82	55.68 ± 2.27	56.31 ± 2.49	57.59 ± 2.60	57.30 ± 2.11
ACCURACY						
SG-G	57.43 ± 2.72	61.79 ± 1.58	64.49 ± 1.13	66.31 ± 1.19	67.69 ± 1.04	70.66 ± 1.08
FG-G	57.05 ± 3.05	60.88 ± 1.94	64.09 ± 1.18	65.92 ± 1.10	67.22 ± 1.13	70.28 ± 0.98
SG-NN	**58.73 ± 2.84**	**62.57 ± 1.95**	64.72 ± 2.54	**67.11 ± 1.31**	**68.34 ± 1.14**	**71.42 ± 1.01**
FG-NN	57.49 ± 3.40	62.28 ± 2.10	**65.22 ± 1.46**	65.37 ± 4.38	68.02 ± 2.29	71.40 ± 0.94
SS-IND-G	50.95 ± 3.59	52.00 ± 3.32	55.89 ± 5.05	59.39 ± 2.92	60.45 ± 2.02	63.29 ± 1.47
MTL-GP	50.82 ± 3.40	52.37 ± 2.10	58.65 ± 1.46	62.42 ± 4.38	63.98 ± 2.29	68.23 ± 0.94
MTL-IND	53.64 ± 3.79	58.78 ± 4.27	62.55 ± 3.31	64.65 ± 2.46	66.52 ± 1.20	68.95 ± 0.90
SS-STL-NN	56.83 ± 4.18	59.54 ± 3.74	62.61 ± 2.28	64.70 ± 2.52	66.27 ± 1.61	69.00 ± 0.93
SS-STL-G	50.37 ± 2.68	56.67 ± 4.63	59.71 ± 3.43	63.00 ± 4.03	65.60 ± 2.73	68.20 ± 2.21
STL-GP	50.01 ± 5.65	53.86 ± 3.35	56.62 ± 2.37	58.38 ± 2.86	60.77 ± 2.82	65.26 ± 1.16
NN	3	4	5	6	7	10

Table B.4: *AUC and Accuracy on the Letters data set.*

METHOD	NO. OF TRAINING DATA POINTS PER TASK 10	20	30	40	50
AUC					
SG-G	82.71 ± 6.12	88.47 ± 1.01	90.98 ± 0.72	91.46 ± 0.81	92.51 ± 0.70
FG-G	84.13 ± 2.10	89.30 ± 1.05	91.61 ± 0.89	92.52 ± 0.62	93.39 ± 0.48
SG-NN	78.04 ± 0.23	86.91 ± 2.22	89.57 ± 6.76	91.15 ± 2.57	92.36 ± 1.89
FG-NN	82.03 ± 4.77	88.28 ± 1.88	90.32 ± 1.35	91.71 ± 1.31	92.73 ± 0.70
SS-IND-G	84.32 ± 1.86	88.36 ± 1.28	90.85 ± 0.91	91.75 ± 0.59	92.75 ± 0.51
MTL-GP	81.33 ± 2.40	85.64 ± 1.89	88.76 ± 1.46	90.46 ± 1.05	91.69 ± 0.90
MTL-IND	**86.09 ± 1.92**	**90.15 ± 0.92**	**92.15 ± 0.77**	**92.90 ± 0.56**	93.70 ± 0.50
SS-STL-NN	80.26 ± 3.38	86.85 ± 3.27	91.07 ± 2.03	92.48 ± 1.04	92.89 ± 1.52
SS-STL-G	79.93 ± 2.71	87.12 ± 2.29	90.77 ± 1.25	91.99 ± 0.97	93.27 ± 0.74
STL-GP	83.85 ± 2.93	88.70 ± 2.77	91.89 ± 1.40	92.88 ± 0.75	**93.77 ± 0.54**
ACCURACY					
SG-G	71.13 ± 3.90	77.50 ± 1.45	80.71 ± 1.31	81.65 ± 1.42	83.96 ± 0.96
FG-G	**74.21 ± 2.87**	**80.68 ± 1.46**	83.55 ± 1.48	**85.08 ± 1.16**	86.37 ± 0.92
SG-NN	66.16 ± 16.59	77.40 ± 1.97	80.96 ± 2.04	83.08 ± 1.84	84.91 ± 1.22
FG-NN	71.82 ± 6.07	79.52 ± 2.29	81.90 ± 1.90	83.85 ± 1.88	85.27 ± 1.27
SS-IND-G	71.50 ± 2.46	77.48 ± 1.43	80.73 ± 1.36	82.34 ± 1.21	83.97 ± 0.87
MTL-GP	69.01 ± 3.15	75.62 ± 2.8	79.22 ± 2.34	81.75 ± 1.69	83.30 ± 1.61
MTL-IND	64.96 ± 6.29	79.60 ± 1.67	83.02 ± 1.73	84.70 ± 1.21	86.16 ± 1.04
SS-STL-NN	63.40 ± 5.81	76.87 ± 5.01	82.05 ± 2.21	83.34 ± 2.46	85.30 ± 1.94
SS-STL-G	66.53 ± 5.54	77.21 ± 3.23	81.47 ± 2.16	84.14 ± 1.55	85.25 ± 2.39
STL-GP	68.56 ± 4.56	78.36 ± 2.85	**82.90 ± 2.48**	85.14 ± 1.60	**86.76 ± 1.10**
NN	3	4	5	6	7

B.3 Useful Matrix identities

The matrix inversion lemma, which is also known as the Woodbury formula is given by (Rasmussen and Williams, 2005),

$$(Z+UWV^T)^{-1}=Z^{-1}-Z^{-1}U(W^{-1}+V^TZ^{-1}U)^{-1}V^TZ^{-1}. \tag{B.32}$$

The inverse of a block matrix is computed as,

$$A=\begin{pmatrix} \mathrm{P} & \mathrm{Q} \\ \mathrm{R} & \mathrm{S} \end{pmatrix}, \qquad A^{-1}=\begin{pmatrix} \tilde{P} & \tilde{Q} \\ \tilde{R} & \tilde{S} \end{pmatrix}, \tag{B.33}$$

where $\tilde{P}$, $\tilde{Q}$, $\tilde{R}$, $\tilde{S}$ are given either by,

$$\tilde{P}=P^{-1}+P^{-1}Q\tilde{S}RP^{-1} \tag{B.34}$$
$$\tilde{Q}=-P^{-1}Q\tilde{S} \tag{B.35}$$
$$\tilde{R}=-\tilde{S}RP^{-1} \tag{B.36}$$
$$\tilde{S}=\left(S-RP^{-1}Q\right)^{-1}, \tag{B.37}$$

or by,

$$\tilde{P}=\left(P-QS^{-1}R\right)^{-1} \tag{B.38}$$
$$\tilde{Q}=-\tilde{P}QS^{-1} \tag{B.39}$$
$$\tilde{R}=-S^{-1}R\tilde{P} \tag{B.40}$$
$$\tilde{S}=S^{-1}+S^{-1}R\tilde{P}QS^{-1}, \tag{B.41}$$

and the two forms are equivalent.

Appendix C

Appendix to Chapter 5

C.1 Approximate Inference for CMTMC model

The section provides the approximate posteriors of the random variables for the upper channel of the CMTMC model $\theta = \{\mathbf{G}, \mathbf{H}^t\}$. The approximate posteriors for the lower channel $\mathbf{f}$, $\mathbf{h}^x$ are the same used in the variational probit model in chapter 3 and can be found in appendix A.3.1, where $\mathbf{h} \equiv \mathbf{h}^x$. As in the previous appendix, we will write K^x without any subscripts to denote the covariance matrix between all training points from all tasks, unless stated otherwise. Note that, most of the computations for the multi-class classifier based on the Variational probit model can be found in Girolami and Rogers (2006); they are only reported here for completeness.

C.1.1 $Q(\mathbf{G})$

The approximate posterior for $Q(\mathbf{G})$ is computed as (Girolami and Rogers, 2006)

$$
\begin{aligned}
Q(\mathbf{G}) &\propto \exp\left\{\mathbb{E}_{Q(\mathbf{h}^t)}\left(\sum_{i=1}^{N}\sum_{j=1}^{M}\log p(h^t_{ij}|g_{ij}) + \log p(\mathbf{g}_j|\mathbf{X})\right)\right\} \\
&\propto \exp\left\{\mathbb{E}_{Q(\mathbf{h}^t)}\left(\sum_{j=1}^{M}\log \mathcal{N}_{\mathbf{h}^t_j}(\mathbf{g}_j, I) + \log \mathcal{N}_{\mathbf{g}_j}(0, K^x)\right)\right\} \\
&\propto \prod_{j=1}^{M}\mathcal{N}_{\tilde{\mathbf{h}}^t_j}(\mathbf{g}_j, I)\mathcal{N}_{\mathbf{g}_j}(0, K^x),
\end{aligned}
$$

which gives that

$$
Q(\mathbf{G}) = \prod_{j=1}^{M} Q(\mathbf{g}_j) = \prod_{j=1}^{M}\mathcal{N}_{\mathbf{g}_j}(\tilde{\mathbf{g}}_j, \Sigma^g), \tag{C.1}
$$

where $\Sigma^g = \left(I + K^{x-1}\right)^{-1} = K^x\left(I + K^x\right)^{-1}$, and $\tilde{\mathbf{g}}_j = \Sigma^g \tilde{\mathbf{h}}^t_j$.

