Statistical Extensions of Multi-Task Learning with Semiparametric Methods and Task Diagnostics

Nikolay Miller

University of New Mexico - Main Campus

Nikolay Miller
Candidate

Mathematics & Statistics
Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Guoyi Zhang , Chairperson

Yan Lu

Fletcher Christensen

Manel Martínez-Ramón

Statistical Extensions of Multi-Task Learning with Semiparametric Methods and Task Diagnostics

by

Nikolay Miller

Bachelor, Economics and Business Administration, Oslo Metropolitan University, 2015

MSc, Financial Mathematics, University of Aberdeen, 2017

DISSERTATION

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Statistics

The University of New Mexico

Albuquerque, New Mexico

July 2022

Dedication

Dedicated to my family.

Acknowledgments

I would like to thank my advisor, Professor Guoyi Zhang, for his tremendous guidance and continuous encouragement at every step of the creation of this work. From the beginning stages of researching the field, to literature review, developing new approaches and designing simulations, his neverending support have always given me strength and regenerated my trust in my project. Professor Zhang's generous cheer and optimism has been lightening up my work in this project during the most difficult moments. It is my honor that as a PhD student I have such a supportive, kind-hearted and understanding advisor as Professor Zhang.

I am tremendously thankful to my committee members Professors Yan Lu, Fletcher Christensen and Manel Martínez-Ramón for their support and review of my dissertation project. Their comments and questions during my research talk have given me important insights to reflect upon as a guidance to continue improving my research work.

I would like to thank the UNM Center for Advanced Research Computing, supported in part by the National Science Foundation, for providing the research computing resources used in this dissertation.

I am grateful to my wife, Krystal, for her encouragement, and for comforting me by baking cookies. She has been my source of enthusiasm and optimism throughout my whole PhD degree. Finally, I want to thank my father Igor, my mother Olga, Howard, and other family members and friends for their continuous support and in particular for our conversations about my research.

Statistical Extensions of Multi-Task Learning with Semiparametric Methods and Task Diagnostics

by

Nikolay Miller

Bachelor, Economics and Business Administration, Oslo Metropolitan University, 2015

MSc, Financial Mathematics, University of Aberdeen, 2017

Ph.D., Statistics, University of New Mexico, 2022

Abstract

In this dissertation, I propose new approaches to multi-task learning, inspired by statistical model diagnostics and semiparametric and additive modeling. The newly designed additive multi-task model framework allows for flexible estimation of multi-task parametric and nonparametric effects by using an extension of the backfitting algorithm. Further, I propose new methods for statistical task diagnostics, which allow for the identification and remedy of outlier tasks, based on task-specific performance metrics and their empirical distributions. I perform a deep examination of the well-established multi-task kernel method and achieve theoretical and experimental contributions. Lastly, I propose a two-step modeling approach to multi-task modeling, where the tasks are modeled differently according to their belongingness to different clusters based on the selected performance criteria. The newly proposed frameworks are examined on a well-known real-world multi-task benchmark dataset and show significant improvement over other modern multi-task learning methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Central theme

Multi-task learning is an area of machine learning which makes use of the separation of the data into tasks. One can train separate models for each task (independent task learning), or train a single model for all tasks altogether (single-task learning), but in many research and application contexts, it will be far more beneficial to consider a setup where all tasks share some common information among each other while being influenced by their own specific task information. Thus, the objective of multi-task learning is to improve the overall accuracy of a machine learning model when compared with cases of single task and individual task learning.

Multi-task learning is widely used in fields such as finance, economics, medicine, and education. For example, in finance and economics forecasting, it is often required to predict the value of many possibly related indicators simultaneously (Fiot and Dinuzzo (2015) [34]); in stock price prediction, stocks are often related to each other in multi-task fashion (Ghosn and Bengio (1996) [36], Bitvai and Cohn (2015) [10]); in bioinformatics, we may want to study tumor prediction from multiple microarray data sets or analyze data from multiple related diseases simultaneously (Zhou et al (2021) [87]); in small area estimation, we may want to study the small area total estimate from multiple areas simultaneously; in web search, search results can be improved when combining information from multiple geographical markets that exhibit similar or distinct search patterns (Bai et al (2009) [8]).

In the context of supervised learning, regularized multi-task learning methods have received significant attention from many researchers. The idea of multi-task mean regularization, where task parameters are penalized individually and with regard to their closeness to the overall mean of all task parameters, was first introduced in Evgeniou and Pontil (2004) [31]. This idea was further generalized to be cast into any L2-regularized single-task learning problem using specialized multi-task kernels in Evgeniou, Michelli, and Pontil (2005) [30]. Its framework is based on Evgeniou, Pontil, and Poggio (2000) [32], where the emphasis is based on regression. In this context, reproducing kernel Hilbert spaces (RKHS) are introduced in great detail by Manton and Amblard (2014) [56]. An extensive overview of vector-valued functions and kernels is given in Alvarez, Rosasco, and Lawrence (2012) [4], and learning these functions in RKHS is studied by Michelli and Pontil [60] (2005). Extension of representer theorem (Kimeldorf and Wahba (1971) [49]) to multi-task regularization was studied in Argyriou, Michelli and Pontil (2009) [7]. An effective application of regularized multi-task learning to log-density estimation is demonstrated in Yamane, Sasaki, and Sugiyama (2016) [80].

The methods of multi-task regularization were shown to be special cases of a larger multi-task clustering framework in Jacob, Vert, and Bach (2009) [42]. Multi-task kernels are proved to have a universal property in Caponnetto, Michelli, Pontil, and Ying (2008) [16]. There is extensive literature on multi-task kernels, which includes works by Amodei (1997) [5], Masani and Burbea (1984) [57], Caponnetto and De Vito (2007) [15], Carmeli, De Vito and Toigo (2006) [17], Reisert and Burkhardt

(2007) [65].

Distinctively, choosing kernels and features for a multi-task combination of support vector machines was studied in Jebara (2004) [44], which is related to the independent task learning procedure covered in this dissertation. Another method to combine task features in a multi-task SVM application was studied by Bonilla, Agakov, and Williams (2007) [11], where task-specific features are chosen using a gating network. In contrast to multi-task regularization, Parameswaran and Weinberger (2010) [61] propose using large margin nearest neighbors algorithm to concentrate task parameters. In the context of multiple linear regression, Lounici, Pontil, Tsybakov, and van de Geer (2009) [55] propose using the Group LASSO method to choose the predictor variables in tasks. Kato, Kashima, Sugiyama, and Asai (2007) [47] introduce a method to differentiate between task parameter closeness depending on their relatedness. Jawanpuria, Lapin, Hein, and Schiele (2015) [43] propose an efficient algorithm to jointly learn tasks and their output kernel.

Lastly, a more extensive and thorough overview of multi-task learning as a field in machine learning can be found in a work by Zhang and Yang (2018) [84], Thung and Wee (2018) [73], and Zhang and Yang (2021) [85]. Although this dissertation approaches multi-task learning from a statistical perspective, it is notable that the use of deep learning models for multi-task learning has become popular; the survey of this subfield is given by Ruder (2017) [67].

1.2 Purpose of the study

In this dissertation, I intend to contribute to the multi-task machine learning field in multiple ways. First, I perform a deep analysis of mean-regularized multi-task kernel methods, contributing both theoretical results about two versions of the kernel and experimental investigation of the Gaussian kernel application. I design new methods for multi-task learning through a two-step modeling approach, which is based on the identification of important tasks and suitable remedial measures. Drawing inspiration from the statistical learning field, I further propose an additive multi-task model, which is designed to be customizable to the needs of a particular problem. I also propose new methods for statistical task diagnostics, which allow the identification of task outliers. The effectiveness of the methods I proposed is demonstrated in their respective experimental analyses.

It is my opinion that the existing research in multi-task learning is performed predominantly from a computer science point of view. There is often a particular focus on the design and implementation of optimization algorithms for solving complex regularization problems. Possibly due to their complexity and programming skills requirements, for many of these models, it is a rare occurrence to find them being applied to solve real-world data analysis problems.

The purpose of this research is to start from the fundamentals, ground zero, and propose new multi-task learning methods which are grounded in simple statistical truths. I make a particular emphasis on interpretability and ease of implementation of the proposed statistical models so that they can be beneficial to the academic community at large. The arguments and definitions in this dissertation often use ideas, results, and methods from the statistical field.

1.3 Summary of chapters

The rest of the dissertation is organized as follows.

In Chapter 2, I perform a review of the existing theory of the topics that are used for further research later in the dissertation. Section 2.2 presents the theory of support vector regression, which is an important method for applications with multi-task kernels. Random forest models are covered in Section 2.3. Then in Section 2.4, I present the explained variance metric, which is often used in the scientific literature to measure the performance of multi-task learning methods. I continue by presenting the famous framework of learning multiple tasks with multi-task kernel methods in Section 2.5. The chapter concludes with a review of the existing work on task outliers and clustering in Section 2.6, which motivates my research in statistical task diagnostics.

In Chapter 3, I perform a deep analysis of regularized multi-task learning, focusing on mean-regularization. In Section 3.1, I do an analysis of parameter choice in Evgeniou et al (2005) [30], and in Section 3.2 I present a simple example to illustrate the inner workings of the data transformation. Then in Section 3.3 I compare two different versions of the mean-regularized multi-task kernel, and show their equivalency both theoretically with a proof and experimentally. I also cover a special case of ridge regression in Sections 3.3.4 and 3.3.5, where the two different versions of the kernel can be made equal with simpler conditions than the general case. In Section 3.4 I perform a series of experiments, where I test the performance of the method in Evgeniou et al (2005) [30], and also examine the case of the Gaussian kernel and its influence on the model performance. The chapter is concluded with an examination of mean-regularized MTL kernel's edge cases of single-task learning and independent task learning in Section 3.5.

The purpose of Chapter 4 is to propose and investigate new methods for multi-task learning, based on a two-step modeling approach. It begins with a thorough explanation of the logic and motivation behind the method in Section 4.1, and a simple applied example is provided in Section 4.2. Then, in Section 4.3 multiple configurations of the two-step multi-task modeling framework are developed and demonstrated, which include methods of explained variance rebalancing approach, best subsets information extraction, other tasks performance approach, and best performance subsets combinations approach.

In Chapter 5 I propose a new method to perform multi-task learning, which I call the additive multi-task model. To accommodate the model, I generalize the idea of explained variance in Section 5.2 and introduce multi-task combined estimation in Section 5.3. The additive multi-task model definition and fitting algorithm are given in Section 5.4. This algorithm is further generalized in Section 5.5. Then, in Section 5.6 the model is customized and applied to the real-world dataset. In Section 5.7 I perform an extension of the additive multi-task model, and in Section 5.8 I demonstrate the effectiveness of the method by comparing two reduced models against a full model.

Chapter 6 contains new inventions in statistical task diagnostics. I begin with an initial analysis in Section 6.2, which gives description of the tasks in ILEA schools data. In Section 6.3 I propose a task influence test, which uses a newly defined deleted explained variance statistic and is performed in a leave-task-out fashion. Then, in Section 6.4 I propose a method of automating the task influence test to use critical values in t-distribution, and in Section 6.5 kernel density estimation is used as task cut-off criteria for the deleted explained variance statistic. Further, in Section 6.6 I introduce a method of measuring task influence on model assumptions.

Lastly, the dissertation finishes with a conclusion in Chapter 7, where I summarize the results achieved in my research and give directions to open questions and future work.

Chapter 2

Background

2.1 Background

The research in this dissertation is based on multi-task kernel methods which are covered in detail. This framework allows the use of regularization networks such as support vector machines, which are also introduced. Explained variance is an often-used metric in performance measurements of multi-task learning models, and is covered in a separate section. Random forest models are used for some of the experiments for the newly introduced methods in this dissertation, therefore this topic is covered as well. Lastly, I present a review of existing research in detecting task outliers in the multi-task learning field. Although the research in this dissertation stands on its own, the overview of this area is necessary for understanding the state of this research area and appreciation of the importance of this work.

2.2 Support vector regression

Support vector machines (SVM) are a class of machine learning algorithms for supervised learning. SVM models can be divided into support vector regression (SVR) for regression problems, and support vector classification (SVC) for classification problems. Support vector machines were invented by Vapnik (1992) [74] and are rooted in statistical learning and Vapnik–Chervonenkis theories (Vapnik and Chervonenkis (1971) [75]). SVR will often be integrated with multi-task learning setups in this dissertation and used extensively both in theory, due to the way it penalizes the norm of its parameters, and in practice. This section focuses on explaining the theory of SVR.

Formulation

Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$ be a set of n observations, where $\mathbf{x}_i \in \mathbb{R}^d$, thus having d predictor variables, and $y_i \in \mathbb{R}$, for all i. The goal of support vector regression models is to learn a function $f(\mathbf{x})$ with a property that the deviation of real data y and model predictions $\hat{y}$ is at most ϵ and that such function is as flat as possible. This defines a concept of **ϵ-tube** (Drucker, Burges, Kaufman, Smola, Vapnik (1996) [27]), such that the algorithm fits a line to the data with a margin of ϵ on both sides of the line using ϵ-insensitive loss function for observation i (Vapnik (1992) [76]):

$$L(y_i, f(\mathbf{x}_i)) = \begin{cases} 0, & \text{if } |y_i - f(\mathbf{x}_i)| < \epsilon \\ |y_i - f(\mathbf{x}_i)| - \epsilon, & \text{otherwise} \end{cases} \tag{2.1}$$

Assuming that $f(\mathbf{x})$ is a linear function, define it as $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$, where $\mathbf{w} \in \mathbb{R}^p$ is a parameter vector, b is bias and $\langle \, , \rangle$ is an inner product in Eucledian space.

