UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Guilherme Henrique Caponetto

Data-driven hierarchical structures in multi-task learning

Estruturas hierárquicas orientadas por dados em aprendizado multi-tarefa

Campinas
2017

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Guilherme Henrique Caponetto

Data-driven hierarchical structures in multi-task learning

Estruturas hierárquicas orientadas por dados em aprendizado multi-tarefa

Dissertation presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, in the area of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na área de Engenharia de Computação.

Supervisor: Prof. Dr. Fernando José Von Zuben

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Guilherme Henrique Caponetto, e orientada pelo Prof. Dr. Fernando José Von Zuben

---

Campinas
2017

Agência(s) de fomento e nº(s) de processo(s): CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

C173d

Caponetto, Guilherme Henrique, 1988-
Data-driven hierarchical structures in multi-task learning / Guilherme Henrique Caponetto. – Campinas, SP : [s.n.], 2017.

Orientador: Fernando José Von Zuben.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Aprendizado de máquina. 2. Mineração de dados. 3. Algoritmos. I. Von Zuben, Fernando José,1968-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Estruturas hierárquicas orientadas por dados em aprendizado multi-tarefa
Palavras-chave em inglês:
Machine learning
Data mining
Algorithms
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Fernando José Von Zuben [Orientador]
Márcio Porto Basgalupp
Levy Boccato
Data de defesa: 09-08-2017
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA – DISSERTAÇÃO DE MESTRADO

Candidato: Guilherme Henrique Caponetto **RA:** 162639

Data da defesa: 9 de agosto de 2017

Título da dissertação: "Data-driven hierarchical structures in multi-task learning (Estruturas hierárquicas orientadas por dados em aprendizado multi-tarefa)"

Prof. Dr. Fernando José Von Zuben (Presidente, FEEC/UNICAMP)
Prof. Dr. Márcio Porto Basgalupp (ICT/UNIFESP)
Prof. Dr. Levy Boccato (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Dedico este trabalho à minha família.

Acknowledgements

Agradeço à minha companheira de vida, Florence, por simplesmente tudo. Seu suporte incondicional foi fundamental para que eu decidisse encarar mais este desafio. Sua compreensão nos momentos em que eu precisava estar ausente para trabalhar no mestrado foi essencial para que eu conseguisse focar no que precisava ser feito para atingir meus objetivos. Sem dúvida, a sua presença tinha o dom de aliviar o fardo das demandas mais exigentes da pesquisa. A certeza da sua companhia, como recompensa nos momentos de descanso, sempre me motivou ainda mais.

Agradeço ao meu orientador, professor Fernando, pela oportunidade, confiança e dedicação desde o início dos estudos até o final do desenvolvimento deste trabalho. Agradeço também pelas aulas repletas de entusiasmo, inspiração e informação, das quais tive o prazer de participar como aluno e que despertaram meu interesse pelas áreas de conhecimento que se interseccionam com o conteúdo deste trabalho.

Agradeço aos meus pais, Francisco e Claudete, por sempre me incentivarem a manter um ritmo constante de aprendizado e por terem me ensinado a trabalhar com muito foco e perseverança para atingir meus objetivos.

Agradeço aos membros da comissão julgadora, professores Levy e Márcio, pelo tempo dedicado e contribuições que ajudaram a melhorar a qualidade deste trabalho.

Agradeço aos colegas Conrado e André, pelas valiosas dicas e conselhos dados durante meus estudos.

Agradeço ao professor Eleri, pela oportunidade que me foi dada. O ótimo exemplo de profissional que é fica evidente em cada atitude.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo suporte financeiro para a realização deste trabalho.

"What we do now echoes in eternity."
Marcus Aurelius

Abstract

In multi-task learning, a set of learning tasks is simultaneously considered during the learning process so that it can leverage performance by exploring similarities among the tasks. In a significant number of approaches, such similarities are encoded as additional information within the regularization framework. Although some sort of structure is taken into account by several proposals, such as the existence of task clusters or a graph-based relationship, others have shown that using a properly defined hierarchical structure may lead to competitive results. Focusing on a hierarchical relationship, the extension pursued in this research is based on the idea of learning it directly from data, enabling a methodology like this to be extended to a wider range of applications. Thus, the hypothesis raised is that obtaining a representative hierarchy-based task relationship from data and using this additional information as a penalty term in the regularization framework would be beneficial, relaxing the necessity of a domain-specific specialist and improving overall generalization predictive performance. Therefore, the novelty of the data-driven hierarchical approaches proposed in this dissertation for multi-task learning is that information exchange among associated real tasks is promoted by auxiliary hypothetical tasks at the upper nodes, given that the real tasks are not directly connected in the hierarchy. Once the main idea involves obtaining a hierarchical structure, several studies were performed focusing on combining both hierarchical clustering and multi-task learning areas. Three promising strategies for automatically obtaining hierarchical structures were adapted to the context of multi-task learning. Two of them are Bayesian-based approaches and one of those two is characterized by non-binary branching. The possibility of cutting edges is also investigated, being a powerful tool to detect outlier tasks. Moreover, a general concept called Hierarchical Multi-Task Learning Framework is proposed, individually grouping modules, which can be easily extended in future research. Extensive experiments are presented and discussed, showing the potential of employing a hierarchical structure obtained directly from task data within the regularization framework. Both synthetic datasets with known underlying relations among tasks and real-world benchmark datasets from the literature are adopted in the experiments, providing evidence that the proposed framework consistently outperforms well-established multi-task learning strategies.

Keywords: Multi-Task Learning; Hierarchical Clustering; Regularization.

Resumo

Em aprendizado multi-tarefa, um conjunto de tarefas é simultaneamente considerado durante o processo de aprendizado de modo a promover ganho de desempenho através da exploração de similaridades entre tarefas. Em um número significativo de abordagens, tais similaridades são codificadas como informação adicional na etapa de regularização. Embora algumas estruturas sejam levadas em consideração em muitas propostas, como a existência de grupos de tarefas ou um relacionamento baseado em grafo, outras propostas mostraram que usar uma estrutura hierárquica corretamente definida poderá guiar a resultados competitivos. Focando em um relacionamento hierárquico, a extensão buscada nesta pesquisa é baseada na ideia de aprender a estrutura diretamente dos dados, possibilitando que a metodologia multi-tarefa possa ser estendida a uma gama mais vasta de aplicações. Assim, a hipótese levantada é que obter um relacionamento representativo dos dados baseado em hierarquia entre tarefas e usar esta informação adicional como um termo de penalização dentro do formalismo de aprendizado regularizado seria benéfico, relaxando a necessidade de um especialista específico de domínio e melhorando o desempenho de predição. Portanto, a novidade em abordagens hierárquicas orientadas por dados propostas nesta dissertação para aprendizado multi-tarefa é que a troca de informação entre tarefas reais associadas é promovida por tarefas hipotéticas auxiliares presentes nos nós mais altos, dado que as tarefas reais não são diretamente conectadas na hierarquia. Uma vez que a ideia principal envolve obter uma estrutura hierárquica, estudos foram feitos com foco em combinar ambas as áreas de clusterização hierárquica e aprendizado multi-tarefa. Três estratégias promissoras para a obtenção automática de estruturas hierárquicas foram adaptadas ao contexto de aprendizado multi-tarefa. Duas delas são abordagens bayesianas, sendo uma caracterizada por ramificações não binárias. A possibilidade de corte na estrutura também é investigada, sendo uma poderosa ferramenta para detecção de tarefas *outliers*. Além disso, um conceito geral chamado *Hierarchical Multi-Task Learning Framework* é proposto, agrupando módulos individualmente, os quais podem ser facilmente estendidos em pesquisas futuras. Experimentos extensivos são apresentados e discutidos, mostrando o potencial da utilização de estruturas hierárquicas obtidas diretamente dos dados para guiar a etapa de regularização. Foram adotados nos experimentos tanto conjuntos de dados sintéticos, com relacionamento entre tarefas conhecido, como conjuntos de dados reais utilizados na literatura, nos quais foi possível observar que o *framework* proposto consistentemente supera estratégias bem estabelecidas de aprendizado multi-tarefa.

Palavras-chaves: Aprendizado Multi-Tarefa; Clusterização Hierárquica; Regularização.

List of Figures

List of Tables

List of abbreviations and acronyms

AGM	Accelerated Gradient Method
BHC	Bayesian Hierarchical Clustering
BRT	Bayesian Rose Trees
CE	Classification Error
DP	Dirichlet Process
DPMM	Dirichlet Process Mixture Model
EM	Expectation–Maximization
HIA	Hierarchy identification algorithm
HMTL	Hierarchical Multi-Task Learning
LASSO	Least Absolute Shrinkage and Selection Operator
LR	Logistic Regression
OLS	Ordinary Least Squares
MAP	Maximum A Posteriori
MTL	Multi-Task Learning
PC	Pairwise Combination
RBHC	Randomized Bayesian Hierarchical Clustering
RMSE	Root Mean Squared Error
RS	Random Search
STL	Single-Task Learning

Contents

Chapter 1

Introduction

Countless problems distributed in distinct areas have been successfully solved by machine learning techniques over the years and, with the exponential growth in the amount of collected data in different domains alongside low-cost computation, a whole set of new problems frequently arises to be tackled, demanding new computationally scalable and high-performance approaches (JORDAN; MITCHELL, 2015).

Particularly, when a set of similar supervised tasks (regression or classification) intends to learn linear predictive models, considering a procedure to jointly learn them with information exchange may guide to better generalization performance. A simple and intuitive use case in this setting would be modeling personalized predictors for a set of users (*e.g.* spam detectors). These predictors may take advantage of similarities among related users, so that better individual models can be produced, mainly when the performance of single-task learning is limited by insufficient training data (XUE *et al.*, 2007).

Associated with the transfer learning concept (see more in Pan & Yang (2010), Xu & Yang (2011) and references therein), multi-task learning (MTL) has attracted attention when multiple related tasks are jointly solved, once common information across them can flow as inductive bias, contributing to improve generalization performance (CARUANA, 1993; CARUANA, 1997). However, task relations are not always attainable and discovering a good representation of their underlying structure may become considerably challenging. In some fortunate cases, a domain expert is available to define the task relationship, but not depending on an expert and letting data themselves tell how the tasks are related would be more desirable (BAXTER, 1997; ZHANG; YEUNG, 2010).

In MTL early years, presuming all tasks being equally related was a popular hypothesis or, even, that they share a common structure. However, if tasks are linked in an improper manner, that is, a relationship is forced when it should not exist, MTL may no longer be beneficial and might even produce negative transfer, characterized by a worse performance than learning each task individually (XU *et al.*, 2015; KANG *et al.*, 2011). Moreover, even in the case where tasks might seem all related to each other, distinct degrees of relationship among tasks may be hidden and, therefore, those distinct degrees would be completely neglected if all tasks are assumed to be equally related (BAKKER;

HESKES, 2003).

In order to address negative transfer, some approaches have been proposed assuming a prior knowledge on the task relationship structure, such as clusters (JACOB *et al.*, 2008), graph models (EVGENIOU *et al.*, 2005) and hierarchy (WIDMER *et al.*, 2010), most of them translated as additional information via the regularization term. Recently, a small number of approaches have been proposed to learn a graph-based relationship among tasks directly from data (GONÇALVES *et al.*, 2014; ARGYRIOU *et al.*, 2013), but learning a hierarchy-based structure in this setting is a gap yet to be filled.

The area of clustering has been actively explored over the past decades in different domains, providing several tools to group objects that present a determined level of similarity. Either divisive or agglomerative, approaches have been proposed particularly aiming at performing a hierarchical clustering on objects. Such generated hierarchical structures provide distinct levels of granularity regarding the cluster organization of data. Essentially, objects to be clustered are considered leaves in the hierarchical structure and internal nodes represent clusters of leaves directly below in the hierarchy (MAIMON; ROKACH, 2010).

Nonetheless, combining MTL and hierarchical clustering is challenging due to the necessity of defining the objects to be clustered that will represent the tasks and also how to measure task similarity in order to build a hierarchical structure. Moreover, classical hierarchical clustering approaches do not provide an effective way of inferring the number of clusters, once the output is just a single structure. Recently, Bayesian-based approaches have been proposed, focused on addressing classic agglomerative hierarchical clustering limitations, such as the lack of a probabilistic model describing the clusters and the problem of inferring the number of clusters (HELLER; GHAHRAMANI, 2005a; BLUNDELL *et al.*, 2012).

Studies performed by Widmer *et al.* (2010) and Shan *et al.* (2012) have shown that exploring a properly defined hierarchical structure may be beneficial to the learning process, which motivates a research to address the scenario where one does not know beforehand this hierarchical structure. Additionally, since the literature devoted to hierarchical clustering has been actively explored, several tools to obtain a hierarchy-based structure among objects may be employed, though its usage in the context of MTL has not been deeply investigated.

Thus, with the purpose of contributing to the MTL community, the main goal of this work is to study and develop methodologies that identify a representative hierarchical relationship among tasks from data, and to take advantage of the obtained structure in the regularization of the learning tasks. In this dissertation, three approaches to obtain a hierarchical structure from tasks are proposed, in which two of them employ the Bayesian-based algorithms proposed by Heller & Ghahramani (2005a) and Blundell *et al.* (2012).

1.1 Structure of the dissertation

The structure of the dissertation is divided into four parts. The first part provides a literature review of the main concepts used in the research. The second part introduces the proposals of the dissertation. The third part deals with the experiments that exercise the proposals. The fourth part concludes the dissertation. A brief description of the chapters is provided as follows:

Part I - Background

- Chapter 2 presents a literature overview regarding MTL focusing on the combination with the regularization framework, and discussing the main approaches and applications found in the literature.
- Chapter 3 presents a literature overview regarding Hierarchical Clustering with emphasis in Bayesian-based approaches advances, since they are directly related to this work.

Part II - Proposals

- Chapter 4 introduces the proposed data-driven hierarchical multi-task learning framework, motivating its conception and detailing its features.

Part III - Experiments

- Chapter 5 presents all the necessary information regarding the experimental setup, such as evaluation metrics, description of the datasets, competing approaches and specific settings of the experiments.
- Chapter 6 presents and discusses the results obtained from all the experiments performed during the research, aiming at assessing the proposed approaches and providing insights for those who want to understand, use or even extend it.

Part IV - Final considerations

- Chapter 7 concludes the dissertation by discussing the final considerations obtained from the entire research, and suggests future directions to be followed in order to extend the proposed approaches.

Part I

Background

Chapter 2

Multi-task learning

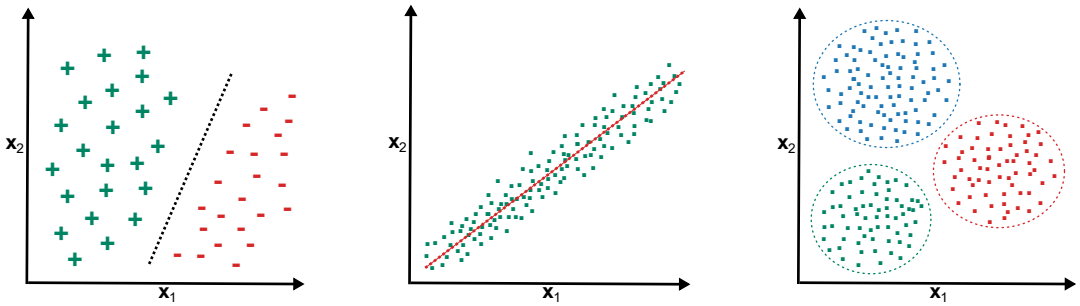
2.1 Overview

Machine learning is evolving as a data-driven research area, offering several tools to extract useful information from data in many distinct domains, characterizing such processes of information extraction as learning tasks. Although there are several types of learning tasks, the most common ones can be basically grouped into supervised learning or unsupervised learning, which depends both on the configuration of the training data and the goal of the task. Other learning scenarios may be encountered as well, such as online learning, reinforcement learning and active learning (MOHRI *et al.*, 2012).

In supervised learning, one has access to labeled data associated with a task, that is, the data is composed of input values with their expected output values, and the goal is to learn a model that captures correlations between input and output values, so that predictions may be made via these correlations. In unsupervised learning, on the other hand, the data is not labeled, so the goal is to learn correlations considering only the input values, which is more challenging. In addition, there is semi-supervised learning, which is characterized when one has access to a small amount of labeled data combined with a large amount of unlabeled data. Figures 2.1(a) and 2.1(b) show two real world examples of supervised learning tasks, whereas Figure 2.1(c) shows a real world example of unsupervised learning task.

In order to work on a learning task, one generally has access to limited observed data, compounding a finite and noisy dataset available for training the predictive model. In practice, when learning these predictive models, one may consider splitting the observed data into training and test sets, representing observed and unseen data, respectively. Thereafter, training techniques are applied using the training set in order to learn the predictive model, and its final performance is evaluated using the test set. Additionally, the training set may be split into training and validation sets in order to perform internal validations during the learning process.

Since currently there are several sources of data due to advances in computing devices, a case where one needs to deal with not only one, but multiple related learning



(a) Spam detection (b) Sales estimates (c) Market segmentation

Figure 2.1 – Examples of learning tasks. (a) shows an example of a binary classification task, where the goal is to classify e-mails into spam or not spam. (b) shows an example of a linear regression task, where the goal is to predict sales for a determined store. (c) shows an example of a clustering task, where the goal is to identify customer groups potentially interested in the same market segment.

tasks could happen. Additionally, it could make more sense to break a large number of collected training data into several possibly related learning tasks, thus preserving task particularities inside each partition.

A naïve approach to deal with many related tasks would be applying traditional single-task learning (STL), that is, learning each task in isolation, though this strategy does not seem appropriate, since similarities across tasks are not exploited whatsoever (EVGENIOU; PONTIL, 2004; ZHOU *et al.*, 2011a). On the other hand, one could simply consider learning a single general model simultaneously incorporating all the tasks, which allows capturing the similarities among tasks, but individual information from each task would be ignored, since the tasks are not identical (XUE *et al.*, 2007). Figure 2.2 depicts these two naïve approaches to deal with many related tasks.

(a) Particularities (b) Similarities

Figure 2.2 – Naïve approaches to solve multiple related tasks. Each circle filled in red represent a task and **w** represent its respective resulting parameter vector after the training step. In (a), the focus is on task particularities, where each task is treated individually during the learning process, so that no information is exchanged among them. In (b), the focus is on the task similarities, where all tasks are incorporated into one general task during the learning process, but task particularities are ignored to satisfy a general model.

Hence, a more intuitive approach would be letting tasks take their individual information into account and also simultaneously learn from one another so that correlations can be used as an inductive bias in the learning process. Such approach is known as multi-

task learning (MTL) (CARUANA, 1993) and has attracted increasing attention over the past years, supported by theoretical studies, such as in Ben-David & Schuller (2003) and Ando & Zhang (2005), and empirical results, such as in Evgeniou *et al.* (2005), Widmer *et al.* (2010) and Xu *et al.* (2015).

2.2 Task relationship

Although the idea of simultaneously learning a bunch of tasks seems attractive, it only makes sense when the tasks exhibit some sort of relationship among them and when one is able to properly identify the existing relationship. Such relationship could be known beforehand, which is not a usual scenario (ZHONG; KWOK, 2012), or introduced by a domain-specific expert (BAXTER, 1997). Either way, imposed relations may be incorrect and depending upon experts tends to be costly (CARUANA, 1993). Those are relevant motivations for letting the machine automatically learn the underlying relationship from data (XUE *et al.*, 2007). In spite of this, a number of approaches were proposed considering that all tasks are somehow close to each other, which may not hold in many real-world applications (XU *et al.*, 2015). Therefore, a key point to get good results when applying MTL is to properly capture task relationship (BEN-DAVID; SCHULLER, 2003; ZHANG; YEUNG, 2010).

Nevertheless, when task relationship is incorrectly modeled, for instance forcing a relationship among dissimilar tasks or failing to detect outlier tasks, MTL may no longer be beneficial and might even degrade performance to the point that learning each task in isolation would be better. This behavior is known as negative transfer (THRUN; O'SULLIVAN, 1996; ZHANG; YEUNG, 2010; KANG *et al.*, 2011; CHEN *et al.*, 2011; ZHONG; KWOK, 2012; XU *et al.*, 2015). In MTL early stages, Caruana (1993) had already approached this subject, pointing out that MTL is a problem-dependent tool that can either help or hurt and, therefore, a deep analysis must be performed concerning the task relationship.

Despite positive correlation, which happens when tasks are similar, Zhang & Yeung (2010) considered negative correlation as well, that is, when dissimilar tasks help to reduce the complexity of the learning stage. They argued that this kind of correlation can be modeled by learning a task covariance matrix and most of the proposed approaches consider that specific tasks only present either positive correlation or no relation. In their work, a formulation is proposed for learning both task parameter vectors and the covariance matrix. Gonçalves *et al.* (2014) also considered negative correlation in their work, but the inverse covariance matrix (precision matrix) was taken into account rather than the covariance matrix.

All this problematic around task relationship has led researchers to propose numerous approaches with a variety of views regarding how tasks relates to each other. Such

approaches range from using a shared hidden layer in neural networks (CARUANA, 1993; CARUANA, 1997), assuming a common prior in Bayesian hierarchical models (BAKKER; HESKES, 2003), assuming a low-rank structure among task parameter vectors (JI; YE, 2009) or assuming that task parameter vectors are structured in distinct clusters (THRUN; O'SULLIVAN, 1996; JACOB *et al.*, 2008; ZHANG; YEUNG, 2010), to name a few. A wide discussion of the main approaches is presented in Section 2.4.

Another important aspect of MTL is its advantage over STL when dealing with insufficient training data. Clearly, model quality is compromised if limited training data is available, which perfectly fits the MTL formalism, since more information is available in multiple related tasks and it also allows tasks to avoid learning individual noise (CARUANA, 1993; EVGENIOU *et al.*, 2005; XUE *et al.*, 2007; WIDMER *et al.*, 2010). Still regarding training data, but dealing with its format in the supervised learning context, the following scenarios can be considered in MTL: all tasks have the same output values, but different input values; all tasks have the same input values, but different output values; all tasks have different input and output values (EVGENIOU; PONTIL, 2004).

2.3 Regularized multi-task learning

Considering the case where training data (input and output) are different among tasks and all tasks use the same feature space, MTL can be more formally defined as follows. Let $\mathcal{S} = \{\mathcal{S}_k : k = 1, \ldots, m\}$ be the set of m supervised learning tasks devoted to learning linear models. The available training data for the k-th task, which has n_k data observations described by d features, is represented by $\mathcal{S}_k = \{(\mathbf{x}_k^i, \mathrm{y}_k^i) : i = 1, \ldots, n_k\}$, where $\mathbf{x}_k^i \in \mathbb{R}^d$ is the input, while $\mathrm{y}_k^i \in \mathbb{R}$ is the corresponding output when dealing with regression problems, and $\mathrm{y}_k^i \in \{0, 1\}$ when dealing with binary classification problems. Therefore, the goal is to learn a parameter vector $\mathbf{w}_k \in \mathbb{R}^d$ for each task k, such that $f(\mathbf{x}_k^i, \mathbf{w}_k) \approx \mathrm{y}_k^i$, $i = 1, \ldots, n_k$ and $k = 1, \ldots, m$. Additionally, parameter vectors that need to be estimated for the tasks are organized in a set denoted by $\mathcal{W} = \{\mathbf{w}_k : k = 1, \ldots, m\}$ for better exposition. Table 2.1 summarizes the components of a multi-task learning problem.

Table 2.1 – Description of a multi-task learning problem.

	Set $\mathcal{S}$	Features	Observations	Training data*	Set $\mathcal{W}$
$Task_1$	$\mathcal{S}_1$	d	n_1	$\{(\mathbf{x}_1^i, \mathrm{y}_1^i)\}_{i=1}^{n_1}$	$\mathbf{w}_1 \in \mathbb{R}^d$
$Task_2$	$\mathcal{S}_2$	d	n_2	$\{(\mathbf{x}_2^i, \mathrm{y}_2^i)\}_{i=1}^{n_2}$	$\mathbf{w}_2 \in \mathbb{R}^d$
...	...	...	...	...	...
$Task_m$	$\mathcal{S}_m$	d	n_m	$\{(\mathbf{x}_m^i, \mathrm{y}_m^i)\}_{i=1}^{n_m}$	$\mathbf{w}_m \in \mathbb{R}^d$

* $\mathbf{x}_k^i \in \mathbb{R}^d$ is the input, while the corresponding output is either $\mathrm{y}_k^i \in \mathbb{R}$ or $\mathrm{y}_k^i \in \{0, 1\}$ for regression or binary classification problems, respectively.

In the STL case when dealing with linear predictive models, a common approach to employ is to fix a convex loss function $\ell(f(\mathbf{x}, \mathbf{w}), \mathrm{y})$, such as squared, logistic or hinge, that measures the cost of the prediction $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$ compared to the correct output y (JACOB *et al.*, 2008). Then, this cost is averaged over all training data according to Equation (2.1), denoted the empirical loss over the single training set:

$$L(X, \mathbf{y}, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}^i, \mathbf{w}), \mathrm{y}^i) \tag{2.1}$$

where n is the number of observations, $X \in \mathbb{R}^{n \times d}$ is the input matrix and $\mathbf{y} \in \mathbb{R}^n$ is the desired output vector. Bringing up the empirical loss to multiple tasks, Equation (2.1) can then be generalized by taking into account all m tasks together, leading to Equation (2.2):

$$L(\mathcal{S}, \mathcal{W}) = \sum_{k=1}^{m} \left[\frac{1}{n_k} \sum_{i=1}^{n_k} \ell(f(\mathbf{x}_k^i, \mathbf{w}_k), \mathrm{y}_k^i) \right] \tag{2.2}$$

Although Equation (2.2) can simultaneously cover the empirical loss of multiple tasks, it is still equivalent to treating each task independently. Moreover, computing the average of the empirical loss of each task ignores that the k-th task may have more information than others, thus $\frac{1}{n_k}$ can be removed from Equation (2.2) to encourage such bias, resulting in the Equation (2.3):

$$L(\mathcal{S}, \mathcal{W}) = \sum_{k=1}^{m} \sum_{i=1}^{n_k} \ell(f(\mathbf{x}_k^i, \mathbf{w}_k), \mathrm{y}_k^i) \tag{2.3}$$

In order to incorporate the relationship among tasks, the well-established regularization framework, which is largely applied to STL, can be generalized to the context of MTL, providing tools to model task relations. Essentially, prior assumptions about task relationship can be enforced in the regularization term R (EVGENIOU; PONTIL, 2004; JACOB *et al.*, 2008; WIDMER *et al.*, 2010; GONG *et al.*, 2012). Hence, the regularized MTL cost function can be represented by Equation (2.4):

$$J(\mathcal{S}, \mathcal{W}) = \underbrace{\sum_{k=1}^{m} \sum_{i=1}^{n_k} \ell(f(\mathbf{x}_k^i, \mathbf{w}_k), \mathrm{y}_k^i)}_{\text{MTL loss function}} + \lambda\, R(\mathcal{W}) \tag{2.4}$$

where $\lambda \geq 0$ is a hyper-parameter that controls the penalty influence and should be optimized by using some technique, such as cross-validation. Therefore, an optimum λ must be found so that it can provide a balance between the terms L and R. This formulation also covers the case where all tasks are indeed independent, thus becoming equivalent to STL (EVGENIOU; PONTIL, 2004).

Finally, the optimization problem boils down to finding the task parameter

vectors $\mathcal{W}$, given training data $\mathcal{S}$, which minimizes the regularized cost function J, such that:

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} J(\mathcal{S}, \mathcal{W}) \tag{2.5}$$

As will be seen in Section 2.4, the class of regularized MTL is very popular and several approaches have been proposed on top of it, where a notion of task similarity is translated into the regularization term. Usually, researchers have pursued convex or, at least, bi-convex formulations for J, so that off-the-shelf optimization techniques can then be applied.

In addition, if the cost function J is smooth, the optimization problem expressed in Equation (2.5) could be efficiently solved by first-order iterative optimization algorithms, such as the gradient descent method, which has convergence rate of $O(1/t)$, where t is the number of iterations. A gradient descent method starts at an initial condition and iteratively explores the search space toward better candidate solutions.

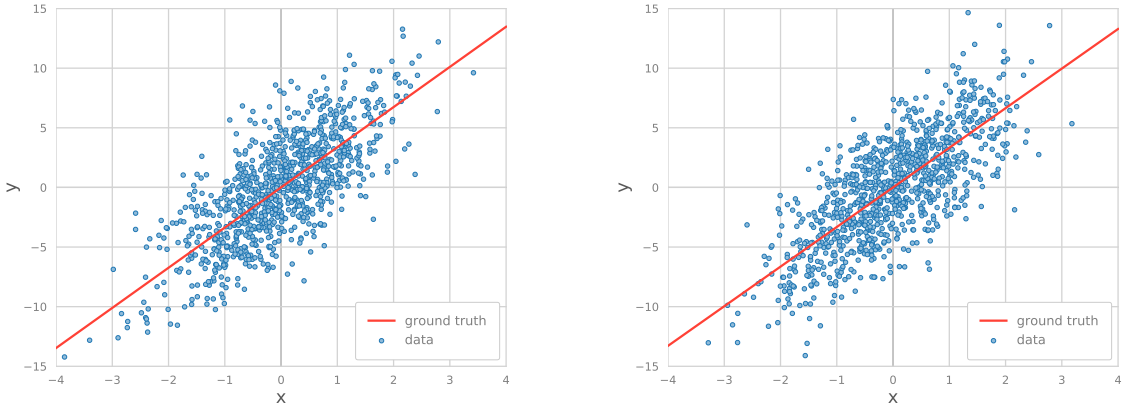
In order to speed up the gradient descent method, Nesterov (1983) proposed an accelerated gradient method (AGM), which has convergence rate of $O(1/t^2)$. Basically, AGM uses a linear combination of the last two candidate solutions to operate as the starting point of the search. Recently, AGM's author also proposed other accelerated methods covering non-smooth convex functions (NESTEROV, 2005) and composite functions (NESTEROV *et al.*, 2007). Although AGM is designed for general purposes, in the MALSAR toolbox (ZHOU *et al.*, 2011b), which stands for Multi-tAsk Learning via StructurAl Regularization, most of the included MTL approaches are implemented via AGM.