C.1.2 $Q(\mathbf{H}^t)$

The approximate posterior of $Q(\mathbf{H}^t)$ can be computed in a similar manner.

$$\begin{aligned}
Q(\mathbf{H}^t) &\propto \exp\left\{\mathbb{E}_{Q(\mathbf{G})}\left(\sum_{i=1}^{N}\log p(y_n^t|\mathbf{h}_n^t)+\log p(\mathbf{h}_n^t|\mathbf{g}_n)\right)\right\}\\
&\propto \exp\left\{\mathbb{E}_{Q(\mathbf{G})}\left(\sum_{i=1}^{N}\log p(y_n^t|\mathbf{h}_n^t)+\log \mathcal{N}_{\mathbf{h}_n^t}(\tilde{\mathbf{g}}_n,I)\right)\right\}\\
&\propto \prod_{n=1}^{N}\mathcal{N}_{\mathbf{h}_n^t}(\tilde{\mathbf{g}}_n,I)\delta(h_{nj}^t>h_{nk}^t\forall k\neq i)\delta(y^t=i).
\end{aligned}\tag{C.2}$$

The concept behind this exposition is that the auxiliary variable h_{ni}^t is the largest if $y_n^t=i$, which can also be seen as that each $\mathbf{h}_n^t$ is distributed according to a truncated multivariate Gaussian that the i^{th} dimension is the largest, which gives that

$$Q(\mathbf{H}^t)=\prod_{n=1}^{N}Q(\mathbf{h}_n^t)=\prod_{n=1}^{N}\mathcal{N}_{\mathbf{h}_n^t}^{y_n^t}(\tilde{\mathbf{g}}_n,I_M),\tag{C.3}$$

where the identity matrix $I_M\in\mathbb{R}^{M\times M}$, and $\mathcal{N}_{\mathbf{h}_n^t}^{y_n^t}(\tilde{\mathbf{g}}_n,I_M)$ denotes a truncated Gaussian with the largest dimension indicated by y_n^t. Taking into account that each component of $Q(\mathbf{H}^t)$ must be normalized appropriately we will have that,

$$Q(\mathbf{h}_n^t)=\frac{1}{Z_n}\prod_{k=1}^{M}\mathcal{N}_{h_{nk}^t}(\tilde{g}_{nk},1),\tag{C.4}$$

where $Z_n=p(\mathbf{h}_n^t\in\mathcal{C})$ and $\mathcal{C}=\{\mathbf{h}_n^t:h_{nj}^t<h_{ni}^t,j\neq i\}$. The normalization constant Z_n will be given by,

$$\begin{aligned}
Z_n&=p(\mathbf{h}_n^t\in\mathcal{C})\\
&=\int_{-\infty}^{+\infty}\mathcal{N}_{h_{ni}^t}(\tilde{g}_{ni},1)\prod_{j\neq i}\int_{-\infty}^{h_{ni}^t}\mathcal{N}_{h_{nj}^t}(\tilde{g}_{nj},1)\mathrm{d}h_{nj}^t\mathrm{d}h_{ni}^t.
\end{aligned}\tag{C.5}$$

To continue we write the product of (M-1) integrals as ,

$$\prod_{j\neq i}\int_{-\infty}^{h_{ni}^t}\mathcal{N}_{h_{nj}^t}(\tilde{g}_{nj},1)\mathrm{d}h_{nj}^t\mathrm{d}h_{ni}^t=\prod_{j\neq i}\int_{-\infty}^{h_{ni}^t}(2\pi)^{-1}\exp\left\{-\frac{1}{2}(h_{nj}^t-\tilde{g}_{nj})^2\right\}\mathrm{d}h_{nj}^t,$$

where setting that $z_j=h_{nj}^t-\tilde{g}_{nj}$, we will have that

$$\begin{aligned}
&h_{nj}^t=-\infty\rightarrow z_j=-\infty,\\
&h_{nj}^t=h_{ni}^t\rightarrow z_j=h_{ni}^t-\tilde{g}_{nj},\ \text{ and}\\
&\mathrm{d}z_j=\mathrm{d}h_{nj}^t,
\end{aligned}$$

which gives that

$$\prod_{j\neq i}\int_{-\infty}^{h_{ni}^t}\mathcal{N}_{h_{nj}^t}(\tilde{g}_{nj},1)\mathrm{d}h_{nj}^t\mathrm{d}h_{ni}^t=\prod_{j\neq i}\int_{-\infty}^{h_{ni}^t-\tilde{g}_{nj}}(2\pi)^{-1}\exp\left\{-\frac{1}{2}z_j^2\right\}\mathrm{d}z_j,$$
$$=\prod_{j\neq i}\Phi(h_{ni}^t-\tilde{g}_{nj}),$$

where Φ is the cumulative distribution. Then,

$$Z_n=\int_{-\infty}^{+\infty}\mathcal{N}_{h_{ni}^t}(\tilde{g}_{ni},1)\prod_{j\neq i}\Phi(h_{ni}^t-\tilde{g}_{nj}),$$
$$=\int_{-\infty}^{+\infty}(2\pi)^{-1}\exp\left\{-\frac{1}{2}(h_{ni}^t-\tilde{g}_{ni})^2\right\}\prod_{j\neq i}\Phi(h_{ni}^t-\tilde{g}_{nj})\mathrm{d}h_{ni}^t,$$

where setting again $u=h_{ni}^t-g_{ni}$ results in $h_{ni}^t=u+\tilde{g}_{ni}$, which in turn gives that,

$$Z_n=\int_{-\infty}^{+\infty}(2\pi)^{-1}\exp\left\{-\frac{1}{2}u^2\right\}\prod_{j\neq i}\Phi(u+\tilde{g}_{ni}-\tilde{g}_{nj})\mathrm{d}u,$$

which can be seen as the expectation of $\Phi\{.\}$ over $p(u)\sim\mathcal{N}(0,1)$, which finally gives that

$$Z_n=\mathbb{E}_{p(u)}\left\{\prod_{j\neq i}\Phi(u+\tilde{g}_{ni}-\tilde{g}_{nj})\right\}. \tag{C.6}$$

Then we have that the posterior expectation of $\tilde{h}_{nk}^t$ for all $k\neq i$ will be computed from,

$$\tilde{h}_{nk}^t=Z_n^{-1}\int_{-\infty}^{+\infty}h_{nk}^t\prod_{j=1}^{M}\mathcal{N}_{h_{nj}^t}(\tilde{g}_{nj},1)\mathrm{d}h_{nj}^t,$$
$$=Z_n^{-1}\int_{-\infty}^{+\infty}\int_{-\infty}^{h_{ni}^t}h_{nk}^t\mathcal{N}_{h_{nk}^t}(\tilde{g}_{nk},1)\prod_{j\neq i,k}\mathcal{N}_{h_{ni}^t}(\tilde{g}_{ni},1)\Phi(h_{ni}^t-\tilde{g}_{nj})\mathrm{d}h_{ni}^t\mathrm{d}h_{nk}^t$$
$$=\tilde{g}_{nk}-Z_n^{-1}\mathbb{E}_{p(u)}\left\{\mathcal{N}_u(\tilde{g}_{nk}-\tilde{g}_{ni},1)\prod_{j\neq i}\Phi(u+\tilde{g}_{ni}-\tilde{g}_{nj})\right\}. \tag{C.7}$$

While the posterior expectation for the i^{th} component will be given by,

$$\tilde{h}_{ni}^t=Z_n^{-1}\int_{-\infty}^{+\infty}h_{ni}^t\mathcal{N}_{h_{ni}^t}(\tilde{g}_{ni},1)\prod_{j\neq i}\Phi(h_{ni}^t-\tilde{g}_{nj})\mathrm{d}h_{ni}^t,$$
$$=\tilde{g}_{ni}-Z_n^{-1}\mathbb{E}_{p(u)}\left\{u\prod_{j\neq i}\Phi(u+\tilde{g}_{ni}-\tilde{g}_{nj})\right\},$$
$$=\tilde{g}_{ni}-Z_n^{-1}\sum_{k\neq i}\mathbb{E}_{p(u)}\left\{\mathcal{N}_u(\tilde{g}_{nk}-\tilde{g}_{ni},1)\prod_{j\neq i}\Phi(u+\tilde{g}_{ni}-\tilde{g}_{nj})\right\},$$
$$=\tilde{g}_{ni}\sum_{k\neq i}(\tilde{g}_{nk}-h_{nk}^t), \tag{C.8}$$

where it has been used that for a random variable $u\sim\mathcal{N}(0,1)$ and for any differentiable function $\zeta(u)$, we have that $\mathbb{E}[u\zeta(u)]=\mathbb{E}[\zeta'(u)]$.