The formulation of support vector regression is to find an optimal set of parameters $\mathbf{w}$ and b in order to ensure the flatness property, which is equivalent to minimizing the length of the parameter vector $\mathbf{w}$. This leads to the following optimization:

$$\begin{aligned}
&\min_{\mathbf{w},b} && \frac{1}{2}\|\mathbf{w}\|^2 && (2.2a)\\
&\text{subject to} && y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \epsilon && (2.2b)\\
& && (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \epsilon && (2.2c)
\end{aligned}$$

We assume that there exists a function f that can satisfy all the above constraints. However, it might not be possible to fit all the data points in the ϵ-tube, especially when searching for linear functions. Thus we can introduce slack variables ξ_i, ξ_i^* for positive and negative deviations, respectively, which leads to the following optimization problem (Alpaydin (2004) [3]):

$$\begin{aligned}
&\min_{\mathbf{w},b} && \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) && (2.3a)\\
&\text{subject to} && y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \epsilon + \xi_i && (2.3b)\\
& && (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \epsilon + \xi_i^* && (2.3c)\\
& && \xi_i, \xi_i^* \geq 0 && (2.3d)
\end{aligned}$$

The parameter $C > 0$ controls the trade-off between the complexity of the function and magnitude of deviations from the ϵ-tube, as given by the sum of slack variables. Its optimal value is usually found using cross-validation. The extensions to nonlinear functions are facilitated using the dual problem (Alpaydin (2004) [3], Agarwal (2020) [2]):

$$\max_{\boldsymbol{\alpha}} \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\mathbf{x}_i^T\mathbf{x}_j \tag{2.4a}$$

$$+\sum_{i=1}^{n} y_i(\alpha_i-\alpha_i^*)-\epsilon\sum_{i=1}^{n}(\alpha_i+\alpha_i^*) \tag{2.4b}$$

$$\text{subject to} \quad \sum_{i=1}^{n}(\alpha_i-\alpha_i^*)=0 \tag{2.4c}$$

$$0\le\alpha_i\le C \tag{2.4d}$$

$$0\le\alpha_i^*\le C \tag{2.4e}$$

Both the primal and dual problems are convex quadratic problems and can be solved directly. The name of support vector machines comes from the fact that the solution depends only on a subset of observations, which are called support vectors. In the case of SVR, support vectors are observations on the boundary and outside of the ϵ-tube. From the slackness conditions (Agarwal (2020) [2]), the observations inside the ϵ-tube have α_i and α_i^* equal to zero, the support vector observations on the ϵ-tube have $0<\alpha_i<C$ or $0<\alpha_i^*<C$, and support vector observations outside the ϵ-tube have $\alpha_i=C$ or $\alpha_i^*=C$. This defines two sets of support vectors: $SV_1=\{i\in\{1,2,...,n\}\mid 0<\alpha_i<C \text{ or } 0<\alpha_i^*<C\}$ and $SV_2=\{i\in\{1,2,...,n\}\mid \alpha_i=C \text{ or } 0<\alpha_i^*=C\}$ with an overall set $SV=SV_1\cup SV_2$. Through the Lagrangian of the dual problem, the vector of parameters can be estimated as

$$\begin{aligned}\hat{\mathbf{w}} &= \textstyle\sum_{i=1}^{n}(\hat\alpha_i-\hat\alpha_i^*)\mathbf{x}_i=\sum_{i\in SV}(\hat\alpha_i-\hat\alpha_i^*)\mathbf{x}_i\\ \hat b &= \frac{1}{|\mathrm{SV}_1|}(\textstyle\sum_{i:0<\hat\alpha_i<C}(y_i-\hat{\mathbf{w}}^T\mathbf{x}_i-\epsilon)+\sum_{i:0<\hat\alpha_i^*<C}(\hat{\mathbf{w}}^T\mathbf{x}_i-y_i-\epsilon))\end{aligned}$$

Therefore, the prediction of a new observation $\mathbf{x}$ can be done by

$$\hat f(\mathbf{x})=\langle\hat{\mathbf{w}},\mathbf{x}\rangle+\hat b=\sum_{i\in SV}(\hat\alpha_i-\hat\alpha_i^*)(\mathbf{x}_i^T\mathbf{x})+\hat b. \tag{2.5}$$

Nonlinearity extension

Not all datasets can be explained well by linear hyperplanes, thus we need to extend the SVR algorithm to incorporate nonlinearities. This can be done with feature mapping, such that we preprocess all observations $\mathbf{x}_i$ with some feature map $\Phi : \mathbb{R}^d \rightarrow \mathcal{F}$, where $\mathcal{F}$ is a feature space. In the theory of SVR, we saw that the prediction of new observation depends only on the inner products of new observation with the observations in the training set. However, after transforming all the data points to the feature space, computing the dot products in the feature space between each other becomes computationally infeasible, as the number of features grows dramatically in the feature space (Smole and Schölkopf (2004) [70]). Luckily, for a feature map Φ it suffices to know its **kernel** function K, such that $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. We can then use the kernel function directly to compute dot products in the dual problem without performing feature mapping first, thus making the problem computationally feasible.

Theoretical underpinnings for classes of kernel functions $K(\mathbf{x}_i, \mathbf{x}_j)$ that correspond to dot products in $\mathcal{F}$ are described in Smole and Schölkopf (2004) [70]. A powerful and popular choice is the family of Gaussian radial basis functions $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$, which have a property of translation invariancy due to $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i - \mathbf{x}_j)$.

Empirical and structural risks

Let $L(y, f(\mathbf{x}))$ be a loss function which measures a numeric cost associated with prediction of response variable y by function f using data $\mathbf{x} \in \mathbf{X}$, with $\mathbf{X} \in \mathbb{R}^{n \times d}$. Define $R(y, f) = \int L(y, f(\mathbf{x})) dP(\mathbf{x}, y)$ to be a risk function, which is an expectation of loss of function f across the joint probability distribution $P(\mathbf{x}, y)$ of the data $y \in \mathbf{y}$ and $\mathbf{x} \in \mathbf{X}$, with $\mathbf{X} \in \mathbb{R}^{n \times d}$. Given a sample of n data points, the data distribution is generally unknown, thus the risk function R can be estimated by the empirical risk function

$$R_{emp}(y, f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))$$

In the framework of empirical risk minimization, we seek to find a function f that minimizes empirical risk, $\hat{f} = \operatorname{argmin}_f R_{emp}(y, f)$.

A sole focus on empirical risk minimization can lead to a function that is overfitting to the training set. A function may become too complex in order to fit the training data, and not be generalizable enough to unseen data. This is *structural* risk $R_{str}(f)$ and it can be added to empirical risk in order to have a regularizing effect in the minimizing criteria for a learning algorithm:

$$R_{emp}(y, f) + \lambda R_{str}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda R_{str}(f) \tag{2.6}$$

Parameter λ controls a trade-off between empirical and structural risk in the minimization criteria. The support vector regression algorithm (2.3) fits the risk minimization framework with the ϵ-insensitive loss function, $R_{str}(f) = \|\mathbf{w}\|^2$ and $\lambda = \frac{1}{2Cn}$. Thus, support vector regression model is tuned by adjusting parameters ϵ and C.

2.3 Random forest

Random forest models are an ensemble learning method for classification, regression, or unsupervised learning. They were first introduced by Breiman (2001) [13]. In this dissertation, the random forest will only be used for regression.

The prediction of random forest regression model $h(\mathbf{x}_i)$ is an unweighted average of multiple regression trees predictions $h(\mathbf{x}_i, \theta_q)$, where θ_q are independent and identically distributed vectors for all $q \in \{1, ..., Q\}$, and Q is the number of trees:

$$h(\mathbf{x}_i) = \sum_{q=1}^{Q} h(\mathbf{x}_i, \theta_q) \tag{2.7}$$

The outcome of the prediction $h(\mathbf{x}_i)$ is controlled by the number of trees parameter Q and the parameters that control the regression trees $h(\mathbf{x}_i, \theta_q)$.

Each individual regression tree is learned from a replacement sample of $\mathbf{X}_{train}$ (Lewis (2000) [52]). This sample is partitioned into binary decision nodes multiple times until the tree reaches terminal nodes. The number of partitions is a model complexity parameter that is tuned by decision cost to avoid overfitting and underfitting.

This partitioning happens with regard to a splitting criterion, such that splitting the tree into nodes captures the information from the sample in the most effective way. In the case of regression, the criterion at each split in a tree is the minimization of sums of squares $(y_i - h(\mathbf{x}_i, \theta_q))^2$ as the sum for both nodes. The search is computationally infeasible in this form, thus a greedy algorithm is usually implemented, which involves minimization of the expected sum of variances for both. The details of this algorithm are given in Hastie et al (2001) [38] and Breiman, Friedman, Olshen, and Stone (2017) [14].

As a result of the training of regression trees, each sample in the training data $\mathbf{X}_{train}$ and $\mathbf{Y}_{train}$ is placed in one of the terminal nodes. The predicted values in each terminal node are the average values of the response variable in that node. Resampling the data for each regression tree and then averaging these results in a random forest model achieves a bootstrapping effect. This is conventionally denoted as “bagging” in machine learning literature (Breiman (1996) [12]). Moreover, random forest models allow for sampling of predictor variables for each tree and node, which achieves regularizing effect.

In this dissertation, the computations of random forest were performed with the function "randomForest" in R package e1071 (Meyer, Dimitriadou, Hornik, Weingessel, Leisch, Chang, and Lin (2021) [58]), which implements the original algorithm by Breiman (2001) [13]. In all computations, all the function parameters are left as default, except the number of trees Q. Its default value is 500, but in some simulations, I adjust this number higher, which improves the performance significantly.

2.4 Explained variance

The explained variance metric is widely used in the context of multi-task learning as a measurement metric of a model's performance; for example, it was applied for regression problems by Bakker et al (2003) [9] and Evgeniou et al (2005) [30]. However, the definitions of the explained variance aren't quantitatively described in the papers. To fulfill this gap, this section rigorously defines the explained variance, which was described in detail by Tom Heskes through direct correspondence. This thorough definition is important for the model comparison and will become particularly necessary in the task diagnostics section.

Definition by Heskes

As previously defined, let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a set of d predictor variables for n observations, and let $\mathbf{Y} \in \mathbb{R}^{n}$ be a set of response variables in a learning problem. A train-test split is performed, such that $\mathbf{X} = \mathbf{X}_{train} \cup \mathbf{X}_{test}$ and $\mathbf{Y} = \mathbf{Y}_{train} \cup \mathbf{Y}_{test}$, with $\mathbf{X}_{train} \in \mathbb{R}^{n_{train} \times d}$, $\mathbf{X}_{test} \in \mathbb{R}^{n_{test} \times d}$, $\mathbf{Y}_{train} \in \mathbb{R}^{n_{train}}$, $\mathbf{Y}_{test} \in \mathbb{R}^{n_{test}}$ such that $n_{train} + n_{test} = n$.

Let $\hat{f}$ be a model for estimating $\hat{y}_i = \hat{f}(x_i)$. $\hat{f}$ is trained using observations in the training set $(\mathbf{X}_{train}, \mathbf{Y}_{train})$ and its performance is measured on the test set

$(\mathbf{X}_{test}, \mathbf{Y}_{test})$. Let $\hat{\mathbf{Y}}_{test} = \{\hat{f}(x_i) : x_i \in \mathbf{X}_{test}, i \in \mathbb{N}_{n_{test}}, \forall i\} \in \mathbb{R}^{n_{test}}$, where $\mathbb{N}_{n_{test}} = \{1, 2, ..., n_{test}\}$, be a set of outputs of the model $\hat{f}$ when applied on the predictor variables of observations in the test set, $\mathbf{X}_{test}$.

Define $MSE(\mathbf{Y}_{test}, \hat{\mathbf{Y}}_{test}) = \frac{\sum_{i=1}^{n_{test}} (y_i - \hat{y}_i)^2}{n_{test}}$ for $y_i \in \mathbf{Y}_{test}$ and $\hat{y}_i \in \hat{\mathbf{Y}}_{test}$, and let $\text{Var}(\mathbf{Y}_{test}) = \frac{\sum_{i=1}^{n_{test}} (y_i - \bar{y})^2}{n_{test}}$ be the variance of response variables in the test set. Then, explained variance $EV(\mathbf{Y}_{test}, \hat{\mathbf{Y}}_{test})$ can be defined as:

$$EV(\mathbf{Y}_{test}, \hat{\mathbf{Y}}_{test}) = \frac{\text{Var}(\mathbf{Y}_{test}) - MSE(\mathbf{Y}_{test}, \hat{\mathbf{Y}}_{test})}{\text{Var}(\mathbf{Y}_{test})} \times 100\% \tag{2.8}$$

Explained variance is the fraction of variance in the response variable Y that is correctly predicted by f using predictor variable X. $EV \in (-\infty, 100\%]$. It can be interpreted as comparing model f to the simplest possible model of predicting all y_i with the mean value of Y. In that case, $MSE(\mathbf{Y}_{test}, \hat{\mathbf{Y}}_{test})$ is equal to $\text{Var}(\mathbf{Y}_{test})$, and $EV = 0\%$. Any improvement in performance of f will lead to decrease in MSE, thus increasing explained variance up to its maximum value 100%, although it can only be equal to 100% if $\mathbf{Y}_{test} = \hat{\mathbf{Y}}_{test}$, so that f achieves perfect prediction with zero error. If $EV < 0\%$, then the model f is performing worse than just using the mean value as a prediction. The possible issues can be the model specification, lack of fit, severe overfitting of f to the training set $(\mathbf{X}_{train}, \mathbf{Y}_{train})$, or that the test set is not representative of the distribution. Percentage explained variance is defined only for regression problems and can be used for cross-validation to measure model performance.