2.3.1 STL and regularized MTL: a comparative example

The simplest scenario in which MTL approaches may be employed is the one composed of only two similar tasks (t_1 and t_2). In this section, a comparative example between STL and regularized MTL considers such scenario, where the goal of each task is fit a line aiming at modeling the linear relationship between the output y and the input x. Moreover, in order to provide a visual motivation of the results, each observation in the employed datasets is described by only one feature, that is, they are simple linear regression tasks. Hence, the resulting figures can be depicted in two dimensions.

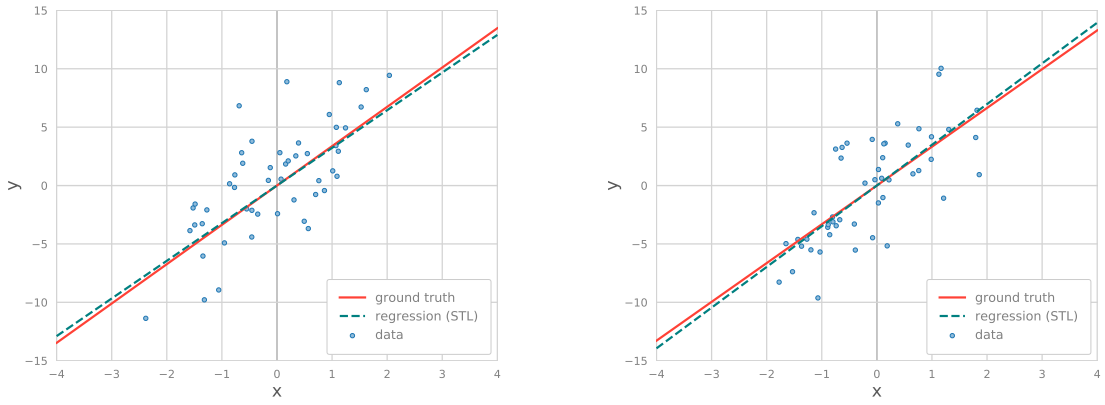
Each dataset employed in this example has a population of 1000 data points randomly generated by $\mathbf{x} \sim \mathcal{N}(0, 1)$ and $\mathbf{y} = \mathbf{x}\mathrm{w} + \mathcal{N}(0, 3)$, where $\mathcal{N}$ is the normal distribution, and the ground truth parameters for t_1 and t_2 are the values $\mathrm{w}_1 = 3.368$ and $\mathrm{w}_2 = 3.321$, respectively. Notice that w_1 and w_2 are similar values, making the tasks properly fitting to be solved by MTL approaches. Since in this example the tasks are going to be solved by regularized MTL, the estimated parameters are going to be encouraged to be similar, having their deviations penalized. Finally, the parameters represent the slope of the regression line, whereas the intercept is fixed to zero.

In Figures 2.3(a) and 2.3(b), both population (1000 points) and ground truth line are shown. From the population, 50 data points were randomly sampled to compose the training set for each task, representing 5% of all data. Thus, after performing a training on these 50 data points, the results for each task are depicted in Figures 2.4(a) and 2.4(b), representing the training via STL, and Figures 2.5(a) and 2.5(b), representing the training via regularized MTL. Notice that the lines obtained by the regularized MTL is closer to the ground truth lines than the ones obtained by STL. Finally, the more similar tasks compound the problem, the higher is the advantage of MTL over STL. Conversely, the easier the problem becomes, the more equivalent STL and MTL tends to be.



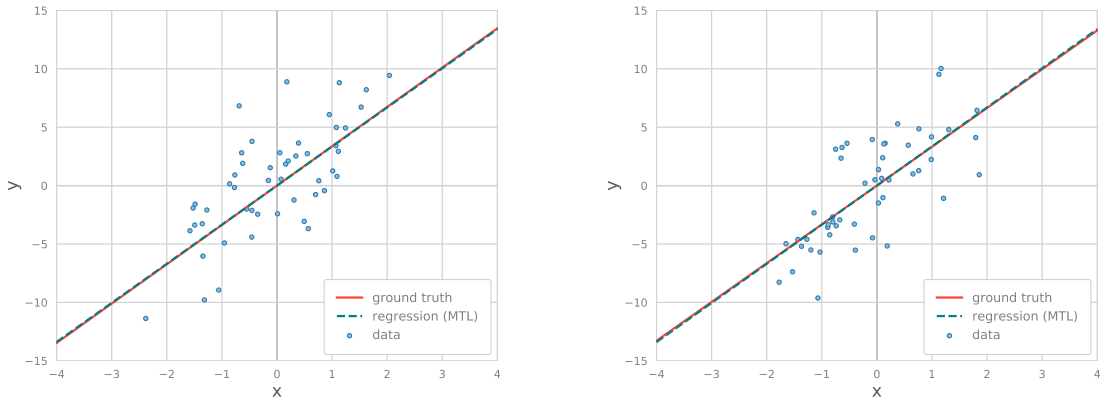
(a) Ground truth slope: 3.368 (t_1) (b) Ground truth slope: 3.321 (t_2)

Figure 2.3 – Comparison between STL and regularized MTL: the ground truth. (a) and (b) show data points and the ground truth line for t_1 and t_2, respectively.



(a) Slope via STL: 3.224 (t_1) (b) Slope via STL: 3.486 (t_2)

Figure 2.4 – Comparison between STL and regularized MTL: STL results. (a) and (b) show limited training data points and the resulting linear regression lines via STL for t_1 and t_2, respectively.



(a) Slope via regularized MTL: 3.347 (t_1) (b) Slope via regularized MTL: 3.350 (t_2)

Figure 2.5 – Comparison between STL and regularized MTL: MTL results. (a) and (b) show limited training data points and the resulting linear regression lines via regularized MTL for t_1 and t_2, respectively. The lines obtained via regularized MTL are closer to the ground truth than lines obtained via STL.

2.4 Main approaches

A common view among the wide number of approaches that have been proposed in the MTL literature is devoted to task relationship. In this manner, the following subsections describe the main approaches, distinguishing the ones that assume that all tasks are related and the ones that assume that task relationship is encoded in some sort of structure (clustered, graph-based and hierarchical). Clearly, the MTL literature comprises a large number of proposals, so the intention with this section is to provide a general view of the main approaches in order to identify research opportunities.

2.4.1 All tasks are related

One of the earliest models introduced in the MTL literature was proposed by Caruana (1993), where task relationship is defined in neural network hidden units and each task corresponds to an output in the output layer. Thus, tasks using the same features in the hidden layer are considered similar. Succeeding and highly popular in STL, regularization–based methods was first extended to the context of MTL by Evgeniou & Pontil (2004), assuming the existence of underlying task parameter vector shared across tasks, where a regularization penalty is imposed on task parameter vectors, so that all of them are constrained to be close to their feature-by-feature average. Thereafter, Argyriou *et al.* (2006) proposed an approach assuming that tasks share a common set of features, where a group sparsity penalty is imposed on task parameter vectors. Furthermore, Abernethy *et al.* (2006) and Ji & Ye (2009) considered imposing a trace norm on task parameter vectors, assuming they share a low-dimensional subspace.

Approaches presented so far do not consider the case where outlier tasks are present, once they assume either all tasks are close to each other or tasks share a common structure. Consequently, negative transfer from an outlier task might easily degrade generalization performance (XU *et al.*, 2015). Aiming at addressing this issue, the following regularized MTL approaches have been proposed, where the regularization penalty imposed on task parameter vectors is decomposed in two components:

- Chen *et al.* (2011) proposed a robust MTL formulation, decomposing task parameter vectors in a component that identifies the task relationship using a low-rank structure (similar to Ji & Ye (2009)) and another component that identifies outlier tasks using a group-sparse structure.

- Gong *et al.* (2012) also proposed a robust MTL formulation for feature learning, decomposing task parameter vectors in a component that captures common features among relevant tasks (similar to Argyriou *et al.* (2006)) and another component that identifies outlier tasks;

Nevertheless, despite all mentioned approaches consider that tasks are somehow related, most advanced approaches relax this constraint, as will be seen in the next sections.

2.4.2 Clustered structure

Several approaches are based on the idea that relationship among tasks follows a clustered structure, so that knowledge transfer is allowed only among tasks within the same cluster. To begin with, Evgeniou *et al.* (2005) showed different approaches on how to incorporate kernel methods and regularization into MTL, and one of such approaches included a task clustering regularization. The initial idea that all task parameter vectors were assumed to be close to their average was extended considering only tasks within the same group, though task membership needed to be known a priori.

In order to avoid the necessity of knowing the task membership, a number of approaches have been proposed to learn a clustered structure directly from data. In MTL early stages, Thrun & O'Sullivan (1996) proposed a methodology to learn task clusters by using a pairwise relationship metric among tasks, where distances are measured based on how well a task model performs when applied to another task. Relying in Bayesian hierarchical models, Bakker & Heskes (2003) applied an approach in neural networks where a mixture distribution is considered rather than a single Gaussian prior, meaning that each Gaussian in the mixture represents a cluster, though the number of mixtures needs to be informed a priori. On the other hand, Xue *et al.* (2007) proposed an approach that automatically identifies the task structure without knowing beforehand the number of clusters, where task similarities are learned based on Dirichlet process.

Jacob *et al.* (2008), in turn, proposed a convex formulation considering that task membership is not known a priori and parameters of tasks in the same cluster are constrained to be similar to each another. Kang *et al.* (2011) and Zhou *et al.* (2011a) also considered this scenario, but applied alternating optimization to simultaneously learn task parameter vectors and perform task clustering. Recently, Xu *et al.* (2015) proposed an approach that assumes a co-clustered structure, arguing that most proposals are restricted by the fact that tasks within the same cluster are close in all features, which might not be always the case. Zhong & Kwok (2012) also approached this scenario in their work, assuming that clusters can vary from feature to feature, but Xu *et al.* (2015) pointed out that feature relationship are completely neglected in this case. On the other hand, the co-clustered-based formulation of Xu *et al.* (2015) is capable of capturing both global similarities among tasks and group-specific similarities, then considering feature relationship.

2.4.3 Graph-based structure

Although assuming a clustered structure seems attractive, once information is not transferred among different clusters, tasks within the same cluster are constrained to have similar parameter vectors, which may not hold in all the cases. Such scenario can be better modeled using a graph-based structure, where weighted edges between tasks can quantify how strong their relations are and, with densely connected subgraphs, even clusters of tasks can be represented (ARGYRIOU *et al.*, 2013).

Given a graph of relations among features, Li & Li (2008) proposed a STL formulation that both performs feature selection and penalizes deviations in a genetic graph structure which employs the Laplacian regularization term. In Zhou *et al.* (2011b), the Laplacian regularization was brought up to the context of MTL, where a graph models relations among task parameter vectors rather than features. A limiting factor on the previously mentioned approaches is that those formulations assume the knowledge of the graph structure beforehand, information not usually available.

In order to address the aforementioned limitation, some efforts have been made in the sense of modeling the task relationship as a graph via alternating optimization, where such structure is learned directly from data alongside task parameter vectors. In this manner, Argyriou *et al.* (2013) considered the case of jointly learning task parameter vectors and their graph Laplacian. Recently, Gonçalves *et al.* (2014) combined MTL and advances in structure learning of probabilistic graphical models, proposing an approach to learn both task parameter vectors and their sparse inverse covariance matrix, which represents a graph of relations among tasks.

2.4.4 Hierarchical structure

A number of approaches in the MTL literature have also taken into account a hierarchical task structure among tasks, comprising cases similar to graph-based, where the structure can either be known a priori or not. Widmer *et al.* (2010) considered the case where tasks are leaves in a given hierarchical structure described by a phylogenetic relationship and proposed two approaches to explore such hierarchy. In the former, a top-down approach is employed by individually learning task parameter vectors for all nodes, where each node is composed of data union from tasks below in the hierarchy and a regularization penalty is imposed, constraining the task parameter vector from a node to be similar to the parameter vector of its parent node. In the latter, all task parameter vectors are simultaneously learned and a regularization penalty is imposed on task parameter vectors via a closeness measure derived from the hierarchy, which is similar to treating the structure as a graph with edge weights calculated by the mentioned closeness measure.

Recently, the idea of decomposing task parameter vectors in two components (as mentioned earlier in the case where outlier tasks are present) was extended, considering multiple components rather than only two. In Han *et al.* (2014), a probabilistic tree sparsity model alongside a product decomposition is employed assuming a given tree structure, where each node corresponds to a component and the product from a leaf to the tree root is the sought parameters for its associated task. In contrast, a sum-based decomposition in a multi-level structure was considered in Han & Zhang (2015), where each level groups tasks via multiple regularization terms in a descending order, that is, the lower the level, the more the number of task groups, constraining task parameter vectors within the same group to be similar with each other.

In a similar direction regarding levels of sharing, Zweig & Weinshall (2013) considered a top-down iterative feature selection structure, where the incentive to share information among tasks is gradually decreased by using a hierarchy of regularization functions. Unlike Han *et al.* (2014), both approaches from Han & Zhang (2015) and Zweig & Weinshall (2013) do not depend upon knowing any structure a priori, though the number of levels does need to be optimized.

2.5 Applications

An important and natural contribution that all researches in the area of MTL have brought is a variety of case studies and insights for applications. Pertaining to the interdisciplinary machine learning field, MTL has been applied in different domains, including bioinformatics, robotics, computer vision, meteorology and medicine. Hence, an overview of some applications is provided as follows.

When training neural networks, Caruana (1993) justified the usage of MTL

by presenting two use cases in computer vision, where the goal is to learn main tasks aided by a number of auxiliary tasks. In the former use case, the main task aimed at predicting steering direction using road images, while simultaneously learning 8 auxiliary tasks, for example: road center location, road surface intensity, and whether the road has one or two lanes. In the latter use case, the main tasks aimed at locating doorknobs and recognizing door types using door images collected by a robot, while simultaneously learning 10 auxiliary tasks, for example: doorway width, right edge horizontal location, and whether the door is single or double. In both cases, MTL outperformed tasks trained individually.

Another example is a real-world problem, whose data have been employed as a benchmark dataset in publications such as Chen *et al.* (2011), Zhou *et al.* (2011b) and Xu *et al.* (2015). The problem consists of predicting exam scores of students that are distributed in many secondary schools. Bringing up to the context of MTL, each school can be seen as one task and each student is a specific observation described by a number of demographic attributes. This configuration motivates the usage of MTL when dealing with data composed of entities that are divided into groups.

MTL has also been applied to predict disease progression. Zhou *et al.* (2011c) brought MTL and Alzheimer's disease progression together by considering cognitive scores prediction at each time point as a task, where each task is related to its neighboring time points. Extensive experimental studies were performed and results showed that the proposed MTL approach could capture the progression trends better than existing methods.

In the context of bioinformatics, Widmer *et al.* (2010) considered the problem of splice site prediction across different organisms. In this setting, each organism is a task and their relations are derived from a given beforehand phylogenetic tree. Thus, a model quality improvement is assumed when using information from such tree, since tasks with sufficient evolutionary distance are expected to help improving one another. In other words, the closer the organisms are in the tree, the more information exchange is expected among them. Results demonstrated that MTL outperformed baseline methods with large margin, suggesting MTL as a potential tool to be used with known biological structures.

Lastly, the problem of temperature prediction considering global climate model outputs is approached by Gonçalves *et al.* (2014) as part of the performance evaluation of their proposed MTL framework. In this setting, each geographical location is a task and their relations are encoded in a sparse graph, which is jointly learned with task parameter vectors. It is worth mentioning that the method is not influenced by the geographical neighborhood among locations, in the sense that the geographical coordinates are not provided as input information. Results confirmed better performance of the proposed approach against baseline methods, and also showed informative correlations among locations captured by the graph.

2.6 Chapter summary

This chapter presented an overview regarding Multi-Task Learning (MTL), pointing out distinct task relationship views, the trend involving MTL and the regularization framework, and case studies employing MTL. Due to the complexity of real world problems, learning a general model considering all training data within a single task or learning individual models without taking into account task similarities is not attractive, mainly when limited training data is available and the tasks exhibit some sort of structure. Besides, the task relationship must be carefully identified (preferably in an automatic manner) rather than assuming that all tasks are related, avoiding the occurrence of negative transfer of information among tasks. Finally, the usage of a given hierarchical structure by Widmer *et al.* (2010) motivates a subsequent research on letting the machine uncover the task hierarchical structure, which is going to be the main contribution of this dissertation.

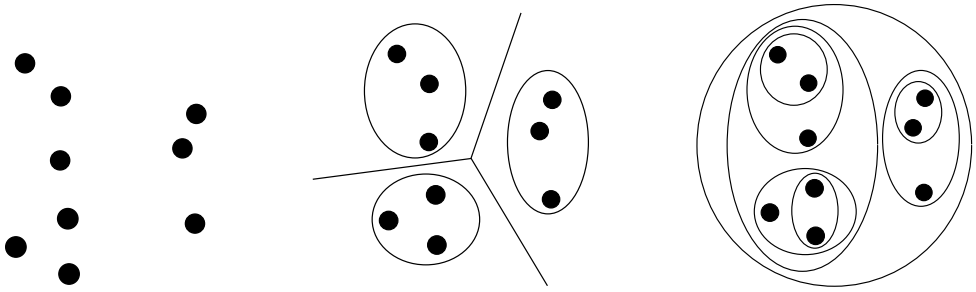
Chapter 3

Hierarchical clustering

3.1 Overview

Chapter 2 discussed the area of MTL focusing on supervised learning tasks, though a limited number of MTL proposals consider addressing unsupervised learning tasks as well. Despite this, unsupervised learning in general context has also been extensively studied over the past decades, resulting in the proposal of several techniques to model the relationship among objects without relying on labeled data, such as association rules, cluster analysis, self-organizing maps, dimensionality reduction, among others (FRIEDMAN *et al.*, 2001). In addition, Section 2.4.2 has shown that unsupervised learning steps may be incorporated into the context of MTL, for instance by clustering similar tasks and letting the occurrence of information exchange only within the cluster.

Specifically dealing with the task of clustering objects in general context, the goal for such tasks is to group similar objects, so that sufficiently distinct objects remain in separate groups. Due to a subjective notion of the organization of objects into clusters, several approaches have been proposed in the clustering literature, basically denoted as partitional and hierarchical algorithms (JAIN *et al.*, 1999). In Figure 3.1, a comparison is shown between partitional and hierarchical clustering views considering the same set of objects.



(a) Dataset (b) Partitional (c) Hierarchical

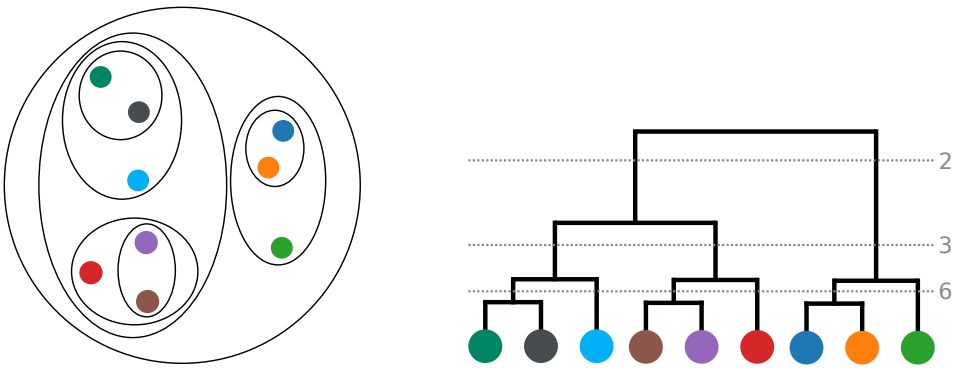
Figure 3.1 – Comparison between partitional and hierarchical clustering views.

Additionally, simpler approaches like the k-means algorithm require the a priori definition of the number of clusters, whereas advanced approaches based on a Dirichlet

Process (infinite mixture) obtains it from data. Related to the research detailed in this dissertation, the class of hierarchical clustering is discussed with more attention and, for a survey on clustering algorithms, the reader is invited to refer to Xu & Wunsch II (2005), Berkhin (2006) and references therein.

In essence, hierarchical clustering algorithms can be classified into divisive (top-down) and agglomerative (bottom-up) strategies. Both strategies provide a tree structure as the output, often referred as *dendrogram*, where the leaves are the original objects to be clustered and internal nodes represent clusters of leaves directly below in the hierarchy. In the divisive approach, all objects start in the same cluster and a divisive strategy is recursively applied, progressively forming sub-clusters down in the hierarchy until each object ends up on its own cluster or a given number of levels is reached. In the agglomerative approach, in turn, each object starts in its own cluster and an agglomerative strategy is recursively applied, progressively forming larger clusters up in the hierarchy until all objects end up in the same cluster or a given number of clusters is reached (MAIMON; ROKACH, 2010). In this chapter, the focus is on the agglomerative approach, which is more frequently studied in the information retrieval area (MANNING *et al.*, 2008).

An attractive advantage of hierarchical clustering algorithms over partitional clustering algorithms is the interpretability that the resulting tree provides. In other words, one can perform analyses taking into account all different levels of granularity exhibited by the nested clusters (BERKHIN, 2006), as exemplified in Figure 3.2.



(a) Nested clusters (b) Hierarchical representation

Figure 3.2 – Hierarchical representation of a set of objects showing different levels of granularity exhibited by the nested clusters. In (b), one can consider having 2, 3 or even 6 plausible clusters.

Although the number of clusters does not need to be specified a priori, inferring it from a hierarchical structure might not be an easy task. Usually, some cut strategy is applied in order to derive clusters from a hierarchical structure, such as cutting at a determined level of similarity, cutting when a large gap between two successive levels is identified or, as pointed out earlier, stop the algorithm when a desired number of clusters

is reached (MANNING *et al.*, 2008).

One downside of hierarchical clustering algorithms, however, is that such strategies tend to be prohibitive when dealing with a large number of objects due to the high cost of building the tree, even restricted to binary trees, which makes partitional algorithms more interesting to be employed (DUDA *et al.*, 1995; JAIN *et al.*, 1999). Moreover, the usage of an incorrect similarity strategy or the presence of noise may contribute to generate uninformative results (KARYPIS *et al.*, 1999). Lastly, a disadvantage exhibited for most hierarchical clustering algorithms is the lack of structure revision during its construction, because node relations are never revisited once created (BERKHIN, 2006).

Aiming at exemplifying the ease of interpretation provided by a hierarchical structure, Figure 3.3 shows undergraduate courses hierarchically organized, where internal nodes are properly labeled as major clusters. Depending on the problem, one could be interested in obtaining clusters by area, which results in three clusters, or by field, to be even more specific. Clearly, the internal nodes represent obvious groups in this example, but real world problems can admit latent grouping levels that only hierarchical clustering algorithms are able to identify. Furthermore, the tree presented in the example is not binary but, since the hypothesis space for binary trees are easier to deal with, several algorithms provide a binary tree as output, though some efforts have been made to relax this constraint.

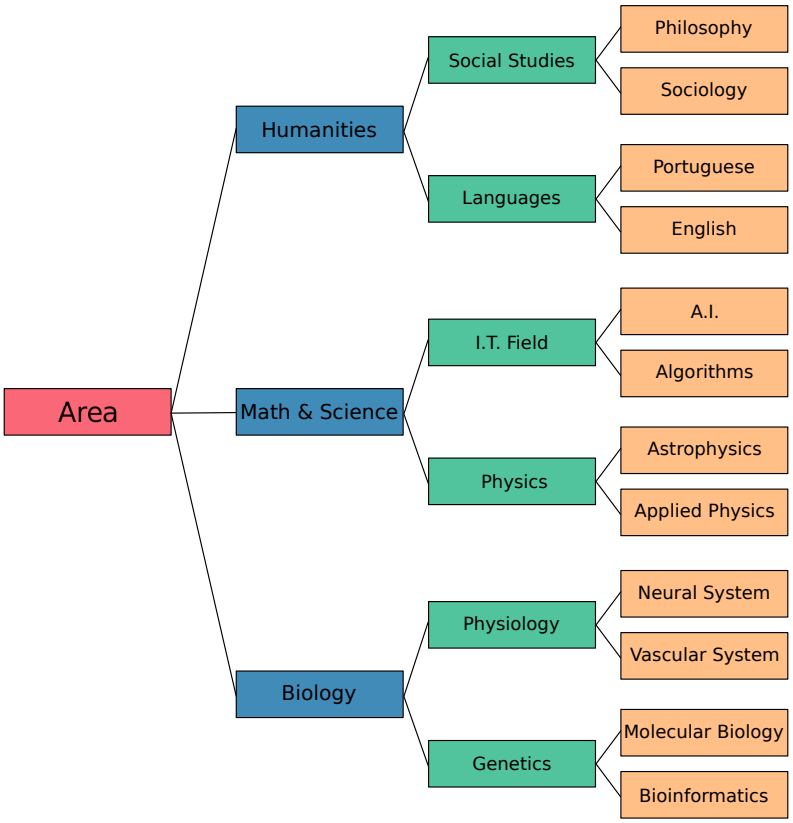


Figure 3.3 – Example of a hierarchical organization of a number of undergraduate courses.

3.2 Agglomerative hierarchical clustering: a binary structure perspective

Treating the problem more formally, let $\mathcal{D} = \{\mathbf{x}_i : i = 1, \ldots, n\}$ be the set of n objects, where each object is described by d features. The goal is to organize the objects in a binary tree structure $\mathcal{T}$, such that leaf nodes in $\mathcal{T}$ represent the objects in $\mathcal{D}$ and internal nodes represent clusters of leaves directly below in the hierarchy. In other words, the i-th internal node can be viewed as a sub tree $\mathcal{T}_i$ composed of $\mathcal{D}_i \subseteq \mathcal{D}$ objects. In addition, a similarity strategy $\mathcal{S}$ must be defined in order to identify the clustering levels. Finally, since each object starts in its own singleton cluster when initializing the algorithm, they also represent n trivial trees $\{\mathcal{T}_i : i = 1, \ldots, n\}$ and, via the similarity metric $\mathcal{S}$, the algorithm progressively identifies the most similar tree pair $(\mathcal{T}_i, \mathcal{T}_j)$ and merges them forming a new internal node $\mathcal{T}_k$, which is composed of the objects $\mathcal{D}_k = \mathcal{D}_i \cup \mathcal{D}_j$. Algorithm 3.1 shows the general procedure for agglomerative hierarchical clustering.

Algorithm 3.1: General procedure of agglomerative hierarchical clustering

Input : A set $\mathcal{D}$ of objects to be clustered; a similarity metric $\mathcal{S}$.
Output : A tree structure $\mathcal{T}$ representing the hierarchical organization of the objects.

begin
  Initialize n trivial trees. Each T_i is composed of $D_i = \{\mathbf{x}_i\}, i = 1, \ldots, n$.
  Initialize $\mathcal{L} \leftarrow \{1, \ldots, n\}$ as labels for available trees to be merged.
  Initialize $k \leftarrow n + 1$ to be the label for the next merged sub tree.
  while $|\,\mathcal{L}\,| > 1$ **do**
    Find the closest tree pair (T_i, T_j) using the similarity metric $\mathcal{S}$, $\{i, j\} \subset \mathcal{L}$.
    Create a new tree node $T_k \leftarrow (T_i, T_j)$ composed of $D_k \leftarrow D_i \cup D_j$.
    Update the labels: $\mathcal{L} \leftarrow \mathcal{L} + \{k\} - \{i, j\}$.
    Update the next label: $k \leftarrow k + 1$.
  end
end

Classic agglomerative hierarchical clustering algorithms include the so-called and widely used linkage methods. Such methods vary in the employed similarity metric, usually distance-based, including Euclidean, Manhattan or cosine, to name a few. The most popular linkage methods are single linkage, complete linkage and average linkage, which consider similar objects by minimum, maximum and average distance, respectively (BERKHIN, 2006). Hence, the selected linkage method is applied as the similarity metric $\mathcal{S}$ in Algorithm 3.1, so that the most similar object pair is selected to be merged at each iteration.

However, the following drawbacks associated with linkage methods are discussed by Heller & Ghahramani (2005a) in order to motivate the proposal of their probabilistic approach. Firstly, it would be necessary to rerun the algorithm when a new object is

available in order to recalculate the distances, once a distance-based similarity metric is employed. Secondly, choosing a distance metric among many options is often hard, especially for structured data, such as images and gene expression profiles. Lastly, linkage methods are not capable of identifying the number of clusters by themselves. Often, it depends on either subjective assumptions by visual identification or arbitrary strategies, such as pruning the tree at a determined distance level.

3.3 Bayesian hierarchical clustering (BHC)

Motivated by linkage limitations, Heller & Ghahramani (2005a) proposed a Bayesian-based approach named Bayesian Hierarchical Clustering (BHC) that, basically, differs from linkage methods in the similarity metric adopted. They argued that such limitations would be addressed by defining a probabilistic model of the data, enabling the incorporation of a new object by computing its predictive distribution without rerunning the algorithm, and also allowing model comparisons with other probabilistic models. Moreover, measuring similarity among objects via statistical tests could identify objects that are most likely part of the same cluster, information ignored by a distance-based similarity metric. Finally, pruning the tree in order to identify the number of clusters would be enabled once verified that objects in the merge operation are more likely to belong to different clusters. Further details regarding the algorithm are provided in Section 3.3.2.

3.3.1 Motivation

A number of interesting examples comparing average-linkage and BHC are provided by Heller & Ghahramani (2005a) in their work, and a particular one was extended in this dissertation in order to motivate the usage of BHC. Basically, a comparison of the generated dendrograms is made among the techniques single-linkage, complete-linkage, average-linkage and BHC.

In Figure 3.4, a plot of the employed dataset is shown, which was obtained from the BHC authors' homepage[1]. In this example, assume all data cross-like arranged in two dimensions and the observed data being represented by those 18 points labeled from 0 to 17. The goal is to identify the most informative dendrogram among the mentioned techniques, the one that best describes the existence of two major clusters composed of labels $[0, 9]$ and $[10, 17]$.