C.2 Lower Bound of the CMTMC model

C.2.1 Lower bound on log marginal likelihodd

The lower bound on the log marginal likelihood is computed by,

$$\mathcal{L}(Q) = \mathbb{E}_{Q(\Theta)}[\log p(\mathbf{y}^t, \mathbf{y}^x, \mathbf{G}, \mathbf{H}^t, \mathbf{f}, \mathbf{h}^x | X, \theta^t, \theta^x)] - \mathbb{E}_{Q(\Theta)}[\log Q(\mathbf{G})Q(\mathbf{H}^t)Q(\mathbf{f})Q(\mathbf{h}^x)] \tag{C.9}$$

where taking into account the independence assumption of the two channels given the hyperparameters θ^x, is decomposed into two set of quantities. The first is due to the multi-class channel and we will refer to it as $\mathcal{L}_t(Q)$ and the second is due to the multi-task channel which we write as $\mathcal{L}_t(Q)$. The quantities that depend on the multi-task classifier is given in appendix A.3.2. Here, we expand the terms that depend on the multi-class classifier. The terms of the lower bound that depend on the multi-class classifier can be found in Girolami and Rogers (2006) and are presented here for completeness,

$$\begin{aligned}\mathcal{L}_t(Q) = &\sum_{j=1}^{M}\sum_{n=1}^{N} \mathbb{E}_{Q(\mathbf{G})Q(\mathbf{H}^t)}\left\{\log p(h^t_{nj}|g_{nj})\right\} + \sum_{j=1}^{M} \mathbb{E}_{Q(\mathbf{G})}\left\{\log p(\mathbf{g}_j|\mathbf{X})\right\} \\ &- \sum_{j=1}^{M} \mathbb{E}_{Q(\mathbf{g}_j)}\left\{\log Q(\mathbf{g}_j)\right\} - \sum_{n=1}^{N} \mathbb{E}_{Q(\mathbf{h}^t_n)}\left\{\log p(\mathbf{h}^t_n)\right\},\end{aligned} \tag{C.10}$$

where following a similar derivation to the lower bound of the multi-task classifier in appendix A.3.2 gives that,

$$\begin{aligned}\mathcal{L}_t(Q) = &-\frac{NM}{2}\log(2\pi) + \frac{N}{2}\log(2\pi) + \frac{NM}{2} - \frac{M}{2}\text{trace}(\Sigma^g) - \frac{1}{2}\sum_m \tilde{\mathbf{g}}_m^T (K^x)^{-1}\tilde{\mathbf{g}}_m \\ &- \frac{M}{2}\text{trace}\left(K^{x^{-1}}\Sigma^g\right)\frac{M}{2}\log|K^x| + \frac{M}{2}\log|\Sigma^g| + \sum_n \log Z^t_n.\end{aligned} \tag{C.11}$$

Putting together equations A.32 and C.11, gives the lower bound on the CMTMC model as,

$$\begin{aligned}\mathcal{L}(Q) = &-\frac{NM}{2}\log(2\pi) + \frac{N}{2}\log(2\pi) + \frac{NM}{2} - \frac{M}{2}\text{trace}(\Sigma^g) - \frac{1}{2}\sum_m \tilde{\mathbf{g}}_m^T K^{x^{-1}}\tilde{\mathbf{g}}_m \\ &- \frac{M}{2}\text{trace}\left(K^{x^{-1}}\Sigma^g\right) - \frac{M}{2}\log|K^x| + \frac{M}{2}\log|\Sigma^g| + \sum_n \log z^t_n \\ &+ \sum_{n=1}^{N}\log z^x_n - \frac{1}{2}\log|I + K^t \otimes K^x| - \frac{1}{2}\tilde{\mathbf{f}}^T(K^t \otimes K^x)^{-1}\tilde{\mathbf{f}},\end{aligned} \tag{C.12}$$

where $Z^t_n = \mathbb{E}_{p(u)}\left\{\prod_{j \neq i}\Phi(u + \tilde{g}_{ni} - \tilde{g}_{nj}\right\})$ is given in equation C.6, and $Z^x_n = \Phi(y^x_n \tilde{f}_n)$ is given in equations A.20 and A.22.

Terms that depend on hyperparameters θ^x and θ^t are:

$$\begin{aligned}\mathcal{L}(Q)_{\theta^x,\theta^t} = & -\frac{M}{2}\text{trace}(\Sigma^g) - \frac{1}{2}\sum_m \tilde{\mathbf{g}}_m^T K^{x^{-1}} \tilde{\mathbf{g}}_m - \frac{M}{2}\text{trace}\left(K^{x^{-1}}\Sigma^g\right) \\ & -\frac{M}{2}\log|K^x| + \frac{M}{2}\log|\Sigma^g| - \frac{1}{2}\log|I + K^t \otimes K^x| - \frac{1}{2}\tilde{\mathbf{f}}^T (K^t \otimes K^x)^{-1}\tilde{\mathbf{f}} \quad \text{(C.13)}\end{aligned}$$

C.2.2 Gradients on lower bound

The gradients with respect to the parameters of the data covariance function K^x are computed from:

$$\begin{aligned}\frac{\partial}{\partial\theta^x}\mathcal{L}(q) = & -\frac{M}{2}\text{trace}\left\{\Omega(I+K^x)^{-1} - K^x(I+K^x)^{-1}\Omega(I+K^x)^{-1}\right\} + \frac{1}{2}\tilde{\mathbf{g}}_m^T K^{x^{-1}}\Omega K^{x^{-1}}\tilde{\mathbf{g}}_m \\ & + \frac{M}{2}\text{trace}\left\{(I+K^x)^{-1}\Omega(I+K^x)^{-1}\right\} - \frac{M}{2}\text{trace}\left\{K^{x^{-1}}\Omega\right\} \\ & + \frac{M}{2}\text{trace}\left\{(I+K^{x^{-1}})^{-1}K^{x^{-1}}\Omega K^{x^{-1}}\right\} + \frac{1}{2}\tilde{\mathbf{f}}^T\left(K^t\otimes K^x\right)^{-1}K^t\otimes\Omega\left(K^t\otimes K^x\right)^{-1}\tilde{\mathbf{f}} \\ & - \frac{1}{2}\text{trace}\left(\left(I+K^t\otimes K^x\right)^{-1}K^t\otimes\Omega\right). \quad \text{(C.14)}\end{aligned}$$

While the gradients with respect to the parameters of the task covariance matrix are computed from:

$$\frac{\partial}{\partial\theta^t}\mathcal{L}(q) = \frac{1}{2}\tilde{\mathbf{f}}^T\left(K^t\otimes K^x\right)^{-1}\Xi\otimes K^x\left(K^t\otimes K^x\right)^{-1}\tilde{\mathbf{f}} - \frac{1}{2}\text{trace}\left(\left(I+K^t\otimes K^x\right)^{-1}\Xi\otimes K^x\right), \quad \text{(C.15)}$$

where $\Omega = \frac{\partial K^x}{\partial\theta^x}$, and $\Xi = \frac{\partial K^t}{\partial\theta^t}$

C.3 Predictions with the CMTMC model

This section presents the computations needed to infer the posterior probability of a point coming from a target task $p(y_*^f = +1|x^t, \mathbf{X}, \mathbf{y}^t, \mathbf{y}^x)$. This quantity involves the computation of two separate terms, the posterior probability of the multi-task classifier $p(y_{*j}^x = +1|x^t, \mathbf{X}, \mathbf{y}^x)$, and the posterior probability of the multi-class classifier $p(y_{*j}^t|x^t, \mathbf{X}, \mathbf{y}^t)$. Computations that are needed to infer the posterior of the multi-task classifier with the Variational probit model are the same used in chapter 3 and we refer to appendix A.3.3 for details. In the following of this section we first present how to make predictions with the multi-class classifier for a single point (P2P) and secondly we discuss the problems encountered when dealing with a batch of test data points (Batch).