As described in this section, explained variance is defined for a model trained on a training set, and measured for its predictive performance on the test set. In this dissertation, there arise circumstances when explained variance needs to be further generalized from its classical definition to be measured not only on the test set. It is helpful in applications when cross-validation for parameter tuning is done on the training set exclusively, without using the test set. There also arise applications where there is no separation between train and test splits, so that no splitting is performed. Both of these situations fall outside of the conventional definition of explained variance, as described in the literature. In order to overcome this, I will seek to generalize and expand the explained variance definition to cover these cases, while keeping the original definition intact to prevent any confusion. Therefore, when there arises a need to apply a new metric based on explained variance, a new definition will be created and explained in detail. In particular, this will be done in Chapter 5.

2.5 Learning multiple tasks with kernel methods

The benefits of learning multiple tasks simultaneously can be achieved by using the multi-task kernels applied to regularized single-task learning models. The methodology of extension of single-task kernel methods, such as support vector machines and regularization networks, to multi-task learning, was developed by Evgeniou et al (2004) [31] and further generalized by Evgeniou et al (2005) [30]. In the following, I will present the main implications of the theory and perform a simple experiment of the proposed methodology on real data.

Multi-task kernel method

Suppose that we have m tasks that are related to each other. Let n_l be a number of observations for task l, such that $\{(\mathbf{x}_{1,l}, y_{1,l}), (\mathbf{x}_{2,l}, y_{2,l}), ..., (\mathbf{x}_{n_l,l}, y_{n_l,l})\}$ is the data for task l. Without loss of generality, we assume that the number of datapoints per task is constant, such that $n_l := n$ for all tasks l. Then, the dataset for all tasks becomes

$\{(\mathbf{x}_{i,l}, y_{i,l}) : i \in \mathbb{N}_n, l \in \mathbb{N}_m\}$. Let the l-th task's input space be $\mathcal{X}_l$ and output space be $\mathcal{Y}_l$ and let $\mathcal{X}_l \in \mathbb{R}^d$ and $\mathcal{Y}_l \in \mathbb{R}$. We are seeking to learn functions $f_l : \mathcal{X}_l \rightarrow \mathcal{Y}_l$ for each task $l \in \mathbb{N}_m$.

If we are searching among linear functions, then $f_l(\mathbf{x}_{i,l}) = \mathbf{u}_l^T \mathbf{x}_{i,l}$ for some parameter vector $\mathbf{u}_l \in \mathbb{R}^d$. Let $\mathbf{u}_l = \mathbf{B}_l^T \mathbf{w}$ for some $p \times d$ matrix $\mathbf{B}_l$ with $p \geq dm$, with $p \in \mathbb{N}$, for each task $l \in \mathbb{N}_m$. Note that feature vector $\mathbf{w} \in \mathbb{R}^p$, as previously defined. Then, the functions can be reparameterized as:

$$f_l(\mathbf{x}_{i,l}) = \mathbf{w}^T \mathbf{B}_l \mathbf{x}_{i,l}$$

Therefore, all functions f_l are parameterized by the same feature vector $\mathbf{w}$. Define $\mathbf{B} := [\mathbf{B}_l : l \in \mathbb{N}_m]$ to be a $p \times dm$ feature matrix, which is created by joining the matrices $\mathbf{B}_l$ for each task. Matrices $\mathbf{B}_l$ can be specified in a desired manner to capture the relationships between the tasks.

Let $f = (f_l : l \in \mathbb{N}_m)$ be a vector-valued function for all tasks. The above feature space viewpoint of transforming the data with matrices $\mathbf{B}_l$ reshapes function f to be a real-valued function $f : (\mathbf{x}, l) \rightarrow \mathbf{w}^T \mathbf{B}_l \mathbf{x}$. Then, function f has the squared norm $\mathbf{w}^T \mathbf{w}$ and input space $\mathbb{R}^d \times \mathbb{N}_m$. The reproducing kernel for the Hilbert space of such functions is

$$K((\mathbf{x}, l), (\mathbf{t}, q)) = \mathbf{x}^T \mathbf{B}_l^T \mathbf{B}_q \mathbf{t}, \quad \mathbf{x}, \mathbf{t} \in \mathbb{R}^d, \quad l, q \in \mathbb{N}_m \tag{2.9}$$

This defines a *linear multi-task kernel* (Evgeniou et al (2005) [30]). Using this kernel and a proper choice of matrices $\mathbf{B}_l$ facilitates recasting the multi-task learning problem as a single-task learning problem with the appropriately transformed data, which corresponds to the minimization criteria:

$$S(\mathbf{w}) = \frac{1}{nm} \sum_{j \in \mathbb{N}_m} \sum_{i \in \mathbb{N}_n} L(y_{i,j}, \mathbf{w}^T \mathbf{B}_l \mathbf{x}_{i,j}) + \gamma \mathbf{w}^T \mathbf{w} \tag{2.10}$$

The averaging of the loss function is over the whole training set for all tasks, and γ is a parameter that controls the trade-off between structural and empirical risk. Thus, this methodology falls within the risk minimization framework in (2.6). Notably, the regularization term is given by a convenient L2 norm.

The choice of matrices $\mathbf{B}_l$ defines the feature space of the multi-task kernel. One possible choice is

$$\mathbf{B}_l^T = [\sqrt{1-\lambda}\mathbf{I}_d, \underbrace{\mathbf{0}, ..., \mathbf{0}}_{l\text{-}1}, \sqrt{\lambda m}\mathbf{I}_d, \underbrace{\mathbf{0}, ..., \mathbf{0}}_{m\text{-}l}], \tag{2.11}$$

where $\mathbf{0}$ is a $d \times d$ matrix of zeroes. The matrix $\mathbf{B}_l$ has dimensions $d(m+1) \times d$, thus the combined feature matrix $\mathbf{B}$ has dimensions $d(m+1) \times dm$ and $\mathbf{w} \in \mathbb{R}^{d(m+1)}$. This choice of matrices $\mathbf{B}_l$ leads to the multi-task kernel

$$K((\mathbf{x}, l), (\mathbf{t}, q)) = (1 - \lambda + \lambda m \delta_{l,q})\mathbf{x}^T\mathbf{t}, \quad l, q \in \mathbb{N}_m, \quad \mathbf{x}, \mathbf{t} \in \mathbb{R}^d, \tag{2.12}$$

where $\delta_{l,q}$ is an indicator function of $[l = q]$. This leads to the multi-task regularizer

$$J(u) = \frac{1}{m}\left(\sum_{l \in \mathbb{N}_m} \|\mathbf{u}_l\|^2 + \frac{1-\lambda}{\lambda} \sum_{l \in \mathbb{N}_m} \|\mathbf{u}_l - \frac{1}{m} \sum_{q \in \mathbb{N}_m} \mathbf{u}_q\|^2 \right), \tag{2.13}$$

where parameter $\lambda \in (0, 1]$ controls trade-off between closeness of parameters to their average and small size of the parameter vectors themselves. If λ is close to 0, the tasks are learned as a single task, and if $\lambda = 1$, the tasks are learned independently. A value of λ between 0 and 1 can be seen as encoding a trade-off between single-task learning and independent learning. In case of SVR, λ controls trade-off of flatness of each task's SVR to the proximity of individual task's parameters to the average SVR [31]. Note that the regularizer (2.13) doesn't appear directly in the minimization criteria (2.10), but is implied in the multi-task learning problem because of the particular choice of matrices $\mathbf{B}_l$.

Multi-task SVR experiment

In order to illustrate the multi-task kernel method experimentally, I run the SVR model with a multi-task kernel for the "Inner London Education Authority" dataset, which replicates the experiment by Evgeniou et al. (2004) [31]. It is a data of examination records of 15362 students from 139 schools. Detailed information about the dataset is given in Appendix A. I treat each school as a separate task. There is one response variable, examination score, and 8 predictor variables. Using one-hot encoding, I create indicator columns to expand categorical variables, resulting in 26 predictor variables. The number of observations per school (n_l) varies from 22 to 251.

I use linear multi-task kernel in support vector regression model and matrices $\mathbf{B}_l$ as described in (2.11). For this data, $d = 26$, $dm = 26 * 139 = 3614$, $d(m+1) = 26 * (139 + 1) = 3640$, $\mathbf{B}_l$ has dimensions $d(m+1) \times d = 3640 \times 26$, the feature matrix $\mathbf{B}$ has dimensions $d(m+1) \times dm = 3640 \times 3614$ and the feature vector $\mathbf{w} \in \mathbb{R}^{d(m+1)} = \mathbb{R}^{3640}$. The data transformation step performs the matrix product $\mathbf{B}_l \mathbf{x}_{i,l}$ of each observation's vector with the matrix $\mathbf{B}_l$ that corresponds to the task of the observation. The resulting transformed dataset has 15362 rows and 3640 columns.

We run cross-validation for the parameter $\lambda \in [0, 0.1, ..., 0.9, 1]$ and record explained variance metric. The train-test split is done with 75% of the data in the train set, and the remaining 25% in the test set. This results in a train set of 11521 observations and a test set of 3841 observations. 10 random train-test splits are performed, and the results of explained variance are averaged across the runs for all λ. For this experiment, we let $\gamma = 1$ in minimization criteria (2.10), and $\epsilon = 0.1$ in the SVR loss function (2.3). Note that the original 10 train-test splits used by Evgeniou et al. (2005) [30] are not publicly available, thus a new set of 10 train-test splits are done for this demonstration.

The model fit was performed using function “svm” from the e1071 package in R (Meyer et al (2021) [58]). The implementation of this function in the package is based on the LIBSVM library by Chang and Lin (2011) [18].

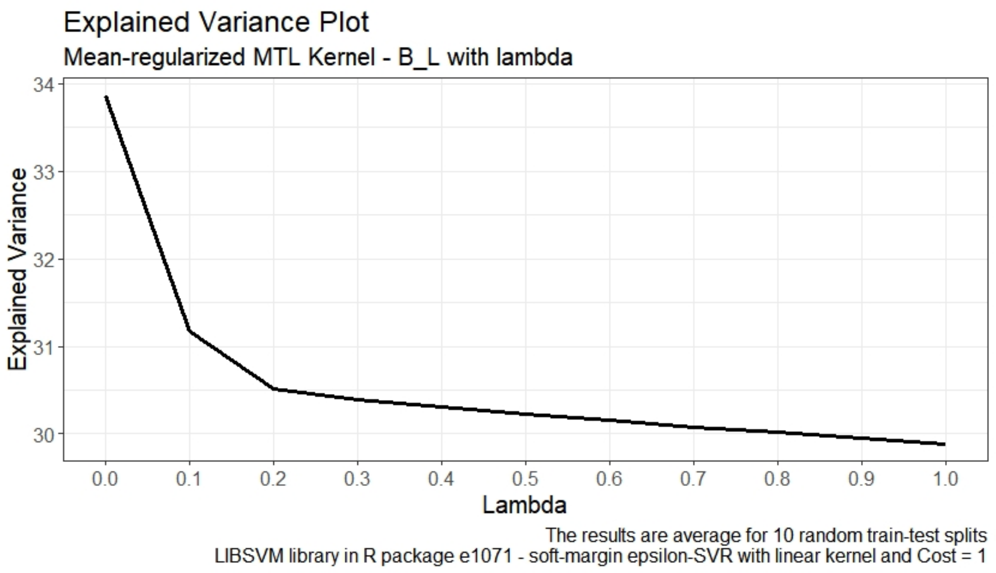

Figure 2.1: Explained variance of multi-task linear SVR by coupling parameter λ

Figure 2.1 displays the results of the best fit to the ILEA data. Note that the explained variance is increasing up to the point of $\lambda = 0$, which indicates that the best performance is achieved when the tasks are learned together, i.e. when the schools are treated as one task and do not differ across the initial tasks partition. The lowest explained variance is at $\lambda = 1$, thus the data doesn’t appear to come from completely separate tasks. Differently from results in Evgeniou et al (2005) [30], where performance was rather flat from $\lambda = 0$ to 0.7, followed by a sharp drop in explained variance closer $\lambda = 1$, we observe that explained performance drops sharply after $\lambda = 0$ and declines slowly in a linear way as λ increases. This difference can possibly be the result of different learning algorithm or package that was used in the experimental section of Evgeniou et al (2005) [30]. It is notable, however, that the results at $\lambda = 0$ are very similar in both of the experiments. This experiment will be studied in more detail in the upcoming chapter, with a thorough analysis of the influence of different parameters and kernels, challenging the consensus assumption that it is a single-task learning dataset.