Figure 3.5 shows dendrograms obtained from linkage-based techniques. Overall, none of them were able to capture two major clusters, showing that employing a distance-based similarity metric fails in scenarios represented by the example. Clearly, the object labeled as 17 always appears closer to the group $[0, 9]$. Assuming an a priori radial

[1] http://www.gatsby.ucl.ac.uk/∼heller/code/bhc

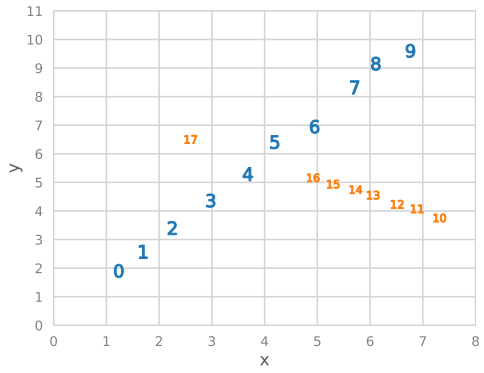
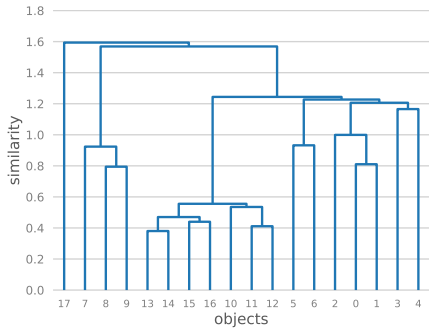
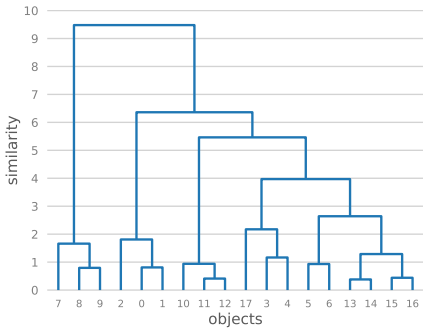


Figure 3.4 – Example dataset for clustering obtained from BHC author's homepage. The objects to be clustered are represented by numbered labels. Consider the existence of two major clusters composed of labels $[0, 9]$ and $[10, 17]$.

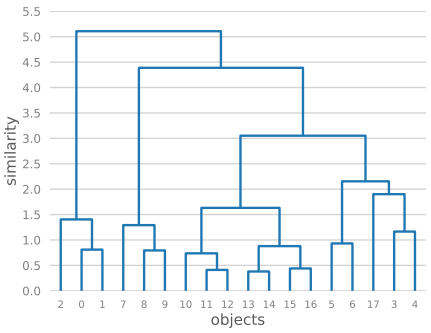
distribution for the data, the effective distribution is never taken into account in a distance-based similarity metric, motivating the usage of a probabilistic-based similarity metric. Although one could easily identify the clusters by just looking to Figure 3.4, practical applications often are composed of a higher number of dimensions and a more complex data distribution, so that visual inspection is no more feasible.



(a) Single linkage



(b) Complete linkage



(c) Average linkage

Figure 3.5 – Hierarchical structures obtained from the example dataset via different linkage strategies. None of them were able to obtain a structure indicating the presence of 2 major clusters.

In Figure 3.6, on the other hand, two clusters can be easily identified after applying the BHC algorithm, which properly represents the expected structure. In this case, object 17 follows the data distribution and appears within the group $[10, 17]$. Moreover, the BHC algorithm would suggest to prune the last node in the structure so that the two clusters be completely independent from each other. Therefore, this example motivates the usage of BHC instead of distance-based approaches, since BHC is capable of properly identifying aspects of the problem.

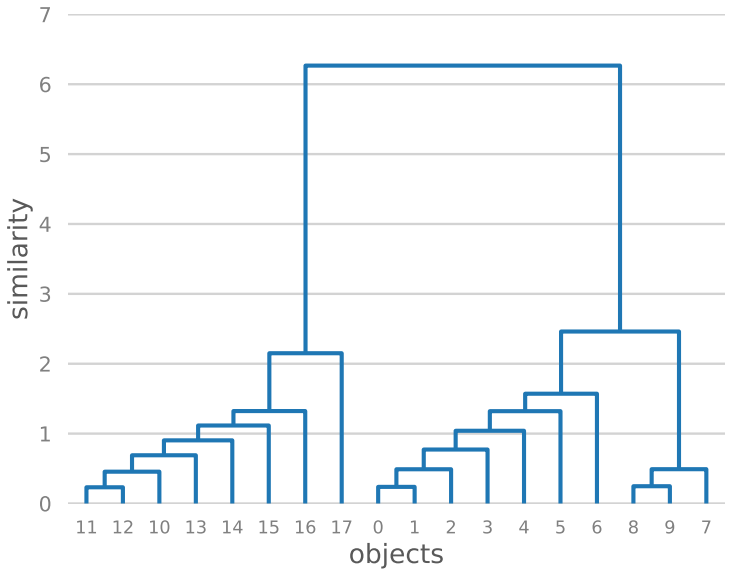


Figure 3.6 – Hierarchical structure obtained from the example dataset via Bayesian Hierarchical Clustering. The obtained hierarchical structure clearly indicates the presence of 2 major clusters. This example is based on Heller & Ghahramani (2005a).

3.3.2 Overview of the BHC algorithm

This section is intended to present an overview of the BHC algorithm, since it is directly related to the research detailed in this dissertation. However, for a comprehensive explanation of the theory behind the BHC proposal, the reader is invited to refer to the original work described in Heller & Ghahramani (2005a).

Essentially, BHC assumes that data to be clustered are generated from a Dirichlet Process Mixture Model (DPMM), which considers an infinite number of components instead of assuming that data are generated from a finite number of components. In practice, clustering through DPMM is more attractive and it has been widely studied, since it does not require the a priori specification of the number of clusters. Moreover, defining beforehand in how many clusters a particular dataset will be divided is often difficult when dealing with real world applications. As presented in Section 2.4.2 related

to MTL, Xue *et al.* (2007) employed a similar strategy to learn the clustered structure among tasks, though a hierarchical structure was not considered by them.

Focusing on the algorithm itself, two intuitive hypotheses are tested for every candidate sub tree pair $(\mathcal{T}_i, \mathcal{T}_j)$ to be merged as $\mathcal{T}_k$, which basically functions as a similarity metric for the agglomerative hierarchical clustering procedure presented in Algorithm 3.1. The first hypothesis, denoted as $\mathcal{H}_1^k$, indicates that data in $\mathcal{D}_k$ come from the same mixture, that is, data were generated from the same probabilistic model $p(x|\boldsymbol{\theta})$ with unknown parameters $\boldsymbol{\theta}$. The alternative hypothesis, denoted as $\mathcal{H}_2^k$, indicates that data come from two or more mixtures.

In order to evaluate the probability of $\mathcal{H}_1^k$, a prior probability $p(\boldsymbol{\theta}|\boldsymbol{\beta})$ over the unknown parameters $\boldsymbol{\theta}$ with hyper-parameters $\boldsymbol{\beta}$ must be specified. Assuming a probabilistic model $p(x|\boldsymbol{\theta})$, the probability of the data $\mathcal{D}_k$ under $\mathcal{H}_1^k$ can be computed as follows:

$$p(\mathcal{D}_k|\mathcal{H}_1^k) = \int p(\mathcal{D}_k|\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\boldsymbol{\beta})\, d\boldsymbol{\theta} \tag{3.1}$$

Depending on the dataset $\mathcal{D}$, distinct distributions for $p(x|\boldsymbol{\theta})$ can be employed, such as multinomial distribution and Gaussian distribution for categorical and continuous data, respectively. In addition, the integral in Equation (3.1) is tractable if $p(x|\boldsymbol{\theta})$ is chosen with a conjugate prior, for instance, the Normal-Inverse-Wishart prior when dealing with Gaussian data. Refer to Murphy (2007) for a thorough conjugate Bayesian analysis of the Gaussian distribution.

When dealing with the alternative hypothesis $\mathcal{H}_2^k$, which means that $\mathcal{D}_k$ come from two or more mixtures, considering all possibilities in order to divide the data into an unknown number of mixtures is certainly intractable. Instead, it can be recursively computed if the hypothesis space is limited to the already built structures in the sub trees $(\mathcal{T}_i, \mathcal{T}_j)$. Thus, the probability of the data $\mathcal{D}_k$ under $\mathcal{H}_2^k$ can be evaluated as follows:

$$p(\mathcal{D}_k|\mathcal{H}_2^k) = p(\mathcal{D}_i|\mathcal{T}_i)\, p(\mathcal{D}_j|\mathcal{T}_j) \tag{3.2}$$

where $p(\mathcal{D}_k|\mathcal{T}_k)$ is the marginal probability of the data $\mathcal{D}_k$ given the tree $\mathcal{T}_k$, recursively evaluated as shown in Equation (3.3). In addition, π_k is defined as the weight representing the prior $p(\mathcal{H}_1)$ for the merged hypothesis, which can be computed considering the DPMM concentration parameter α during the tree construction, as shown in Heller & Ghahramani (2005a).

$$p(\mathcal{D}_k|\mathcal{T}_k) = \pi_k p(\mathcal{D}_k|\mathcal{H}_1^k) + (1 - \pi_k)\, p(\mathcal{D}_k|\mathcal{H}_2^k) \tag{3.3}$$

Finally, the pursued posterior probability $p(\mathcal{H}_1^k|\mathcal{D}_k)$, which denotes the probability of the merged hypothesis given the data, is obtained through the Bayes rule and is defined as r_k, as shown in Equation (3.4). Consequently, the candidate tree pair $(\mathcal{T}_i, \mathcal{T}_j)$ which maximizes r_k is selected to be merged as the new node $\mathcal{T}_k$. Thus, r_k can be interpreted

as the similarity metric $\mathcal{S}$ for the agglomerative hierarchical clustering algorithm.

$$r_k = \frac{\pi_k p(\mathcal{D}_k|\mathcal{H}_1^k)}{p(\mathcal{D}_k|\mathcal{T}_k)} \tag{3.4}$$

In order to derive the number of clusters from the obtained tree, BHC authors suggest pruning the tree when $r_k < 50\%$, which makes sense, once the statistical test is indicating that objects from the best selected candidate pair $(\mathcal{T}_i, \mathcal{T}_j)$ are more likely to belong to different mixtures in this case. In fact, it can be used as a stopping criteria for the algorithm, since r_k is a monotonically decreasing function.

Regarding the hyper-parameters associated with BHC, which are the concentration parameter α from the mixture model and the prior hyper-parameters $\boldsymbol{\beta}$, BHC authors suggest that the marginal probability $p(\mathcal{D}|\mathcal{T})$ may be also used to optimize these hyper-parameters. In this sense, different hyper-parameter settings act on such probability values and may work as a function to be maximized in order to find the best hyper-parameters. Thus, a procedure similar to expectation-maximization (EM) can be performed during the construction of the tree, where the best tree is obtained using the current hyper-parameters and then the best hyper-parameters are obtained for the current tree, and so on.

In summary, the BHC algorithm is quite similar to the classic agglomerative hierarchical clustering algorithm, especially for being greedy and achieving time complexity of order $O(n^2)$. The only difference, though, is that BHC evaluates marginal likelihoods of a probabilistic model in order to find the best tree pair to merge, instead of using a distance-based similarity metric. Furthermore, as stated by BHC authors, the BHC algorithm can also be interpreted as a fast bottom-up approximate inference method for a DPMM, functioning as an alternative to Markov chain Monte Carlo approximations.

3.3.3 Applications

Heller & Ghahramani (2005a) provided experimental results considering four real problems originally for classification and compared with classic agglomerative hierarchical clustering algorithms, such as single, complete and average linkage. The datasets are composed of data for classifying e-mail, glass, digit and text. For the majority of the tested datasets, BHC outperformed all classic algorithms.

In addition, a small number of applications employing BHC have been found in the literature, mainly focused on microarray gene expression data analysis. Essentially, such analysis considers clustering patterns of expressions of different genes. The expected result is a set of clusters, each one grouping genes in terms of related functionality, allowing the identification of genes influenced by a common factor. Usually, classic agglomerative hierarchical clustering is employed, but its earlier discussed disadvantages motivated the usage of BHC on this context (SAVAGE *et al.*, 2009; COOKE *et al.*, 2011; SIRINUKUNWATTANA

et al., 2013).

Savage *et al.* (2009) implemented an R/Bioconductor port for general usage of BHC and applied it in gene expression clustering, considering single time point microarray observations. As an extension, Cooke *et al.* (2011) employed BHC in a more challenging setting composed of microarray time series with replicates and outlier measurements. Recently, Sirinukunwattana *et al.* (2013) applied BHC for studying cancer gene expression data. All of these studies are composed of extensible experiments showing the potential of BHC compared against classic agglomerative hierarchical clustering algorithms. Finally, the promising but scarce number of conducted studies emphasizes that BHC needs to be further explored in other contexts.

3.3.4 Extensions

Contrasting with the powerful advantages that BHC provides and being a frequent issue to be addressed in agglomerative hierarchical clustering algorithms, BHC disadvantage in terms of time complexity is due to the necessity of pairwise comparisons among all objects, especially in lower levels of the tree, making BHC applicability not interesting to datasets composed of a large number of objects. Thus, BHC authors extended their original approach aiming at allowing BHC to be ran on very large datasets.

Therefore, by taking advantage of randomized algorithms, Heller & Ghahramani (2005b) proposed two strategies to accelerate BHC so that the time complexity is improved, allowing their usage on large datasets. Similarly to the original BHC, both proposed algorithms take the object set as input and return a tree $\mathcal{T}$ as output, where each node in $\mathcal{T}$ is a Bayesian mixture component. As result, the proposed algorithms achieve time complexity of order $O(n\,log(n))$ and $O(n)$.

Referred as Randomized Bayesian Hierarchical Clustering (RBHC), the first proposed algorithm applies BHC on a randomly chosen subset of objects rather than all objects, resulting in a tree structure $\mathcal{T}'$ that describes the subset of objects. Thus, $\mathcal{T}'$ is assumed to be a good approximation of the sought tree $\mathcal{T}$. Afterwards, the remaining objects are recursively filtered through $\mathcal{T}'$, that is, the probability of each object belonging to the left and right sub trees are computed and then each object is added into the sub tree with the highest probability. In addition, if the number of levels L is limited to a value that not necessarily makes the algorithm reach the point where each object is in their own cluster, the time complexity reduces to $O(nL)$.

The second proposed RBHC extension is based on the EM algorithm and referred to as EMBHC. Firstly, objects of a randomly chosen subset are assigned as unit clusters and the remaining objects are randomly put into those clusters. Then, the cluster assignments are improved by running k steps of the EM algorithm. Finally, BHC is applied to cluster the clusters, resulting in a tree structure where the leaves are composed of the

clusters obtained in the EM step.

Despite proposing the randomized extensions and discussing their time complexity, Heller & Ghahramani (2005b) did not show any experiments aiming at comparing RBHC and BHC. Recently, Darkins *et al.* (2013) applied the first mentioned RBHC algorithm for a large microarray time series dataset. In their work, several experiments are discussed, showing a substantial improvement of RBHC over the original BHC in terms of run-time at a small cost to the final clustering performance, which motivates further research regarding RBHC applied to large datasets.

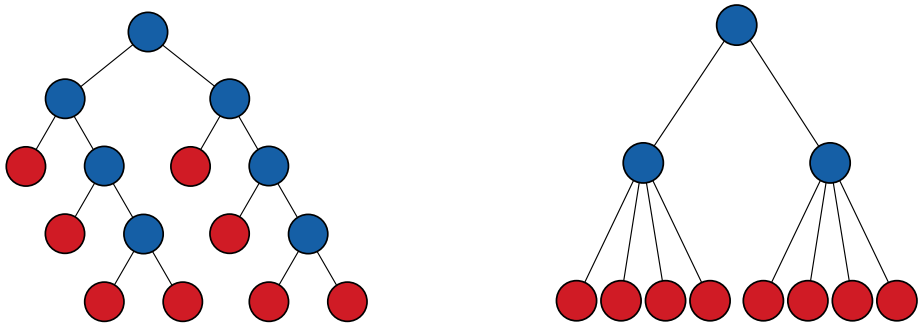
3.4 Bayesian rose trees (BRT)

Previous section showed BHC as a powerful agglomerative hierarchical clustering algorithm that mainly leverages from probabilistic models in order to select objects most likely to belong to the same cluster. Although computationally convenient, the resulting tree is strictly binary, which may not properly describe the hierarchical relationship among objects, once spurious relations in the shape of cascades are sometimes imposed. Aiming at addressing this issue, Blundell *et al.* (2012) extended BHC to an efficient and flexible approach named Bayesian Rose Trees (BRT). Basically, BRT is a mixture model that has all features provided by BHC with the additional property of allowing objects to be hierarchically represented in an arbitrary branching structure (rose tree), having the binary structure as just one of the possible configurations.

3.4.1 Motivation

The promotion of non-binary structures is achieved by the logical principle of Occam's razor, given that the simplest representation of the data might be associated with a non-binary structure, avoiding unnecessary cascades (BLUNDELL *et al.*, 2012). When dealing with large datasets, for instance, the assumption of unnecessary spurious relations, such as the imposition of a binary structure, may contribute to make the obtained clustering structure more complex than it should be.

Figure 3.7 depicts a comparison example between a binary and a non-binary tree built from a simple problem composed of 8 objects that must be organized in two clusters. In the binary version depicted in Figure 3.7(a), 7 internal nodes were necessary to represent the data and the clusters ended up being organized as cascades. In Figure 3.7(b), on the other hand, only 3 internal nodes were necessary to represent the data in a non-binary tree, resulting in a much more informative structure when compared to the binary version.



(a) Binary tree (b) Rose tree

Figure 3.7 – Hierarchical structure of two object clusters modeled as (a) binary tree and (b) rose tree. Red nodes represent the original objects to be clustered, while blue nodes represent internal nodes. This example is an adaptation from Blundell *et al.* (2012).

3.4.2 Comparison between BHC and BRT

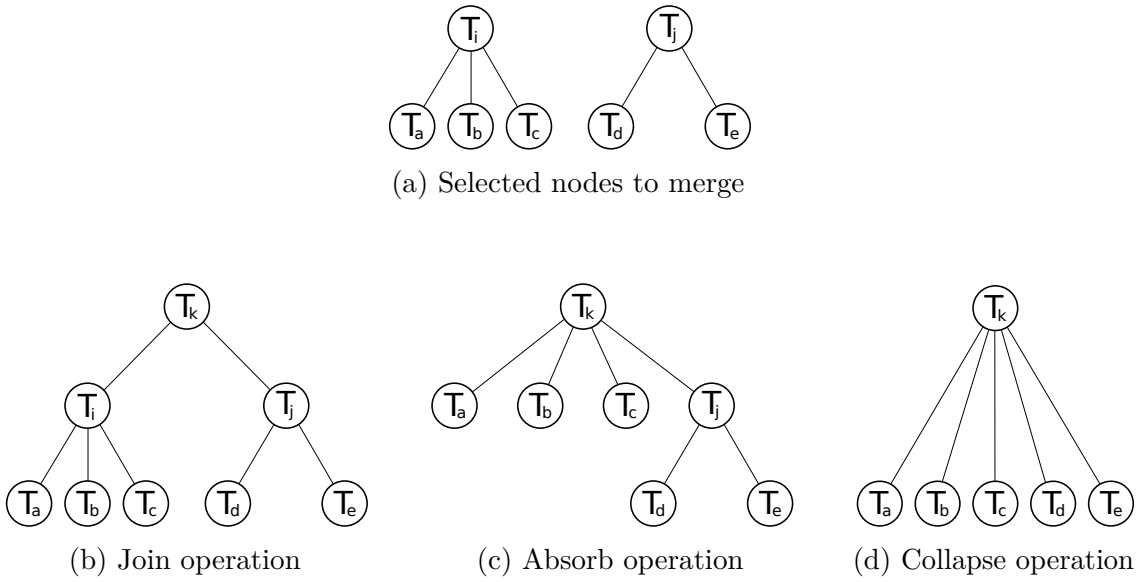
One of the main differences between BHC and BRT lies on the merge operation occurring at each iteration after identifying the best node pair $(\mathcal{T}_i, \mathcal{T}_j)$ to combine into $\mathcal{T}_k$. In the BHC algorithm, $(\mathcal{T}_i, \mathcal{T}_j)$ is simply combined as child nodes of $\mathcal{T}_k$, operation referred to as *Join*. In BRT, on the other hand, three operations are evaluated before performing the merge itself and the one yielding the best likelihood ratio between $p(\mathcal{D}_k|\mathcal{T}_k)$ and $p(\mathcal{D}_i|\mathcal{T}_i)\,p(\mathcal{D}_j|\mathcal{T}_j)$ is then selected. The operations are described as follows:

- **Join**: As in BHC, this operation just combines the nodes $(\mathcal{T}_i, \mathcal{T}_j)$ as child nodes of $\mathcal{T}_k$, meaning they are considered related but their individual structures must be kept untouchable. Figure 3.8(b) exemplifies this operation.

- **Absorb**: This operation considers incorporating $\mathcal{T}_j$ as a child node of $\mathcal{T}_i$, becoming $\mathcal{T}_k$ afterwards, meaning that the child nodes from $\mathcal{T}_i$ are considered similar to $\mathcal{T}_j$ but keeping untouchable the structure of $\mathcal{T}_j$ is better than collapsing it. Clearly, incorporating $\mathcal{T}_i$ as a child node of $\mathcal{T}_j$ is also evaluated. Figure 3.8(c) exemplifies this operation.

- **Collapse**: This operation removes the nodes $(\mathcal{T}_i, \mathcal{T}_j)$ and relates their child nodes to one single parent node $\mathcal{T}_k$, meaning that the child nodes are considered sufficiently indistinguishable. Figure 3.8(d) exemplifies this operation.

These operations offer the opportunity of performing local corrections in the hierarchical structure during its construction, aiming at improving the representation of the data and removing the binary restriction imposed on the resulting tree. Conversely, both BHC and classic hierarchical clustering algorithms do not allow such flexibility, that is, links between nodes can not be undone during the process. Notice that, regardless of the selected operation in BRT, the new node $\mathcal{T}_k$ is always composed of the same leaves



(a) Selected nodes to merge

(b) Join operation (c) Absorb operation (d) Collapse operation

Figure 3.8 – Bayesian Rose Trees merge operations. This figure is based on Blundell *et al.* (2012).

(*e.g.* leaf nodes T_a, T_b, T_c, T_d, T_e in Figure 3.8). BRT is thus a generalization of BHC, given that the binary structure is always considered as a candidate at each branching decision.

In addition, the probability of data $\mathcal{D}_k$ under $\mathcal{H}_2^k$, which is shown in Equation (3.2) for BHC and means that data come from different mixtures, is generalized in BRT by considering two or more child sub trees, resulting in:

$$p(\mathcal{D}_k|\mathcal{H}_2^k) = \prod_{\mathcal{T}_i \in ch(\mathcal{T}_k)} p(\mathcal{D}_i|\mathcal{T}_i) \tag{3.5}$$

where $ch(\mathcal{T}_k)$ denotes the set of child sub trees of the sub tree $\mathcal{T}_k$. In other words, Equation (3.5) covers binary trees, where $ch(\mathcal{T}_k)$ is composed of $\mathcal{T}_i$ and $\mathcal{T}_j$, and non-binary trees, where $ch(\mathcal{T}_k)$ is composed of more than two child sub trees. Thus, the marginal probability of the data $\mathcal{D}_k$ given the tree $\mathcal{T}_k$ can be also evaluated using Equation (3.3), but now considering Equation (3.5).

However, as discussed by the authors and unlike BHC, BRT can not be viewed as an approximate inference method for DPMM. Thus, the following proposal is suggested by the authors to compute the weight π_k in Equation (3.3):

$$\pi_k = 1 - (1 - \alpha)^{n_\mathcal{T} - 1} \tag{3.6}$$

where α is a hyper-parameter controlling the proportion of the partitions and $n_\mathcal{T}$ is the number of child nodes of $\mathcal{T}$.

Additionally, Blundell *et al.* (2012) employs the same EM-like procedure suggested by Heller & Ghahramani (2005a) in order to optimize BRT hyper-parameters, which are α, from Equation (3.6), and the prior hyper-parameters $\boldsymbol{\beta}$. However, they found that the hyper-parameters are sensible to initial conditions when dealing with binary data,

which motivates the application of other hyper-parameter optimization strategies.

As a final remark, BHC and BRT are greedy algorithms, which do not guarantee finding the optimal tree. Nonetheless, results have shown that they provide more informative structures than classic agglomerative hierarchical clustering algorithms. For a comprehensive explanation of the theory behind the BRT proposal as well as comparative experiments between BHC and BRT, the reader is invited to refer to the original work described in Blundell *et al.* (2012) and Mengersen *et al.* (2011).

3.5 Chapter summary

This chapter presented an overview of clustering methods focusing on hierarchical clustering, which is directly related to the proposal of Chapter 4. A discussion was provided involving characteristics, pros and cons of classic agglomerative hierarchical clustering algorithms. The advantages of Bayesian-based approaches are then highlighted, with the description of BHC and BRT algorithms. Only a few number of conducted studies were identified in the literature that employ BHC, most of them related to bioinformatics. It seems relevant to extend the application of BHC to other domains as well as the combination with other techniques. Finally, BRT may be interpreted as an extension of BHC to admit non-binary trees, which motivates its application in the context of MTL, as will be described in Chapter 4.

Part II

Proposals

Chapter 4

Proposals for data-driven hierarchical multi-task learning

As seen in Section 2.4, several MTL approaches assume some kind of structure among tasks and encode this additional information in regularization terms. Additionally, the MTL community initially explored the usage of a pre-defined structure and, then, considered the possibility of learning such structure directly from data. For instance, Evgeniou *et al.* (2005) showed that using a known clustered structure among tasks provided promising results and, afterwards, Xue *et al.* (2007) proposed an approach that jointly learns the task parameter vectors and their corresponding clustered structure.

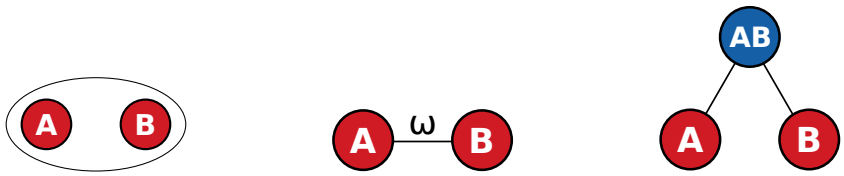
Following the trend of letting the machine responsible for providing the structure among tasks rather than depending on domain experts or specific a priori knowledge, a hierarchical MTL framework is proposed in this dissertation, which incorporates a hierarchical structure into the learning process via regularization. Moreover, once the structure is obtained, users may use such structural information on additional analysis of the problem.

4.1 Motivation

Differently from assuming a clustered structure, where parameter vectors of tasks within the same cluster are constrained to be similar, or assuming a graph-based structure, where parameter vectors of linked tasks are constrained to be similar, the idea of using a hierarchical structure is based on the hypothesis that parameter vectors of tasks sharing the same upper-level parameter vector will be constrained to be similar to that upper-level parameter vector rather than directly with each other. Such strategy is intuitive to the context of MTL, since it considers both that tasks have their own particularities and that similarities can be captured by a general model simultaneously incorporating more than one task.

A setting composed of tasks A and B is illustrated in Figure 4.1, exemplifying those previously mentioned structure assumptions. In the clustered case represented in

Figure 4.1(a), a regularized MTL formulation would constrain task parameter vectors from A and B to be similar, since the tasks belong to the same cluster. In the case represented in Figure 4.1(b), where the task relationship is modeled by a graph, a regularized MTL formulation would also constrain task parameter vectors from A and B to be similar, since the tasks are linked in the graph by the relationship strength ω. Lastly, since tasks A and B are directly close in the hierarchy represented in Figure 4.1(c), a regularized MTL formulation would constrain task parameter vectors from both A and B to be similar to the parameter vector of the upper-level task AB. Moreover, the task AB is referred to as a hypothetical task, once it is created during the learning process and it is not part of the original tasks.



(a) Cluster (b) Graph (c) Hierarchy

Figure 4.1 – Different task relationship assumptions. (a) Tasks A and B belong to the same cluster, constraining their parameter vectors to be similar. (b) Tasks A and B are linked in the graph, constraining their parameter vectors to be similar, but the relationship strength is weighted by ω. (c) Tasks A and B are directly close in the hierarchy, constraining their parameter vectors to be similar to the parameter vector of the upper-level hypothetical task AB.

In the context of ecology and probabilistic matrix factorization, as a final remark to motivate the usage of hierarchical structures, Shan *et al.* (2012) proposed a methodology to deal with high percentage of missing data applied to plant trait prediction, resulting in significant accuracy improvement over other methods. Considering hierarchical phylogenetic correlation of plants, which is a known biological information derived from many years of intensive research (*e.g.* species, genus and family), data combinations were created following such structure, forcing the algorithm to learn parameters according to the hierarchy levels. At each level L, corresponding parameters were constrained to be close to the ones at level $L+1$. Moreover, as presented in Section 2.4, a similar idea was employed by Widmer *et al.* (2010). Therefore, using a hierarchical structure combined with this kind of regularization motivates a research applied to a more general form, characterized by removing the necessity of knowing the hierarchical structure a priori.

4.2 Hierarchical multi-task learning framework

The proposal is denoted as *Hierarchical Multi-Task Learning Framework* and it is illustrated in Figure 4.2. Each module of the proposed framework is independent of the others, facilitating studies of extensions aiming at improving specific components without

affecting the others, such as alternative proposals for obtaining the hierarchical structure and novel strategies to define the cost function. Moreover, the framework also provides the obtained hierarchical structure, allowing a thorough analysis of the problem, making it possible to identify, for example, clusters of tasks and outlier tasks. Further details of each component are presented in what follows.