C.3.1 Point to Point (P2P) predictions

Inferring the task label y^t_* for a single test point x^t, using the multinomial probit model, involves the computation of several integrals given below:

$$p(y^t_* = k|x^t, \mathbf{X}, \mathbf{y}^t) = \int p(y^t_* = k|\mathbf{h}^t_*)p(\mathbf{h}^t_*|x^t, \mathbf{X}, \mathbf{y}^t)\mathrm{d}\mathbf{h}^t_* \tag{C.16}$$

$$p(\mathbf{h}^t_*|x^t, \mathbf{X}, \mathbf{y}^t) = \int p(\mathbf{h}^t_*|\mathbf{g}_*)p(\mathbf{g}_*|x^t, \mathbf{X}, \mathbf{y}^t)\mathrm{d}\mathbf{g}_*, \tag{C.17}$$

$$\begin{aligned} p(\mathbf{g}_*|x^t, \mathbf{X}, \mathbf{y}^t) &= \int p(\mathbf{g}_*|\mathbf{g})Q(\mathbf{G})\mathrm{d}\mathbf{G} \\ &= \prod_{m=1}^{M} \int p(g_{*m}|\mathbf{g}_m)Q(\mathbf{g}_m)\mathrm{d}\mathbf{g}_m. \end{aligned} \tag{C.18}$$

Where from standard GP results we have that,

$$p(g_{*m}|\mathbf{g}_m) \sim \mathcal{N}_{\mathbf{g}_m}\left((K^x_{\mathbf{X},x^t})^T (K^x_{\mathbf{x},\mathbf{x}})^{-1}\mathbf{g}_m, k^x_{x^t,x^t} - (\mathbf{k}^x_{\mathbf{X},x^t})^T (K^x_{\mathbf{X},\mathbf{X}})^{-1}(\mathbf{k}^x_{\mathbf{x},x^*})\right) \tag{C.19}$$

Then the mean and the variance of $p(\mathbf{g}_{*m}|x^t, \mathbf{X}, \mathbf{y}^t) = \mathcal{N}(\mu^g_{*m}, \sigma^g_{*m})$ are given by:

$$\mu^g_{*m} = (\mathbf{k}^x_{\mathbf{X},x^*})^T \left(I + K^x\right)^{-1} \tilde{\mathbf{h}}^t_m, \tag{C.20}$$

$$\sigma^g_{*m} = k^x_{x^t,x^t} - (\mathbf{k}^x_{\mathbf{X},x^t})^T \left(I + K^x\right)^{-1} (\mathbf{k}^x_{\mathbf{X},x^t}). \tag{C.21}$$

Then equation C.18 is given by:

$$p(\mathbf{g}_*|x^t, \mathbf{X}, \mathbf{y}) = \prod_{m=1}^{M} \mathcal{N}(\mu^g_{*m}, \sigma^g_*). \tag{C.22}$$

Returning to equation C.17, we have that:

$$p(\mathbf{h}^t_*|x^t, \mathbf{X}, \mathbf{y}^t) = \prod_{m=1}^{M} \int \mathcal{N}_{h^t_{*m}}(g_{*m}, 1)\mathcal{N}_{g_{*m}}(\mu^g_{*m}, \sigma^g_{*m})\mathrm{d}\mathbf{g}_{*m}. \tag{C.23}$$

Marginalizing out g_{*m} yields that $p(h^t_{*m}|x^t, \mathbf{X}, \mathbf{y}^t) = \mathcal{N}(\mu^g_{*m}, \upsilon^2_{*m})$ where:

$$\upsilon^2_{*m} = 1 + \sigma^g_{*m} \tag{C.24}$$

Returning to equation C.16 , and using the shorthand $p(y^t_* = k|h^{t*}) = \delta(h^t_{*k} > h^t_{*i}\ \forall\ i \neq k)\delta(y^t_* = k) \equiv \delta_{*k}$ we have that:

$$\begin{aligned} p(y^t_* = k|x^t, \mathbf{X}, \mathbf{y}^t) &= \int p(y^t_* = k|\mathbf{h}^t_*)p(\mathbf{h}^t_*|x^t, \mathbf{X}, \mathbf{y}^t)\mathrm{d}\mathbf{h}^t_* \\ &= \int \delta_{*k} \prod_{m=1}^{M} \mathcal{N}_{h^t_{*m}}(\mu^g_{*m}, \upsilon^2_{*m})\mathrm{d}h^t_{*m} \qquad \text{(C.25)} \\ &= \int_{-\infty}^{+\infty} \mathcal{N}_{h^t_{*k}}(\mu^g_{*k}, \upsilon^2_{*k}) \prod_{m \neq k} \int_{-\infty}^{h^t_{*k}} \mathcal{N}_{h^t_{*m}}(\mu^g_{*m}, \upsilon^2_{*m})\ \mathrm{d}h^t_{*m}\ \mathrm{d}h^t_{*k}, \qquad \text{(C.26)} \end{aligned}$$

which integral can efficiently be evaluated using numerical integration as:

$$p(y_*^t = k|x^t, \mathbf{X}, \mathbf{y}^t) = \mathbb{E}_{p(u)} \left\{ \prod_{j \neq k} \Phi\left(\frac{1}{\upsilon_{*j}} \left[u\upsilon_{*k} + \mu_{*k}^g - \mu_{*j}^g \right] \right) \right\}, \tag{C.27}$$

where $u \sim \mathcal{N}_u(0,1)$.

C.3.2 Batch predictions

The task predictive distribution for a batch of test points $p(y_*^t|X^t, \mathbf{X}, \mathbf{y}^t)$ involves the computation of the following integrals,

$$\begin{aligned} p(\mathbf{y}_*^t = k|X^t, \mathbf{X}, \mathbf{y}^t) &= \int p(\mathbf{y}_*^t = k|\mathbf{h}^{t*}) p(\mathbf{h}_*^t|X^t, \mathbf{X}, \mathbf{y}^t) \mathrm{d}\mathbf{h}_*^t && \text{(C.28)} \\ &= \int \delta_k^* \prod_{m=1}^{M} \mathcal{N}_{\mathbf{h}_m^{t*}}(M_m^{g*}, \Upsilon^*) \mathrm{d}\mathbf{h}_m^{t*} && \text{(C.29)} \\ &= \int_{-\infty}^{+\infty} \mathcal{N}_{\mathbf{h}_k^{t*}}(M_k^{g*}, \Upsilon^*) \prod_{m \neq k} \int_{-\infty}^{\mathbf{h}_k^{t*}} \mathcal{N}_{\mathbf{h}_m^{*}}(M_m^{g*}, \Upsilon^*) \, \mathrm{d}\mathbf{h}_m^{t*} \, \mathrm{d}\mathbf{h}_k^{t*}. && \text{(C.30)} \end{aligned}$$

The second part of equation C.30, $\int_{-\infty}^{\mathbf{h}_k^{t*}} \mathcal{N}_{\mathbf{h}_m^{t*}}(M_m^{g*}, \Upsilon^*) \, \mathrm{d}\mathbf{h}_m^{t*}$, is the multivariate cumulative distribution function which is decomposed as:

$$\begin{aligned} \int_{-\infty}^{\mathbf{h}_k^{t*}} \mathcal{N}_{\mathbf{h}_m^{t*}}(M_m^{g*}, \Upsilon^*) \, \mathrm{d}\mathbf{h}_m^{t*} &= \frac{1}{(2\pi)^{\frac{n^*}{2}}} \frac{1}{|\Upsilon^*|^{\frac{1}{2}}} \\ &\int_{-\infty}^{h_{1k}^{t*}} \cdots \int_{-\infty}^{h_{n^*k}^{t*}} \exp\left\{ -\frac{1}{2} \left(\mathbf{h}_m^{t*} - M_m^{g*} \right)^T \Upsilon^{*^{-1}} \left(\mathbf{h}_m^{t*} - M_m^{g*} \right) \right\} \, \mathrm{d}\mathbf{h}_m^{t*}, \end{aligned} \tag{C.31}$$

where the last equation implies that $\mathbf{h}_{im}^{t*} - M_{im}^{g*} \leq \mathbf{h}_{ik}^{t*} \ \ \forall \ i$. As already discussed in chapter 5 computing multivariate cumulative distributions is an extremely hard task, which compelled us to use the approximation presented in equation 5.24.

Appendix D

Automatic Classification of Arrhythmic Beats using Gaussian Processes

Automatic Classification of Arrhythmic Beats Using Gaussian Processes

G Skolidis, RH Clayton, G Sanguinetti

Department of Computer Science, University of Sheffield, Sheffield, United Kingdom

Abstract

We propose a novel approach to the automated discrimination of normal and ventricular arrhythmic beats. The method employs Gaussian Processes, a non-parametric Bayesian technique which is equivalent to a neural network with infinite hidden nodes. The method is shown to perform competitively with other approaches on the MIT-BIH Arrhythmia Database. Furthermore, its probabilistic nature allows to obtain confidence levels on the predictions, which can be very useful to practitioners.

1. Introduction

Cardiac arrhythmias are one of the major causes of morbidity and mortality in the Western world. Their early diagnosis is often reliant on an analysis of electrocardiogram (ECG) traces, generally involving time-consuming manual annotation by expert physicians. Because of this, several automated methods to detect arrhythmic beats have been proposed, often achieving very good levels of performance [1–3].