2.6 Task outliers and clustering

In multi-task learning, we assume that all tasks are somehow related to each other, and it is the objective of the multi-task learning model to learn these task relationships. Tasks can be similarly or differently distributed or have clusters of similarly distributed tasks. There may also arise situations where most tasks are similar to each other, while some few tasks differ in their distributions. We denote the latter as outlier tasks. The vast distributional differences may affect the predictive power of the model fit, such that the model may predict poorly on the majority class tasks and on outlier tasks. It is therefore desirable to identify such tasks and to address them specifically to improve the model performance.

Identification of outlier tasks has been investigated significantly from an algorithmic perspective. One popular framework is robust multi-task learning, initially proposed in Chen, Zhou, and Ye (2011) [21]. Gong, Ye and Zhang (2012) [37] proposed working with parameters of each task's parametric models, which are defined as $\mathbf{f}_i(\mathbf{x}_j^{(i)}) = (\mathbf{x}_j^{(i)})^T \mathbf{w}_i$, where $\mathbf{w}_i$ are parameter vector of task i. The parameter vectors are then combined as columns in a matrix $W \in \mathbf{R}^{d \times m}$ and decomposed as $W = P + Q$, and separate penalties are imposed on these matrices, such that P encodes the shared features among tasks and Q captures the outlier tasks. The latter step is done in order to simplify the optimization problem. Thus, this method learns both the shared features and outlier tasks. The loss minimization is then performed using a modified version of gradient descent. Gong et al (2012) [37] also decomposed the matrix W with group sparsity, while Pu, Jiang, Wang, and Xue (2013) [62] proposed to solve a similar problem by using the accelerated proximal method. Chen, Liu, and Ye (2012) [20] introduced a multi-task model which is robust to influence of outlier tasks. Kumar and Hal (2012) [50] proposed that there exist some basis tasks, and all other tasks can be expressed as linear combinations of these tasks. Zhong, Pu, Jiang, Feng, and Xue (2016) [86] further relax the tasks group structures assumptions to identify them instead. Jeong and Jun (2018) [45] attempt a similar approach by optimizing two coefficient matrices based on a low-rank assumption. It should be noted, however, that a general limitation of these frameworks is that they can only accommodate parametric models, which limits their applicability to nonparametric relationships.

Many other frameworks apply the idea of task clustering in order to group tasks by using algorithmic methods. Thrun and O'Sullivan (1996) [72] propose the TC algorithm for binary classification, which creates a hierarchy of tasks by maximizing their generalization accuracy, and transfers the knowledge selectively across the related tasks when predicting. The strength of this method is that it can deal effectively with tasks that were not used during the model training. Jacob et al (2008) [41] proposed a framework of task clustering by using the linear model parameters matrix W and by using a customized spectral norm. In the Bayesian paradigm, Xue, Liao, Carin, and Krishnapuram (2007) [79] introduced an application of Dirichlet processes to multi-task logistic regression in two formulations. In the symmetric formulation, all the tasks are learned jointly, while in the asymmetric formulation, learning a new task doesn't require access to data of all the previous tasks. In another work in the Bayesian framework, Xiong, Bi, Rao, and Cherkassky (2007) [78] propose a probabilistic model to identify relevant tasks by identifying feature patterns among the tasks. Kang, Grauman, and Sha (2011) [46] proposed using parameter matrices W_g for task group g out of totally G separate task groups, and solving this multi-task feature learning problem by iterative procedure with mixed integer programming. Hernandez-Lobato D., Hernandez-Lobato J.M., and Ghahramani (2015) [39] propose a probabilistic modeling approach to identify both task outliers and feature outliers. Zhang Xiaotong, Zhang Xianchao, Liu H., and Liu X. (2018) [83] define partially related tasks and identify them by constructing a similarity matrix with clustering algorithms. Liu, Zheng, Wu, Yu, and Wong (2020) [54] propose an algorithm to identify correlations among task features with graph-clustering.

It is generally the case that the existing works in task diagnostics for multi-task learning are highly algorithmic in their nature, not motivated by the statistical groundwork, and have a major focus on the development of optimization procedures. There is a significant focus on linear and parametric models, limiting their applicability to more complex datasets. Another distinct tendency is the assumption of all tasks being equally related to each other in every cluster. The focus is mostly on clustering the tasks into groups, not on identifying the outlier tasks and performing remedial measures. In this dissertation, I propose methods that seek to overcome these limitations, have a statistical foundation, and also to generalize the algorithmic clustering methods.

Chapter 3

Analysis of regularized multi-task learning

The purpose of this section is to perform deeper analysis and research in the framework of mean-regularized multi-task kernel regularization. Firstly, I introduce a different version of the mean-regularized multi-task kernel, which uses task coupling parameter c and works differently from a version with λ. I establish the conditions to make these versions to be equivalent both for ridge regression and support vector machine models, proving these results theoretically and demonstrating the equivalency through experiments. I also study the experimental conditions in Evgeniou et al (2005) [30] to make the experiments easier to replicate and study the nonlinear Gaussian kernel's influence on the performance of the mean-regularized multi-task model.

3.1 Setting up $\gamma = 1$

Evgeniou et al (2005) [30] perform an experiment on ILEA schools data with mean-regularized multi-task regularization, using linear kernel and $\gamma = 1$. For more information about the dataset, see Appendix A. Recall the minimization criteria (2.10):

$$S(\mathbf{w}) = \frac{1}{nm} \sum_{j \in \mathbb{N}_m} \sum_{i \in \mathbb{N}_n} L(y_{i,j}, \mathbf{w}^T \mathbf{B}_l \mathbf{x}_{i,j}) + \gamma \mathbf{w}^T \mathbf{w} \tag{3.1}$$

In comparison, in the modern package for support vector machines LIBSVM [18], the minimization criteria for regression is somewhat different, where it is a cost parameter C that controls the trade-off of the loss and penalty:

$$C \sum_{i=1}^{n} (\xi_i + \xi_i^*) + \frac{1}{2} \mathbf{w}^T \mathbf{w} \tag{3.2}$$

where the slack variables ξ_i and ξ_i^* play the role of a loss function.

To achieve $\gamma = 1$ in the LIBSVM's SVR minimization criteria, we need to choose C to average over all samples and to compensate for $\frac{1}{2}$ in the regularizer. Therefore, choose $C = \frac{1}{2nm} = \frac{1}{2n_{test}}$ where n_{test} is the number of samples in the test set for all tasks. Then, the penalties differ only by a factor of $\frac{n_{test}}{2}$, which is a multiplicative constant in both of the terms and does not affect the results of the optimization.

Using the mean-regularized multi-task kernel and LIBSVM through R package e1071, the parameters for cross-validation are mean-regularization task coupling parameter λ, and SVR parameters ϵ and C for the case of the linear kernel. This result will be used in the experimental part of this chapter.

3.2 Analysis of a simple case of mean-regularized MTL

To further investigate the inner workings of the mean-regularized MTL kernel, I design a simple simulation study. I control the data-generating process to make every step of the algorithm easy to demonstrate.

Let there be $m = 2$ tasks with each task having a shared input space $\mathbf{X}^d \in \mathbb{R}^d$ with $d = 2$, i.e. there are two predictor variables. Define the predictor variables and error terms as:

$$x_1 \sim U(5,8),\ x_2 \sim U(15,25),\ \epsilon \sim N(0,7)$$

The tasks 1 and 2 are defined as:

$$\begin{aligned} y_1 &= 5x_1 + 12x_2 + \epsilon \\ y_2 &= 5x_1 + 8x_2 + \epsilon \end{aligned}$$

Thus, the dataset consists of observations from 2 tasks, with observations described by 2 quantitative predictor variables with an error term, and 1 quantitative response variable. The response variable y is a linear function of the predictor variables and the error term, but the linear relationship is slightly different for each task. The coefficients for x_1 are the same in both tasks, but they differ for x_2. Because the individual coefficients are close to each other, it becomes reasonable to model the data by using a mean-regularized MTL kernel.

For this simple case with $m = 2$ tasks and $d = 2$ predictor variables, consider the case that we have only 4 observations per task. With the $75 - 25\%$ train-test split, it leaves 3 observations in the training set and 1 observation in the test set per task. Thus, the training and test sets have 6 and 2 observations in total, respectively.

To be consistent for all the steps, the following randomly generated dataset and train-test split will be used:

- Training set:

y	task	x_1	x_2
330.5173	1	5.404407	24.66323
250.9219	1	7.866185	17.47271
302.9691	1	5.492322	22.65991
205.5379	2	5.070766	15.91489
219.7720	2	6.021867	16.28367
260.8539	2	5.136080	20.49508

- Test set:

y	task	x_1	x_2
220.6514	1	5.933868	16.58836
217.8197	2	5.586355	15.56803

First, consider the $\mathbf{B}_l$ version with λ for mean-regularized multi-task learning as originally developed by Evgeniou (2005) [30]. As described in Chapter 2.5, $\mathbf{B}_l$ is a $(m+1)d \times d = 6 \times 2$ matrix that captures the relationships between tasks:

$$\mathbf{B}_l^T = [\sqrt{1-\lambda}\mathbf{I}_d, \underbrace{\mathbf{0}, ..., \mathbf{0}}_{l\text{-}1}, \sqrt{\lambda m}\mathbf{I}_d, \underbrace{\mathbf{0}, ..., \mathbf{0}}_{m\text{-}l}] = [\sqrt{1-\lambda}\mathbf{I}_2, \underbrace{\mathbf{0}, ..., \mathbf{0}}_{l\text{-}1}, \sqrt{2\lambda}\mathbf{I}_2, \underbrace{\mathbf{0}, ..., \mathbf{0}}_{m\text{-}l}]$$

For illustration, consider the following cases:

- Task $l = 1$, $\lambda = 0$:

$$\mathbf{B}_1^T = [\sqrt{1-0}\mathbf{I}_2, \sqrt{0 \times 2}\mathbf{I}_2, \mathbf{0}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Task $l = 2$, $\lambda = 0.2$:

$$\mathbf{B}_2^T = [\sqrt{0.8}\mathbf{I}_2, \mathbf{0}, \sqrt{2 \times 0.2}\mathbf{I}_2] = \begin{bmatrix} 0.8944 & 0 & 0 & 0 & 0.6324 & 0 \\ 0 & 0.8944 & 0 & 0 & 0 & 0.6324 \end{bmatrix}$$

- Task $l = 2$, $\lambda = 1$:

$$\mathbf{B}_2^T = [\sqrt{1-1}\mathbf{I}_2, \mathbf{0}, \sqrt{2 \times 1}\mathbf{I}_2] = \begin{bmatrix} 0 & 0 & 0 & 0 & 1.4142 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.4142 \end{bmatrix}$$

The data transformation step involves pre multiplying the x vectors for each observation by the $\mathbf{B}_l$ matrix of the observation's task. In particular, denote $x_l = [x_{1,l}, x_{2,l}]^T$ as the data vector for an observation from task l. Then, the data transformation step is $\mathbf{B}_l\mathbf{x}_l$. This product of a 6×2 matrix and 2×1 vector results in a 6×1 vector that encodes the data transformation for the observation. In other words, a 2×1 vector is mapped to a 6×1 vector. In the statistical literature it is traditional to write observation vectors as row vectors, while in machine learning literature, the observation vectors are column vectors. The 6×1 column vector of the transformed data above is still encoded in the machine learning notation, so we transpose it into a 1×6 row vector, such that it takes up a row of the transformed dataset.

Continuing with the examples for the matrices $\mathbf{B}_l$ above, applied to the training set data using the λ version:

- Task $l = 1$, $\lambda = 0$:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 5.404407 & 24.66323 \\ 7.866185 & 17.47271 \\ 5.492322 & 22.65991 \end{bmatrix} \right)^T$$

$$= \left(\begin{bmatrix} 5.404407 & 24.66323 & 0 & 0 & 0 & 0 \\ 7.866185 & 17.47271 & 0 & 0 & 0 & 0 \\ 5.492322 & 22.65991 & 0 & 0 & 0 & 0 \end{bmatrix} \right)^T$$

- Task $l = 2$, $\lambda = 0.2$:

$$\begin{bmatrix} 0.8944 & 0 \\ 0 & 0.8944 \\ 0 & 0 \\ 0 & 0 \\ 0.6324 & 0 \\ 0 & 0.6324 \end{bmatrix} \left(\begin{bmatrix} 5.070766 & 15.91489 \\ 6.021867 & 16.28367 \\ 5.136080 & 20.49508 \end{bmatrix} \right)^T$$

$$= \left(\begin{bmatrix} 4.535431 & 14.23471 & 0 & 0 & 3.207034 & 10.06546 \\ 5.386121 & 14.56456 & 0 & 0 & 3.808563 & 10.29870 \\ 4.593849 & 18.33135 & 0 & 0 & 3.248342 & 12.96222 \end{bmatrix} \right)^T$$

- Task $l = 2$, $\lambda = 1$:

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1.4142 & 0 \\ 0 & 1.4142 \end{bmatrix} \left(\begin{bmatrix} 5.070766 & 15.91489 \\ 6.021867 & 16.28367 \\ 5.136080 & 20.49508 \end{bmatrix} \right)^T$$

$$= \left(\begin{bmatrix} 0 & 0 & 0 & 0 & 7.171146 & 22.50705 \\ 0 & 0 & 0 & 0 & 8.516206 & 23.02859 \\ 0 & 0 & 0 & 0 & 7.263514 & 28.98441 \end{bmatrix} \right)^T$$

It becomes evident that in the case of single-task learning of $\lambda = 0$, the mean-regularized MTL method merely reshapes the design matrix, adding additional columns of zeroes, which are merely placeholders of the method for other λ values, and do not have any influence on the model solution for the problem due to being null.