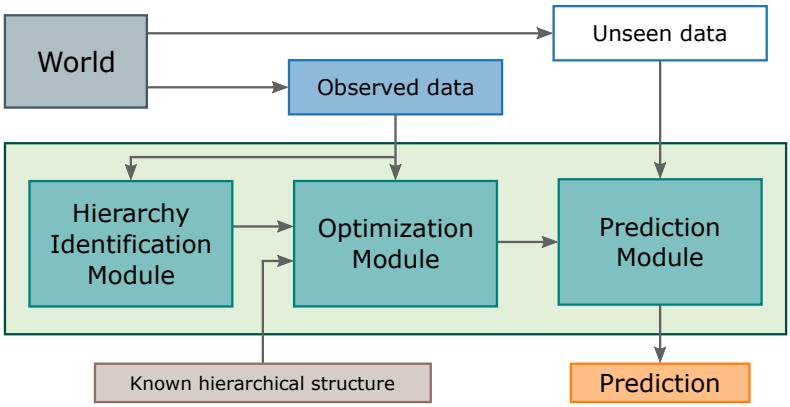


Figure 4.2 – Structure of the proposed framework.

The *Hierarchy Identification Module* receives training data of all tasks and it is responsible for obtaining a representative hierarchical structure that maintains similar tasks close to each other. Once the structure is fully obtained, this module returns an arc set, which describes the obtained structure. In Section 4.3, three alternative strategies are proposed to obtain the hierarchical structure from tasks.

The *Optimization Module* receives both training data of all tasks and the hierarchical structure obtained by the *Hierarchy Identification Module*. In possession of this information, this module is responsible for optimizing the defined cost function, which considers learning parameter vectors for all nodes in the hierarchy, including hypothetical tasks at the internal nodes. In Section 4.5, one cost function is proposed, which incorporates the hierarchical structure in the regularization term. Alternatively, if the hierarchical structure is known due to a prior research, one could use this knowledge directly as input in the *Optimization Module*, skipping the step of obtaining the hierarchical structure from task data.

After completing the optimization procedure, the *Optimization Module* returns only the parameter vectors of the original tasks, whereas parameter vectors for hypothetical tasks are discarded, once they are useful only during the optimization. Notice that the number of hypothetical tasks is $O(n)$, where n is the number of original tasks. Moreover, the number of hypothetical tasks is further reduced when non-binary structures are considered, when compared to binary ones. Finally, the *Prediction Module* receives the task parameter vectors learned from the *Optimization Module*, and, then, this module is able to make predictions on data not observed during the training phase. Figures 4.3, 4.4 and 4.5 depict these three mentioned modules considering an example composed of four tasks.

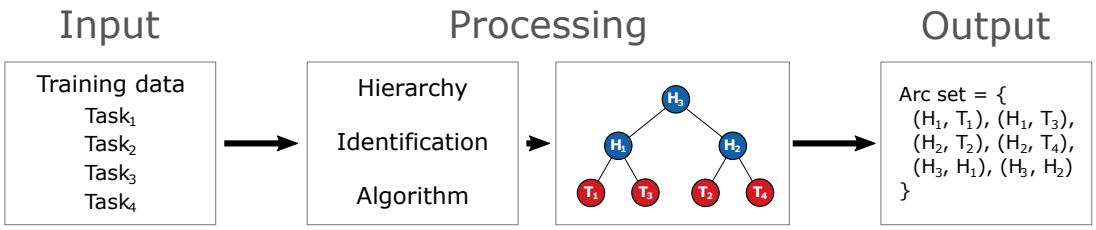


Figure 4.3 – Hierarchy identification module. A hierarchical structure described by an arc set is obtained from training data of all tasks via a hierarchy identification algorithm. In this example, the problem is originally composed of four original tasks, but the obtained hierarchical structure has the original tasks (leaves in red) plus three hypothetical tasks (internal nodes in blue).

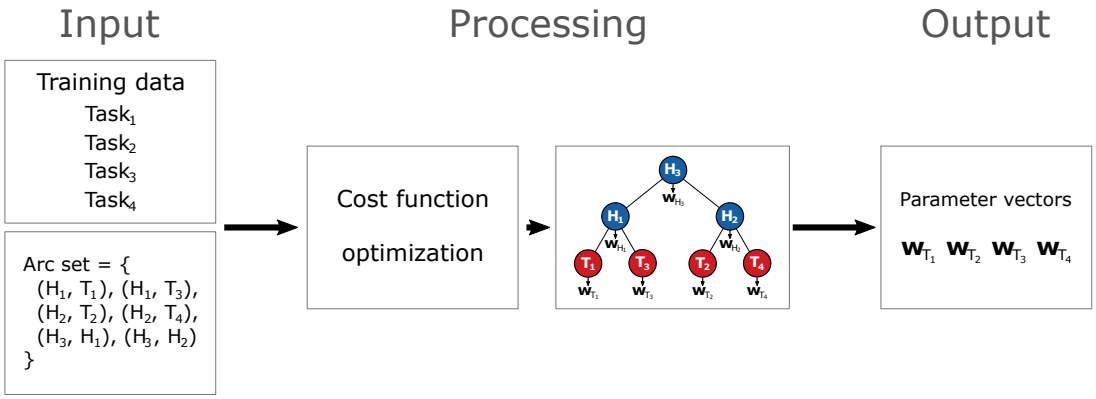


Figure 4.4 – Optimization module. In this example, parameter vectors $\mathbf{w}$ are learned for all nodes in the given hierarchical structure, including for hypothetical tasks at the internal nodes, but, in the end, only the parameter vectors for original tasks matter, whereas parameter vectors for hypothetical tasks are discarded, once they are useful only during the optimization.

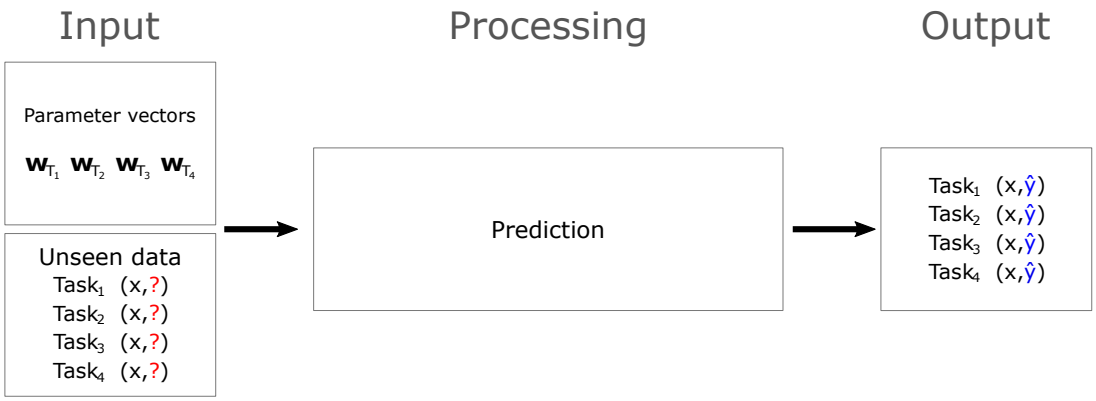


Figure 4.5 – Prediction module. Predictions are made on task data not observed during the training phase.

4.3 Identification of the hierarchical structure

In this section, three alternative strategies that identify a representative hierarchical structure from task data are proposed. Basically, such strategies have been developed following the idea of an agglomerative hierarchical clustering algorithm. Reflecting a bottom-up model, each object starts in its own cluster and, through a similarity metric, the most similar cluster pair at each iteration is combined until the root node is reached.

The resulting structure of applying a hierarchical clustering algorithm assumes that each node corresponds to a particular cluster. In other words, a leaf node is a cluster composed of one object, while an internal node is a cluster composed of the leaves directly below in the hierarchy. Bringing up to the context of MTL, objects to be clustered are original tasks and internal nodes are treated as hypothetical tasks, that is, an internal node is not simply a cluster, but also a generic task that should simultaneously generalize for all data on the leaves.

Following the definition presented in Section 2.3, suppose there are m linear predictive tasks, each of which associated with a training dataset $\mathcal{S}_k$, $k = 1, \ldots, m$. Additionally, let $\mathcal{G}(\mathcal{V}, \mathcal{A})$ be a directed graph, where $\mathcal{V} = \{1, \ldots, \mathrm{M}\}$ denotes the set of M vertices and $\mathcal{A} = \{(p, c) : p, c \in \mathcal{V}\}$ denotes the set of arcs which describes the existence of a connection between vertices p and c. In this scenario, the graph $\mathcal{G}$ is assumed to have a tree topology, such that leaves represent original tasks and internal nodes represent hypothetical tasks. The goal, then, is to define strategies to obtain the set of arcs $\mathcal{A}$ so that similar tasks remain close in the hierarchical structure.

Each strategy that is going to be proposed has its own internal characteristics, which are going to be discussed in the sections that follow. The basic difference among them, though, concentrates on the similarity metric taken. From now on, the proposed approaches to obtain the hierarchical structure are referred to as Hierarchy Identification Algorithm (HIA).

4.3.1 Pairwise task combination

In this proposed approach, the similarity metric taken is the predictive performance when pairing tasks, so that the pair with best performance collapses, that is, the training sets from that task pair are combined, transforming two tasks into a single hypothetical task. At each iteration, STL is applied to all possible hypothetical task candidates and the obtained performance of the training process is kept. At the end of each iteration, all available candidates have been tested, and the one yielding the best performance remains as a new final hypothetical task with its children left out until the structure is completely learned. Furthermore, arcs are created between new hypothetical tasks and their children, which compose the set $\mathcal{A}$. With this intuition in mind, Algorithm 4.1 is

proposed to fulfill this job (comments reinforce what each instruction is performing), based on Algorithm 3.1. Notice that this process of evaluating all task pairing candidates is $O(n^2)$, where n is the number of original tasks.

Algorithm 4.1: Pairwise task combination

```
// task list copy: contains task training data
Input  : S = {S_k : k = 1, ..., m}
// arc list: describes the hierarchical structure
Output : A
1  begin
       // I holds available task ids to be combined
2      I ← {1, ..., |S|}
       // c holds the new task id
3      c ← |S| + 1
       // A holds arcs that describe the hierarchy
4      A ← ∅
       // iterate until the structure is fully identified
5      while |I| > 1 do
           // performance evaluation for all task pairing ij using STL and
           // k-fold cross-validation to obtain an average performance
6          P_ij ← cross_validation(S_i ∪ S_j, k), ∀ i, j ∈ I, i < j
           // identify the task pair which yielded the best performance
7          i, j ← arg max_{i,j} P_ij
           // combine data of the best pair and add into S as the new task data
8          S ← S + {S_i ∪ S_j}
           // remove children ids i and j from I and add the new task id c
9          I ← I − {i, j} + {c}
           // add arcs from c to its children i and j
10         A ← A + {(c, i), (c, j)}
           // increase new task id
11         c ← c + 1
12     end
13 end
```

Since applying STL and measuring the performance are internal operations of the proposed algorithm, it is necessary to split the training data of the combined tasks into training and validation sets. Thus, STL is applied using the training set and the performance is measured using the validation set. Such technique is known as holdout, which is simple to implement and indicated when the sample size is large.

However, tasks possessing limited training data, which is the situation where MTL stands out, might generalize poorly, providing an inappropriate performance value to be used as similarity metric. In order to overcome such scenario, the k-fold cross-validation technique is proposed to be applied rather than holdout. This way, the resulting value is an average performance value over the folds, which is more suitable to be incorporated

into the proposed algorithm.

The proposed pairwise task combination strategy, which is simply identified as HMTL-PC in the sections that follow, presents itself as an intuitive procedure and, as will be shown in the experiments, provides promising results. However, HMTL-PC does not follow a probabilistic model and it depends totally on the performance obtained by task combinations, which might not provide a structure that best generalizes, especially in the presence of noisy data. Besides, the computational cost is high due to the necessity of learning parameter vectors for all possible task combinations, which may be impracticable if the number of tasks is large. These aspects motivated further studies, resulting on the usage of Bayesian-based hierarchical clustering methods already discussed in Chapter 3.

4.3.2 Bayesian hierarchical clustering

As seen in Chapter 3, the absence of a probabilistic model on classic agglomerative hierarchical clustering algorithms is one motivation for Heller & Ghahramani (2005a) having developed the BHC algorithm. Moreover, an ad-hoc distance metric (*e.g.* Euclidean or Manhattan) and a criterion (*e.g.* single or complete) are necessary to be specified for classic algorithms, making it difficult to achieve the inherent structure behind the task relationship. Thus, Bayesian-based hierarchical clustering algorithms are favored in this research instead of relying on linkage methods.

In order to incorporate the BHC algorithm into the context of MTL, it is proposed that the objects to be clustered are task parameter vectors obtained from STL. Although such strategy has showed itself very competent during the research development, it is not limited to STL. In other words, given the independence of the framework modules depicted in Figure 4.2, other strategies to derive objects to be hierarchically clustered from tasks may be proposed, including employing MTL approaches. Furthermore, task parameter vectors derived from systems already running may also be employed as objects to be clustered by BHC in order to verify the obtained hierarchical structure, which may provide new insights regarding the entire task set.

Algorithm 4.2 describes the procedure of using BHC combined with MTL to obtain a hierarchical structure. Although BHC was conceived for general purposes, its combination with MTL fits well into the proposed HMTL framework and, on the sections that follow, their joint usage is referred to as HMTL-BHC and HMTL-BHCc, when the cut is not allowed and when the cut is allowed, respectively. In practice, allowing the cut means that BHC stops the hierarchy construction as soon as it identifies that the probability of hypothesis $\mathcal{H}_1$ (all objects belong to the same mixture) of the newly constructed node is less than 50%. At the end of the algorithm, the arc set $\mathcal{A}$ is available, which will be used afterwards in the *Optimization Module*.

Algorithm 4.2: BHC for obtaining a hierarchical structure from tasks.

```
// task list copy, probabilistic model, prior, α and cut indicator
Input  : 𝒮, model, prior, α > 0, cut
// arc list: describes the hierarchical structure
Output : 𝒜
1 begin
2     for i ← 1 to |𝒮| do
          // apply STL for task i obtaining the parameter vector w_i
          // each w_i is viewed as an object to be clustered
3         data[i] ← w_i ← STL(task_i)
4     end
      // obtain the hierarchical structure from task parameter vectors
5     structure ← bhc(data, model, prior, α, cut)
      // convert bhc output to a list of arcs
6     𝒜 ← convert(structure)
7 end
```

Finally, Section 3.3 discussed the importance of having a probabilistic model applied to agglomerative hierarchical clustering when a new data point comes out. In the context of MTL, therefore, it means that there is no need to recalculate the hierarchical structure when a new task is available, since the entire hierarchical structure is supported by a probabilistic model.

4.3.3 Bayesian rose trees

The main difference between BHC and BRT algorithms lies in the decision taken when the best object pair to be combined is found. In BHC, the best pair is simply combined, operation called join. In BRT, on the other hand, two additional operations are also considered (absorb and collapse), as described in Section 3.4.2. Thus, instead of BHC, employing BRT is also proposed in this dissertation as an HIA within the HMTL framework. Similarly to BHC, the sections that follow refer to the usage of BRT as HMTL-BRT when the cut is not allowed, and HMTL-BRTc when the cut is allowed.

The BRT approach tends to obtain parsimonious structures and, like BHC, it also allows the structure to be cut. Consequently, there will be less hypothetical nodes to be optimized in the HMTL framework, resulting in a faster optimization process. Moreover, the resulting structure may be more informative to users, once spurious relations forced by binary structures are removed. On the other hand, depending on the MTL problem, it may be more interesting to have binary relations among tasks rather than allowing various tasks being regularized by a single parent node.

It is worth mentioning that employing BRT allows but does not force the resulting structure to be non-binary. As BRT is a generalization of BHC, obtaining a binary structure is perfectly plausible. Moreover, allowing the cut also does not force the resulting structure to be presented in more than one tree. The inherent characteristics of the problem and the choice made for the hyper-parameters will define the shape of the resulting structure.

Algorithm 4.3 presents the usage of BRT within the HMTL framework. Notice that both Algorithm 4.2 and Algorithm 4.3 are identical, except for the highlighted area, where BRT is employed instead of BHC. Such simple procedure allows other HIAs to be proposed as well by just changing the highlighted area.

Algorithm 4.3: BRT for obtaining a hierarchical structure from tasks.

```
// task list copy, probabilistic model, prior, α and cut indicator
Input  : 𝒮, model, prior, α > 0, cut
// arc list: describes the hierarchical structure
Output : 𝒜
1 begin
2     for i ← 1 to |𝒮| do
          // apply STL for task i obtaining the parameter vector w_i
          // each w_i is viewed as an object to be clustered
3         data[i] ← w_i ← STL(task_i)
4     end
      // obtain the hierarchical structure from task parameter vectors
5     structure ← brt(data, model, prior, α, cut)
      // convert brt output to a list of arcs
6     𝒜 ← convert(structure)
7 end
```

Either applying BHC or BRT, the set of necessary hyper-parameters are composed of the hyper-parameter α and the hyper-parameters $\boldsymbol{\beta}$ of the chosen prior, as discussed in Sections 3.3 and 3.4. Since the objects to be hierarchically clustered by BHC and BRT (refer to Algorithm 4.2 and Algorithm 4.3) are the m task parameter vectors in d dimensions, which are organized in the matrix $data \in \mathbb{R}^{m \times d}$, the task parameter vectors are modeled using Gaussians and, consequently, the Normal-inverse-Wishart distribution is an appropriate conjugate prior to be employed. Such prior has the following four hyper-parameters:

- $\boldsymbol{\mu_0}$: The prior on the mean, which is set $\boldsymbol{\mu_0} = \overline{data}$, $\boldsymbol{\mu_0} \in \mathbb{R}^d$.
- S: The scatter matrix, which is set $S = I_d g$, $S \in \mathbb{R}^{d \times d}$, where I_d is the identity matrix and g is a constant that should be optimized.
- ν: The degrees of freedom, which is set to a value $\nu \geq d$. In the experiments, the value used is $\nu = d + 2$.

- r: The scaling factor, $r \in \mathbb{R}$, that should be optimized as well.

Hence, the necessary hyper-parameters to be optimized is denoted $\boldsymbol{\gamma} = \{r, \alpha, g\}$. Both Heller & Ghahramani (2005a) and Murphy (2007) provide the equations for MAP estimates of the prior hyper-parameters $\boldsymbol{\beta}$, which are used in the implementation of both BHC and BRT. In addition, the cut indicator could also be treated as a hyper-parameter to be optimized, though it is going to be individually assessed in the experiments.

4.4 Prior hyper-parameters optimization

Heller & Ghahramani (2005a) suggested optimizing BHC hyper-parameters in an EM-like procedure, where the best tree structure is obtained given the current hyper-parameters and then the best hyper-parameters are obtained given the current tree structure, via maximizing the marginal likelihood that data within the newly created node belong to the same mixture. Nevertheless, authors stated that only one hyper-parameter was optimized using such procedure.

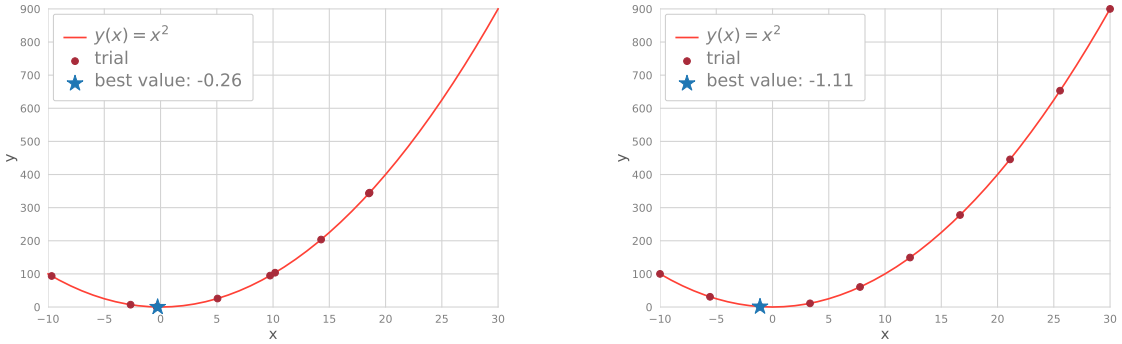
In Blundell *et al.* (2012), the same procedure was applied for BRT hyper-parameters, though they pointed out that optimizing in such manner converges to a local optimum since the marginal likelihood function is typically not convex, and, consequently, the hyper-parameters are sensible to initial conditions. Experiments using this EM-like procedure were performed in this research, and, indeed, the hyper-parameters presented themselves very sensible to initial conditions. In addition, it was observed that manually setting the hyper-parameters in a random manner yielded favorable results with only few trials.

This scenario motivated the usage of the Random Search (RS) technique to optimize both BHC and BRT hyper-parameters $\boldsymbol{\gamma}$. Shed light by Bergstra & Bengio (2012), RS is simple to implement either sequentially or in parallel, though a Python library[1] is available to be incorporated into any Python project (BERGSTRA *et al.*, 2013). Furthermore, Bergstra & Bengio (2012) showed theoretically and empirically that RS is more interesting than the widely used grid search strategy, especially when there are lots of hyper-parameters and the search space is awkwardly not convex.

In order to apply RS, the following components are necessary to be specified: (i) an objective function taking hyper-parameter candidates randomly chosen by RS as input, and returning a value that indicates the quality of the tested hyper-parameter candidates; (ii) the space over which to search for the hyper-parameters; (iii) the number of trials. The selected hyper-parameters are the ones that yield the best output value of the objective function within the number of trials.

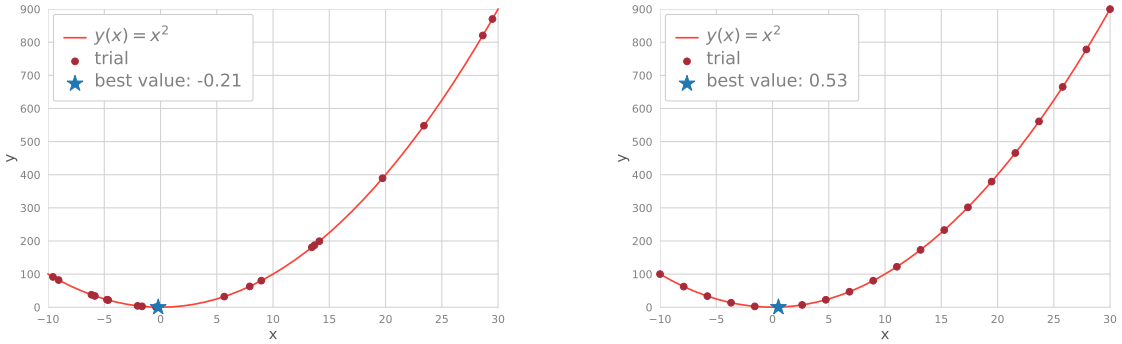
[1] http://hyperopt.github.io/

Aiming at motivating RS, Figures 4.6, 4.7, 4.8 and 4.9 exemplify a comparison between RS and grid search with the number of trials fixed in 10, 20, 50 and 100, respectively. In this example, suppose that $y(x) = x^2$ defined in the interval $[-10, 30]$ is the objective function to be minimized, where the optimum value is zero, so that the closer to zero a value is, the better. RS tries to randomly find the best value within the search space $[-10, 30]$, whereas grid search tries to find the best value considering a grid of n possible values uniformly spaced within the interval $[-10, 30]$, where n is the number of trials. Notice that, in all runs depicted in Figures 4.6, 4.7, 4.8 and 4.9, RS approached more the optimum point than grid search.



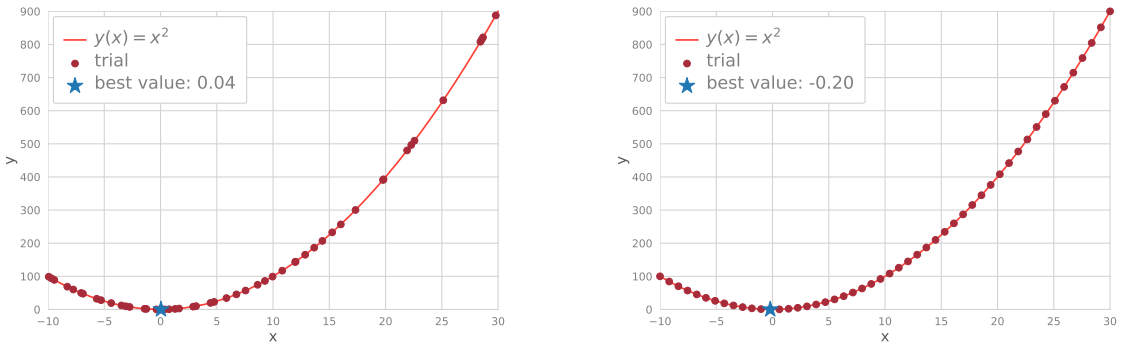
(a) Random search (b) Grid search

Figure 4.6 – Comparison between random search and grid search with 10 trials. Compared to grid search, random search found the best value, that is, a value closer to the optimum.



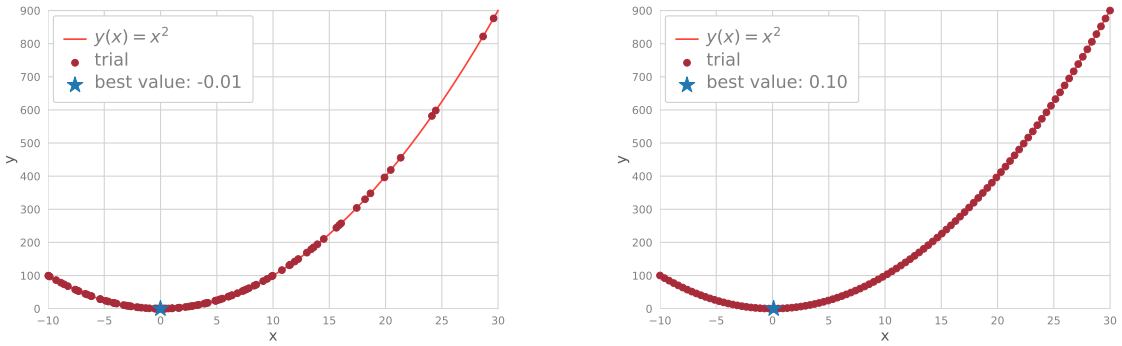
(a) Random search (b) Grid search

Figure 4.7 – Comparison between random search and grid search with 20 trials. Compared to grid search, random search found the best value, that is, a value closer to the optimum.



(a) Random search (b) Grid search

Figure 4.8 – Comparison between random search and grid search with 50 trials. Compared to grid search, random search found the best value, that is, a value closer to the optimum.



(a) Random search (b) Grid search

Figure 4.9 – Comparison between random search and grid search with 100 trials. Compared to grid search, random search found the best value, that is, a value closer to the optimum.

Bringing up RS and the HMTL framework together, the proposed objective function is defined simply as a function returning the task set predictive error average by means of k-fold cross-validation when applying the obtained hierarchical structure with the hyper-parameter candidates. In other words, this function measures how the hyper-parameter candidates are suitable to derive a hierarchical structure consistent with the context of MTL. Such approach internally exercises the entire framework, but clearly considers only training and validation sets. In addition, the search space and the number of trials are defined in Chapter 5, which deals with the experimental setup. Finally, after optimizing the hyper-parameters $\boldsymbol{\gamma}$ in the described procedure, the final hierarchical structure is obtained, which is employed in the *Optimization module*.

4.5 Hierarchy-based regularization

Once the hierarchy is obtained and encoded in $\mathcal{A}$, a regularization term similar to the one employed by Widmer *et al.* (2010) and Shan *et al.* (2012) is proposed, where parameter vectors from nodes in the hierarchy and their respective children/parent are constrained to be close to each other, taking the form presented in Equation (4.1):

$$R(\mathcal{W}, \mathcal{A}) = \sum_{(p,c)\in\mathcal{A}} \|\mathbf{w}_c - \mathbf{w}_p\|_2^2 \tag{4.1}$$

where a task parameter vector $\mathbf{w}_p$ is treated as the parent of the task parameter vector $\mathbf{w}_c$ and $\|\cdot\|_2$ is the Euclidean norm. That way, this regularization function penalizes deviations between a particular parent parameter vector and its children parameter vectors.

Although the intention is to learn parameter vectors not only for original tasks but also for hypothetical tasks, the set $\mathcal{S}$, which includes only original task data, may be used in the loss term together with the presented regularization term. Thus, even though internal nodes have no data associated with them, which means their loss is 0, learning their parameter vectors will indirectly consider their children node data due to the necessity of the agreement in terms of similarity between a parent parameter vector and their children parameter vectors.

Finally, considering the regularized MTL framework presented in Section 2.3, the complete regularized cost function to be minimized is represented by Equation (4.2):

$$J(\mathcal{S}, \mathcal{W}, \mathcal{A}) = \underbrace{\sum_{k=1}^{M}\sum_{i=1}^{n_k} \ell(f(\mathbf{x}_k^i, \mathbf{w}_k), \mathrm{y}_k^i)}_{L(\mathcal{S},\mathcal{W})} + \lambda \underbrace{\sum_{(p,c)\in\mathcal{A}} \|\mathbf{w}_c - \mathbf{w}_p\|_2^2}_{R(\mathcal{W},\mathcal{A})} \tag{4.2}$$

where $\lambda \geq 0$ is a hyper-parameter controlling the regularization influence. As additional material, a Bayesian perspective regarding the problem derivation to obtain the cost function is presented in Appendix A.

4.6 Chapter summary

This chapter presented the Hierarchical Multi-Task Learning framework. Motivated by the literature, where there is a trend on letting the machine identify the task relationship as well as the lack of studies merging MTL and hierarchical clustering areas, three approaches to obtain a representative hierarchical structure from task data are proposed. Moreover, a cost function is proposed in order to translate such structural information into the regularization function. Finally, as the Bayesian-based HIAs have hyper-parameters to be specified, their optimization via Random Search is also proposed. In summary, these are the main contributions made during the development of this research.

Part III

Experiments

Chapter 5

Experimental setup

In this chapter, common aspects regarding experimental settings are presented. First, the evaluation metrics are introduced, followed by datasets that are going to be employed, including both synthetic and real world ones. Then, selected competitors that are going to be put to test on the same conditions to the proposed framework are presented. As the proposed framework uses specific settings to be properly executed, such settings are also covered in detail. Finally, the experimental plan is summarized, closing the chapter.

5.1 Evaluation metrics

As will be seen in Section 5.2, the employed datasets in the experiments are composed of either regression or binary classification tasks. Focusing on these two types of tasks, this section presents the evaluation metrics used both to perform model selection and to measure the generalization performance of learned task models.