We present a novel approach for the automatic classification of arrhythmic versus normal beats from ECG signals based on recent developments in Machine Learning. We use the framework of Gaussian Process (GP) classification [4], a non-parametric Bayesian technique which has been shown to be highly accurate on non-linear classification tasks while controlling complexity and avoiding the pitfalls of overfitting. GPs are a natural way to define probability distributions over spaces of functions; they can be viewed as a generalization of Neural Networks where the number of hidden nodes (basis functions) tends to infinity [5]. A key feature of GPs is their probabilistic nature, which means that predictions are always accompanied by an estimate of the associated uncertainty. This is a key advantage over standard non-linear classifiers such as neural networks which generally can only provide a hard assignment.

The method uses as input the spectral or wavelet transform of segmented individual beats from a recording, which requires much less manual annotation than methods based on interval estimation. We use an Automatic Relevance Determination (ARD) kernel for the classifier to automatically reduce dimensionality and extract the most discriminant features by optimising weights.

We test the model on the MIT-BIH arrhythmia data set on the two class problem of discriminating normal and premature ventricular contraction beats (PVC). The results we report show that the method is competitive with the state of the art, obtaining predictive accuracies on test data which are frequently above 90%. This can be further increased by thresholding over posterior probabilities and retaining only predictions with high confidence; the model consistently has a higher accuracy for prediction made with higher posterior probability, indicating that the discriminant obtained from the training data mirrors the structure of the whole data set.

The rest of the paper is organised as follows: in the first section, we briefly review Gaussian Process classification. In the second section, we discuss the beat segmentation algorithm and the feature selection procedure. We then present our results on real ECG data, and conclude the paper with a discussion of the strengths and weaknesses of the method, as well as the possible future extensions.

2. Methods

2.1. Gaussian Processes for classification

In this section we briefly review the statistical foundations of our approach; for a thorough review, the reader is referred to [4]. A Gaussian Process (GP) is a (finite or infinite) collection of random variables any finite subset of which is distributed according to a multivariate normal distribution. As a random function $f(\mathbf{x})$ can be seen as a collection of random variables indexed by its input argument, GPs are a natural way of describing probability distributions over function spaces. A GP is characterised by its *mean function* $\mu(\mathbf{x})$ and *covariance function* $k(\mathbf{x}, \mathbf{x}')$, a symmetric function of two variables which has to satisfy the Mercer conditions ([4]). In formulae, the definition of GP can be written as

$$\mathbf{f} \sim \mathcal{GP}(\mu, \mathcal{K}) \Leftrightarrow [\mathbf{f}(\mathbf{x_1}), \dots, \mathbf{f}(\mathbf{x_N})] \sim \mathcal{N}([\mu(\mathbf{x_1}), \dots, \mu(\mathbf{x_N})], \mathcal{K}) \quad (1)$$

for any finite set of inputs $\mathbf{x_1}, \dots, \mathbf{x_N}$. Here

$$K_{ij} = k(\mathbf{x_i}, \mathbf{x_j}).$$

The choice of mean and covariance functions is largely determined by the problem under consideration. In this paper, we will use a zero mean GP with ARD covariance function, in order to automatically select the most relevant features from a high dimensional input space [6]

$$k(\mathbf{x_i}, \mathbf{x_j}) = exp\left(-\frac{1}{2}(\mathbf{x_i} - \mathbf{x_j})^T \Lambda (\mathbf{x_i} - \mathbf{x_j})\right), \quad (2)$$

where Λ is diagonal, with Λ_{ii} denoting the precision (inverse characteristic length-scale) of each feature of the input matrix.

Given some observations $\mathbf{y}$ of the function $\mathbf{f}$ at certain input values X, and given a noise model $p(\mathbf{y}|\mathbf{f}, \mathbf{X})$, one can use Bayes' theorem to obtain a posterior over the function values at the inputs

$$p(\mathbf{f}|\mathbf{y}, \mathbf{X}, \theta) = \frac{p(\mathbf{y}|\mathbf{f}, \mathbf{X}, \theta) p(\mathbf{f}|\mathbf{X}, \theta)}{p(\mathbf{y}|\mathbf{X}, \theta)} \quad (3)$$

where θ denotes the parameters of the GP prior (ARD parameters). One can then obtain a predictive distribution for the function value f^* at a new input point $\mathbf{x}^*$ by averaging the conditional distribution of $p(f^*|\mathbf{f})$ under the posterior (3)

$$p(f^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*, \theta) = \int p(\mathbf{f}^*|\mathbf{f}, \mathbf{X}, \mathbf{x}^*, \theta) p(\mathbf{f}|\mathbf{y}, \mathbf{X}, \theta) d\mathbf{f}.$$

If the noise model $p(\mathbf{y}|\mathbf{f})$ is Gaussian, then we are dealing with a regression problem and one can obtain an analytical expression for the posterior (3). In classification, the noise model is non-Gaussian; in this paper, we will take the noise model to be the logistic function

$$p(y = 1|f) = \frac{1}{1 + \exp(-f)}.$$

In this case, the denominator of equation (3) cannot be computed analytically and one must seek approximate solutions. In this paper, we use the Laplace approximation [7]. This computes a second order Taylor expansion of the un-normalised posterior $p(\mathbf{y}|\mathbf{f}, \mathbf{X}, \theta) p(\mathbf{f}|\mathbf{X}, \theta)$ about its mode and then approximates the true posterior distribution with a Gaussian centered at the true mode and with covariance given by the Hessian of the un-normalised posterior.

2.2. Experimental setup

In this study experimental data were taken from the MIT-BIH Arrhythmia database [8], for training and evaluation purposes of the proposed classifier. Specific recordings were selected according to the exhibited type of arrhythmia; each recording was sampled at $360Hz$, and had sufficient amount of Normal and premature ventricular contraction (PVC) beats, for training and evaluating the model. Annotation provided by the database was used to separate the beats before any preprocessing.

2.3. Data processing

Two different types of transforms were considered in the analysis of the ECG signal. The first one is based on the Fourier Transform while the second one on the Wavelet Transform.

2.3.1. Fast Fourier Transform (FFT)

Each beat segment, consisting of 360 data points (one minute), was transformed into the frequency domain using a Fast Fourier Transform with a Hanning window. The frequency based representation of each beat consisted of 180 frequencies, since it was sampled at $360Hz$.

2.3.2. Wavelet Transform (WT)

The second type of features were obtained by the Discrete Wavelet Transform. The Continuous Wavelet Transform (CWT) of a signal $x(t)$ is defined as:

$$W_a x(b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} x(t) \psi\left(\frac{t-b}{a}\right) \mathrm{dt}, \ \mathrm{a} > 0. \quad (4)$$

The discrete wavelet transform uses a dyadic scale factor $a = 2^k$ for $k \in \mathbf{Z}^+$. The wavelet that was used in this work was from the Daubechie family [9]. It is noted that the high frequency phenomena of a signal are captured at the smallest scales, namely 2^2 and 2^1,while most of the details of the signal are contained from the third to the fifth scale. Consequently, coefficients from these three scales were selected as input features for the classification, resulting in 900 features.

3. Results

3.1. Feature selection

To determine which frequencies are more relevant for classification, for the features acquired from FFT, ARD was used. Using ARD the five most relevant frequencies were identified and then the characteristic length scale of

Table 1. Classifier Performance in terms of Accuracy (%)(**HC**:High Confidence)

	FFT			WT		
Recording	**Accuracy Thresh. 0.5 (%)**	**Accuracy Thresh. 0.8 (%)**	**Data of HC(%)**	**Accuracy Thresh. 0.5 (%)**	**Accuracy Thresh. 0.8 (%)**	**Data of HC(%)**
106	93.2	95.8	89.23	98.61	99.53	95.58
119	100	100	100	99.84	99.89	99.49
200	98.32	99.08	97.61	97.61	98.53	95.97
203	87.9	93.37	80.79	96.9	98.53	95.55
221	96.16	96.7	97.18	96.16	96.92	95.69
223	88.43	96.8	71.43	90.67	96.54	81.59
228	99.8	99.8	99.8	99.11	99.65	98.43
233	97.96	98.78	96.98	96.19	99.35	90.94

each input feature was optimized for each recording. Thorough experimental research indicate that using two features (frequencies) achieves better performance, instead of using a larger number of features.

The features obtained from the Wavelet transform, were projected into a two-dimensional space, using Principal Component Analysis (PCA). After PCA, the characteristic length scale of each feature was estimated again using ARD.

3.2. Performance evaluation

The performance measure that was used for the evaluation of the classifier is simple misclassification error. In the Gaussian Process framework, the misclassification error is computed by setting a threshold over the posterior probabilities, since GPs produce a measure of uncertainty instead of giving hard assignments to a class. This concept can be further cultivated, by setting a threshold of 0.8 and retaining test samples that have been assigned with posterior probabilities higher than 0.8. In this way, a measure of high confidence is obtained to evaluate the classifier.

For example figure 1 and 2 show the decision boundaries along with the data points in the Euclidean space,that were created by GPs of the wavelet transformed input data of recording 223. It is clearly observable, that high confidence regions are produced in the input space where the density, of the training data of each class, is high.