When $\lambda = 0.2$, the transformation scales and transfers the data into the first block matrix with multiplier $\sqrt{1 - \lambda} = 0.8944$ and into the third block matrix with multiplier $\sqrt{\lambda m} = 0.6324$. The first block matrix serves the role of the mean regularization of all model parameters together, while the task-specific block (here, the third block matrix) is situated in columns that are exclusive to the task itself, thus facilitating learning of the task-specific parameters.

In the case of independent task learning of $\lambda = 1$, we see that the task data is scaled with $\sqrt{\lambda m} = 1.4142$ and transferred into a block matrix. This block matrix has columns that are zero for all other tasks in the dataset and for the first mean-regularization block matrix. Therefore, the result works as a block-diagonal matrix where the blocks are task-specific. Per Evgeniou et al (2005) [30], fitting a model on this transformed data will result in learning all tasks independently, so that there is no information transfer between the tasks.

3.3 Equivalence of two different versions

Another definition of mean-regularized multi-task kernel was presented in Michelli and Pontil (2004) [59]. It is different in the way the data is transformed to multi-task kernel feature space, and creates a different regularization function. The new task coupling parameter is $c > 0$, and the matrices B_l for each task l are constructed as:

$$\mathbf{B}_l^T = [c^{-1}\mathbf{I}_d, \underbrace{\mathbf{0}, ..., \mathbf{0}}_{l\text{-}1}, \mathbf{I}_d, \underbrace{\mathbf{0}, ..., \mathbf{0}}_{m\text{-}l}]] \tag{3.3}$$

As the task coupling parameter c is any positive number, note that there is never the case of completely independent task learning, as the first block identity matrix, which encodes the mean of the task parameters, never fully disappears. In the subsequent sections, I perform a deeper investigation of the new task coupling parameter c.

The new version of the mean-regularized kernel leads to the following regularization term:

$$J(\mathbf{u}) = \frac{c^2}{m + c^2} \sum_{l \in \mathbb{N}_m} \|\mathbf{u}_l\|^2 + \frac{m}{m + c^2} \sum_{l \in \mathbb{N}_m} \|\mathbf{u}_l - \frac{1}{m} \sum_{q \in \mathbb{N}_m} \mathbf{u}_q\|^2 \tag{3.4}$$

Note that the regularized quantities are the same as in Equation 2.13. However, there is a difference in the way the trade-off is achieved. Moreover, data premultiplied with the new matrices B_l with task coupling parameter c is different from the same data premultiplied with the version using λ. In the following subsections, I introduce and prove the conditions necessary for the equivalency of the two penalties, which are demonstrated through a simple example. Then, I further simplify the conditions for the case of ridge regression.

3.3.1 The equivalence of the two penalties

Assume that

$$\lambda = \frac{c^2}{m + c^2} \tag{3.5}$$

.

Then,

$$c = \sqrt{\frac{\lambda m}{1 - \lambda}} \tag{3.6}$$

$$1 - \lambda = \frac{m}{m + c^2} \tag{3.7}$$

As $c > 0$, $\lambda \in (0, 1)$. $\lambda = 1$ corresponds to $c = \infty$ and implied division by 0, therefore the case of independent task learning is never fully achieved, but is approximated as c becomes sufficiently large. The case of $\lambda = 0$ is not included in the original definition of the mean-regularized multi-task kernel but is further examined in subsequent sections. At this point, note that setting $\lambda = 0$ into the equivalency Equation 3.6 leads to $c = 0$, which violates the condition that $c > 0$, and leads to division by 0 in Equations 3.3 of B_l, thus it isn't achievable either.

Let

$$A = \sum_{l \in \mathbb{N}_m} \|\mathbf{u}_l\|^2$$

$$B = \sum_{l \in \mathbb{N}_m} \|\mathbf{u}_l - \frac{1}{m} \sum_{q \in \mathbb{N}_m} \mathbf{u}_q\|^2$$

Denote $J_1(\mathbf{u})$ to be the penalty of the λ version of mean-regularized multi-task kernel, and $J_2(\mathbf{u})$ to be the penalty of the version with c. Then:

$$J_1(\mathbf{u}) = \frac{1}{m}\left(A + \frac{1-\lambda}{\lambda}B\right)$$

$$J_2(\mathbf{u}) = \frac{c^2}{m+c^2}A + \frac{m}{m+c^2}B$$

Replacing with parameter λ by using equations 3.5 and 3.7:

$$J_2(\mathbf{u}) = \lambda A + (1-\lambda)B = m\lambda\frac{1}{m\lambda}(\lambda A + (1-\lambda)B) = m\lambda\frac{1}{m}(A + \frac{1-\lambda}{\lambda}B) = m\lambda J_1(\mathbf{u})$$

It follows that $J_2(\mathbf{u}) = m\lambda J_1(\mathbf{u})$ and $J_1(\mathbf{u}) = \frac{J_2(\mathbf{u})}{m\lambda}$. Therefore, the penalty equivalence is achieved when $\lambda = \frac{c^2}{m+c^2}$. It remains to establish the equivalence of minimizing criteria, and I begin with a continuation of the demonstrative example.

3.3.2 Continuation of the simple example

In the following, I extend the simple example of the Section 3.2 with the c-version of the mean-regularized multi-task kernel.

The dimensionality of the matrices $\mathbf{B}_l$ of λ and c versions is the same, and we've shown previously that the relationship between c and λ is given by $c = \sqrt{\frac{\lambda m}{1-\lambda}}$. The number of tasks $m = 2$, and let sequence of λ run from 0 to 1 in increments of 0.1. As $c > 0$, the edge cases of $\lambda = 0$ and 1 are limiting cases, and approximations to these cases were taken as λ close to 0 (0.00001) and 1 (0.99999). This yields in the sequence: $c \in \{0.0045, 0.4714, 0.7071, 0.9258, 1.1547, 1.4142, 1.7321, 2.1602, 2.8284, 4.2426, 447.2114\}$

The following illustrating examples are created for the equivalent cases as above for the λ version:

- Task $l = 1$, $c = 0.0045$ (equivalent to $\lambda = 0.00001$):

$$\mathbf{B}_1^T = [0.0045^{-1}\mathbf{I}_2, \mathbf{I}_2, \mathbf{0}] = \begin{bmatrix} 222.22 & 0 & 1 & 0 & 0 & 0 \\ 0 & 222.22 & 0 & 1 & 0 & 0 \end{bmatrix}$$

- Task $l = 2$, $c = 0.7071$ (equivalent to $\lambda = 0.2$):

$$\mathbf{B}_2^T = [0.7071^{-1}\mathbf{I}_2, \mathbf{0}, \mathbf{I}_2] = \begin{bmatrix} 1.4142 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1.4142 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Task $l = 2$, $c = 447.2114$ (equivalent to $\lambda = 0.99999$):

$$\mathbf{B}_2^T = [447.2114^{-1}\mathbf{I}_2, \mathbf{0}, \mathbf{I}_2] = \begin{bmatrix} 0.0022 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.0022 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The single-task learning and independent task learning examples above are defined as $c \to 0$ and $c \to \infty$, respectively, for the coupling parameter $c > 0$. Those limits are equivalent to $\lambda = 0$ and $\lambda = 1$, as discussed previously.

As $c \to 0$, the first block matrix explodes in size, compared to the task block of the identity matrix. This leads to a stronger influence of the average component and reduces the influence of the task component. Therefore, when used in the data transformation step, the impact on the data is comparable to the impact of the λ version. The only difference is that the values get multiplied by a large number in the c version, and realistically, even for a very small c, there is still some numerical influence of the task block matrix, which ideally should not happen in the single-task learning.

As $c \to \infty$, the first block matrix vanishes away, and the inner mechanism is a reverse of the previous example. And similarly to the above, for a numeric approximation of a very large c, the mean component never fully vanishes away.

For the cases of λ between 0 and 1, there is a deterministic relationship between c and λ. As an example, for the case of $\lambda = 0.2$ and $c = 5.895$, note that the $\mathbf{B}_2$ matrices are quite different. In the λ version, the first block matrix has a slightly higher influence with a multiple of 0.8944 than the task matrix with a multiple of 0.6324. In the c version, the first block matrix multiple is 0.1696 and the task matrix is an identity matrix. Thus, in the c version, the influence of the task data is much stronger than in the equivalent λ case. In other words, turning up c has a much stronger push to include the task information separately, compared to the equivalent task coupling parameter values in λ version, where this increase is more gradual.

Consider also a transformation of the same data using the mean-regularized MTL kernel version with the coupling parameter c:

- Task $l = 1$, $c = 0.0045$ (equivalent to $\lambda = 0.00001$):

$$\begin{bmatrix} 222.22 & 0 \\ 0 & 222.22 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 5.404407 & 24.66323 \\ 7.866185 & 17.47271 \\ 5.492322 & 22.65991 \end{bmatrix} \right)^T$$

$$= \left(\begin{bmatrix} 1200.979 & 5480.718 & 5.404407 & 24.66323 & 0 & 0 \\ 1748.041 & 3882.824 & 7.866185 & 17.47271 & 0 & 0 \\ 1220.516 & 5035.536 & 5.492322 & 22.65991 & 0 & 0 \end{bmatrix} \right)^T$$

- Task $l = 2$, $c = 0.7071$ (equivalent to $\lambda = 0.2$):

$$\begin{bmatrix} 1.4142 & 0 \\ 0 & 1.4142 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 5.070766 & 15.91489 \\ 6.021867 & 16.28367 \\ 5.136080 & 20.49508 \end{bmatrix} \right)^T$$

$$= \left(\begin{bmatrix} 7.171215 & 22.50726 & 0 & 0 & 5.070766 & 15.91489 \\ 8.516287 & 23.02881 & 0 & 0 & 6.021867 & 16.28367 \\ 7.263583 & 28.98469 & 0 & 0 & 5.136080 & 20.49508 \end{bmatrix} \right)^T$$

- Task $l = 2$, $c = 447.2114$ (equivalent to $\lambda = 0.99999$):

$$\begin{bmatrix} 0.0022 & 0 \\ 0 & 0.0022 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 5.070766 & 15.91489 \\ 6.021867 & 16.28367 \\ 5.136080 & 20.49508 \end{bmatrix} \right)^T$$

$$= \left(\begin{bmatrix} 0.01133863 & 0.03558694 & 0 & 0 & 5.070766 & 15.91489 \\ 0.01346537 & 0.03641157 & 0 & 0 & 6.021867 & 16.28367 \\ 0.01148468 & 0.04582861 & 0 & 0 & 5.136080 & 20.49508 \end{bmatrix} \right)^T$$

The version with c works through a trade-off of the mean regularization and individual task learning, thus it generally replicates the mechanism of the λ version. As expected, in cases of single task and independent task learning, the dataset in the c version has a different structure than λ versions due to the approximation of c values. The single-task and independent task learning cannot be explicitly implemented in the c version, as they are defined as limit cases.

Consider the case of $\lambda = 0.2$ and $c = 0.7071$, and note that even though the datasets look different at first glance, they have the same structure and they only differ by the multiple of 1.5811, such that the c data matrix equals to 1.5811 multiplied element-wise with the λ data matrix. This relationship holds also for other equivalent values of λ and c, not illustrated here, although the proportionality constant changes. Therefore, even though λ and c were made equivalent to make the penalty terms equal, they still differ in how the data itself is transformed.

3.3.3 Dataset equivalency

It was previously shown that the kernel function for mean-regularized kernel versions with λ and c encodes their equivalence by $c = \sqrt{\frac{\lambda m}{1-\lambda}}$. Although this equation connects the penalties encoded by the two versions, we note also that the datasets of the two versions differ by proportionality constant. As demonstrated above, this constant depends on the value of the task coupling parameter, and we expect it to differ by the total number of tasks.

The raw data is always kept constant in our experiments, therefore, both transformed datasets will equal to each other when their transformation matrices $\mathbf{B}_l$ are equal. In the following, I establish equivalence between these matrices for the λ and c versions.