5.1.1 Regression problems

The goal when dealing with regression tasks is to predict real values, such as exam scores based on information from students, house prices based on features from houses, and so on. In order to compare the predicted output with the actual one, quantifying the difference between these two values is common in the literature by means of an error metric. Thus, in the experiments of this dissertation, the root mean squared error (RMSE) is employed as an error metric for regression tasks.

Basically, RMSE applied in MTL takes the average of the squared difference between actual values and predicted values considering all tasks, followed by taking the squared root. This error metric punishes more intensively larger errors and produces a score to be minimized. Focusing only on the context of MTL, several authors like Jacob *et al.* (2008), Kang *et al.* (2011), Kumar & III (2012), Gonçalves *et al.* (2014) and Xu *et al.* (2015) have used RMSE to compare results produced by existing approaches with the one

achieved by their own proposals. Equation (5.1) formally defines RMSE.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\text{y}_t - \hat{\text{y}}_t)^2} \tag{5.1}$$

where y_t and $\hat{\text{y}}_t$ are the actual value and the predicted value for observation t, respectively, and n is the total number of observations.

5.1.2 Binary classification problems

Unlike regression, the goal in classification is to predict discrete values, that is, assigning objects to a finite number of classes. These tasks range from classification of e-mails into spam or not spam, assignment of a particular sentiment based on a product review, distinguishing objects in images, and so on.

Classification tasks may be divided in either binary classification (in which there are only two classes) or multiclass classification (in which there are multiple classes), so that both consider assigning each object to only one class. There is also multilabel classification, which allows an object to be assigned to a subset of labels. In the experiments of this dissertation, only binary classification tasks are considered. Given that multiclass classification tasks may be converted into multiple binary classification tasks, they could have been considered as well.

In order to measure performance of binary classification tasks, one may consider building the so-called confusion matrix. Such matrix quantifies correct and incorrect predicted values compared to actual values. Assuming {False, True} as the set of classes, there are only four possible situations to account, as depicted in Table 5.1: (1) the actual value is True and the classifier correctly classified the object as True (True Positive); (2) the actual value is True, but the classifier wrongly classified the object as False (False Negative); (3) the actual value is False and the classifier correctly classified the object as False (True Negative); (4) the actual value is False, but the classifier wrongly classified the object as True (False Positive).

Table 5.1 – Confusion matrix for a binary classification task. An ideal classifier would have no FP/FN entries.

		Actual value	
		True	False
Predicted value	True	True Positive (TP)	False Positive (FP)
	False	False Negative (FN)	True Negative (TN)

Once the confusion matrix is built by counting the number of occurrences in each situation, several metrics may be calculated from it, such as accuracy, classification

error (CE), precision, recall and f-score. For a wide discussion regarding classification metrics, the reader is invited to refer to Powers (2011).

In the experiments of this dissertation, CE and recall are employed as evaluation metrics. Essentially, recall is used as a similarity metric for HMTL-PC in order to decide the best task pair to combine and CE is used to compare final results obtained from the proposed framework and competitors.

Focusing on CE, this metric gives the ratio of wrong predictions in relation to total predictions made, which is equivalent to the complement of the accuracy metric. As CE measures error, the closer to zero, the better. Equation (5.2) formally defines CE as follows:

$$\text{CE} = \frac{\text{wrong predictions}}{\text{total predictions}} = \frac{\sum FP + \sum FN}{\sum TP + \sum TN + \sum FP + \sum FN} \tag{5.2}$$

Finally, the recall metric gives the ratio of objects correctly classified as True in relation to objects that should actually be classified as True. Also known as sensitivity, this metric provides more discriminant values than CE to be used as a similarity metric for HMTL-PC, though other metrics may be employed as well. Values provided by Recall range in the interval $[0, 1]$ and the closer to one, the better. Equation (5.3) formally defines Recall as follows:

$$\text{Recall} = \frac{\text{objects correctly classified as True}}{\text{objects that should actually be classified as True}} = \frac{\sum TP}{\sum TP + \sum FN} \tag{5.3}$$

5.2 Description of the datasets

Four synthetic datasets named Ds_1, Ds_2, Ds_3 and Ds_4 were generated in order to assess the proposed approaches, each one having a particular structure to reproduce different scenarios, making it possible to perform a thorough analysis on whether the approaches are behaving as expected or not. Additional experiments are also considered involving the proposed approaches, but focusing on real world datasets used in the MTL literature. Hence, four real world datasets were selected to be employed in the experiments as well. Both synthetic and real world datasets are detailed in the sections that follows and summarized in Table 5.2.

5.2.1 Synthetic tasks

Inspired by similar procedures adopted by Xu *et al.* (2015) and Zhong & Kwok (2012), each synthetic dataset was conceived to correspond to $m = 12$ regression tasks (labeled from 0 to 11) with $d = 30$ features and $n = 500$ data observations randomly generated by $\mathrm{X} \sim \mathcal{N}(0, 1)$ and $\mathbf{y} = \mathrm{X}\mathbf{w} + \mathcal{N}(0, 10)$, where $\mathcal{N}$ is the normal distribution and each task parameter vector set $\mathcal{W}$ was designed to simulate the following structures:

- $\boldsymbol{Ds_1}$: All tasks are independent, such that $\mathbf{w}^i \sim \mathcal{U}(0, 10)$, where $\mathcal{U}$ is the uniform distribution and $\mathbf{w}^i$ is the parameter vector of the i-th task, $i = 1, \ldots, m$. Notice that independent does not mean distinct. The generated centers could be close by random chance.

- $\boldsymbol{Ds_2}$: All tasks are similar, such that $\mathbf{w}^i \sim \mathcal{N}(\mu, I)$ and $\mu \sim \mathcal{U}(0, 10)$.

- $\boldsymbol{Ds_3}$: 3 clusters of 4 similar tasks, such that $\mathbf{w}^i_j \sim \mathcal{N}(\mu^j, I)$ and $\mu^j \sim \mathcal{U}(0, 10)$, where j is the j-th cluster. The clusters are composed of tasks {0, 3, 6, 9}, {1, 4, 7, 10} and {2, 5, 8, 11}.

- $\boldsymbol{Ds_4}$: 2 clusters of 5 similar tasks composed of tasks {0, 2, 4, 6, 8} and {1, 3, 5, 7, 9}, plus 2 independent tasks labeled as 10 and 11. The generative procedures used in Ds_3 and Ds_1 were applied to this scenario in order to generate the clusters and the independent tasks, respectively.

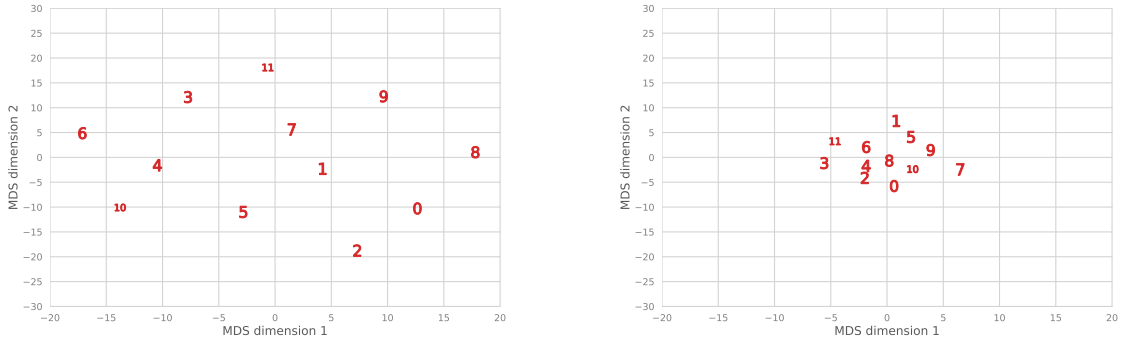
In order to validate the generative process of the desired scenarios, each task parameter vector set $\mathcal{W}$ was projected in 2 dimensions using the multidimensional scale (MDS) technique (BORG; GROENEN, 2005) as illustrated in Figure 5.1, making it possible to have a perspective view of their structures. Although the generated task parameter vectors (ground truth) lie in the 30-dimensional space, MDS places the objects in a lower dimension preserving their relative distances as much as possible. Such projections suffice to provide insights into the pursued structures.

In Figure 5.1(a) and Figure 5.1(b), which represent independent and similar tasks, respectively, it is possible to observe that task parameter vectors were spread over the space as expected, since they do not present a specific structure, such as clusters. On the other hand, Figure 5.1(d) shows the clusters and independent tasks distant from each other. Moreover, Figure 5.1(c) shows that two of the three clusters are closer, information that will be explored on the results.

Clearly, if the task structure of dataset Ds_3 was not known beforehand, Figure 5.1(c) could guide to misleading interpretations regarding the relations. For example, tasks 1 and 5 could be assumed to be part of the same cluster, which is not true. Therefore, it is of great relevance to implement an automatic structural learning mechanism to identify the underlying relationship among tasks, thus removing the dependency of limited human assumptions.

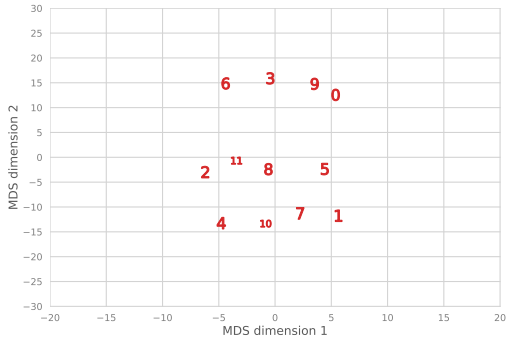
5.2.2 Real world tasks

Synthetic datasets will provide several important observations derived from the results due to the prior knowledge regarding their task relationship structure. In case of real world problems, on the other hand, one does not know such structural information.

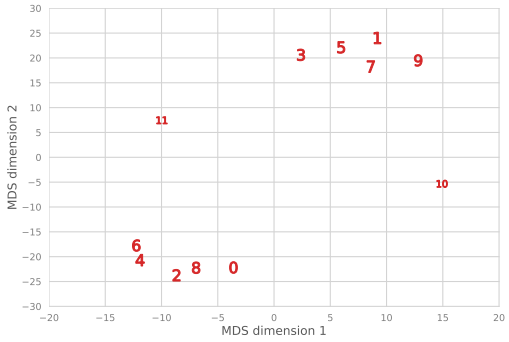


(a) Ds_1

(b) Ds_2



(c) Ds_3



(d) Ds_4

Figure 5.1 – Two-dimensional projection of the ground truth task parameter vectors from synthetic datasets. (a) and (b) do not show a clear structure, once their structures represent independent and similar tasks, respectively. (c) shows each expected cluster horizontally arranged, while (d) clearly shows the expected clusters and the independent tasks.

Nevertheless, the intention when employing real world datasets is to show the proposed framework effectiveness, competing with other approaches in the literature. Additionally, as all synthetic datasets are composed of regression tasks, the experiments with real world datasets are focused mostly on binary classification problems. In summary, the datasets are composed of one regression problem and three binary classification problems, described as follows:

- **School**: This problem is composed of 139 regression tasks. Each task gathers observations from a specific London's secondary school and the goal is to predict students' exam scores. Each observation is described by 27 features, which are extracted from the schools and their students, such as percentage of students eligible for free meals, gender and ethnic group (GOLDSTEIN, 1991). The dataset was obtained from the MALSAR toolbox (ZHOU *et al.*, 2011b).

- **Spam3**: This problem is composed of 3 binary classification tasks. Each task gathers observations from a specific user and the goal is to classify an e-mail as either (1)

spam or (0) non-spam. The dataset is originally derived from ECML 2006 discovery challenge[1], but the pre-processed dataset provided by Gonçalves *et al.* (2014) was used in the experiments of this research.

- **Spam15**: This problem is composed of 15 binary classification tasks. Similarly to Spam3, the goal is also to classify users' e-mails and the differences lie in the number of tasks and in the sample size per task. Gonçalves *et al.* (2014) provided the dataset as well.

- **Landmine**: This problem is composed of 19 binary classification tasks. Each task gathers observations from a specific landmine field and the goal is to classify the field as either (1) landmine or (0) clutter. Each observation is described by 9 features extracted from radar images, in which four of them are moment-based, three are correlation-based, one is an energy ratio and one is a spatial variance (XUE *et al.*, 2007).

Although the task relationship structures of real world datasets are not known a priori, the MTL literature has identified important characteristics of each problem. The *School* dataset has been explored as a problem where all the tasks are similar, properly fitting into MTL (BAKKER; HESKES, 2003). Xue *et al.* (2007) and Gonçalves *et al.* (2014) showed that the *Landmine* dataset is composed of two major clusters of tasks. The datasets *Spam3* and *Spam15* are described as problems composed of related tasks. This information will be important when discussing the results obtained for each dataset. As a final remark, all datasets were standardized to have zero mean and unit variance.

Table 5.2 – Summary of the datasets. In case of School and Landmine datasets, the tasks have a varied sample size within the defined interval. Notice that *Synthetics* represent the 4 versions presented in Section 5.2.1.

Dataset	**Type**	**Tasks**	**Features**	**Sample size per task**
Synthetics	Regression	12	30	500
School	Regression	139	27	[22, 251]
Spam3	Binary classification	3	500	2500
Spam15	Binary classification	15	500	400
Landmine	Binary classification	19	9	[445, 690]

5.3 Competing approaches

Experiments involving the proposed framework will also show comparative results obtained by STL and four regularized approaches. If any MTL approach loses

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

to STL in terms of generalization performance, it can indicate that negative transfer is occurring, which is not interesting. Moreover, the regularized competitors were specifically chosen due to similarities with the proposed cost function, including: (i) they are built on top of the regularization framework; (ii) their regularization function has only one hyper-parameter to optimize; (iii) their optimization problem are solved using AGM, whose implementation is derived from the MALSAR toolbox (ZHOU *et al.*, 2011b). A brief description of each competitor is presented as follows:

- **Single-Task Learning (STL)**: Considers learning each task isolated from the others, applying Ordinary Least Squares (OLS) and Logistic Regression (LR) when dealing with regression and binary classification problems, respectively.
- **LASSO**: Well-known regularization method introduced by Tibshirani (1996), which imposes the minimization of the ℓ_1-norm on the parameter vectors in order to perform feature selection.
- **Ridge**: Another well-known regularization method, which imposes the minimization of the ℓ_2-norm on the parameter vectors in order to control the complexity of the model. This method is also known as Tikhonov regularization.
- **MTFL** (ARGYRIOU *et al.*, 2006): Considers that similar tasks share a common set of features by the imposition of $\ell_{2,1}$-norm restrictions on the parameter vectors.
- **LowRank** (JI; YE, 2009): Considers that parameter vectors of similar tasks share a low dimensional subspace, when trace norm minimization is imposed on the parameter vectors.

5.4 Specific settings

All the proposed approaches within the HMTL framework admit some sort of specific settings. Particularly, the proposed hierarchy identification algorithms (HIAs) are flexible and different strategies may be employed in some internal procedures. In this section, suitable settings identified for the experiments are presented, though alternative promising configurations could have been considered as well.

5.4.1 HMTL-PC

Focusing on HMTL-PC, this HIA needs the number k of folds for k-fold cross-validation in order to obtain an average STL performance for each task combination. In all experiments, the value $k = 3$ was employed, which provided consistent results to situations either with plenty or scarce training data. Clearly, users may consider using other types of cross-validation as well, such as leave-one-out and leave-p-out.

A particular STL strategy and evaluation metric also need to be specified, that is, how to perform the training on the task combinations and what performance metric will be used in order to decide which task combination is the best choice to merge. Thus, when dealing with regression tasks, the STL steps in HMTL-PC correspond to applying OLS and the goal at each iteration is to find the task pair which minimizes the RMSE metric. On the other hand, when dealing with binary classification tasks, STL corresponds to applying logistic regression (LR) and the goal at each iteration is to find the task pair which maximizes the Recall metric.

5.4.2 Bayesian-based HIAs

Recall that Bayesian-based HIAs are HMTL-BHC and HMTL-BRT (and their variations to enable the cut operation), which are simply hierarchical clustering algorithms of general purpose but adapted to the context of MTL. As seen in Section 4.3.2 and Section 4.3.3, STL is performed for each task in order to learn their parameter vectors, which, in turn, are used as objects to be clustered by the hierarchical clustering algorithms. Hence, a STL strategy needs to be defined.

Although OLS and LR may be used as STL to obtain the task parameter vectors to be clustered for regression and binary classification problems, respectively, such strategies might not behave very well when dealing with scarce training data. Therefore, there is a need to employ more advanced techniques for such situations.

In order to cover both scarce and plenty training data, OLS and LR were employed, but regularized by the LASSO (TIBSHIRANI, 1996). In this manner, task parameter vectors are regularized and only relevant features are selected, resulting in parameter vectors more appropriated to be used as objects. Moreover, the regularization influence of the LASSO is controlled by a hyper-parameter λ, which is optimized via k-fold cross-validation ($k = 3$) among the candidate values $\{0.01, 0.1, 1, 10, 100\}$.

Hyper-parameters

As seen in Section 4.4, the Random Search (RS) technique was chosen to optimize the hyper-parameters $\boldsymbol{\gamma} = \{r, \alpha, g\}$ of Bayesian-based HIAs. Recall that RS needs three settings to be employed: the objective function to minimize (already discussed in Section 4.4), the search space and the number of trials. For the experiments, 100 trials were set, though the more trials are available, the more is the chance to obtain better values, but, consequently, the higher is the computational budget required. Regarding the search space, a wide space was defined as shown in Table 5.3.

Table 5.3 – Search space for the hyper-parameters $\boldsymbol{\gamma} = \{r, \alpha, g\}$. Log-uniform returns a value, with its logarithm being uniformly distributed according to $\exp(\mathcal{U}(\text{a, b}))$. Log-normal returns a positive value, with its logarithm being normally distributed according to $\exp(\mathcal{N}(\mu, \sigma))$. Refer to hyperopt documentation for a thorough description of all available options.

hyper-parameter	distribution	a	b
r	log-uniform	$1e-6$	1
α	log-uniform	$1e-6$	1
g	log-normal	0	1

5.4.3 Cost function regularization parameter

Not limited to the proposed cost function presented in Section 4.5, all employed algorithms built on top of the regularization framework, which includes the competitors, use the same configuration for optimizing their cost function regularization parameter: k-fold cross-validation ($k = 3$) is applied in the experiments to select the best regularization parameter among the values $\{0.01, 0.1, 1, 10, 100\}$. In addition, since all presented competitors implemented in the MALSAR toolbox use AGM to optimize the task parameter vectors, the same procedure is performed in the *Optimization Module* of the HMTL framework, though other convex optimization techniques could have been considered as well.

5.5 Experimental plan

Once having the evaluation metrics, synthetic and real world datasets, competitors from the literature and specific settings of the proposed framework specified, this section closes the experimental setup by summarizing the experiments that are going to be discussed in Chapter 6.

Structural analysis (Section 6.1)

In this experiment, the intention is to assess the obtained hierarchical structure when applying each HIA, including the versions that allow the structure to be cut (HMTL-BHCc and HMTL-BRTc). All four synthetic datasets are considered, providing insights into the adequacy of each HIA, since a prior knowledge on task relations is available. Of course, this prior knowledge is only considered after concluding the learning process. On each hierarchical structure that will be depicted in the resulting figures, the leaf nodes (in red) represent original tasks, while internal nodes (in blue) represent hypothetical tasks generated during the process of obtaining the structure. Moreover, labels on blue nodes

also indicate the order in which they were generated, and the absence of a specific value on the ordering indicates that the node was removed during the process (applicable only for HMTL-BRT and HMTL-BRTc).

Performance analysis: Varying training data size (Section 6.2.1)

In this experiment, the intention is to assess the predictive performance varying the number of observations available for training when applying each HIA, including the versions that allow the structure to be cut (HMTL-BHCc and HMTL-BRTc). All four synthetic datasets are considered, aiming at showing that the HIAs provide promising results in different scenarios when compared to competitors of the literature presented in Section 5.3. More specifically, 40% of all training data is reserved exclusively for performance evaluation (test set) in order to provide representative performance results and, with the rest, different quantities of data for each task are gathered for training, which lie on the interval $[100, 200]$ and is incremented with step 25. The resulting figures (bar plots) will depict performance comparisons between the competitors and the proposed HIAs in terms of average RMSE and standard deviation in the test set over 20 independent runs.

Performance analysis: Random structures (Section 6.2.2)

In this experiment, the intention is to evaluate the effect of employing a randomly generated structure (simply referred to as *Random* in the results) on different scenarios provided by the synthetic datasets. More specifically, 40% of all training data is reserved exclusively for performance evaluation (test set) in order to provide representative performance results and, with the rest, different quantities of data for each task are gathered for training, which lie on the interval $[100, 200]$ and is incremented with step 25. The resulting figures (bar plots) will depict performance comparisons among *Random*, STL and HMTL-PC in terms of average RMSE and standard deviation in the test set over 20 independent runs.

Performance analysis: Each task (Section 6.2.3)

In this experiment, the intention is also assessing the predictive performance of the HIAs, but the focus is on analyzing the results of each task rather than averaging the entire task set. All four synthetic datasets are considered, aiming at bringing different known scenarios to be discussed. More specifically, 40% of all training data is reserved exclusively for performance evaluation (test set) in order to provide representative performance results and, with the rest, quantities of size 100 are gathered for training over 20 independent

runs. The resulting figures (box-plots) will depict performance comparisons in terms of RMSE on the test set. Furthermore, results for each synthetic dataset are divided in two figures for best exposition: the first one compares STL and HMTL-PC; and the second one compares HMTL-PC and Bayesian-based HIAs.

Performance analysis: Real world datasets (Section 6.2.4)

In this experiment, the intention is to assess the predictive performance of each HIA, including the versions that allow the structure to be cut (HMTL-BHCc and HMTL-BRTc), when applied to real world problems, and compare with the competitors presented in Section 5.3. In this case, different portions of the dataset are considered for training (80%, 60%, 40% and 20%) and the rest is gathered for performance evaluation (test set). The resulting tables will show the average error and standard deviation over 20 independent runs. The best values are highlighted in bold face and the symbol "*" indicates statistical improvement over the competitors measured by paired t-tests at 5% significance level. Finally, for regression and binary classification problems, the error metric is RMSE and CE, respectively.

Running time analysis (Section 6.3)

In this experiment, the intention is to assess the running time of each HIA (HMTL-PC, HMTL-BHC and HMTL-BRT) varying the number of objects to be clustered, that is, varying the number of tasks. In this case, only the procedures which obtain the hierarchical structure are considered and their necessary hyper-parameters are given a priori. In order to perform the evaluation, synthetic datasets are generated on the fly with the number of tasks within the interval $[10, 100]$ and increasing in a 10-step increment. Each task is composed of 100 observations described by 10 features and the entire set was generated using the same generative process done for the synthetic dataset Ds_2. The results will be presented in terms of average running time for each HIA over 10 independent runs.

Cost function analysis (Section 6.4)

In this experiment, the intention is to assess the convergence behavior of the proposed cost function presented in Section 4.5. A comparison is made between LowRank and HMTL-PC, given that HMTL-PC is the HIA which guides to the higher number of hypothetical tasks. The results are presented in terms of evolution of the cost function over iterations during the optimization of the task parameter vectors via AGM. In this experiment, the HMTL framework is simply referred to as HMTL and the Landmine dataset is the selected problem.

Regularization parameter analysis (Section 6.5)

In this experiment, the intention is to assess the influence of the regularization parameter λ in the proposed cost function presented in Section 4.5. All four synthetic datasets are considered, allowing different scenarios to be analyzed and compared. It is expected a higher regularization influence on cases where tasks are similar. Additionally, only HMTL-PC is employed in this experiment, that is, the evaluation will be performed considering a single hierarchical structure. More specifically, 40% of all training data are reserved exclusively for performance evaluation (test set) in order to provide representative performance results and, with the rest, quantities of size 100 are gathered for training over 10 independent runs. Regarding the λ candidates, a grid search composed of 100 values uniformly spaced within the interval $[0.01, 200]$ is employed. The results will be presented in terms of average RMSE on the test set obtained from the runs when varying λ and the best selected λ is highlighted in each case.

5.6 Chapter summary

This chapter presented all the necessary information regarding the experimental setup. Since the focus of the experiments is on regression and binary classification problems, three evaluation metrics used for these types of problems were defined, where two of them are employed for measuring performance (RMSE and CE) and one is employed as a similarity metric (Recall). Moreover, synthetic and real world datasets were detailed, and all the competitors that are going to participate in the experiments are briefly described. Finally, specific settings of the proposed approaches are suggested followed by a summary of the experimental plan.

Chapter 6

Results and discussion

The proposed HMTL framework is analyzed in this chapter, where several experiments are performed following the experimental plan discussed in Section 5.5. Since the framework is based on obtaining a hierarchy directly from task data, performing a structure validation represents a reasonable first step. After that, different experiments focusing on performance evaluation are discussed in detail. Finally, analyses are shown regarding the running time to obtain the hierarchical structure by each HIA, the behavior of the proposed cost function and the regularization influence on distinct scenarios.

6.1 Structural analysis

6.1.1 Ds_1 - Independent tasks

As presented in Figures 6.1, 6.2(a) and 6.3(a), HIAs forcing a single structure identified their own view of task relationship. Some tasks were put close by all HIAs, like occurred with pairs (7,9) and (0,2). Although dataset Ds_1 is composed of independent tasks, building a single structure keeps the most similar tasks close to each other, or, in this case, keeps the least different tasks close to each other, once the tasks are not equally independent among each other. Therefore, a minor influence of task relations is expected when employing a single hierarchical structure as regularization to the problem composed of independent tasks.

Figures 6.2(b) and 6.3(b), on the other hand, show that HIAs admitting cuts in the structure correctly identified that all tasks are independent, information that may be helpful for users interested in performing a structural analysis of the problem. Clearly, such behavior will cause each task to be learned independently, that is, the framework will not apply any structural regularization. So, MTL frameworks endowed with such structural identification mechanisms can be interpreted as having STL as a particular case, thus working properly for datasets characterized by the existence of independent tasks. Moreover, negative transfer is expected to be mitigated, since the tasks would not be directly connected, but linked to an upper-level hypothetical task whose parameter vector agrees with all its child parameter vectors after the cost function optimization.

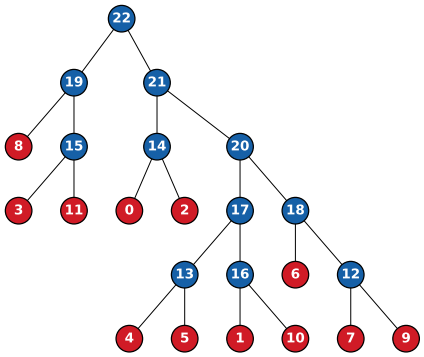
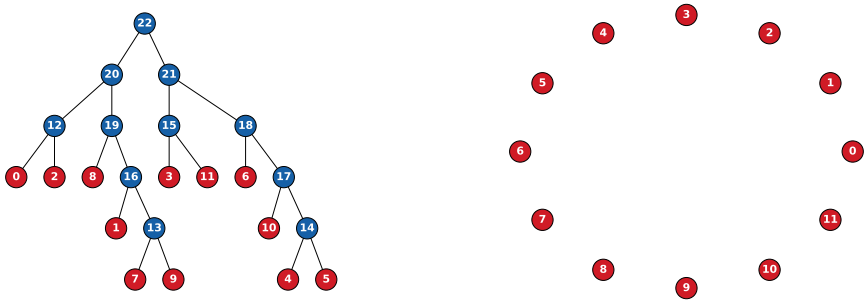
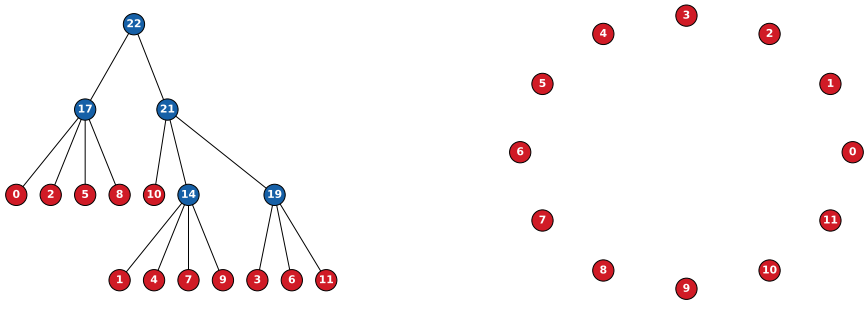


Figure 6.1 – Hierarchical structure produced by HMTL-PC for dataset Ds_1. Although the tasks are independent, HMTL-PC identified the best possible relations according to its own strategy.



(a) HMTL-BHC (b) HMTL-BHCc

Figure 6.2 – Hierarchical structures produced by HMTL-BHC and HMTL-BHCc for dataset Ds_1. A single structure is obtained in (a), whereas the cut is allowed in (b). Notice that HMTL-BHCc ultimately considered all tasks independent, once no relation was identified.



(a) HMTL-BRT (b) HMTL-BRTc

Figure 6.3 – Hierarchical structures produced by HMTL-BRT and HMTL-BRTc for dataset Ds_1. A single structure is obtained in (a), whereas the cut is allowed in (b). Similarly to HMTL-BHC, (b) shows that all tasks are ultimately treated as independent tasks.

6.1.2 Ds_2 - Similar tasks

As already verified in the case of dataset Ds_1, each HIA identified their own view of task relationship for dataset Ds_2 as well. Focusing on HMTL-PC, such approach preferred to progressively aggregate the tasks in a cascade structure as shown in Figure 6.4, which makes sense to the proposed algorithm, since adding more data from similar tasks to those already existent is the best option at each iteration.