Table 1 shows the performance of the classifier, with optimized hyperparameters, in terms of the accuracy the test data set achieves. The first column of each transform indicates the accuracy the classifier achieves with a threshold of 0.5. The other two columns show the accuracy of the classifier when a threshold of 0.8 is used and the proportion of data that have been assigned with probabilities higher than 0.8.

Each recording was trained and tested four times. The beats in each recording was separated into four disjoint subsets, preserving the initial prior probabilities of each class. In each run three data subsets were used for training

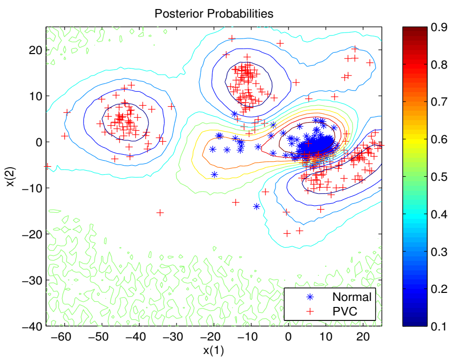


Figure 1. Decision Boundaries of recording 233(contour plot) - Scatter plot; where x(1), x(2) represent the features obtained from the wavelet transform after PCA

and one for testing. The accuracy each recording achieves in table 1 is the mean accuracy of the four trainings.

Figure 3 illustrates the effect the increase of the threshold has on the accuracy (solid line), and the proportion of the test data set, that has been assigned probabilities higher than the threshold (dashed line). It is noticed that the accuracy remains high but the proportion of the data that have been assigned with probabilities higher than the value of the threshold, decreases as the threshold increases.

4. Discussion and conclusions

In this paper we propose the use of Gaussian Processes for automatically classifying ECG signals into normal and ventricular beats. Furthermore, Automatic Relevance Determination was applied for the identification of the frequencies that were most relevant for classification, and then for the optimisation of the hyperparameters of the covariance function. A different methodology was followed for the features extracted by the wavelet transform of the raw ECG signal, where first PCA was applied to reduce the dimensionality of the inputs, and then ARD was used

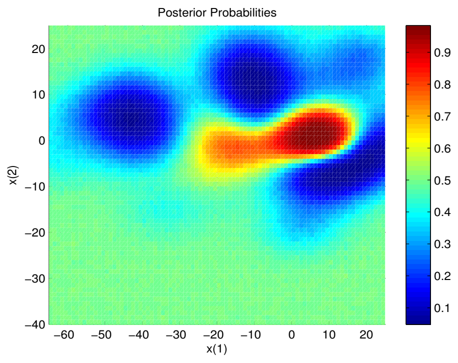


Figure 2. Decision Boundaries of recording 233

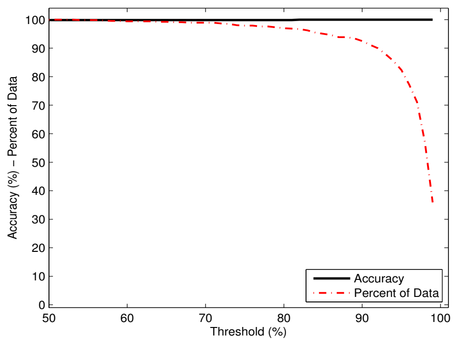


Figure 3. Model Accuracy as the threshold increases - Percent of test data set larger than the threshold

again to optimise the hyperparameters. On both types of features obtained, results indicate that Gaussian Processes are shown to perform with high precision, with an average accuracy above 90%. Moreover, a measure of performance exceeding 95% of accuracy is achieved, by considering only posterior probabilities of high confidence above a certain threshold.

Future work will emphasize on the use of different features for classifying beat hearts with Gaussian Processes, as well as the investigation of applying different approximation methods to the non-Gaussian likelihood. Moreover, future work will focus on extending the predictions to different subjects.

Acknowledgements

The authors would like to thank EPSRC for providing financial support for this project, under the grant R/115710-11-1.

References

[1] S. AFA, I. H. Combined wavelet transformation and radial basis neural network for classifying life-threatening cardiac arrhythmias. Med Bio Eng Comput 1999;37(1):566–573.

[2] Minami K, Nakajima H, Toyoshima T. Real-time discrimination of ventricular tachyarrhythmia with fourier-transform neural network. Biomedical Engineering IEEE Transactions on Feb 1999;46(2):179–185. ISSN 0018-9294.

[3] de Chazal P, Reilly R. Automatic classification of ecg beats using waveform shape and heart beat interval features. Acoustics Speech and Signal Processing 2003 Proceedings ICASSP 03 2003 IEEE International Conference on April 2003;2:II–269–72 vol.2. ISSN 1520-6149.

[4] Rasmussen CE, Williams CK. Gaussian Processes for Machine Learning. MIT press, 2005.

[5] Neal RM. Bayesian Learning for Neural Networks. Ph.D. thesis, University of Toronto, Canada, 1994.

[6] Neal RM. Bayesian Learning for Neural Networks. Springer, 1996. Lecture Notes in Statistics 118.

[7] Williams CKI, Barber D. Bayesian classification with gaussian processes. IEEE Trans Pattern Analysis Machine Intelligence 1998;20(12):1342–1351.

[8] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation 2000 (June 13);101(23):e215–e220. Circulation Electronic Pages: http://circ.ahajournals.org/cgi/content/full/101/23/e215.

[9] Daubechies I. Ten lectures on wavelets. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992. ISBN 0-89871-274-2.

Address for correspondence:

Grigorios Skolidis
Dept. of Computer Science / University of Sheffield
Regent Court, 211 Portobello
Sheffield, S1 4DP, United Kingdom

Bibliography

Adams, R. and Ghahramani, Z. (2009). Archipelago: Nonparametric Bayesian semi-supervised learning. In Bottou, L. and Littman, M., editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 1–8.

Adams, R., Murray, I., and MacKay, D. (2009). The Gaussian Process Density Sampler. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 9–16.

Albert, J. H. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679.

Alvarez, M. and Lawrence, N. (2011). Computationally Efficient Convolved Multiple Output Gaussian Processes. *Journal of Machine Learning Research*, 12:1425–1466.

Alvarez, M. and Lawrence, N. D. (2009). Sparse Convolved Gaussian Processes for Multi-output Regression. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 57–64.

Alvarez, M., Luengo, D., and Lawrence, N. (2009). Latent force models. In *Proceedings of the 12th International Workshop on Artificial Intelligence and Statistics*, volume 5, pages 9–16.

Alvarez, M., Luengo-Garcia, D., Titsias, M., and Lawrence, N. (2011a). Efficient multioutput Gaussian processes through variational inducing kernels. In *Proceedings of the 13th International Workshop on Artificial Intelligence and Statistics*, volume 9, pages 25–32.

Alvarez, M., Rosasco, L., and Lawrence, N. (2011b). Kernels for vector-valued functions: a review. *In preparation*.

Ando, R. and Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

Argyriou, A., Evgeniou, T., and Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, 73(3):243–272.

Arnold, A. (2009). *Exploiting domain and task regularities for robust named entity recognition*. PhD thesis, School of Computer Science, Machine Learning Department, Carnegie Mellon University.

Arnold, A., Nallapati, R., and Cohen, W. (2007). A comparative study of methods for transductive transfer learning. *Proceedings of the 7th IEEE International Conference on Data Mining Workshops*, pages 77–82.

Bakker, B. and Heskes, T. (2003). Task clustering and gating for Bayesian multitask learning. *The Journal of Machine Learning Research*, 4:83–99.

Barber, D. (2011). *Bayesian Reasoning and Machine Learning*. Cambridge University Press. In press.

Barry, R. and Jay, M. (1996). Blackbox kriging: spatial prediction without specifying variogram models. *Journal of Agricultural, Biological, and Environmental Statistics*, pages 297–322.

Baxter, J. (1997). A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198.

Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396.

Belkin, M. and Niyogi, P. (2003). Using manifold structure for partially labeled classification. In *Advances in Neural Information Processing Systems 15*, pages 929–936.

Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2007). Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 19*, pages 137–145.

Ben-David, S. and Borbely, R. S. (2008). A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine Learning*, 73(3):273–287.

Ben-David, S., Gehrke, J., and Schuller, R. (2002). A theoretical framework for learning from a pool of disparate data sources. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 443–449.

Ben-David, S., Luu, T., Lu, T., and Pál, D. (2010). Impossibility Theorems for Domain Adaptation. In *Proceedings of the 13th International Workshop on Artificial Intelligence and Statistics*, volume 13, pages 129–136.

Ben-David, S. and Schuller, R. (2003). Exploiting task relatedness for multiple task learning. In *COLT*.

Bickel, S., Bogojeska, J., Lengauer, T., and Scheffer, T. (2008). Multi-task learning for HIV therapy screening. In *Proceedings of the 25th International Conference on Machine Learning*, pages 56–63.