Lemma 3.3.1. *The equivalence of datasets transformed with mean-regularized multi-task kernels with λ and c versions is established when the transformation matrices $\boldsymbol{B}_l$ are connected as $\boldsymbol{B}_l^{<\lambda>} = \sqrt{\lambda m}\, \boldsymbol{B}_l^{<c>}$ for the equivalent values of λ and c.*

This relation holds for any number of tasks and value of λ and c, when equation $c = \sqrt{\frac{\lambda m}{1-\lambda}}$ is used to establish equivalency between the mean-regularized kernels. For the simple example of two tasks considered above, when $\lambda = 0.2$ and, equivalently, $c = 0.7071$:

- Task $l = 2$, $\lambda = 0.2$:

$$\mathbf{B}_l^{<\lambda>}\mathbf{x}_l = \left(\begin{bmatrix} 4.535431 & 14.23471 & 0 & 0 & 3.207034 & 10.06546 \\ 5.386121 & 14.56456 & 0 & 0 & 3.808563 & 10.29870 \\ 4.593849 & 18.33135 & 0 & 0 & 3.248342 & 12.96222 \end{bmatrix} \right)^T$$

- Task $l = 2$, $c = 0.7071$:

$$\begin{aligned}\mathbf{B}_l^{<c>}\mathbf{x}_l &= \left(\begin{bmatrix} 7.171215 & 22.50726 & 0 & 0 & 5.070766 & 15.91489 \\ 8.516287 & 23.02881 & 0 & 0 & 6.021867 & 16.28367 \\ 7.263583 & 28.98469 & 0 & 0 & 5.136080 & 20.49508 \end{bmatrix} \right)^T \\ &= \frac{1}{\sqrt{\lambda n}}\mathbf{B}_l^{<\lambda>}\mathbf{x}_l \\ &= \frac{1}{\sqrt{0.2 \times 2}}\left(\begin{bmatrix} 4.535431 & 14.23471 & 0 & 0 & 3.207034 & 10.06546 \\ 5.386121 & 14.56456 & 0 & 0 & 3.808563 & 10.29870 \\ 4.593849 & 18.33135 & 0 & 0 & 3.248342 & 12.96222 \end{bmatrix} \right)^T \\ &= \left(\begin{bmatrix} 7.171146 & 22.50705 & 0 & 0 & 5.070766 & 15.91489 \\ 8.516206 & 23.02859 & 0 & 0 & 6.021867 & 16.28367 \\ 7.263514 & 28.98441 & 0 & 0 & 5.136080 & 20.49508 \end{bmatrix} \right)^T\end{aligned}$$

with the minor differences due to the rounding error. The multiple is $\frac{1}{\sqrt{0.2 \times 2}} = 1.5811$, just as we observed in the previous section.

Therefore, we see that both mean-regularized kernels are equivalent, with their datasets differing by a constant of proportionality $\sqrt{\lambda m}$ and the equivalency between the coupling parameters given by $c = \sqrt{\frac{\lambda m}{1-\lambda}}$.

3.3.4 Ridge regression equivalence

In this section, I seek to investigate the inner workings of the ridge regression estimator with mean-regularized multi-task kernel versions with λ and c task coupling parameters. Ridge regression fits perfectly into the framework of Evgeniou et al (2005) [31], as its minimization criterion contains the L2 penalty term. The solution of ridge regression parameters has a closed form, which eliminates randomness differences that could happen when the solution has to be found sequentially. Compared to support vector regression, the advantage of this approach is that all model fitting steps are tractable. The ridge regression estimator and fitted values are defined as

$$\hat{\beta}(\lambda_{RR}) = (\mathbf{X}^T\mathbf{X} + \lambda_{RR}\mathbf{I}_p)^{-1}\mathbf{X}^T Y \tag{3.8}$$

$$\hat{Y}(\lambda_{RR}) = \mathbf{X}\hat{\beta}(\lambda_{RR}) \tag{3.9}$$

First, let's consider the behavior of the estimator for equivalent values of λ and c, without transforming the data. For both versions of $\mathbf{B}_l$, I use the same value of the smoothing parameter $\lambda_{RR} = 1$ and do not fit intercept as it wasn't used to generate the data. The models were fitted separately on each of the previously demonstrated matrices, and the fitted values were computed for the test set for each model fit (recall that $n_{test} = 2$):

y_{test}	220.6514	217.8197
$\lambda = 0$	226.0392	212.2036
$c = 0.0045$	231.4299	207.2989
$\lambda = 0.2$	231.3304	206.7853
$c = 0.7071$	231.3596	207.171
$\lambda = 1$	231.2249	207.0967
$c = 447.2114$	231.1072	206.7309

Note that the fitted values differ for the equivalent values of λ and c. This is to be expected, as we've previously established that the matrices B_l differ by a constant of proportionality, as described in Lemma 3.3.1. This was done intentionally for the demonstration purposes, as in the following I establish that in the case of ridge regression the two versions of the mean-regularized multi-task kernel can be made equivalent without data manipulations per Lemma 3.3.1.

Recall that we have defined $\mathbf{X}$ to be a design matrix with d predictor variables, n to be the number of tasks, and $\mathbf{B}_l^{<\lambda>}, \mathbf{B}_l^{<c>}$ to be the mean-regularized multi-task kernel transformation matrices for task l using λ and c versions, respectively. The data transformation is achieved by concatenating matrix products $\mathbf{B}_l^{<\lambda>}\mathbf{X}_l$, where $\mathbf{X}_l$ are the rows of matrix $\mathbf{X}$ corresponding to task l.

In the following, let $\mathbf{X}^{<\lambda>}$ be matrix that contains the data after the multi-task kernel transformation. Then, $\mathbf{X}^{<\lambda>}$ and $\mathbf{X}^{<c>}$ each have dimensions $nm \times d(n+1)$. From previous results, we know that $\mathbf{X}^{<\lambda>} = \sqrt{\lambda n}\mathbf{X}^{<c>}$, which follows from the fact that $c = \sqrt{\frac{\lambda m}{1-\lambda}}$. Let $\hat{w}_\lambda = (\mathbf{X}^{<\lambda>^T}\mathbf{X}^{<\lambda>} + \gamma_\lambda I)^{-1}\mathbf{X}^{<\lambda>^T}\mathbf{Y}$ be the ridge regression estimator on the datasets transformed using λ version, defined similarly for the c version.

Theorem 3.3.2. *The ridge regression estimators for λ and c versions of mean-regularized MTL kernel yield equal fitted values on their respective datasets $\boldsymbol{X}^{<\lambda>}$ and $\boldsymbol{X}^{<c>}$, given that $\gamma_\lambda = \lambda m \gamma_c$.*

Proof.

$$\begin{aligned}
\hat{w}_c &= (\mathbf{X}^{<c>^T}\mathbf{X}^{<c>} + \gamma_c I)^{-1}\mathbf{X}^{<c>^T}\mathbf{Y} \\
&= (\frac{1}{\sqrt{\lambda m}}\mathbf{X}^{<\lambda>^T}\frac{1}{\sqrt{\lambda m}}\mathbf{X}^{<\lambda>} + \gamma_c I)^{-1}\frac{1}{\sqrt{\lambda m}}\mathbf{X}^{<\lambda>^T}\mathbf{Y} \\
&= \sqrt{\lambda m}(\mathbf{X}^{<\lambda>^T}\mathbf{X}^{<\lambda>} + \lambda m \gamma_c I)^{-1}\mathbf{X}^{<\lambda>^T}\mathbf{Y}
\end{aligned}$$

If we let $\gamma_\lambda = \lambda m \gamma_c$, then $\hat{w}_c = \sqrt{\lambda m}\hat{w}_\lambda$.

Ridge regression estimates yield in fitted values by $\hat{\mathbf{Y}} = \mathbf{X}\hat{w}$. For the λ version of the kernel, $\hat{\mathbf{Y}} = \mathbf{X}^{<\lambda>}\hat{w}_\lambda = \sqrt{\lambda m}\mathbf{X}^{<c>}\frac{1}{\sqrt{\lambda m}}\hat{w}_c = \mathbf{X}^{<c>}\hat{w}_c$. Therefore, the ridge regression fitted values of both transformed datasets are equal, provided that $\gamma_\lambda = \lambda m \gamma_c$ and $c = \sqrt{\frac{\lambda m}{1-\lambda}}$. □

It becomes clear that the methods are fully equivalent when the cost is adjusted appropriately. Notably, this adjustment varies by the value of λ. Thus we have established the conditions necessary to make both versions equivalent in practice, without the need to make the datasets equivalent like in the general case. Note that it was made possible by the fact that the ridge regression estimator has a closed-form solution, so we were able to solve our way to the relationship between regularizing constants. For support vector regression no close form solution exists, therefore such a relationship cannot be easily established, although we can reasonably expect that such a relationship between costs may exist. It is an open research question to examine this further.

3.3.5 Ridge regression demonstration

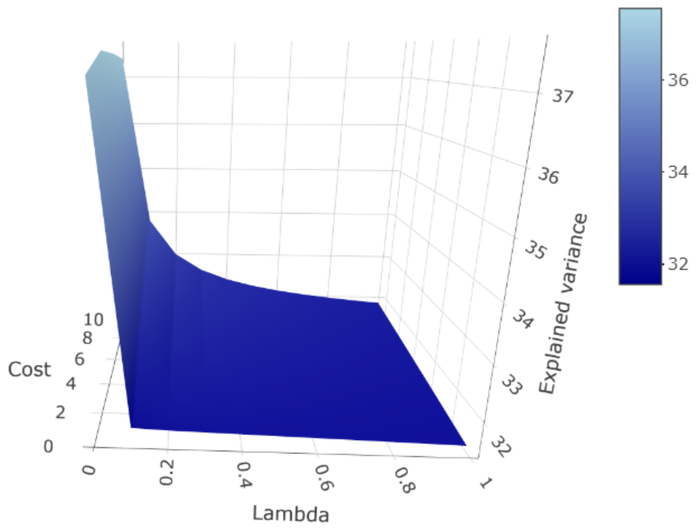

Figure 3.1: Explained variance of ridge regression on the ILEA schools data, applied with mean-regularized multi-task kernel λ version for a range of λ from 0.000001 to 0.99.

Using the ILEA schools data, we can fit the ridge regression model to investigate the theoretical result. We consider scale of coupling λ from 0.000001 to 0.99 in

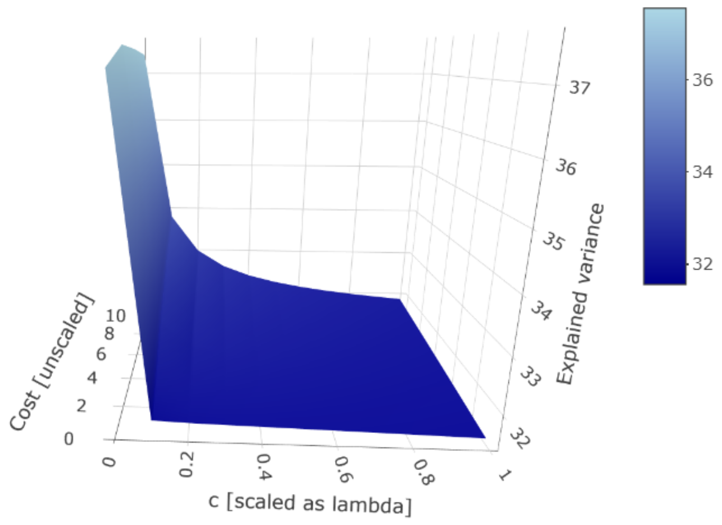

Figure 3.2: Explained variance of ridge regression on the ILEA schools data, applied with mean-regularized multi-task kernel c version for a range of c from 0.012 to 117.307, which are equivalent to λ values in Figure 3.1.

increments of 0.1, and its equivalent scale of c. We explicitly exclude the edge cases of λ being 0 and 1, as the c version cannot be made equivalent to these cases.

We consider an arbitrary scale of γ_λ from 0.1 to 10 in increments of 3.3. γ_c is then created for every value of γ_λ and λ for the fit of c-version. Theoretically, using the respective cost values, we expect that the fitted values will be exactly the same for two independent model fits on both datasets and therefore explained variance is expected to be the same. The results of model fits are illustrated as explained variance surfaces in Figures 3.1 and 3.2.

On Figure 3.2, c is scaled with the same intervals as λ, and γ_c is spaced similarly as γ_λ. The reason is that, while γ_λ is regularly spaced for each λ, γ_c is not regularly spaced for each c. Yet, values of γ_c are equivalent to their respective values of γ_λ, and thus it is visually appealing to scale the axis γ_c with the same intervals as on the axis of γ_λ.

The surfaces look identical, thus we can observe experimentally that the methods are indeed equivalent. Also, I have verified numerically that fitted values and explained variances are equal for both datasets.

These experiments were performed on the same fixed train-test split of data as was previously used in the experiments with a support vector machine. The ridge regression model achieves much higher maximum values of explained variance for $\lambda = 0.000001$. For the illustration, consider also values of 0 and 1, and redraw the explained variance surface with these values included in the λ version of mean-regularized MTL kernel, illustrated in Figure 3.3.

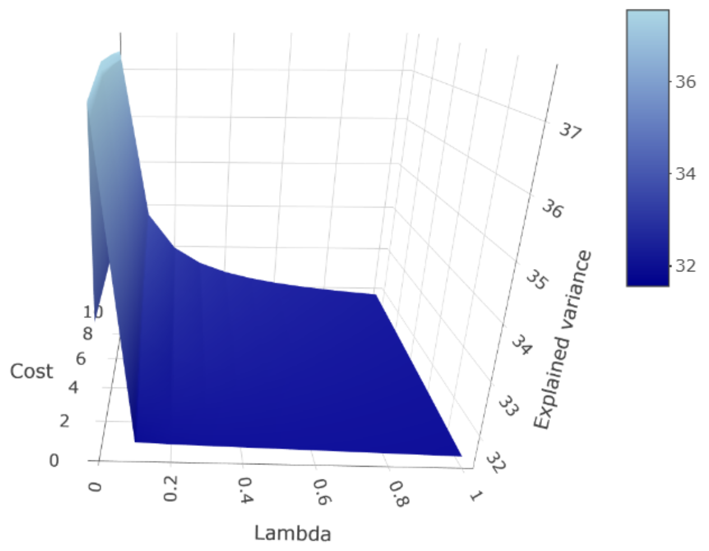

Figure 3.3: Explained variance of ridge regression on the ILEA schools data, applied with mean-regularized MTL kernel λ version for a full range of λ.