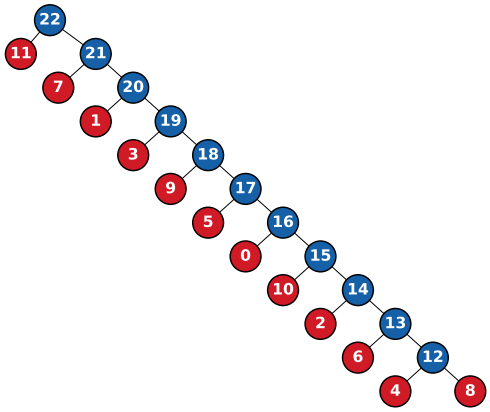
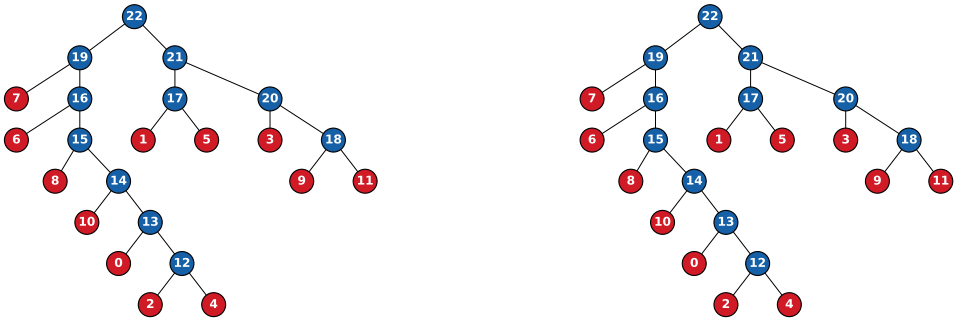


Figure 6.4 – Hierarchical structure produced by HMTL-PC for dataset Ds_2. A cascade was identified by HMTL-PC, since adding more data from similar tasks to those already existent is the best option at each iteration.

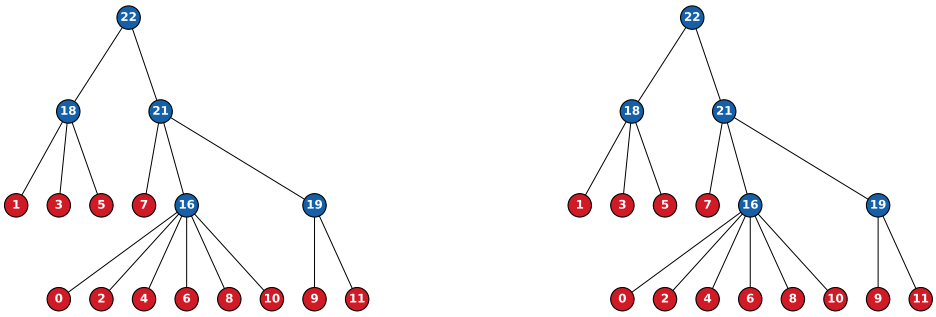
Figures 6.5(b) and 6.6(b) show that HIAs admitting cuts in the structure chose to keep a single structure rather than applying cuts, since the tasks are similar and splitting them is not appropriate. Such behavior occurs due to validations performed when optimizing the hyper-parameters for Bayesian-based HIAs, which also validates the obtained structure within the context of MTL, not only within the context of hierarchical clustering. Additionally, the same structures were obtained by Bayesian-based HIAs that force a single structure, as presented in Figures 6.5(a) and 6.6(a), compared to Figures 6.5(b) and 6.6(b), respectively. Obviously, such behavior occurs once HMTL-BHCc becomes equivalent to HMTL-BHC when the cut is not used, similar to what occurs with BRT-based HIAs.

Finally, it is possible to observe that HIAs obtaining a binary structure needed 11 internal nodes to describe the hierarchical structure, whereas BRT-based HIAs, which are not restricted to a binary structure, needed only 5 internal nodes to perform the same job, a much simpler view of task relations following the Occam's razor principle. Nevertheless, performance comparison will show that there are circumstances where a higher granularity provided by a binary structure is better than employing a simpler non-binary structure, since there are more levels of sharing to leverage.



(a) HMTL-BHC (b) HMTL-BHCc

Figure 6.5 – Hierarchical structures produced by HMTL-BHC and HMTL-BHCc for dataset Ds_2. (a) obtains a single structure, whereas the cut is allowed in (b). Notice that HMTL-BHCc chose to keep a single hierarchical structure, once the tasks are similar.



(a) HMTL-BRT (b) HMTL-BRTc

Figure 6.6 – Hierarchical structures produced by HMTL-BRT and HMTL-BRTc for dataset Ds_2. (a) obtains a single structure, whereas the cut is allowed in (b). Similarly to HMTL-BHCc, (b) shows that HMTL-BRTc chose to keep all tasks in the same structure.

6.1.3 Ds_3 - Clusters of tasks

Focusing on the dataset Ds_3, it is clear that all HIAs correctly identified the presence of 3 clusters of tasks. As the resulting structures are necessarily binary, HIAs illustrated in Figures 6.7 and 6.8(a) joined 2 clusters before composing with the third one, matching the spatial observation seen in the 2D projection illustrated in Figure 5.1(c). Small differences are detected when comparing the order in which the leaves were clustered. On the other hand, Figure 6.9(a) shows that HMTL-BRT considered all clusters equally in the same level. Finally, Figures 6.8(b) and 6.9(b) show that HIAs admitting cuts in the structure performed a cut as expected, resulting in a forest of structures rather than a single one.

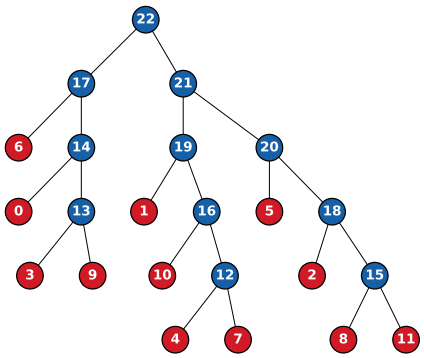
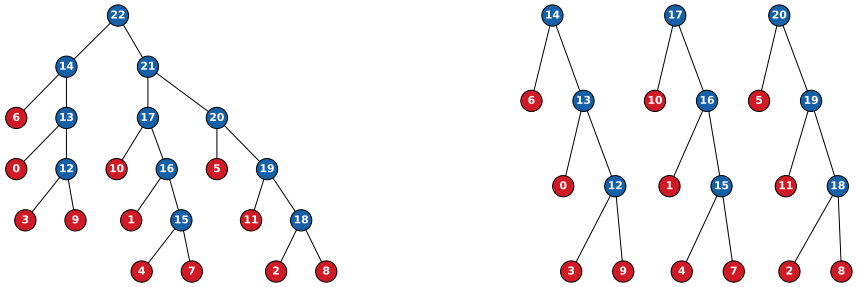
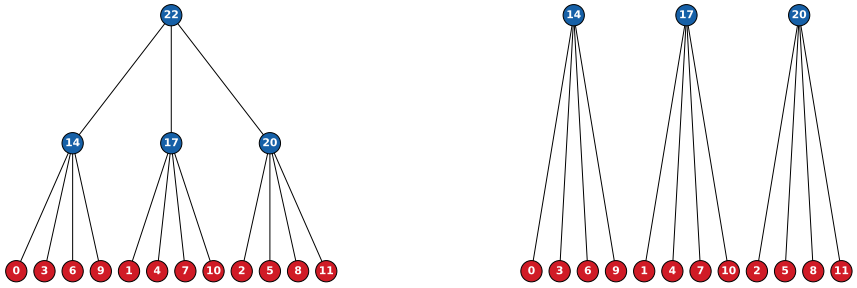


Figure 6.7 – Hierarchical structure produced by HMTL-PC for dataset Ds_3. Even though it is a single structure, tasks belonging to the same cluster are closer to each other, thus revealing the presence of three clusters.



(a) HMTL-BHC (b) HMTL-BHCc

Figure 6.8 – Hierarchical structures produced by HMTL-BHC and HMTL-BHCc for dataset Ds_3. (a) obtains a single structure, whereas the cut is allowed in (b). Notice that HMTL-BHCc correctly applied the cut in the structure, splitting the clusters so that each cluster has its own structure.



(a) HMTL-BRT (b) HMTL-BRTc

Figure 6.9 – Hierarchical structures produced by HMTL-BRT and HMTL-BRTc for dataset Ds_3. (a) obtains a single structure, whereas the cut is allowed in (b). Similarly to HMTL-BHCc, (b) shows that HMTL-BRTc applied the cut as expected.

In a scenario similar to Ds_3, it is expected that properly splitting the clusters guide to better predictive performance compared to the usage of single structures, since the influence of the hypothetical tasks higher in the hierarchy is removed, which also removes the attraction between distinct clusters during the cost function optimization. Although HMTL-BRTc provided a much simpler structure than HMTL-BHCc, only the experiments will tell us which one is the better: many levels of sharing provided by HMTL-BHCc or a single level of sharing provided by HMTL-BRTc.

6.1.4 Ds_4 - Clusters of tasks plus independent tasks

Similarly to dataset Ds_3, all HIAs correctly identified the structural configuration of dataset Ds_4, that is, tasks belonging to the same cluster were kept closer to each other and the independent tasks were the last ones to be attached to the hierarchical structure. In spite of this, Figures 6.10, 6.11(a) and 6.12(a) show that HIAs forcing a single structure identified distinct ways to position the independent tasks. Figures 6.11(b) and 6.12(b), on the other hand, show that HIAs admitting cuts in the structure correctly separated the clusters and the independent tasks, confirming that they are able to both identify the clusters and the independent tasks. In a real scenario, independent tasks may act as outliers, and users could take actions regarding outlier tasks after analyzing the obtained hierarchical structure. Moreover, employing the structures produced by HIAs allowing cuts will treat separately each independent task, demanding specific regularization techniques in order to avoid overfitting.

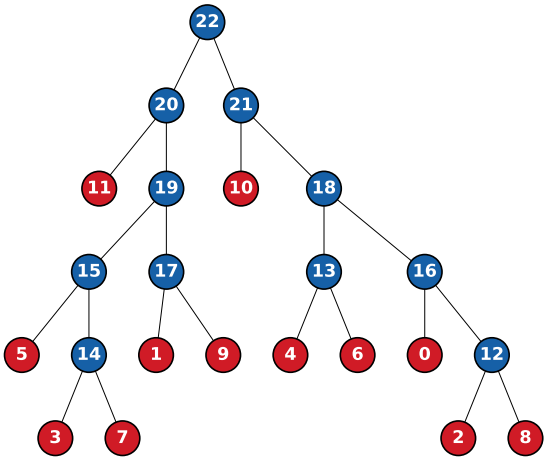
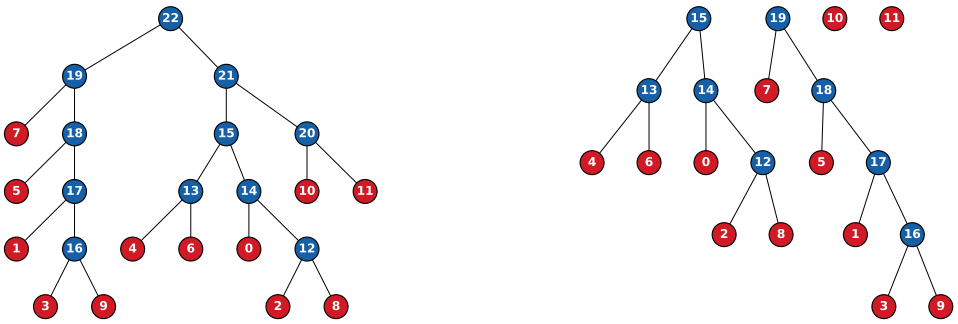
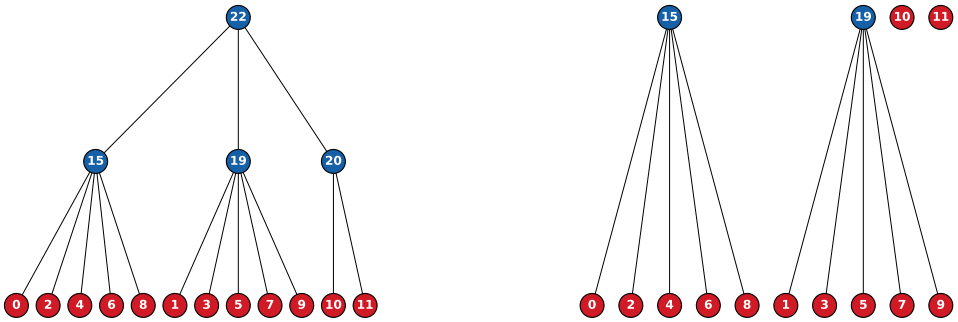


Figure 6.10 – Hierarchical structure produced by HMTL-PC for dataset Ds_4. Tasks from the same cluster are kept closer to each other, while independent tasks are considered only at the last iterations.



(a) HMTL-BHC (b) HMTL-BHCc

Figure 6.11 – Hierarchical structures produced by HMTL-BHC and HMTL-BHCc for dataset Ds_4. (a) obtains a single structure, whereas the cut is allowed in (b). Notice that (b) correctly isolated the independent tasks and the clusters.



(a) HMTL-BRT (b) HMTL-BRTc

Figure 6.12 – Hierarchical structures produced by HMTL-BRT and HMTL-BRTc for dataset Ds_4. (a) obtains a single structure, whereas the cut is allowed in (b). Similarly to HMTL-BHCc, (b) shows that HMTL-BRTc applied the cut as expected.

6.2 Performance analysis

Previous section showed distinct views of task relationship produced by the HIAs, considering synthetic datasets with different configurations. In this section, the entire HMTL framework is exercised, that is, the experiments not only consider obtaining the hierarchical structure, but also use such structural information in the regularization of the learning tasks. Both synthetic and real world datasets are employed in the experiments, and the goal is to evaluate performance in terms of predictive error on data unseen during training. The results are presented and discussed, pointing out important insights for those who want to understand, use or even extend the proposed HMTL framework.

6.2.1 Varying training data size

6.2.1.1 Ds_1 - Independent tasks

With a quick scan over Figure 6.13, it is possible to observe that all regularized formulations yielded better performance than training each task isolated, in the case where the tasks are independent. The performance improvement is not large and such behavior is expected in this scenario, since exchanging dissimilar information would not help the tasks achieving better models. Additionally, negative transfer is not expected to occur, since the regularization influence of the methods are controlled by a hyper-parameter. Nevertheless, all HIAs that obtain a single structure consistently presented better performance than popular regularization techniques, showing that structural regularization pays off more than arbitrarily performing feature selection (LASSO) or controlling the model complexity (Ridge), behavior that will be confirmed in all further scenarios. In addition, HIAs allowing the structure to be cut correctly identified the independence of the tasks, since their performance is equivalent to STL. In practice, results like this may be helpful to users devoted to a deeper understanding of the problem.

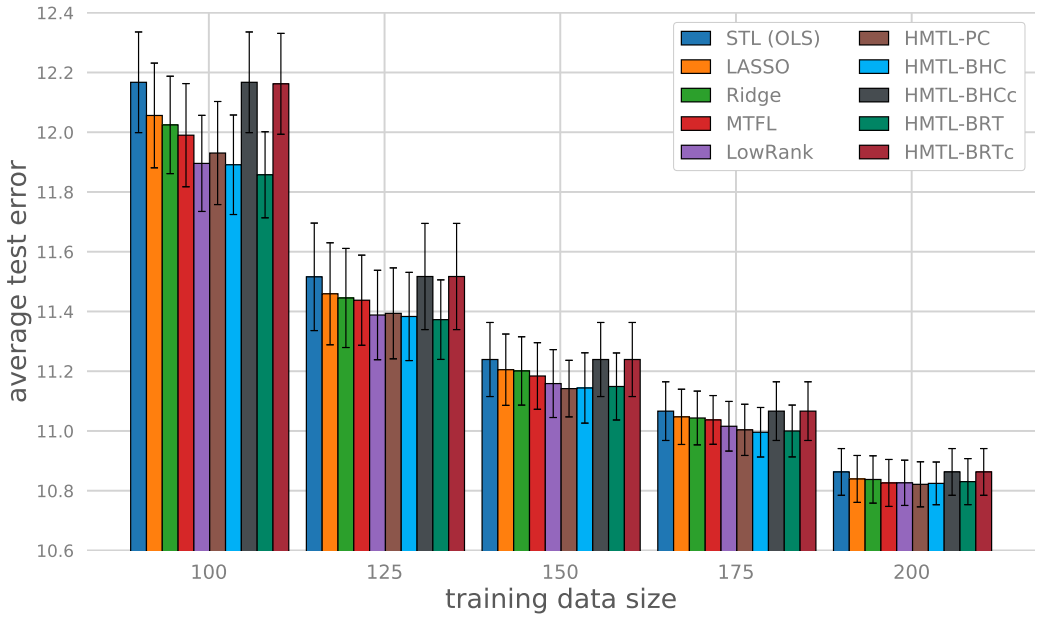


Figure 6.13 – Performance comparison in terms of average RMSE and standard deviation in the test set over 20 independent runs focusing on all tasks of dataset Ds_1. HIAs allowing cuts are equivalent to STL, whereas HIAs forcing a single structure present better predictive performance than competitors.

6.2.1.2 Ds_2 - Similar tasks

Results presented in Figure 6.14 show that using a hierarchical structure as regularization largely outperformed competitors in the case where all tasks are similar. Clearly, a single structure is more suitable than a forest of structures, once all tasks are similar and may positively influence each other in order to produce better models. Moreover, only the competitor LowRank presented a significant improvement over STL, but it was not capable of outperforming the HIAs, especially HMTL-BRT, which presented the best average performance. Finally, all HIAs yielded similar performance, including the ones that allow cuts in the structure. As seen in Section 6.1.2 when dealing with dataset Ds_2, both HMTL-BHCc and HMTL-BRTc preferred not to cut the hierarchy. In practice, such behavior strongly indicates that tasks are similar, since the cut is not used.

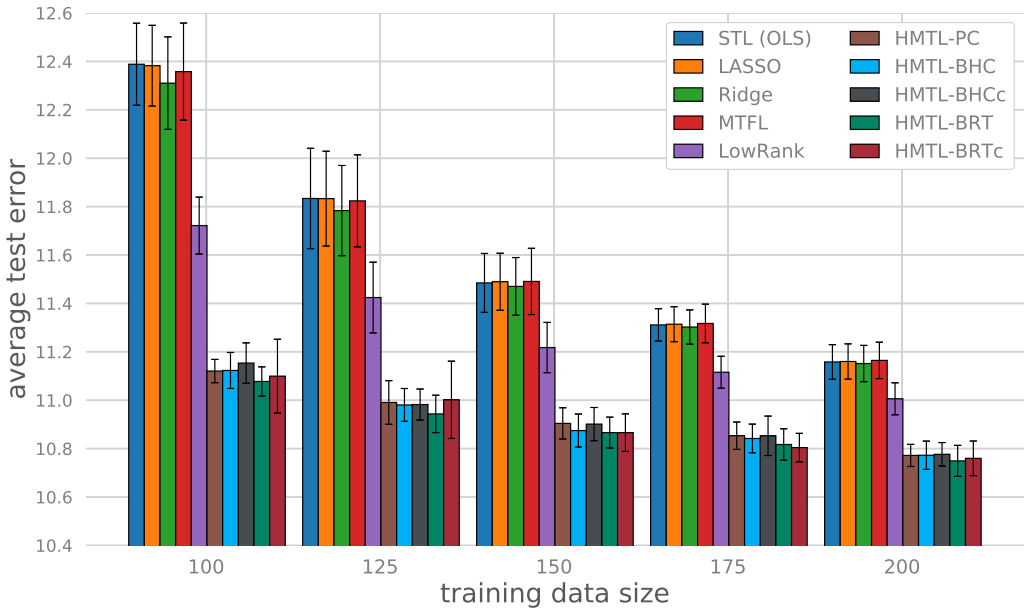


Figure 6.14 – Performance comparison in terms of average RMSE and standard deviation in the test set over 20 independent runs focusing on all tasks of dataset Ds_2. All HIAs present equivalent performance and a clear performance improvement over the competitors.

6.2.1.3 Ds_3 - Clusters of tasks

In general, Figure 6.15 shows that using a single hierarchical structure yielded better results than competitors from the literature, even considering that dataset Ds_3 has clusters of tasks. Similarly to results presented for datasets Ds_1 and Ds_2, all HIAs largely presented better performance than both STL and the well-known regularization techniques (Ridge and LASSO). Additionally, it is possible to observe that HIAs allowing cuts in the structure yielded better results than the other HIAs, since the influence of

higher nodes in the hierarchy is removed and the clusters are properly separated. Finally, a consistent performance improvement is presented by HMTL-BHCc over competitors, validating the usage of a forest rather than a single structure when dealing with problems composed of clusters of tasks, and also indicating that using a binary structure in this case is more favorable than a non-binary structure.

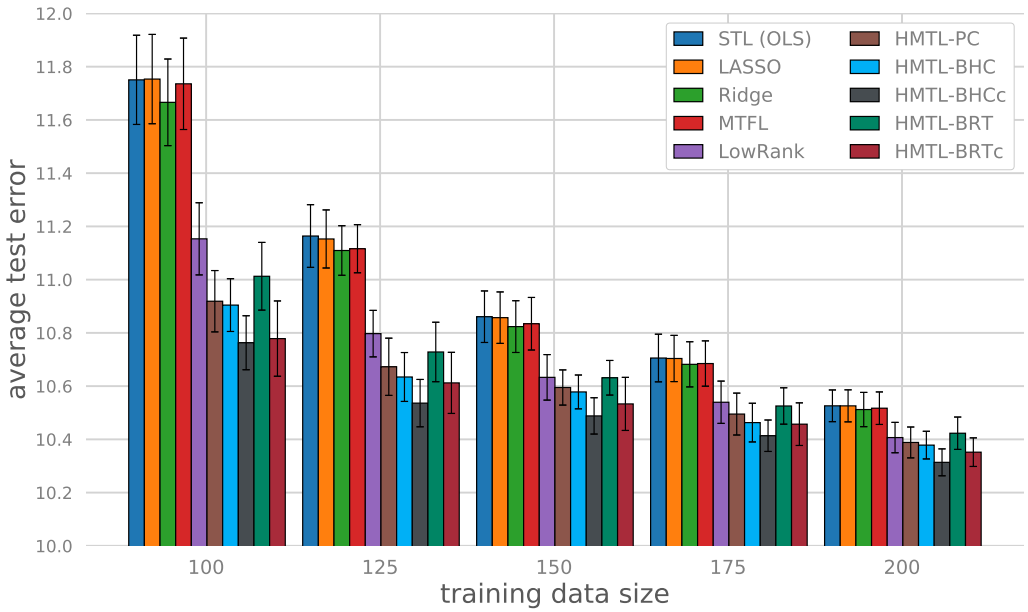


Figure 6.15 – Performance comparison in terms of average RMSE and standard deviation in the test set over 20 independent runs focusing on all tasks of dataset Ds_3. As expected, HIAs allowing the structure to be cut present better performance than other HIAs. Moreover, a binary structure is better for this particular problem than a non-binary structure.

6.2.1.4 Ds_4 - Clusters of tasks plus independent tasks

Similarly to dataset Ds_3, results presented in Figure 6.16 show that using a single structure yielded good results in the case where independent tasks are present. In spite of this, correctly identifying the aspects of the problem through the cuts yielded even better results. Moreover, the difference between all HIAs and STL is the lowest, when compared to this difference for datasets Ds_2 and Ds_3. It indicates that individual tasks (tasks 10 and 11) are degrading the overall performance. Finally and once more, properly identifying the clusters of tasks and independent tasks yielded the best results compared to the competitors. In practice, the structure produced by HIAs that allow the structure to be cut may provide important insights to the user, including a better management of the independent tasks identified via the cuts.

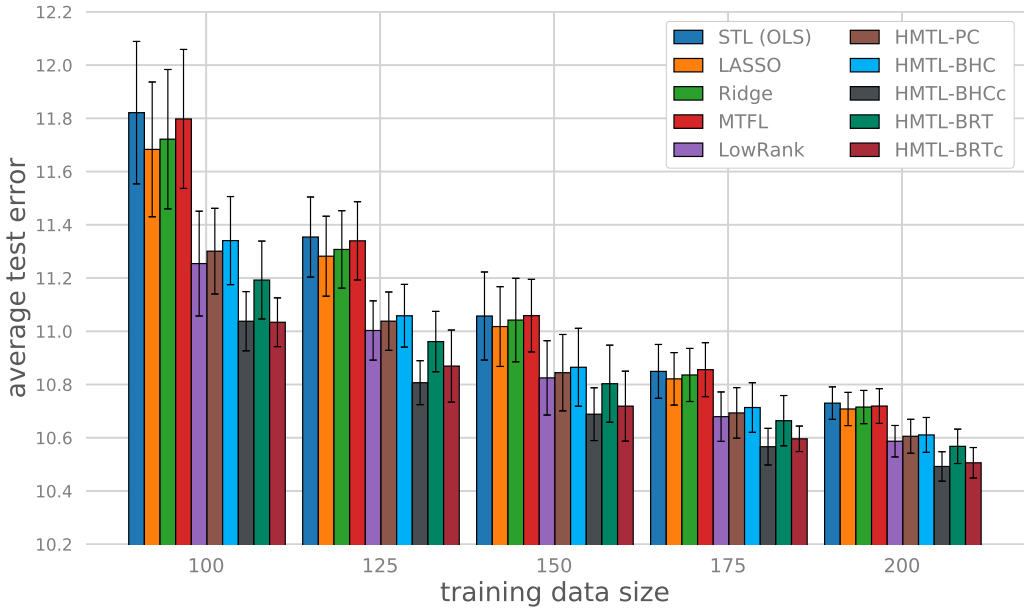


Figure 6.16 – Performance comparison in terms of average RMSE and standard deviation in the test set over 20 independent runs focusing on all tasks of dataset Ds_4. As expected, HIAs allowing the structure to be cut present better performance than other HIAs. Similarly to Ds_3, a binary structure is better for Ds_4 than a non-binary structure.

6.2.2 Random structures

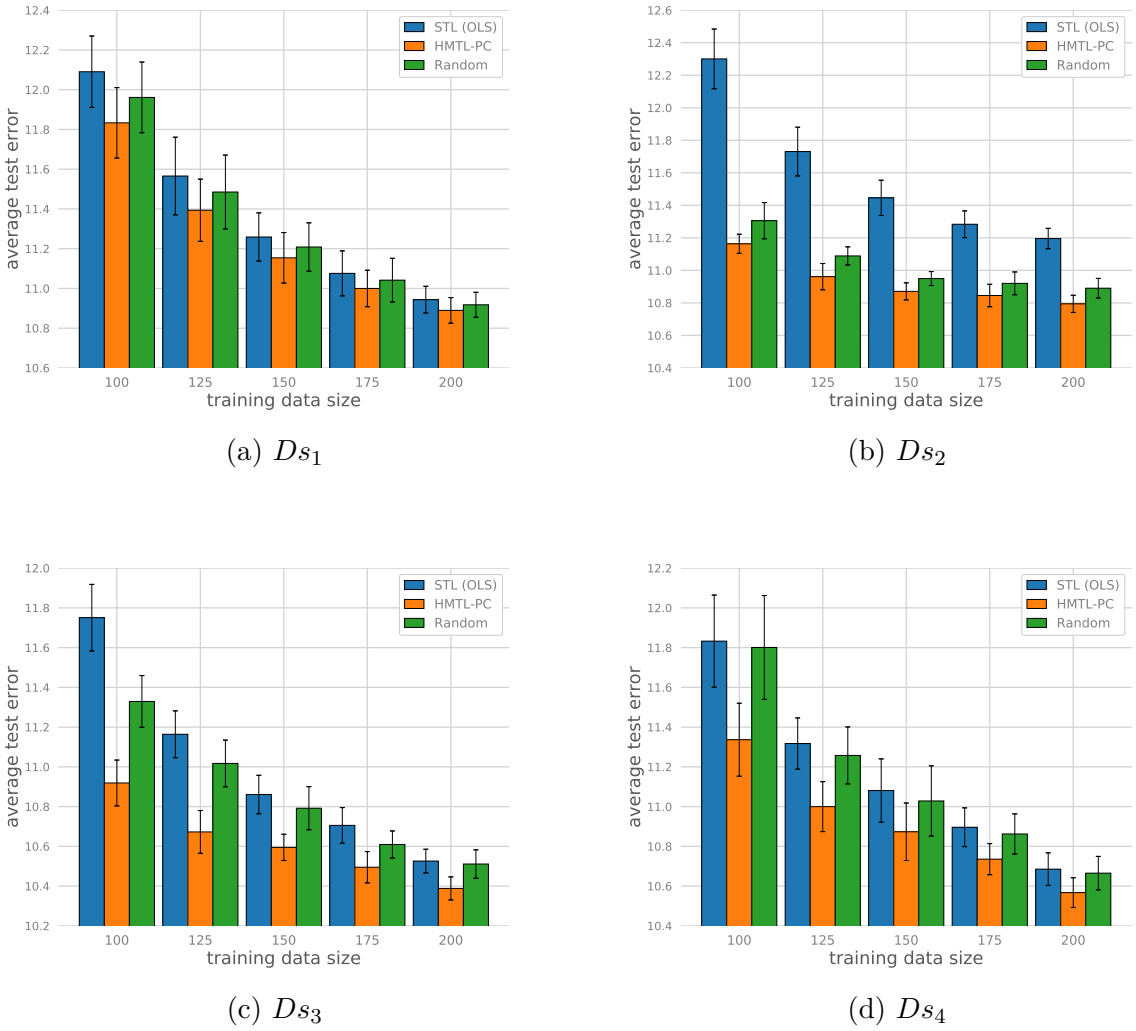
Previously discussed results showed a performance improvement over STL when employing a single hierarchical structure, even in situations where the true task relationship exhibit sparse relations, as in datasets Ds_1, Ds_3 and Ds_4. Moreover, distinct hierarchical structures obtained by the HIAs yielded similar performance on datasets Ds_1 and Ds_2.

Although each HIA identified its own task structure due to different similarity metrics, all single structures have, at least, the root node in common. Hence, a scenario with different structures yielding similar performance motivates an experiment to assess a randomly generated structure, since it also presents a single structure and the root node is the same as in the HIAs.

Overall, Figure 6.17 shows *Random* appearing in between HMTL-PC and STL. The worst performance is observed in Figures 6.17(c) and 6.17(d), since datasets Ds_3 and Ds_4 are composed of specific tasks that should be together and others that should not. Despite this, *Random* presented better performance than STL in all scenarios, indicating that regularizing the tasks by a random structure acted positively.