Bickel, S., Brückner, M., and Scheffer, T. (2009). Discriminative learning under covariate shift. *The Journal of Machine Learning Research*, 10:2137–2155.

Bickel, S. and Scheffer, T. (2007). Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems 19*, pages 161–168.

Birlutiu, A., Groot, P., and Heskes, T. (2010). Multi-task preference learning with an application to hearing aid personalization. *Neurocomputing*, 73(7-9):1177–1185.

Bishop, C. (2006). *Pattern recognition and machine learning*, volume 4. Springer New York:.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2008). Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems 20*, pages 129–136.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 440.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965.

Bonilla, E., Chai, K. M., and Williams, C. (2008). Multi-task Gaussian Process Prediction. In *Advances in Neural Information Processing Systems 20*, pages 153–160.

Bonilla, E. V., Agakov, F. V., and Williams, C. K. I. (2007). Kernel multi-task learning using task-specific features. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*.

Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526.

Boyle, P. and Frean, M. (2005). Dependent Gaussian Processes. In *Advances in Neural Information Processing Systems 17*, pages 217–224.

Breiman, L. and Friedman, J. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(1):3–54.

Brodersen, K., Ong, C., Stephan, K., and Buhmann, J. (2010). The binormal assumption on precision-recall curves. In *2010 International Conference on Pattern Recognition*, pages 4263–4266.

Caruana, R. (1997). Multi-task learning. *Machine Learning*, 28:41–75.

Chai, K. M. (2009). Generalization Errors and Learning Curves for Regression with Multi-task Gaussian Processes. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 279–287.

Chai, K. M. A. (2010). *Multi-task Learning with Gaussian Processes*. PhD thesis, School of Informatics, Institute for Adaptive and Neural Computation,University of Edinburgh, Edinburgh.

Chai, K. M. A., Williams, C. K. I., Klanke, S., and Vijayakumar, S. (2009). Multi-task gaussian process learning of robot inverse dynamics. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 265–272.

Chang, M., Connor, M., and Roth, D. (2010). The necessity of combining adaptation methods. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 767–777.

Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

Chu, W., Sindhwani, V., Ghahramani, Z., and Keerthi, S. S. (2007). Relational Learning with Gaussian Processes. In *Advances in Neural Information Processing Systems 19*, pages 289–296.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167.

Conti, S. and O'Hagan, A. (2010). Bayesian emulation of complex multi-output and dynamic computer models. *Journal of statistical planning and inference*, 140(3):640–651.

Cortes, C., Mohri, M., Riley, M., and Rostamizadeh, A. (2010). Sample selection bias correction theory. In *Algorithmic Learning Theory*, pages 38–53. Springer.

Crammer, K., Kearns, M., and Wortman, J. (2008). Learning from multiple sources. *The Journal of Machine Learning Research*, 9:1757–1774.

Cressie, N. A. (1993). *Statistics for spatial data*. John Wiley & Sons. New York. US.

Csató, L., Fokoué, E., Opper, M., Schottky, B., and Winther, O. (2000). Efficient approaches to Gaussian process classification. In Sara A. Solla, T. K. L. and Müller, K.-R., editors, *Advances in Neural Information Processing Systems 12*, pages 251–257. MIT Press, Cambridge, MA.

Csató, L. and Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668.

Dai, W., Chen, Y., Xue, G.-R., Yang, Q., and Yu, Y. (2009). Translated learning: Transfer learning across different feature spaces. In *Advances in Neural Information Processing Systems 21*, pages 353–360.

Dai, W., Yang, Q., Xue, G., and Yu, Y. (2008). Self-taught clustering. In *Proceedings of the 25th International Conference on Machine Learning*, pages 200–207.

Daumé, H. (2007). Frustratingly easy domain adaptation. In *Annual Meeting of the Association for Computational Linguistics*, volume 45, pages 256–263.

Daumé, H. I., Kumar, A., and Saha, A. (2010). Co-regularization based semi-supervised domain adaptation. In *Advances in Neural Information Processing Systems 23*, pages 478–486.

Daumé III, H. (2009). Bayesian multitask learning with latent hierarchies. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 135–142. AUAI Press.

Daumé III, H. and Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240.

Deak, I. (1980). Three digit accurate multiple normal probabilities. *Numerische Mathematik*, 35(4):369–380.

Dinuzzo, F., Pillonetto, G., and De Nicolao, G. (2008). Client-server multitask learning from distributed datasets. *IEEE Transactions on Neural Networks*, (99):1–14.

Do, C. and Ng, A. (2006). Transfer learning for text classification. In *Advances in Neural Information Processing Systems 18*, pages 299–306.

Duane, A. et al. (1987). Hybrid monte carlo. *Physics letters B*, 195(2):216–222.

Engel, Y., Mannor, S., and Meir, R. (2005). Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 201–208.

Evgeniou, T., Micchelli, C., and Pontil, M. (2006). Learning multiple tasks with kernel methods. *The Journal of Machine Learning Research*, 6(1):615.

Fürnkranz, J. and Hüllermeier, E. (2010). *Preference learning*. Springer-Verlag New York Inc.

Gassmann, H. I., Deak, I., and Szantai, T. (2002). Computing multivariate normal probabilities: A new look. *Journal of Computational and Graphical Statistics*, 11(4):920–949.

Gelman, A., Carlin, J. B., Stern, H. S., and B.Rubin, D. (2004). *Bayesian Data Analysis*. Chapman & Hall/CRC.

Genz, A. (1992). Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics*, 1(2):141–149.

Girolami, M. and Rogers, S. (2006). Variational Bayesian multinomial probit regression with Gaussian process priors. *Neural Computation*, 18(8):1790–1817.

Goldberg, P., Williams, C., and Bishop, C. (1998). Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems 10*, pages 493–499, Cambridge, MA. MIT press.

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220.

Gupta, A. K. and Nagar, D. K. (2000). *Matrix Variate Distributions*. Chapman & Hall/CRC.

Hanley, J. A. and Mcneil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36.

Higdon, D. (2002). Space and space-time modeling using process convolutions. *Quantitative methods for current environmental issues*, 3754.

Hinton, G. (1989). Connectionist learning procedures. *Artificial intelligence*, 40(1-3):185–234.

Huang, F. and Yates, A. (2009). Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 495–503.

Huang, J., Smola, A. J., Gretton, A., Borgwardt, K. M., and Schlkopf, B. (2007). Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems 19*, pages 601–608.

Jacobs, R., Jordan, M., Nowlan, S., and Hinton, G. (1991). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Jebara, T. (2004). Multi-task feature and kernel selection for svms. In *Proceedings of the twenty-first International Conference on Machine Learning*, page 55.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209.

Journel, A. G. and Huijbregts, C. J. (1978). *Mining Geostatistics*. Academic Press, London.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases*, volume 30, pages 180–191.

Kim, H. and Lee, J. (2007). Clustering based on Gaussian processes. *Neural computation*, 19(11):3088–3107.

Kuncheva, L. (2002). A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286.

Lafferty, J. and Wasserman, L. (2008). Statistical analysis of semi-supervised regression. In *Advances in Neural Information Processing Systems 20*, pages 801–808.

Lafferty, J. and Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 111–119.

Lawrence, N. and Platt, J. (2004). Learning to learn with the informative vector machine. In *Proceedings of the 21st International Conference on Machine Learning*, pages 65–72.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast Sparse Gaussian Process Methods: The Informative Vector Machine. In *Advances in Neural Information Processing Systems 15*, pages 609–616.

Lawrence, N. D. (2004). Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 329–336. MIT Press, Cambridge, MA.

Lawrence, N. D. and Jordan, M. I. (2005). Semi-supervised Learning via Gaussian Processes. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 753–760. MIT Press, Cambridge, MA.

Lawrence, N. D., Sanguinetti, G., and Rattray, M. (2007). Modelling transcriptional regulation using Gaussian Processes. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 785–792. MIT Press, Cambridge, MA.

Lázaro-Gredilla, M. and Titsias, M. (2011). Variational Heteroscedastic Gaussian Process Regression. In Bottou, L. and Littman, M., editors, *Proceedings of the 28th International Conference on Machine Learning*, pages 1–8.

Lee, A. C. and Crippen, G. M. (2009). Predicting pka. *Journal of Chemical Information and Modeling*, 49(9):2013–2033.

Lewis, D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. *Machine Learning: ECML-98*, pages 4–15.

Lewis, D. and Jones, K. (1996). Natural language processing for information retrieval. *Communications of the ACM*, 39(1):92–101.

Liu, Q., Liao, X., Li, H., Stack, J. R., and Carin, L. (2009). Semisupervised multitask learning. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 31(6):1074–1086.

MacKay, D. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009a). Domain adaptation: Learning bounds and algorithms. In Dasgupta, S. and Klivans, A., editors, *Proceedings of the Conference on Learning Theory*.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009b). Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems 21*, pages 1041–1048.