We see that there is a very sharp drop in explained variance at $\lambda = 0$ down to the lower range of 33%. This demonstrates that the mean-regularized MTL kernel is indeed useful for this data, and, in the case of ridge regression, we are much better off using a model that is close to single-task learning, but incorporates some small weight to the individual task components as well. As our previous experiments only considered a rough scale of λ, its values close to 0 weren't checked. This finding also partially explains why the c kernel has higher explained variance than λ kernel at $\lambda = 0$; it happened simply because the lowest value of c corresponded to a very low positive value of λ, and thus achieved higher performance than explicit single-task learning.

Notably, Evgeniou et al (2005) [31] use the SVR model but don't check the values of λ very close to 0. We find that, in the case of ridge regression, there is a very significant performance boost when ILEA data is rather considered multi-task data. it can be reasonable to expect that this is the case for SVR model as well. As such, our experiment demonstrates that the conclusion that ILEA schools data is single-task data was not correct.

3.4 Simulations of mean-regularized MTL kernel

The purpose of this section is to demonstrate experimentally the performance of the multi-task kernel models on ILEA schools data and to investigate the influence of the Gaussian kernel. Recall that in the experiment with ILEA data in Evgeniou et al (2005) [30], the mean-regularized multi-task kernel was used with parameter λ in the matrices $\mathbf{B}_l$, where $\lambda \in (0, 1]$ is the multi-task coupling parameter. In Micchelli et al (2004) [59], the authors redefine matrices $\mathbf{B}_l$ for the mean-regularized multi-task kernel, such that $c > 0$ is the multi-task coupling parameter.

Note that parameter γ takes the same value for both definitions of $\mathbf{B}_l$. In Evgeniou et al (2005) [30], the researchers take simplification in the experiments that $\gamma = 1$, which implies $C = \frac{1}{2nm} = \frac{1}{2n_{test}}$ in support vector regression model, as discussed previously.

The following illustrates a series of experiments for the ILEA data, expanding the experiments in Evgeniou et al (2005) [30]. Assume that $\epsilon = 0.1$ in soft-margin ϵ-SVR model, and the train-test split is $75\% - 25\%$ separately within each task. Note that those details weren't explicitly mentioned in Evgeniou (2005) [30], so I am operating under these conventional assumptions, which may cause inconsistencies in the results achieved. Hard-margin SVR, searching for ϵ, and train-test splits for the whole data at once (not per task) were tried but yielded equal or worse results than the results under the chosen assumptions.

For the experiments to be comparable to each other, the 10 fixed train-test splits were created and were kept constant across all experiments. Therefore, the only differences in performance across experiments are not due to randomness, but due to the models and their parameters.

In all experiments, I consider a range of values of $\lambda \in [0, 1]$ and their equivalent scale in c, as mentioned previously. The following experiments are performed:

1. λ-version of the linear mean-regularized multi-task kernel in SVR, search for optimal C.
2. λ-version of the radial basis function (Gaussian) mean-regularized multi-task kernel in SVR, search for optimal C and γ_{RBF}.

The assumption that $\gamma = 1$ in Evgeniou et al (2005) [30], such that $C = \frac{1}{2n_{test}}$, is too restrictive and is not indicative of the true potential performance of the model. It is highly unlikely to be the parameter value that achieves the global maximum of the explained variance. Moreover, any differences in how the SVR minimization criteria were implemented in Evgeniou et al (2005) [30] from its theoretical formulas, would make it impossible to replicate the results in the paper. Indeed, the experimental results, which are not included for brevity, show that the performance is unusually low under $C = \frac{1}{2n_{test}}$ when working with the LIBSVM library. Therefore, I search for the optimal C parameter, starting from a large grid, then narrowing it down to the values of C that yield the highest explained variance. Such simulations were performed in Experiment 1.

Additionally, I seek to investigate the benefit of relaxing the assumption of a linear kernel in SVR in order to improve the results. A linear kernel may prove to be too simple if the data is not linearly separable. As there are multiple predictor variables, this assumption is not possible to check graphically. Therefore, Experiment 2 was conducted. With the new parameter γ_{RBF} in the RBF kernel, the grid search will be expanded to cover this parameter to find its optimal values. This expansion to nonlinear kernels is done according to the theory of Evgeniou et al (2005) [30], such that the RBF kernel is applied to the data that is already transformed with matrices $\mathbf{B}_l$.

The interpretation of the results will be done with respect to the demonstration in Evgeniou et al (2005) [30]. For the ILEA data with mean-regularized multi-task linear kernel SVR using B_l with λ, the researchers observe a maximum point of approx. 34% explained variance at $\lambda = 0$. Expanding to a range of λ values, the explained variance stays at similar levels with a very slight decline until $\lambda = 0.6$, when it starts to drop gradually, being around 30% at $\lambda = 0.9$ and dropping substantially down to approx. 5% explained variance at $\lambda = 1$. It indicates that the data is best interpreted as coming from the single task, although quite marginally compared to the other low values of λs, and the substantial drop in performance at $\lambda = 1$ implies that all schools should not be considered as completely separate tasks.

In the Section 3.3, it was shown that the λ and c versions of the mean-regularized multi-task kernel can be made fully equivalent with the equivalence relations. As expected, I've also found experimentally that the c version produces exactly the same results as the λ version, and all the cross-validation figures that were created for the λ version are the same for the c version. Thus these aren't included for brevity.

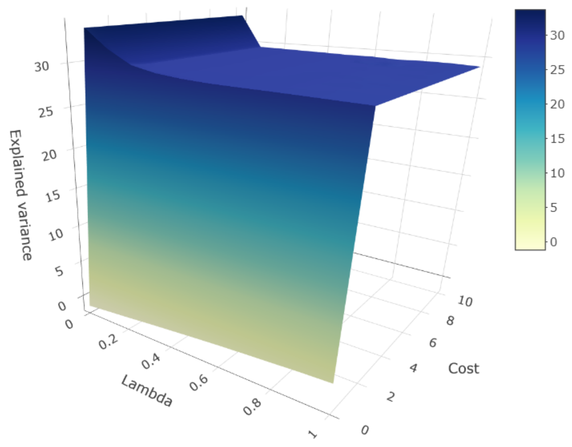

Figure 3.4: Experiment 1. Explained variance surface of mean-regularized MTL linear SVR for a range of values of C (Cost) by task coupling parameter λ

Expanding the experiment of Evgenious et al (2005) [30], I perform a grid search for values of C, which was first done on a large grid, and then narrowed down to a finer grid of values as seen in Figure 3.4. Across all λ, there is a sharp increase in explained variance when the value of C is increased. For a very low value of C, the explained variances are very low for all λ. As C is increased, the results are not sensitive to its value from 0.5 up to 10. We observe that the explained variance is very high for all λ. The maximum of explained variance is approx. 34%, in close agreement with the result in Evgeniou et al (2005) [30]. However, we observe a slight drop in explained variance as λ increases to 0.1 and beyond, which is different from the results in the paper, when the explained variance stayed relatively flat. On the good side, the explained variance doesn't drop at all $\lambda = 0.7$ to 1.0, therefore even when we consider the data as completely separate tasks, the explained variance is at a quite high level, compared to a drop to approx 5% in Evgeniou et al (2005) [30].

It is a notable finding that the C choice in Evgeniou et al (2005) [30] was actually optimal, even though it was seemingly chosen at random. The initial motivation for grid search was to find a value of C that replicates those results, and we see in Figure 3.4 that no such C exists. On the other hand, any other choice of C yields in a curve of explained variance that is significantly higher for bigger λ, and slightly lower for smaller λ.

In one of the experiments, $c = 0.012$ was attempted as an equivalent value for $\lambda = 0.000001$ task coupling parameter, since $c = 0$ is not defined for that definition. I've found that the maximum performance reaches to about 37% explained variance. This finding means that the dataset should not be viewed as a single-task learning dataset. Such a low value of λ encodes the fact that most of the information is shared between the tasks, and there is some minor information that plays an important role in some of the tasks, which is important for a stronger performance. Therefore, the ILEA schools dataset should not be seen as a single-task dataset.

In the Experiment 2, the objective is to investigate whether the Gaussian kernel improves the performance of the mean-regularized MTL kernel, compared to the results in the Experiment 1. There, the highest explained variance was approx. 34% for $\lambda = 0$. In general, the Gaussian kernel is better suited to model nonlinearities in the data. Even though it is not visualizable whether ILEA schools data is linearly separable, when the data is linear, the Gaussian kernel is usually expected perform at least as well as the linear kernel.

A grid of parameter combinations is displayed in Figure 3.5, where the 3 axes are λ, C, and γ_{RBF}. The explained variance achieved in a parameter combination is indicated by the depth of the red color. The highest values of explained variance are for $\lambda = 0$, with the best parameter combination of $C = 10$ and $\gamma_{RBF} = 0.037$, which yields explained variance of 35.53%. It is also notable that the explained variance is not linearly increasing in C and γ_{RBF}. Therefore, a fine grid search is not straightforward, is time-consuming, and is likely to be overfitting to the test sets, even though 10 train-test splits are used.

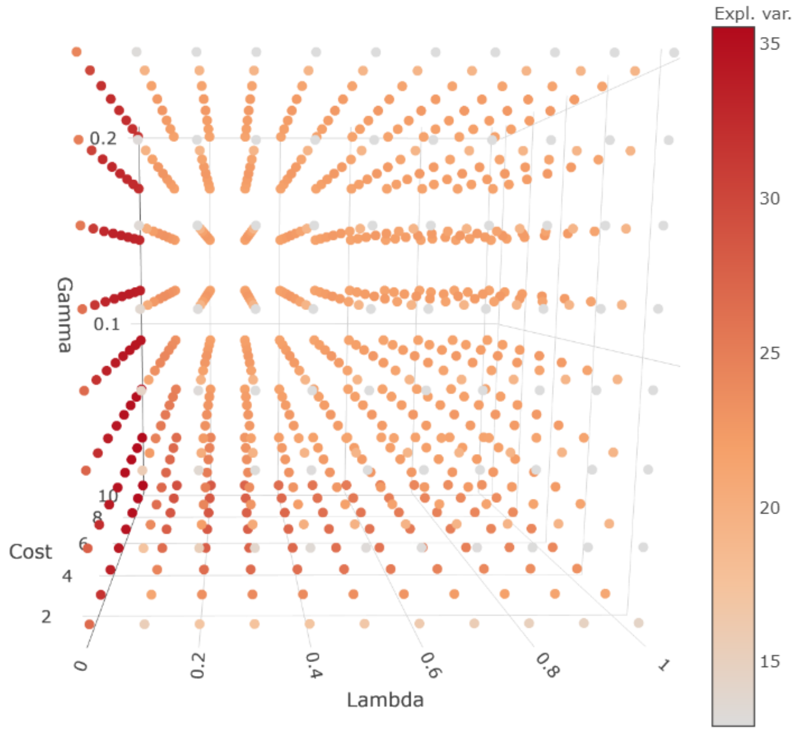

Figure 3.5: Experiment 2. Explained variance surface of mean-regularized MTL Gaussian SVR for a range of values of C (Cost) and γ_{RBF} by task coupling parameter λ.

The most important finding is that there is a performance improvement of approx. 1.5% compared to using the linear kernel for the same value of $\lambda = 0$. This suggests that the data is not linearly separable, and there is a significant benefit in using a nonlinear kernel. Even though the improvement is rather incremental, it shows that there exists a potential to improve the performance further by tuning kernels, which may include investigating other RBF types of kernels.

3.5 Explicit cases of single-task and independent task learning

In this section I perform experiments of single-task and independent task learning as explicit cases, meaning I do not use the mean-regularized multi-task kernel, but model these cases explicitly: merge all data together for single-task learning and create separate models for each task for independent task learning. Then, in order to learn more about the behavior of the mean-regularized multi-task learning with the kernel method, I compare their results with the results of the explicit cases.

Consider the simple case of mean-regularized B_l with coupling parameter λ and special cases of $\lambda = 0$ and $\lambda = 1$. The former is the case when all tasks are learned together as a single task, thus the data for all tasks is concatenated and the task variable is dropped. The latter is the case of each task learning independently from the other tasks. As such, we obtain a "separate" model for each task, in the sense that the design matrix is sparse after transformation, and contains non-zero blocks for each task. This mechanism has been shown in the simple example in Section 3.2.

To investigate this question, I model the data with ϵ-SVR with linear kernel in LIBSVM through R package e1071. The same fixed 10 train-test splits from the previous sections are used to make the results consistent, and explained variance performance is averaged over the splits to increase the level of confidence.