In Figures 6.17(a) and 6.17(b), on the other hand, it is possible to observe that *Random* presented its performance closer to HMTL-PC. Once the tasks are either all independent or all similar in datasets Ds_1 and Ds_2, respectively, relating any task

with others yielded reasonable performance, but HMTL-PC always presented superior performance, indicating that identifying the relationship via a similarity metric is better than the random structure, even considering that both strategies have the root node in common.



(a) Ds_1 (b) Ds_2

(c) Ds_3 (d) Ds_4

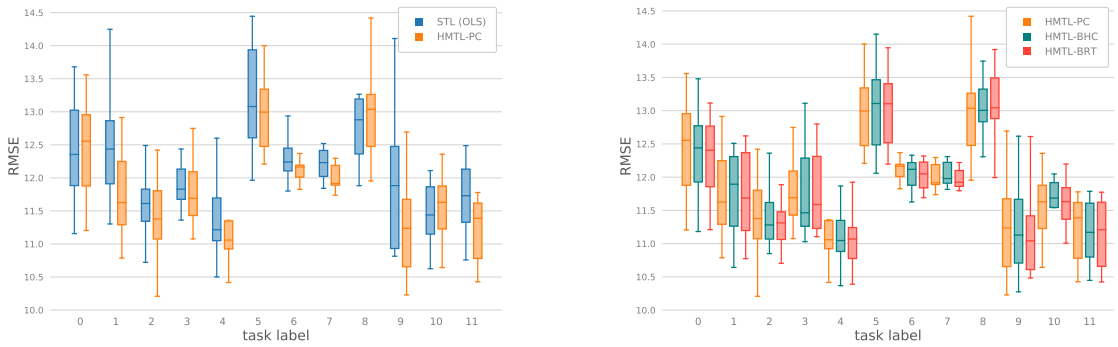
Figure 6.17 – Performance comparison in terms of average RMSE and standard deviation in the test set over 20 independent runs focusing on the synthetic datasets to assess the behavior of randomly generated structures. Although both HMTL-PC and *Random* obtain a single structure and have the root node in common, HMTL-PC presents the best performance due to the possibility of capturing task relations in the structure.

This experiment also illustrates that regularizing tasks by any hierarchical structure yields performance either equivalent or superior to what is achieved by isolated training, motivating the proposition of distinct HIAs. Moreover, once the regularization influence is controlled by a penalty parameter, it is expected that a random structure will yield a performance equivalent to the one achieved by STL, avoiding compromising performance by negative transfer.

6.2.3 Each task

6.2.3.1 Ds_1 - Independent tasks

Figure 6.18(a) shows that the slight performance improvement obtained by HMTL-PC when compared to STL comes from some tasks (*e.g.* tasks 2 and 11), while the performance of others are equivalent to what is achieved by STL (*e.g.* tasks 5 and 10). Thus, employing the hierarchy as regularization contributes even in the case where tasks are independent, because the structure may identify tasks that benefit from regularization hints provided by other tasks. Figure 6.18(b) shows that HIAs admitting a single structure yielded equivalent performance, even considering that each HIA produced a different view of the hierarchical structure, as seen in Section 6.1.



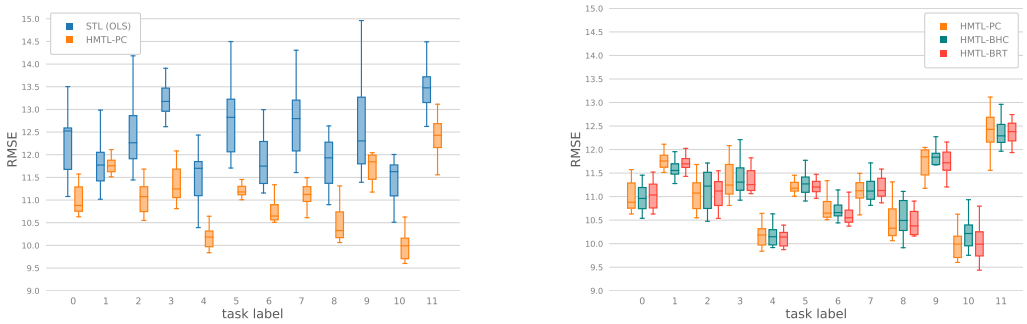
(a) (b)

Figure 6.18 – Performance comparison in terms of RMSE in the test set over 20 independent runs focusing on each task from dataset Ds_1. Even for independent tasks, employing a single hierarchical structure yields a slightly performance improvement over STL.

6.2.3.2 Ds_2 - Similar tasks

In Figure 6.19(a), the performance improvement of HMTL-PC over STL is clear for all tasks except task 1. Nevertheless, this is the most favorable situation to employ a single hierarchical structure, since all tasks are related. Similarly to Ds_1, Figure 6.19(b) shows that HIAs characterized by a single hierarchical structure yielded equivalent performance even founded on distinct hierarchical structures.

Although employing distinct hierarchies produced similar results, all structures share, at least, the same root node. Thus, any HIA (not limiting to the ones proposed in this dissertation) may provide competitive results, once all tasks are similar and can influence each other positively. Despite this, considering all tasks as being similar is not a popular hypothesis anymore and the MTL literature tends to identify a sparse task relationship, as discussed in Section 2.4.



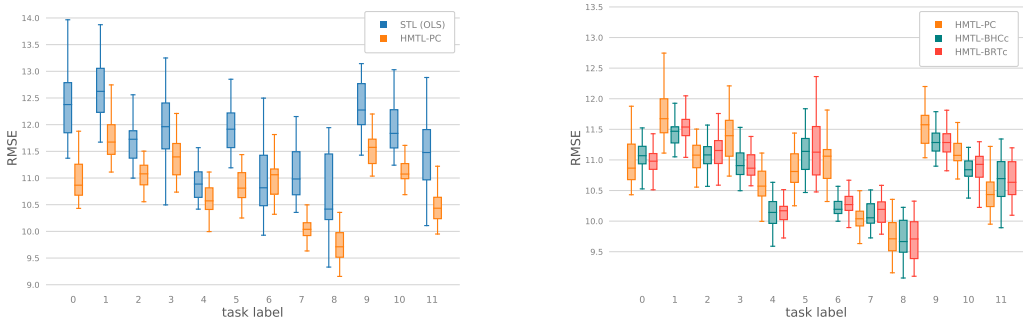
(a) (b)

Figure 6.19 – Performance comparison in terms of RMSE in the test set over 20 independent runs focusing on each task from dataset Ds_2. Even all HIAs having obtained distinct hierarchical structures, in this scenario they yield equivalent performance for each task, as shown in (b).

6.2.3.3 Ds_3 - Clusters of tasks

Figure 6.20(a) shows that using a single structure yielded better performance than STL for all tasks except task 6. Clearly, the performance improvement is not as large as in dataset Ds_2, because this setting is composed of clusters of tasks and all tasks ended up being linked in the same hierarchy. In spite of this, HMTL-PC keeps the tasks from the same cluster closer in the hierarchical structure, which contributes to the performance improvement of individual tasks.

Figure 6.20(b) shows the comparison among HMTL-PC, HMTL-BHCc and HMTL-BRTc to evaluate the behavior when the cut is permitted. It is possible to observe that, overall, HIAs that allow the cut yielded better performance for some tasks (*e.g.* tasks 3, 4 and 6), since tasks from different clusters do not influence each other, while others presented equivalent performance among the tested HIAs (*e.g.* tasks 2, 7, and 8).



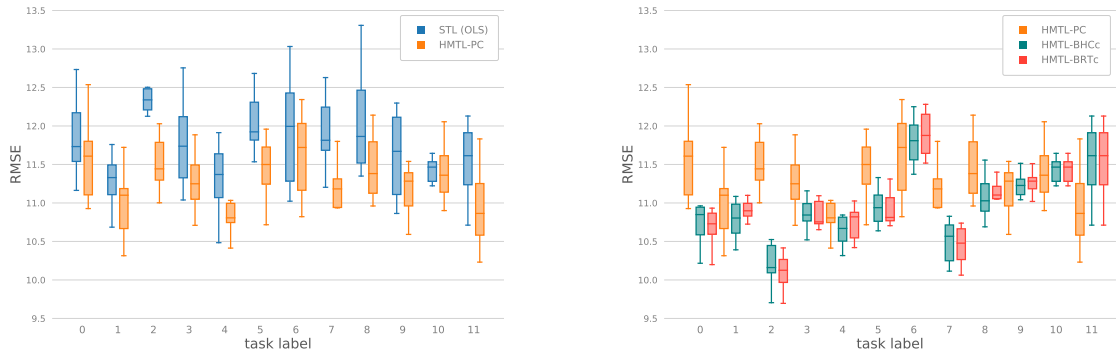
(a) (b)

Figure 6.20 – Performance comparison in terms of RMSE in the test set over 20 independent runs focusing on each task from dataset Ds_3. In general, HIAs that allow the structure to be cut present a better performance than HMTL-PC, since the clusters are properly separated, with a subtle advantage for HMTL-BHCc.

6.2.3.4 Ds_4 - Clusters of tasks plus independent tasks

Once more, Figure 6.21(a) shows that using a single hierarchical structure obtained by HMTL-PC yielded better performance when compared to STL, even considering the scenario composed of clusters of tasks and independent tasks. Similarly to Section 6.2.3.3, Figure 6.21(b) shows that properly applying the cuts yielded better performance for some tasks (*e.g.* tasks 0, 2 and 7), contributing in providing the best results for HIAs that allow the structure to be cut, while others presented equivalent performance among the tested HIAs (*e.g.* tasks 1, 6 and 9).

Regarding the performance of the independent tasks (tasks 10 and 11) for HMTL-BHCc and HMTL-BRTc, it is possible to notice, comparing Figures 6.21(a) and 6.21(b), that they are equivalent to the STL performance, which makes sense, since they are treated as isolated tasks. Finally, task 11 presented better performance for HMTL-PC than both STL and HIAs admitting cuts, even being independent, because the hierarchical structure worked positively in the regularization, behavior not observed for task 10. This last observation motivates the investigation of novel approaches for MTL, characterized by directed relations among tasks, so that task 11 may be regularized by the hierarchical structure, but tasks in the hierarchical structure are not influenced by task 11.



(a) (b)

Figure 6.21 – Performance comparison in terms of RMSE in the test set over 20 independent runs focusing on each task from dataset Ds_4. In this case, HIAs allowing the structure to be cut properly separate the clusters and independent tasks, resulting in performance improvement for some tasks when compared to HMTL-PC.

6.2.4 Real world datasets

6.2.4.1 School dataset

The School dataset comprises a scenario composed of several regression tasks with different sample size per task. In Table 6.1, a consistent and significant performance improvement of the HIAs over the competitors is shown. Moreover, HIAs allowing the structure to be cut did not present a performance improvement over the HIAs that obtain a single structure, indicating that the tasks are similar and all of them should be related together in the same hierarchical structure. Results also suggested that a binary structure is more suitable for this problem, once HMTL-BHC consistently presented the best results. Finally, the differences between results of HMTL-PC and HMTL-BHC, both having a binary structure, showed that a Bayesian strategy was able to capture the task relations better than the strategy used by HMTL-PC, highlighting the advantages of a probabilistic approach.

Table 6.1 – Performance comparison in terms of average RMSE and standard deviation in the test set over 20 independent runs focusing on the School dataset. Different portions of the dataset were considered as training data and the rest was employed as test data. The best values are highlighted in bold face and the symbol "*" indicates statistical improvement over the literature competitors measured by paired t-tests at 5% significance level.

School dataset				
Algorithm	Training data			
	80%	60%	40%	20%
STL (OLS)	10.49 (±0.16)	10.64 (±0.09)	11.00 (±0.08)	12.07 (±0.14)
LASSO	10.43 (±0.16)	10.56 (±0.09)	10.83 (±0.07)	11.50 (±0.07)
Ridge	10.48 (±0.16)	10.63 (±0.09)	10.95 (±0.08)	11.77 (±0.08)
MTFL	10.39 (±0.16)	10.50 (±0.08)	10.71 (±0.07)	11.16 (±0.07)
LowRank	10.37 (±0.16)	10.47 (±0.08)	10.68 (±0.07)	11.11 (±0.06)
HMTL-PC	10.32 (±0.13)*	10.40 (±0.08)*	10.56 (±0.08)*	10.89 (±0.06)*
HMTL-BHC	**10.23 (±0.15)***	**10.28 (±0.07)***	**10.41 (±0.07)***	**10.68 (±0.07)***
HMTL-BHCc	10.26 (±0.14)*	10.33 (±0.11)*	10.51 (±0.12)*	10.84 (±0.12)*
HMTL-BRT	10.24 (±0.15)*	10.31 (±0.09)*	10.44 (±0.08)*	10.73 (±0.08)*
HMTL-BRTc	10.31 (±0.15)*	10.42 (±0.15)*	10.59 (±0.14)*	10.87 (±0.18)*

6.2.4.2 Spam3 dataset

The Spam3 dataset comprises a scenario composed of only three binary classification tasks with a large amount of available data per task. Clearly, the hypothesis space of hierarchical structures is small, once only few hierarchical structures are possible

to be derived from three objects. Thus, it is expected a small difference among results of the HIAs. Table 6.2 shows that HMTL-BRT provided the best results on most cases, indicating that a non-binary structure is more suitable for this problem. On the other hand, HIAs that obtain a binary structure yielded better performance when limiting the training data to 20%. Overall, the Bayesian approaches presented similar or better results than LowRank, the best competitor. Finally, the results indicated that all tasks are related, since HIAs allowing cuts in the structure did not present a performance improvement over HIAs that obtain a single structure.

Table 6.2 – Performance comparison in terms of average CE and standard deviation in the test set over 20 independent runs focusing on the Spam3 dataset. Different portions of the dataset were considered as training data and the rest was employed as test data. The best values are highlighted in bold face and the symbol "*" indicates statistical improvement over the literature competitors measured by paired t-tests at 5% significance level.

Spam3 dataset				
Algorithm	Training data			
	80%	60%	40%	20%
STL (LR)	2.29 (±0.36)	2.63 (±0.29)	3.09 (±0.31)	3.86 (±0.57)
LASSO	1.94 (±0.26)	2.13 (±0.22)	2.52 (±0.16)	3.40 (±0.35)
Ridge	1.97 (±0.31)	2.13 (±0.25)	2.44 (±0.21)	3.21 (±0.22)
MTFL	1.95 (±0.29)	2.13 (±0.22)	2.53 (±0.19)	3.38 (±0.28)
LowRank	1.86 (±0.29)	2.00 (±0.26)	2.37 (±0.20)	3.01 (±0.39)
HMTL-PC	1.92 (±0.26)	2.04 (±0.22)	2.33 (±0.23)	2.80 (±0.41)*
HMTL-BHC	1.85 (±0.31)	2.00 (±0.21)	2.33 (±0.22)	**2.79 (±0.33)***
HMTL-BHCc	1.85 (±0.31)	2.00 (±0.21)	2.33 (±0.22)	**2.79 (±0.33)***
HMTL-BRT	**1.80 (±0.34)**	**1.92 (±0.19)**	**2.27 (±0.23)**	2.82 (±0.38)
HMTL-BRTc	**1.80 (±0.32)**	1.95 (±0.21)	2.28 (±0.23)	2.84 (±0.36)

6.2.4.3 Spam15 dataset

The Spam15 dataset comprises a scenario composed of a reasonable number of binary classification tasks with the number of features higher than the sample size per task. Similarly to results obtained for Spam3, Table 6.3 shows that admitting a non-binary structure is more suitable for this problem, though HMTL-PC presented the best performance when limiting the training data to 20%. In this case, such behavior suggests that performing a hierarchical clustering of task parameter vectors obtained from scarce training data is not able to properly identify the best relations. On the other hand, HMTL-PC leverages from limited training data, since the hierarchy levels are build from

dataset combinations. Once more, results indicated that all tasks are related, since the best results are derived from HIAs that obtain a single structure.

Table 6.3 – Performance comparison in terms of average CE and standard deviation in the test set over 20 independent runs focusing on the Spam15 dataset. Different portions of the dataset were considered as training data and the rest was employed as test data. The best values are highlighted in bold face and the symbol "*" indicates statistical improvement over the literature competitors measured by paired t-tests at 5% significance level.

Spam15 dataset				
Algorithm	Training data			
	80%	60%	40%	20%
STL (LR)	6.33 (±0.61)	7.23 (±0.60)	8.23 (±0.61)	11.03 (±0.63)
LASSO	5.69 (±0.69)	6.52 (±0.63)	7.58 (±0.59)	10.31 (±0.62)
Ridge	5.71 (±0.75)	6.57 (±0.58)	7.53 (±0.61)	9.71 (±0.52)
MTFL	5.66 (±0.70)	6.49 (±0.64)	7.54 (±0.58)	10.18 (±0.65)
LowRank	4.82 (±0.64)	5.43 (±0.49)	6.01 (±0.46)	7.87 (±0.56)
HMTL-PC	4.77 (±0.66)	5.38 (±0.47)	5.76 (±0.47)*	**7.10 (±0.76)***
HMTL-BHC	4.80 (±0.61)	5.37 (±0.63)	5.85 (±0.52)	7.27 (±0.54)*
HMTL-BHCc	4.84 (±0.73)	5.39 (±0.58)	5.85 (±0.52)	7.30 (±0.53)*
HMTL-BRT	**4.61 (±0.60)***	**5.13 (±0.58)***	**5.47 (±0.38)***	7.16 (±0.70)*
HMTL-BRTc	4.67 (±0.61)	5.21 (±0.55)*	5.52 (±0.48)*	7.22 (±0.67)*

6.2.4.4 Landmine dataset

Differently from all the other real world datasets, where the prior knowledge indicated that all tasks are related, the Landmine dataset is supposed to present a structure with two clusters of tasks. Thus, it is expected that HIAs allowing cuts in the structure outperform the others. Indeed, such behavior occurred in most cases and the results are presented in Table 6.4. Similarly to the other classification problems, HMTL-PC presented the most favorable result when limiting the training to 20%, reinforcing that dataset combinations should be promoted in scenarios characterized by scarce data available for training. Additionally, HIAs that obtain a single structure presented equivalent performance for 20% of training data compared to their corresponding versions that allow cuts, indicating that all tasks were treated in the same structure.

Table 6.4 – Performance comparison in terms of average CE and standard deviation in the test set over 20 independent runs (Landmine dataset). Different portions of the dataset were considered as training data and the rest was employed as test data. The best values are highlighted in bold face and the symbol "*" indicates statistical improvement over the literature competitors measured by paired t-tests at 5% significance level.

Landmine dataset				
Algorithm	Training data			
	80%	60%	40%	20%
STL (LR)	5.77 (±0.28)	6.02 (±0.21)	6.45 (±0.21)	8.40 (±0.47)
LASSO	5.58 (±0.19)	5.68 (±0.11)	5.75 (±0.13)	5.96 (±0.14)
Ridge	5.54 (±0.17)	5.69 (±0.13)	5.71 (±0.10)	5.85 (±0.11)
MTFL	5.51 (±0.15)	5.69 (±0.11)	5.76 (±0.12)	5.92 (±0.10)
LowRank	5.50 (±0.15)	5.66 (±0.10)	5.68 (±0.09)	5.77 (±0.08)
HMTL-PC	5.49 (±0.19)	5.64 (±0.10)	5.66 (±0.09)	**5.75 (±0.12)**
HMTL-BHC	5.49 (±0.23)	5.59 (±0.14)	5.72 (±0.10)	5.90 (±0.12)
HMTL-BHCc	**5.40 (±0.22)***	**5.55 (±0.14)***	**5.65 (±0.13)**	5.90 (±0.15)
HMTL-BRT	5.51 (±0.22)	5.66 (±0.17)	5.76 (±0.09)	6.01 (±0.18)
HMTL-BRTc	5.44 (±0.27)	5.56 (±0.17)*	5.68 (±0.13)	6.01 (±0.21)

In Figures 6.22, 6.23 and 6.24, the hierarchical structures produced by all HIAs when the training data of the Landmine dataset is limited to 80% are presented. With a quick scan over the structures, it is possible to observe that all the HIAs produced structures so that tasks from the groups $[0, 9]$ and $[10, 18]$ appear close to each other, thus revealing the presence of two clusters of tasks.

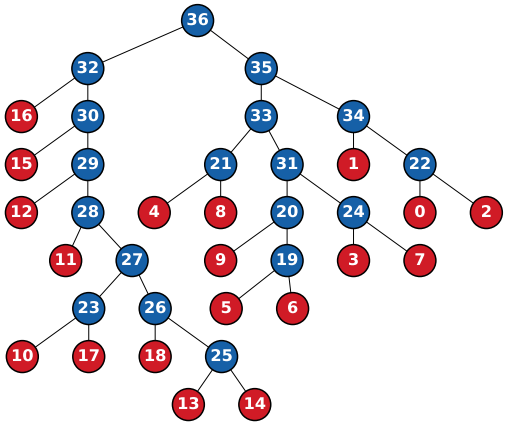
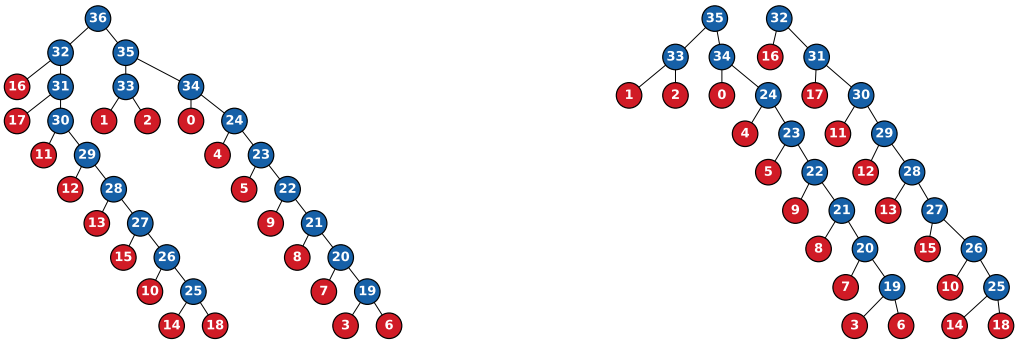
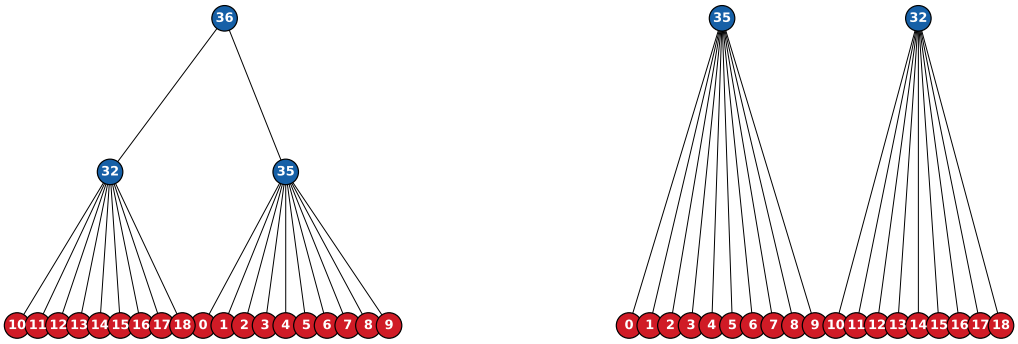


Figure 6.22 – Hierarchical structure produced by HMTL-PC for the Landmine dataset.



(a) HMTL-BHC (b) HMTL-BHCc

Figure 6.23 – Hierarchical structures produced by HMTL-BHC and HMTL-BHCc for the Landmine dataset. (a) obtains a single structure, whereas the cut is allowed in (b).



(a) HMTL-BRT (b) HMTL-BRTc

Figure 6.24 – Hierarchical structures produced by HMTL-BRT and HMTL-BRTc for the Landmine dataset. (a) obtains a single structure, whereas the cut is allowed in (b).

Overall, experiments on real world datasets showed promising results associated with the proposed framework. Indeed, regularizing task parameter vectors by means of a hierarchical structure provided the best results on distinct scenarios when compared with various competitors from the literature. As seen in the results, however, no HIA prevailed in all cases.

In the cases where one does not have any prior knowledge about the task relationship, it is indicated to test all HIAs and a careful analysis be performed in order to identify the most indicated HIA. Even though some HIAs presented better performance than others, all of them yielded good results for most of the scenarios, validating the HMTL framework.

6.3 Running time analysis

Referring to Figure 6.25, it is clear that HMTL-PC presented a higher growth rate than HMTL-BHC and HMTL-BRT, becoming impractical on situations where the number of tasks is large. In fact, such high cost is caused by the fact that HMTL-PC builds level by level the hierarchical structure, testing all possible task pairs via estimating predictors and evaluating their performance, which consumes lots of computational time. Nevertheless, as seen in the previous experiments, HMTL-PC yielded high-quality results, especially when the task set is composed of similar tasks and scarce data are available for training.

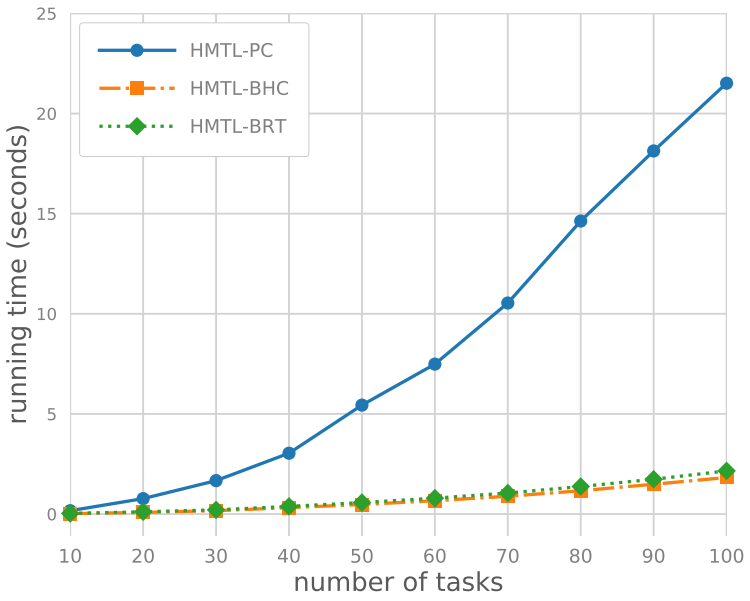


Figure 6.25 – Running time comparison among the HIAs. Bayesian-based HIAs run faster than the strategy used by HMTL-PC.

The running time presented by HMTL-BHC and HMTL-BRT, on the other hand, motivates their usage in situations where the number of tasks is large, but their repeated execution for optimizing the hyper-parameters may require additional computational time, though it can be done in parallel. Considering that the hyper-parameters have already been determined, HMTL-BHC and HMTL-BRT run fast and may be applicable under more challenging conditions. Additionally, as presented in Section 3.3.4, Heller & Ghahramani (2005b) proposed extensions to deal with a large number of objects to be clustered, which may be incorporated into the HMTL framework.

Finally, HMTL-BHC presented slightly better results than HMTL-BRT in terms of running time. This is due to the fact that HMTL-BRT performs some additional operations when deciding how to combine the nodes and, clearly, it will take more running time to be completed.

6.4 Cost function analysis

Figure 6.26 shows that LowRank converged faster than HMTL, which required twice as many iterations. However, this comparison is not totally fair. The Landmine dataset has 19 tasks and, consequently, LowRank is optimizing task parameter vectors for the same 19 tasks. On the other hand, HMTL optimizes task parameter vectors for $2n - 1$ tasks, where n is the number of original tasks. Therefore, HMTL is optimizing task parameter vectors for 37 tasks.

Thus, this experiment shows that even optimizing task parameter vectors for all tasks including the hypothetical ones, the convergence for HMTL occurs rapidly, properly exploring the convexity of the cost function. Besides, the optimization process tends to be even faster when employing hierarchical structures obtained by HMTL-BRT and HIAs that allow the structure to be cut, since the obtained hierarchical structures produced by such HIAs have a smaller number of hypothetical tasks.

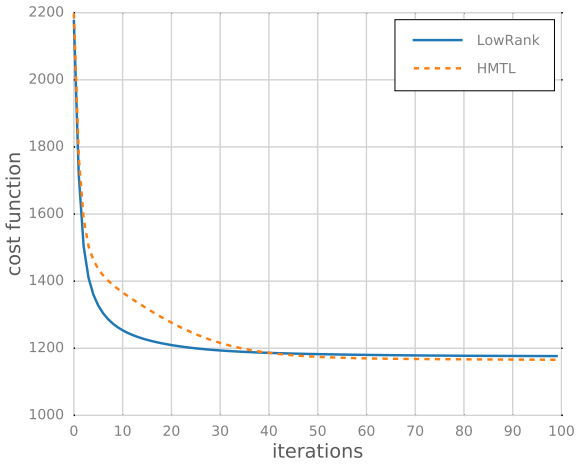
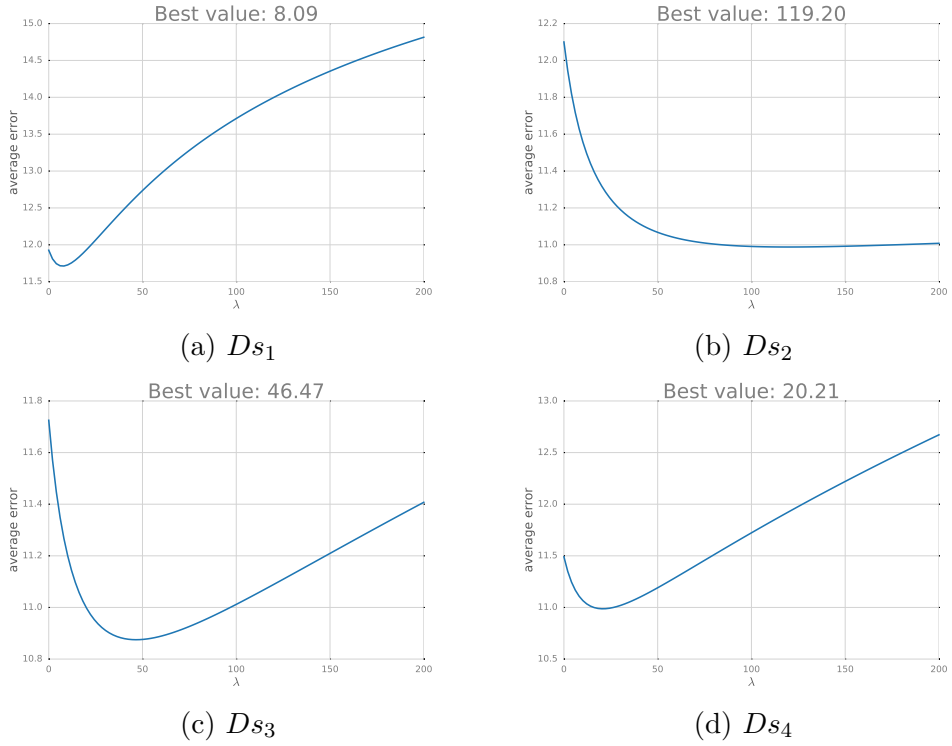


Figure 6.26 – Comparison between HMTL and LowRank applied to the Landmine dataset focusing on the cost function evolution over iterations. LowRank converges faster than HMTL, but HMTL considers both original and hypothetical tasks.