Matheron, G. (1973). The intrinsic random functions and their applications. *Advances in applied probability*, 5(3):439–468.

Menzefricke, U. (2000). Hierarchical modeling with gaussian processes. *Communications in Statistics-Simulation and Computation*, 29(4):1089–1108.

Minka, T. (2001). Expectation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence*, volume 17, pages 362–369.

Minton, S. and Knoblock, C. (2002). Active+ semi-supervised learning= robust multi-view learning. In *Proceedings of the 19th International Conference on Machine Learning*.

Murray, I., Adams, R. P., and MacKay, D. J. (2010). Elliptical slice sampling. *Journal of Machine Learning Research*, 9:541–548.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer. Lecture Notes in Statistics 118.

Nigam, K., Lafferty, J., and McCallum, A. (1999). Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67.

Obozinski, G., Wainwright, M., and Jordan, M. (2009). High-dimensional support union recovery in multivariate regression. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1217–1224.

O'Hagan, A. and Kingman, J. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B (Methodological)*, 40(1):1–42.

Opper, M. and Winther, O. (2000). Gaussian processes for classification: Mean-field algorithms. *Neural Computation*, 12(11):2655–2684.

Osborne, M., Roberts, S., Rogers, A., Ramchurn, S., and Jennings, N. (2008). Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 109–120.

Pan, S., Tsang, I., Kwok, J., and Yang, Q. (2009). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, (99):1–12.

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

Parameswaran, S. and Weinberger, K. (2010). Large margin multi-task metric learning. In *Advances in Neural Information Processing Systems 23*, pages 1867–1875.

Philippe, R. (2007). Generalization error bounds in semi-supervised classification under the cluster assumption. *The Journal of Machine Learning Research*, 8:1369–1392.

Pillonetto, G., Dinuzzo, F., and De Nicolao, G. (2010). Bayesian online multitask learning of Gaussian processes. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 32(2):193–205.

Qi, Y., Tastan, O., Carbonell, J., Klein-Seetharaman, J., and Weston, J. (2010). Semi-supervised multi-task learning for predicting interactions between HIV-1 and human proteins. *Bioinformatics*, 26(18):i645.

Quattoni, A., Collins, M., and Darrell, T. (2008). Transfer learning for image classification with sparse prototype representations.

Quinonero-Candela, J., Rasmussen, C., and Williams, C. (2007). Approximation methods for gaussian process regression. *Large-scale kernel machines*, pages 203–224.

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. (2009). *Dataset shift in machine learning*. The MIT Press.

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. (2007). Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, pages 759–766.

Raina, R., Ng, A., and Koller, D. (2006). Constructing informative priors using transfer learning. In *International Conference on Machine Learning*, pages 713–720.

Rasmussen, C. and Ghahramani, Z. (2001). Infinite mixtures of Gaussian Process experts. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 881–888, Cambridge, MA. MIT Press.

Rasmussen, C. E. and Kuss, M. (2004). Gaussian Processes in Reinforcement Learning. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 751–759. MIT Press, Cambridge, MA.

Rasmussen, C. E. and Williams, C. K. (2005). *Gaussian Processes for Machine Learning*. MIT press.

Rebonato, R. and Jäckel, P. (2000). The most general methodology to create a valid correlation matrix for risk management and option pricing purposes. *Journal of Risk*, 2(2).

Rupp, M., Körner, R., and Tetko, I. V. (2010). Estimation of acid dissociation constants using graph kernels. *Molecular Informatics*, 29(10):731–740.

Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors (1999). *Advances in kernel methods: support vector learning*. MIT Press, Cambridge, MA, USA.

Schwaighofer, A., Tresp, V., and Yu, K. (2005). Learning Gaussian Process Kernels via Hierarchical Bayes. In *Advances in Neural Information Processing Systems 17*, pages 1209–1216.

Seeger, M. (2001). Learning with labeled and unlabeled data. Technical report, Institute of Adaptive and Neural Computation, University of Edinburgh.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244.

Silver, D. and Mercer, R. (2001). Selective functional transfer: Inductive bias from related tasks. In *IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2001)*, pages 182–189.

Silverman, B. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(1):1–52.

Sindhwani, V. (2007). *On semi-supervised Kernel methods*. PhD thesis, Department of Computer Science, University of Chicago, Illinois.

Sindhwani, V., Chu, W., and Keerthi, S. S. (2007). Semi-supervised Gaussian process classifiers. In *International Joint Conference on Artifical Intelligence*, pages 1059–1064.

Sindhwani, V., Niyogi, P., and Belkin, M. (2005). Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 824–831.

Singh, A., Nowak, R., and Zhu, X. (2009). Unlabeled data: Now it helps, now it doesn't. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1513–1520.

Skolidis, G., Clayton, R., and Sanguinetti, G. (2008). Automatic classification of arrhythmic beats using gaussian processes. In *IEEE Transaction on Computers in Cardiology, 2008*, pages 921–924, Bologna, Italy.

Smola, A. and Bartlett, P. (2001). Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625.

Smola, A. and Schölkopf, B. (2002). *Learning with kernels*, volume 3. MIT press.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264.

Storkey, A. J. and Sugiyama, M. (2007). Mixture regression for covariate shift. In *Advances in Neural Information Processing Systems 19*, pages 1337–1344.

Sugiyama, M., Krauledat, M., and Müller, K. (2007). Covariate shift adaptation by importance weighted cross validation. *The Journal of Machine Learning Research*, 8:985–1005.

Sutton, C. and McCallum, A. (2005). Composition of conditional random fields for transfer learning. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 748–754.

Taylor, M. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685.

Teh, Y., Seeger, M., and Jordan, M. (2005). Semiparametric Latent Factor Models. In *Workshop on Artificial Intelligence and Statistics*, volume 10.

Thrun, S. and Pratt, L. (1998). *Learning to learn*. Kluwer Academic Pub.

Titsias, M., Lawrence, N. D., and Rattray, M. (2009). Efficient Sampling for Gaussian Process Inference using Control Variables. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1681–1688.

Tresp, V. (2000). Mixtures of Gaussian Processes. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 654–660, Cambridge, MA. MIT Press.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

Vapnik, V. (1998). *Statistical learning theory*. Wiley-Interscience.

Vapnik, V. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999.

Ver Hoef, J. and Barry, R. (1998). Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Planning and Inference*, 69(2):275–294.

Vilalta, R. and Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review.*, 18(2):77–95.

Von Luxburg, U., Belkin, M., and Bousquet, O. (2008). Consistency of spectral clustering. *The Annals of Statistics*, 36(2):555–586.

Wackernagel, H. (2003). *Multivariate geostatistics: an introduction with applications.* Springer Verlag.

Wang, Y., Khardon, R., and Protopapas, P. (2010). Shift-invariant grouped multi-task learning for Gaussian processes. *Machine Learning and Knowledge Discovery in Databases*, pages 418–434.

Wang, Z., Song, Y., and Zhang, C. (2008). Transferred dimensionality reduction. *Machine Learning and Knowledge Discovery in Databases*, pages 550–565.

Waterhouse, S. (1997). *Classification and regression using mixtures of experts.* PhD thesis, Department of Engineering, Cambridge University.

Weinberger, K. and Saul, L. (2009). Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244.

Williams, C. and Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351.

Williams, C. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688.

Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian Processes for Regression. In S.Touretzky, D., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT press.

Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. (2007). Multi-task learning for classification with Dirichlet process priors. *The Journal of Machine Learning Research*, 8:35–63.

Yang, Q., Chen, Y., Xue, G., Dai, W., and Yu, Y. (2009). Heterogeneous transfer learning for image clustering via the social web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 1–9. Association for Computational Linguistics.

Yu, K. and Chu, W. (2008). Gaussian process models for link analysis and transfer learning. In *Advances in Neural Information Processing Systems 20*, pages 1657–1664.

Yu, K., Chu, W., Yu, S., Tresp, V., and Xu, Z. (2007a). Stochastic relational models for discriminative link prediction. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 1553–1560. MIT Press, Cambridge, MA.

Yu, K., Tresp, V., and Schwaighofer, A. (2005). Learning Gaussian processes from multiple tasks. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 1012–1019.

Yu, S., Tresp, V., and Yu, K. (2007b). Robust multi-task learning with t-processes. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1103–1110.

Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In *Proceedings of the 21st International Conference on Machine Learning*, page 114.

Zhang, Y. and Yeung, D. (2009). Semi-supervised multi-task regression. *Machine Learning and Knowledge Discovery in Databases*, pages 617–631.

Zhang, Y. and Yeung, D.-Y. (2010). Multi-task learning using generalized t process. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 964–971.

Zhang, Y., Yeung, D.-Y., and Xu, Q. (2010). Probabilistic multi-task feature selection. In *Advances in Neural Information Processing Systems 23*, pages 2559–2567.

Zhu, S., Yu, K., and Gong, Y. (2009). Stochastic relational models for large-scale dyadic data using mcmc. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1993–2000.