6.5 Regularization parameter analysis

The experimental analysis can be discussed in a general way by comparing the selected λs and taking into consideration the task structure of each synthetic dataset. In Figure 6.27, the highest selected λ value was used for dataset Ds_2 (Figure 6.27(b)), which makes sense, since all tasks are similar and regularizing with the hierarchical structure should have a higher influence when compared to the other scenarios. As opposed to the problem where tasks are similar, the lowest selected λ value was used for dataset Ds_1 (Figure 6.27(a)), since tasks are independent and the regularization influence should not be dominant.

Intermediary results are presented for dataset Ds_3 and Ds_4 in Figures 6.27(c) and 6.27(d), respectively. Although both datasets admit clusters of tasks, the highest selected λ value between them was used for dataset Ds_3, showing that the presence of independent tasks on dataset Ds_4 may have contributed to reduce the regularization influence, since a single hierarchical structure is connecting both the clusters and the independent tasks.



(a) Ds_1 (b) Ds_2

(c) Ds_3 (d) Ds_4

Figure 6.27 – Average RMSE when varying the regularization parameter λ. The highest regularization influence (λ best value implies minimum average error) is observed in the case where all tasks are similar, whereas the lowest influence is observed in the case where all tasks are independent.

6.6 Chapter summary

This chapter presented results of several experiments in order to assess the proposed HMTL framework. The different and controlled scenarios provided by synthetic datasets contributed to the analyses involving the obtained structures and performance comparisons. It was possible to observe that employing a hierarchical structure as regularization yielded promising results, motivating further research focusing on extensions. The entire study is interpreted as an important contribution to MTL and hierarchical clustering areas, especially focusing on Bayesian-based algorithms (BHC and BRT), since, to the best of the author's knowledge, both areas have not been studied together.

Part IV

Final considerations

Chapter 7

Conclusions and future directions

7.1 Concluding remarks

The overview presented in Chapter 2 showed the evolution of MTL approaches over the years and highlighted characteristics of the technique, especially focusing on the importance of properly identifying the task relationship to avoid the occurrence of negative transfer among tasks. In addition, the motivation behind MTL is to take advantage of both task similarities and their particularities to achieve better models.

Although the MTL literature is rich in distinct task relationship assumptions, the majority can be basically grouped into those in which all tasks are considered somehow related and those assuming a task relationship structure, such as cluster-based, graph-based and hierarchy-based. Since real world problems tend to exhibit sparse relations among tasks, it seems to be more attractive to resort to a task relationship structure.

The well-known regularization framework, which is usually applied to STL, has been largely explored in recent MTL proposals as well. In the case of MTL, distinct task relationship assumptions can be incorporated into the regularization function, commonly attracting together the parameter vectors of similar tasks. Therefore, the key point is to define which tasks are similar. Moreover, since the regularization influence can be parameterized, the mutual influence among the related tasks can be properly tuned.

The evolution of the area of MTL is characterized by a sequence of different assumptions for the task relationship structure. Proposals considering either cluster-based or graph-based structures, for example, started by assuming a predefined structure and evolved to the point of learning both task parameter vectors and their underlying relationship directly from data. In the case of assuming a hierarchical structure, however, the MTL literature has a lack of proposals founded on obtaining such structure from data. Thus, a research opportunity was identified, with the main goal of obtaining a hierarchical structure to represent the relationship among tasks and, then, employing such structural information within the regularization framework.

Since the goal is based on obtaining a hierarchical structure directly from task data, the established hierarchical clustering area was investigated. Such important area

brought insights on how to obtain a hierarchical structure for the tasks. Essentially, the most challenging step was to derive a similarity metric that could identify related tasks. As a proof of concept, a pairwise task combination approach was proposed, showing the potential of the strategy and highlighting points for improvement, such as the computational cost, the incorporation of a probabilistic model and the removal of some limitations on the resulting structure. Such limitations included the fact that all tasks ended up related in a single structure, which may not be interesting for problems where either clusters of tasks or outlier tasks are present.

Having in view recent advances in agglomerative hierarchical clustering, two Bayesian-based approaches named Bayesian Hierarchical Clustering (BHC) and Bayesian Rose Trees (BRT), proposed by Heller & Ghahramani (2005a) and Blundell *et al.* (2012), respectively, were extended to the context of MTL. Essentially, BRT generalizes BHC by allowing an internal node to have more than two child nodes, that is, the resulting structure is not restricted to be binary. Both algorithms are more suitable than the developed proof of concept for problems composed of larger number of tasks and allow the structure to be cut without depending on domain specialists. At this point, three hierarchy identification algorithms were available to be evaluated.

Regardless of many advantages provided by Bayesian-based algorithms, one drawback is related to hyper-parameters. Depending on the dataset, different probabilistic models and their respective conjugate prior must be employed. In the end, a set of hyper-parameters must be optimized. An EM-like approach suggested by Heller & Ghahramani (2005a) was firstly employed, but, as pointed out by Blundell *et al.* (2012), the hyper-parameters were sensible to initial conditions. As an alternative, the Random Search technique shed light by Bergstra & Bengio (2012) was employed, which yielded consistent and high-quality results.

Inspired by approaches that consider incorporating a given hierarchical structure within the regularization framework, a cost function was also proposed. Basically, the information provided by the obtained hierarchical structure is used to constrain task parameter vectors sharing the same upper-level parameter vector to be similar. Such idea follows the trend in MTL that employs the regularization framework to penalize deviations between parameter vectors associated with linked tasks. Moreover, due to the convexity of the cost function, the optimization problem can be easily solved by off-the-shelf algorithms, such as the gradient descent.

Usually, established regularized MTL approaches consider both task particularities and information exchange among tasks. However, such approaches fail to cover the information from a single general model simultaneously incorporating more than one task. In this scenario, a hierarchical structure can cover such information via hypothetical tasks. Thus, hypothetical tasks can be viewed as auxiliary tasks, with the elementary purpose of helping achieving better models for original tasks. Notice that the usage of auxiliary tasks

was motivated by Caruana (1993) at the very beginning of MTL ideas.

All the strategies employed in this research resulted in the proposal of the Hierarchical Multi-Task Learning framework. Each module of the proposed framework is independent of the others, facilitating studies of extensions involving specific components without affecting the remaining parts, such as alternative proposals to obtain the hierarchical structure and novel strategies to define the cost function. Moreover, the framework also provides the obtained hierarchical structure, allowing a thorough analysis of the problem, making it possible to identify, for example, clusters of tasks and outlier tasks.

Empirical results obtained from the experiments indicated that the proposed framework is capable of overcoming well-established MTL strategies derived from the literature. Overall, employing a hierarchical structure as regularization outperformed all competitors in terms of predictive error, which motivates further research on the subject, since a scarce number of approaches exploring hierarchical structures in the context of MTL is available in the literature.

The experiments carried out over synthetic datasets provided important insights regarding the behavior of each component in the proposed framework, revealing that distinct hierarchical structures can yield similar performance in scenarios where all tasks are similar. However, Bayesian-based strategies yielded the best results in the majority of the experiments, highlighting advantages of relying in a probabilistic approach to model task similarities. Such advantages include the strategy to identify similar tasks, the possibility of having a forest of structures via smart and automatic cuts in a previously single structure, the possibility of having internal nodes with more than two child nodes and the ease of accommodating new tasks. Furthermore, all proposed strategies provided consistent structures compared to what was expected.

In the case of real world datasets, performance experiments confirmed the raised hypothesis in this dissertation. The proposed framework consistently outperformed competitors in scenarios characterized by a varying amount of available training data. Favorable results were presented by the pairwise task combination strategy when scarce training data was available, while BHC was more suitable for some scenarios and BRT for others. Moreover, applying the cut for the Landmine dataset was crucial to obtain the best performance, once the clusters were properly split. Therefore, the adequacy of each hierarchy identification algorithm depends on the problem.

High demand of computational cost in the pairwise task combination approach was one factor that motivated further research. As seen in the results, employing BHC and BRT algorithms is much cheaper when hyper-parameters were previously defined. However, repeated executions for hyper-parameter tuning may become costly. In spite of this, applying the random search technique provided hyper-parameters that satisfactorily met expectations better than simply making a grid search.

Finally, the reached conclusion is that obtaining a hierarchical structure from

task data and using this additional information within the regularization framework is promising and suits the problem of simultaneously learning a set of tasks, even if clusters or outliers are present. The proposed approach not only considers task particularities during the learning process but also considers exchanging information among similar tasks via auxiliary tasks, which covers general information contained in two or more tasks. Moreover, the dependence of domain experts is relaxed, once underlying relations among tasks are automatically uncovered by the machine, enabling such a methodology to be employed in a wider range of applications.

7.2 Future directions

Despite promising results obtained via employing the proposed HMTL framework, both in terms of performance and interpretability, some sources of improvement were identified during the course of the research that can be further explored in the future. Thus, a list of suggested directions are provided as follows:

- **Settings:** In Section 5.4, specific settings for the proposed HMTL framework were defined. Nevertheless, different settings can be also analyzed, including different similarity metrics, strategies to obtain all preliminary task parameter vectors that are clustered by BHC/BRT, and strategies for hyper-parameter optimization.
- **Hierarchical clustering:** The results presented in Chapter 6 showed that, depending on the problem, different hierarchical structures may yield either similar or distinct performance in terms of predictive error. In addition, none of the proposed HIAs provided the best results in all scenarios, motivating the proposal of alternative HIAs more suitable to specific problems. Thus, the incorporation of different HIAs in the proposed HMTL framework is strongly encouraged.
- **Cost function:** Although the usage of the proposed cost function has provided favorable results, extensions focused on the regularization term can be incorporated and further explored, such as the usage of different norms (*e.g.* ℓ_1 and $\ell_{2,1}$). Moreover, weighted edges can also be considered, possibly including distinct weights for bidirectional links connecting pairs of tasks. These changes, however, would require additional hyper-parameters and the application of different optimization techniques.
- **Jointly learning the task parameter vectors and hierarchical structure:** Section 2.4 presented MTL approaches that employ alternating optimization to simultaneously learn the structure and task parameter vectors, mostly focusing on either cluster-based or graph-based structures. Such strategy can be investigated for hierarchical structures as well.

Bibliography

ABERNETHY, J. D.; BACH, F. R.; EVGENIOU, T.; VERT, J. Low-rank matrix factorization with attributes. *CoRR*, abs/cs/0611124, 2006. Cited on page 28.

ANDO, R. K.; ZHANG, T. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, v. 6, p. 1817–1853, 2005. Cited on page 23.

ARGYRIOU, A.; CLÉMENÇON, S.; ZHANG, R. Learning the graph of relations among multiple tasks. *Technical Report*, 2013. Cited on pages 18 and 30.

ARGYRIOU, A.; EVGENIOU, T.; PONTIL, M. Multi-task feature learning. In: *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada*. 2006. p. 41–48. Cited on pages 28, 29, and 69.

BAKKER, B.; HESKES, T. Task clustering and gating for Bayesian multitask learning. *Journal of Machine Learning Research*, v. 4, p. 83–99, 2003. Cited on pages 18, 24, 29, and 68.

BAXTER, J. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, v. 28, n. 1, p. 7–39, 1997. Cited on pages 17 and 23.

BEN-DAVID, S.; SCHULLER, R. Exploiting task relatedness for multiple task learning. In: *Learning Theory and Kernel Machines*. 2003. p. 567–580. Cited on page 23.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, v. 13, p. 281–305, 2012. Cited on pages 58 and 100.

BERGSTRA, J.; YAMINS, D.; COX, D. D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA*. 2013. p. 115–123. Cited on page 58.

BERKHIN, P. A survey of clustering data mining techniques. In: *Grouping Multidimensional Data - Recent Advances in Clustering*. 2006. p. 25–71. Cited on pages 35, 36, and 37.

BLUNDELL, C.; TEH, Y. W.; HELLER, K. A. Bayesian rose trees. *CoRR*, abs/1203.3468, 2012. Cited on pages 18, 44, 45, 46, 47, 58, and 100.

BORG, I.; GROENEN, P. J. F. *Modern Multidimensional Scaling: Theory and Applications (Springer Series in Statistics)*. 2. ed. 2005. ISBN 0387251502. Cited on page 66.

CARUANA, R. Multitask learning: A knowledge-based source of inductive bias. In: *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA*. 1993. p. 41–48. Cited on pages 17, 23, 24, 28, 31, and 101.

CARUANA, R. Multitask learning. *Machine Learning*, v. 28, n. 1, p. 41–75, 1997. Cited on pages 17 and 24.

CHEN, J.; ZHOU, J.; YE, J. Integrating low-rank and group-sparse structures for robust multi-task learning. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA*. 2011. p. 42–50. Cited on pages 23, 29, and 32.

COOKE, E. J.; SAVAGE, R. S.; KIRK, P. D. W.; DARKINS, R.; WILD, D. L. Bayesian hierarchical clustering for microarray time series data with replicates and outlier measurements. *BMC Bioinformatics*, v. 12, p. 399, 2011. Cited on pages 42 and 43.

DARKINS, R.; COOKE, E. J.; GHAHRAMANI, Z.; KIRK, P. D.; WILD, D. L.; SAVAGE, R. S. Accelerating Bayesian hierarchical clustering of time series data with a randomised algorithm. *PloS one*, v. 8, n. 4, 2013. Cited on page 44.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification and scene analysis*. 2. ed. 1995. Cited on page 36.

EVGENIOU, T.; MICCHELLI, C. A.; PONTIL, M. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, v. 6, p. 615–637, 2005. Cited on pages 18, 23, 24, 29, and 49.

EVGENIOU, T.; PONTIL, M. Regularized multi–task learning. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA*. 2004. p. 109–117. Cited on pages 22, 24, 25, and 28.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. *The elements of statistical learning*. : Springer series in statistics Springer, Berlin, 2001. v. 1. Cited on page 34.

GOLDSTEIN, H. Multilevel modelling of survey data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, v. 40, n. 2, p. 235–244, 1991. Cited on page 67.

GONÇALVES, A. R.; DAS, P.; CHATTERJEE, S.; SIVAKUMAR, V.; ZUBEN, F. J. V.; BANERJEE, A. Multi-task sparse structure learning. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China*. 2014. p. 451–460. Cited on pages 18, 23, 30, 32, 63, and 68.

GONG, P.; YE, J.; ZHANG, C. Robust multi-task feature learning. In: *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China*. 2012. p. 895–903. Cited on pages 25 and 29.

HAN, L.; ZHANG, Y. Learning multi-level task groups in multi-task learning. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, USA*. 2015. p. 2638–2644. Cited on page 31.

HAN, L.; ZHANG, Y.; SONG, G.; XIE, K. Encoding tree sparsity in multi-task learning: A probabilistic framework. In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec City, Québec, Canada.* 2014. p. 1854–1860. Cited on page 31.

HELLER, K. A.; GHAHRAMANI, Z. Bayesian hierarchical clustering. In: *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany*. 2005. p. 297–304. Cited on pages 18, 37, 38, 40, 41, 42, 46, 55, 58, and 100.

HELLER, K. A.; GHAHRAMANI, Z. Randomized algorithms for fast Bayesian hierarchical clustering. *Technical Report*, 2005. Cited on pages 43, 44, and 95.

JACOB, L.; BACH, F. R.; VERT, J. Clustered multi-task learning: A convex formulation. In: *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada*. 2008. p. 745–752. Cited on pages 18, 24, 25, 30, and 63.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: A review. *ACM Comput. Surv.*, v. 31, n. 3, p. 264–323, 1999. Cited on pages 34 and 36.

JI, S.; YE, J. An accelerated gradient method for trace norm minimization. In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada*. 2009. v. 382, p. 457–464. Cited on pages 24, 28, 29, and 69.

JORDAN, M.; MITCHELL, T. Machine learning: Trends, perspectives, and prospects. *Science*, v. 349, n. 6245, p. 255–260, 2015. Cited on page 17.

KANG, Z.; GRAUMAN, K.; SHA, F. Learning with whom to share in multi-task feature learning. In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA*. 2011. p. 521–528. Cited on pages 17, 23, 30, and 63.

KARYPIS, G.; HAN, E.; KUMAR, V. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, v. 32, n. 8, p. 68–75, 1999. Cited on page 36.

KUMAR, A.; III, H. D. Learning task grouping and overlap in multi-task learning. In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK*. 2012. Cited on page 63.

LI, C.; LI, H. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, v. 24, n. 9, p. 1175–1182, 2008. Cited on page 30.

MAIMON, O.; ROKACH, L. Introduction to knowledge discovery and data mining. In: *Data Mining and Knowledge Discovery Handbook, 2nd ed.* 2010. p. 1–15. Cited on pages 18 and 35.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to information retrieval*. 2008. ISBN 978-0-521-86571-5. Cited on pages 35 and 36.

MENGERSEN, K.; ROBERT, C.; TITTERINGTON, M. *Mixtures: estimation and applications*. 2011. v. 896. Cited on page 47.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of Machine Learning*. : MIT Press, 2012. (Adaptive computation and machine learning). ISBN 978-0-262-01825-8. Cited on page 21.

MURPHY, K. P. Conjugate Bayesian analysis of the Gaussian distribution. *Technical Report*, 2007. Cited on pages 41 and 58.

NESTEROV, Y. A method of solving a convex programming problem with convergence rate O(1/k2). In: *Soviet Mathematics Doklady*. 1983. v. 27, n. 2, p. 372–376. Cited on page 26.

NESTEROV, Y. Smooth minimization of non-smooth functions. *Mathematical programming*, v. 103, n. 1, p. 127–152, 2005. Cited on page 26.

NESTEROV, Y. *et al.* Gradient methods for minimizing composite objective function. 2007. Cited on page 26.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, Piscataway, NJ, USA, v. 22, n. 10, p. 1345–1359, 2010. ISSN 1041-4347. Cited on page 17.

POWERS, D. M. Evaluation: from precision, recall and f-measure to ROC, informedness, markedness and correlation. 2011. Cited on page 65.

SAVAGE, R. S.; HELLER, K. A.; XU, Y.; GHAHRAMANI, Z.; TRUMAN, W. M.; GRANT, M.; DENBY, K. J.; WILD, D. L. R/BHC: fast Bayesian hierarchical clustering for microarray data. *BMC Bioinformatics*, v. 10, 2009. Cited on pages 42 and 43.

SHAN, H.; KATTGE, J.; REICH, P. B.; BANERJEE, A.; SCHRODT, F.; REICHSTEIN, M. Gap filling in the plant kingdom - trait prediction using hierarchical probabilistic matrix factorization. In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK*. 2012. Cited on pages 18, 50, and 61.

SIRINUKUNWATTANA, K.; SAVAGE, R. S.; BARI, M. F.; SNEAD, D. R.; RAJPOOT, N. M. Bayesian hierarchical clustering for studying cancer gene expression data with unknown statistics. *PloS one*, v. 8, n. 10, 2013. Cited on pages 42 and 43.

THRUN, S.; O'SULLIVAN, J. Discovering structure in multiple learning tasks: The TC algorithm. In: *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy*. 1996. p. 489–497. Cited on pages 23, 24, and 29.

TIBSHIRANI, R. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, p. 267–288, 1996. Cited on pages 69 and 70.

WIDMER, C.; LEIVA, J.; ALTUN, Y.; RÄTSCH, G. Leveraging sequence classification by taxonomy-based multitask learning. In: *Research in Computational Molecular Biology, Proceedings of the 14th Annual International Conference, RECOMB 2010, Lisbon, Portugal*. 2010. v. 6044, p. 522–534. Cited on pages 18, 23, 24, 25, 31, 32, 33, 50, and 61.

XU, L.; HUANG, A.; CHEN, J.; CHEN, E. Exploiting task-feature co-clusters in multi-task learning. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, Austin, Texas, USA*. 2015. p. 1931–1937. Cited on pages 17, 23, 29, 30, 32, 63, and 65.

XU, Q.; YANG, Q. A survey of transfer and multitask learning in bioinformatics. *JCSE*, v. 5, n. 3, p. 257–268, 2011. Cited on page 17.

XU, R.; Wunsch II, D. C. Survey of clustering algorithms. *IEEE Trans. Neural Networks*, v. 16, n. 3, p. 645–678, 2005. Cited on page 35.

XUE, Y.; LIAO, X.; CARIN, L.; KRISHNAPURAM, B. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, v. 8, p. 35–63, 2007. Cited on pages 17, 22, 23, 24, 29, 41, 49, and 68.

ZHANG, Y.; YEUNG, D. A convex formulation for learning task relationships in multi-task learning. In: *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI'2010), Catalina Island, CA, USA*. 2010. p. 733–442. Cited on pages 17, 23, and 24.

ZHONG, W.; KWOK, J. T. Convex multitask learning with flexible task clusters. In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK*. 2012. Cited on pages 23, 30, and 65.

ZHOU, J.; CHEN, J.; YE, J. Clustered multi-task learning via alternating structure optimization. In: *Advances in Neural Information Processing Systems 24: Proceedings of the 25th Annual Conference on Neural Information Processing Systems. Granada, Spain.* 2011. p. 702–710. Cited on pages 22 and 30.

ZHOU, J.; CHEN, J.; YE, J. MALSAR: Multi-task learning via structural regularization. *Technical Report*, 2011. Cited on pages 26, 30, 32, 67, and 69.

ZHOU, J.; YUAN, L.; LIU, J.; YE, J. A multi-task learning formulation for predicting disease progression. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA*. 2011. p. 814–822. Cited on page 32.

ZWEIG, A.; WEINSHALL, D. Hierarchical regularization cascade for joint learning. In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA*. 2013. v. 28, p. 37–45. Cited on page 31.

Appendix

APPENDIX A

HMTL cost function as a maximum a posteriori estimation

The proposed cost function of the HMTL framework described in Section 4.5 can be derived in a Bayesian perspective. In this sense, a Maximum a posteriori (MAP) can be written as Equation (A.1), assuming the likelihood as the conditional distribution of the output $\mathrm{y} \in \mathbb{R}$ given the input vector $\mathbf{x} \in \mathbb{R}^d$ and the task parameter vector $\mathbf{w} \in \mathbb{R}^d$ for M tasks, and the prior as a multivariate Gaussian distribution over the task parameter vector $\mathbf{w}$ taking the relations described by the set of arcs $\mathcal{A}$ into account. Essentially, it is assumed that the task parameter vector $\mathbf{w}_c$ has mean $\mathbf{w}_p$ (task parameter vector of the parent node) and covariance matrix $q_p I_d$, where I_d is the identity matrix making all d features having the same variance q_p.

$$\underbrace{p(\mathcal{W} \mid \mathcal{S}, \mathcal{A})}_{\text{Posterior probability}} \propto \underbrace{\prod_{k=1}^{M} \prod_{i=1}^{n_k} p(\mathrm{y}_k^i | \mathbf{x}_k^i, \mathbf{w}_k)}_{\text{Likelihood}} \underbrace{\prod_{(p,c)\in\mathcal{A}} p(\mathbf{w}_c | \mathbf{w}_p, q_p I)}_{\text{Prior probability}} \tag{A.1}$$

Focusing on the least squares regression and assuming the likelihood taking a normal distribution, the posterior over $\mathcal{W}$ is developed step-by-step in Equation (A.2). Similarly, the same process can be done for logistic regression, assuming the likelihood taking a Bernoulli distribution, which applies to binary classification problems.

$$
\begin{aligned}
P(\mathcal{W} \,|\, \mathcal{S}, \mathcal{A}) &\propto \prod_{k=1}^{M} \prod_{i=1}^{n_k} \mathcal{N}(\mathrm{y}_k^i | \mathbf{w}_k^\top \mathbf{x}_k^i, \sigma_k^2) \prod_{(p,c)\in\mathcal{A}} \mathcal{N}(\mathbf{w}_c | \mathbf{w}_p, q_p I_d) \\
&\propto \prod_{k=1}^{M} \prod_{i=1}^{n_k} (2\pi\sigma_k^2)^{-1/2} \exp\left(-\frac{1}{2\sigma_k^2} (\mathrm{y}_k^i - \mathbf{w}_k^\top \mathbf{x}_k^i)^2 \right) \\
&\quad \prod_{(p,c)\in\mathcal{A}} (2\pi)^{-d/2} \left| q_p I_d \right|^{-1/2} \exp\left(-\frac{1}{2} (\mathbf{w}_c - \mathbf{w}_p)^\top (q_p I_d)^{-1} (\mathbf{w}_c - \mathbf{w}_p) \right) \\
&\propto \prod_{k=1}^{M} \prod_{i=1}^{n_k} (2\pi\sigma_k^2)^{-1/2} \exp\left(-\frac{1}{2\sigma_k^2} (\mathrm{y}_k^i - \mathbf{w}_k^\top \mathbf{x}_k^i)^2 \right) \\
&\quad \prod_{(p,c)\in\mathcal{A}} (2\pi)^{-d/2} (q_p)^{-d/2} \exp\left(-\frac{1}{2} (\mathbf{w}_c - \mathbf{w}_p)^\top (q_p I_d)^{-1} (\mathbf{w}_c - \mathbf{w}_p) \right) \\
&\propto \prod_{k=1}^{M} \prod_{i=1}^{n_k} (2\pi\sigma_k^2)^{-1/2} \exp\left(-\frac{1}{2\sigma_k^2} (\mathrm{y}_k^i - \mathbf{w}_k^\top \mathbf{x}_k^i)^2 \right) \\
&\quad \prod_{(p,c)\in\mathcal{A}} (2\pi q_p)^{-d/2} \exp\left(-\frac{1}{2} (\mathbf{w}_c - \mathbf{w}_p)^\top (q_p I_d)^{-1} (\mathbf{w}_c - \mathbf{w}_p) \right) \\
&\overset{\log}{\propto} \sum_{k=1}^{M} \sum_{i=1}^{n_k} -\frac{1}{2} \log(\sigma_k^2) - \frac{1}{2\sigma_k^2} (\mathrm{y}_k^i - \mathbf{w}_k^\top \mathbf{x}_k^i)^2 + \\
&\quad \sum_{(p,c)\in\mathcal{A}} -\frac{d}{2} \log(q_p) - \frac{1}{2} (\mathbf{w}_c - \mathbf{w}_p)^\top (q_p I_d)^{-1} (\mathbf{w}_c - \mathbf{w}_p) \\
&\propto -\sum_{k=1}^{M} \sum_{i=1}^{n_k} \log(\sigma_k^2) + \frac{1}{\sigma_k^2} (\mathrm{y}_k^i - \mathbf{w}_k^\top \mathbf{x}_k^i)^2 - \\
&\quad \sum_{(p,c)\in\mathcal{A}} d \log(q_p) + (\mathbf{w}_c - \mathbf{w}_p)^\top (q_p I_d)^{-1} (\mathbf{w}_c - \mathbf{w}_p) \\
&\propto -\sum_{k=1}^{M} \sum_{i=1}^{n_k} \log(\sigma_k^2) + \frac{1}{\sigma_k^2} (\mathrm{y}_k^i - \mathbf{w}_k^\top \mathbf{x}_k^i)^2 - \\
&\quad \sum_{(p,c)\in\mathcal{A}} d \log(q_p) + \frac{1}{q_p} (\mathbf{w}_c - \mathbf{w}_p)^\top I_d^{-1} (\mathbf{w}_c - \mathbf{w}_p) \\
&\propto -\sum_{k=1}^{M} \sum_{i=1}^{n_k} \log(\sigma_k^2) + \frac{1}{\sigma_k^2} (\mathrm{y}_k^i - \mathbf{w}_k^\top \mathbf{x}_k^i)^2 - \\
&\quad \sum_{(p,c)\in\mathcal{A}} d \log(q_p) + \frac{1}{q_p} (\mathbf{w}_c - \mathbf{w}_p)^\top I_d (\mathbf{w}_c - \mathbf{w}_p) \\
&\propto -\sum_{k=1}^{M} \sum_{i=1}^{n_k} \log(\sigma_k^2) + \frac{1}{\sigma_k^2} (\mathrm{y}_k^i - \mathbf{w}_k^\top \mathbf{x}_k^i)^2 - \sum_{(p,c)\in\mathcal{A}} d \log(q_p) + \frac{1}{q_p} \|\mathbf{w}_c - \mathbf{w}_p\|_2^2
\end{aligned}
\tag{A.2}
$$

A MAP inference on $\mathcal{W}$ can be performed by maximizing the logarithm of the posterior developed in Equation (A.2), which is equivalent to minimizing the negative logarithm of the posterior. Assuming $\sigma_k^2 = 1$ and $q_p = 1$, though their values may be learned as well, and adding a hyper-parameter $\lambda \geq 0$ to control the regularization influence, the optimization problem boils down to minimizing the regularized squared loss, as written in Equation (A.3).

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} \sum_{k=1}^{M} \sum_{i=1}^{n_k} (\mathrm{y}_k^i - \mathbf{w}_k^\top \mathbf{x}_k^i)^2 + \lambda \sum_{(p,c)\in\mathcal{A}} \|\mathbf{w}_c - \mathbf{w}_p\|_2^2 \tag{A.3}$$