The optimal cost is searched for on a logarithmic scale in each case, narrowing it down to the values that yield the highest explained variance. This way, these results are comparable to Experiment 1 of simulations with mean-regularized MTL. Because the edge cases $\lambda = 0$ and $\lambda = 1$ correspond to single-task and independent task learning, respectively, the experiments considered in this section theoretically mirror the results in these edge cases.

First, consider the case of $\lambda = 0$ which corresponds to single-task learning. In Experiment 1 of the previous section, for $\lambda = 0$, the explained variance is approximately at the level of 34% and is not sensitive to the SVR parameter C, as long as $C > 0.5$. In the explicit single-task learning experiment the average explained variance is approx. 34%, and is not significantly different from the results when using the mean-regularized multi-task kernel. This result confirms the theoretical connection between single-task learning and mean-regularized MTL with $\lambda = 0$, such that the method is learning a mutual parameter vector for all tasks and not the individual parameters. Notably, the cases of $\lambda = 0$ and explicit single task learning differ in their sensitivity to changes in cost parameter C. For explicit single-task learning, the explained variance is high as long as the $C > 0.1$ (approx.) and it is not sensitive to increases of C for bigger values. In the case of the mean-regularized multi-task kernel, high values of explained variance and its insensitivity is achieved only for $C > 0.5$.

In the case of $\lambda = 1$, the results of Experiment 1 have shown that the performance of the mean-regularized MTL is not significantly reduced even though the tasks are independent. Note that in Evgeniou et al (2005) [30] the case of $\lambda = 0$ achieves explained average explained variance of approx. 5%, with the choice of C to satisfy $\gamma = 1$.

The results of the experiment for the independent task learning case are summarized in Table 3.5 for a range of costs, with the optimal value of C at approx. 2.3 with the explained variance of 12.62%.

The result of 12.6% explained variance is quite low by itself, however, it is to be expected as we've already seen that the ILEA dataset is best explained as being close to a single-task. The explained variance at independent task learning was increased as C increased, although not optimally in the global sense. And in the example at hand, we search for C only for the explicit case of independent task learning.

C	**Explained variance**
1.000000	10.55645
1.444444	12.01658
1.888889	12.52255
2.333333	12.62515
2.777778	12.59228
3.222222	12.56416
3.666667	12.47304
4.111111	12.24841
4.555556	12.09087
5.000000	11.89426

Table 3.1: Average explained variance by C for the explicit case of independent learning.

Interestingly, the explained variance is lower than the maximum that was achieved with the mean-regularized multi-task kernel in Experiment 1.

The results of this section are in accordance with results in Evgeniou (2005) [30] and are slightly improved because of the search for optimal C, so it is sensible that a better result is achieved in this section. However, we observe a difference between performance in the explicit independent task learning case and the case of $\lambda = 1$ in Experiment 1 in Section 3.2. There might be several possible reasons for this. But, as the data splits of the ILEA schools data were kept constant across all the experiments and simulations, theoretically it was expected that all the results would be consistent with each other. A possible reason for this disparity may be the fact that the support vector machine algorithm in the LIBSVM package is affected by how the data is presented and is not behaving consistently to replicate the results in this setup. The optimization with the whole data together, even though all tasks are in separate block matrices, may lead to unstable behavior of the package and not properly separate between the tasks, which leads to the high performance at $\lambda = 1$ in Experiment 2. Moreover, as all tasks have different sample sizes, the classifier may implicitly prioritize some tasks over others, which may further distort the results.

Chapter 4

Two-step modeling approach to multi-task learning

4.1 Introduction

In this chapter, I propose a two-step modeling approach to multi-task learning, which is highly customizable. Furthermore, I examine its performance on the ILEA schools data for several framework customizations.

The main idea is to split the tasks into two or more clusters depending on their per-task performance and modify the predictions for particularly bad clusters. For example, we may identify a cluster of tasks yields explained variance that is negative when predicted and measured individually on these tasks' separate training and test sets, respectively. From the definition of explained variance in 2.8, it is obvious that in such cases predicting using an average value of the test set for each particular task is going to increase the explained variance to 0% in such tasks. And for the tasks with positive explained variance, we may keep the initial model. Modeling data differently for different tasks would increase the overall performance of the model.

To further generalize this way of thinking, I propose a two-step model, with an initial diagnostic modeling step and, after the proper modifications, creation of the final model. The diagnostic remedial step is highly customizable in several of its steps. As a result, the overall procedure can range from rather simple to complex. Therefore, the general strategy is to arrive at a procedure that gives the best performance and is the most suitable depending on a particular scenario or dataset.

Notationally, let $\mathbf{Y}^{[i]}$ and $\mathbf{X}^{[i]}$ be data for task i only, $i \in \{1, 2, ..., m\}$, where m is the number of tasks. Let K be the number of clusters identified in the diagnostics procedures. In this dissertation the only considered cases are $1 \leq K \leq m$, i.e. the maximum number of clusters is bounded above by the total number of tasks. Let $\mathbf{Y}^{\{k\}}$ and $\mathbf{X}^{\{k\}}$ be the data of tasks that belong to cluster k.

In the following, we will go through the steps of proposed modifications for the multi-task learning algorithm. For the purposes of explanation, the initial starting point is the mean-regularized MTL kernel with λ used with support vector regression described in Section 2.5. The examples will be explained in terms of the previously covered ILEA schools data.

4.1.1 Algorithm of two-step modeling for multi-task learning

In the following, I start by presenting the algorithm of a two-step model for multi-task learning. The brief summary of the algorithm is to identify the clusters of tasks and model them differently. As the initial assumptions of this framework are rather general, the Algorithm 1 is defined in loose terms to enable this.

Algorithm 1: two-step model for multi-task learning

```
Input: performance metric function, clustering criteria

Step 1
Train initial diagnostics models on $\mathbf{Y}_{train}$ and $\mathbf{X}_{train}$;
for each task $i$ do
    Select $\mathbf{Y}^{[i]}_{test}$ and $\mathbf{X}^{[i]}_{test}$, data of only task $i$;
    Predict on $\mathbf{X}^{[i]}_{test}$ with the initial diagnostics models;
    Measure the performance metric based on actual responses $\mathbf{Y}^{[i]}_{test}$ and
     predictions $\hat{\mathbf{Y}}^{[i]}_{test}$
Rank the per-task metrics;
Cluster the tasks into $K$ clusters per criteria;

Step 2
for each cluster $k \in \{1, ..., K\}$ do
    Based on Step 1, determine the best way to model the tasks in the
     cluster;
    Apply this model and obtain $\hat{\mathbf{Y}}^{\{k\}}_{test}$;
Combine $\hat{\mathbf{Y}}^{\{k\}}_{test}$ for all $k \in \{1, ..., K\}$ to arrive at the final predictions;
```

In the follow-up sections, I motivate the procedure step-by-step with explanations and reasoning for its use.

4.1.2 Task clusters

The first step is to define the criteria for a split of tasks into clusters. For diagnostic model fits, the performance is measured on each task's $\mathbf{Y}^{[i]}$ and $\mathbf{X}^{[i]}$ individually, such that the models are used for prediction separately on $\mathbf{X}^{[i]}_{test}$ for each task. Thus, for each fit, a performance metric is computed, such as explained variance EV, and m sets of statistics are obtained, which allows ranking of the tasks according to the performance of the diagnostic models.

The clustering of tasks is made on the basis of models that are fitted, and the criteria used to cluster the tasks. For example, the cases independent task learning and single-task learning models (equivalent to $\lambda = 1$ and 0, respectively). Or, the model is fitted for multiple values of λ between 0 and 1. For each model, we test its performance for each of the tasks on their respective test sets and record the explained variance EV obtained in each case.

The further customizable steps consider a number of clusters, and criteria to split tasks. There may be two or more clusters, and every task must fall in only one cluster. The primary purpose to split tasks into clusters is to later modify the predictions in some clusters in order to increase the explained variance when the model is tested on a combined test set for all tasks. Therefore, one should plan in advance how to modify the predictions in every cluster and split the tasks accordingly. More clusters may complicate the procedure but can result in a more customized, better performing overall model.

As an example, we can create initial model fits of only independent task learning and single-task learning and split tasks into two clusters. Cluster one has tasks that have positive explained variance in both models, while the second cluster tasks have negative explained variance either in independent task learning model, single-task learning model, or both. Another splitting criteria may be to split tasks according to the improvement or worsening of tasks' explained variance when comparing individual and single-task learning performances. And it is possible to combine several criteria to split the tasks into the clusters. A simple starting point to create such criteria is to rank the tasks according to the per-task explained variances.

In this step, the customizability comes in the number of clusters and splitting criteria that are used, and in how many models fits one wants to consider.

4.1.3 Diagnostic initial parameters

We need to take special care of the initial model parameters for the diagnostic procedure above. For every diagnostic model, cross-validation needs to be performed in order to find the optimal parameters.

For example, in support vector regression with linear kernel, the only parameter of interest is cost C. Initially, the optimal values of C are not known, and can possibly differ across different values of the task coupling parameter λ. Ideally, one would want to use the optimal model parameters for every λ used to cluster the tasks. However, the process of finding optimal parameters with a grid search for each λ is time-consuming. Moreover, the optimal parameters for the final model will differ from the optimal parameters in the diagnostic models. Therefore, it is reasonable to devise simplified search procedures for optimal model parameters in the diagnostic models.

In the above example of two clusters and only single-task and independent task learning diagnostic models, we would intially seek to find the optimal C values of the SVR models. As a simplified procedure, single-task learning ($\lambda = 0$) can be used for a search for optimal C on a test set consisting of data from all tasks. Then, this value of C can be used in both single-task and independent task learning diagnostic models.

4.1.4 Designing the remedial measures

At this point of the modeling process, we are working with K clusters of tasks, each cluster k is defined by certain predetermined criteria. The clusters may generally be divided into good and bad ones per the criteria used and explained variance in the cluster. Then, various remedial measures can be performed in each cluster, except perhaps the clusters where the performance is good as is.

As an example of remedial measures, in clusters of tasks with negative explained variances, for each task, we can replace the predicted values with the mean response values in the respective test sets. If a cluster already has positive explained variances in each task, then improvement may be achieved by replacing the initial model, say SVR, with another model, for example ridge regression, LASSO, or random forest. In such cases, special consideration needs to be done to the way the optimal parameters are found for these replacement models, and successful cross-validation will likely require using the whole training set.

Separate consideration is required for the way the values in each task are replaced. Conventionally, all response values for a task are replaced at the same time. It can be reasonable to consider a sequential replacement, such that components $\hat{\mathbf{Y}}^{[i]}_{test}$ get replaced one by one until the point where the performance is maximized, as measured by explained variance EV.

4.1.5 Finalizing the model

In the final part of the overall modeling procedures, step 2 of the Algorithm 1 combines the previous results to arrive at the final model. Therefore, all the procedures together form a two-step multi-task model.

Each cluster is considered differently. The observations in good clusters that were decided to be untouched during the diagnostics procedures, can be predicted with a model that is fit on the training set of all tasks, as was done during the diagnostics steps (except in cases of the models based on individual task learning). Then, the observations in other clusters are replaced per the steps that were discovered during their diagnostics.

In order to find the optimal model for each cluster, we do cross-validation of the models inside their own clusters. Note that the final model requires different parameters compared to diagnostic models. The cluster k training data is likely to be distributionally different from the overall data $\mathbf{X}_{train}$ and $\mathbf{Y}_{train}$. Thus the model parameters that were found to be optimal on the overall data, may not be a good choice for the clusters.

4.2 A simple example of the two-step model

In the next section, I demonstrate a simple example of a two-step multi-task model. For simplicity, assume that $\lambda \in \{0, 1\}$. Because these are the edge cases, there is no need to apply the mean-regularized MTL kernel, although the results would not change much if it was. The baseline model will be SVR.

The tasks are split into two clusters so that $K = 2$. The good cluster ($\mathbf{Y}^{[1]}$, $\mathbf{X}^{[1]}$) consists of the tasks that have positive explained variance on their own test sets for both independent and single-task learning. The other tasks will be put into a bad cluster ($\mathbf{Y}^{[2]}$, $\mathbf{X}^{[2]}$), which contains tasks that have negative explained variance on their test sets either on independent task learning model, single-task learning model, or both.

The categorization of the tasks to the clusters will be done using the SVR model. From the previous experience, when comparing $\lambda = 0$ and 1, the single-task learning yields good performance on the ILEA data, and the optimal $C \approx 2.3$ for $\lambda = 0$. To simplify the procedure, this is the parameter of the SVR model that will be used for all diagnostic fits. Note that if we didn't have experience analyzing this data, we would have to come up with another good approximation of C.

In order to improve the performance in the bad cluster, the predictions of the

SVR model for each of the tasks are to be replaced with the average score value of its test set. Alternatively, the response variable means could be replaced by regression the predicted values to the actual values in the test set, thus obtaining Best Linear Predictor based on the predicted values, rather than on the data itself. This process is called linearizing the predictor, and their property is that they cannot have negative R^2, which means that they cannot have negative EV by proxy. More details are given in Christensen (2020) [23].

The second step of the model will combine the fitted values of good and bad clusters together. In the good cluster, the fitted values will be based on the SVR model trained on all tasks, as was done during the diagnostics step. In the bad cluster, we replace the predicted values with averages within each task ($\bar{y}_l$) as described above. Then, we measure an explained variance EV on the combined set of predicted scores. Thus, we obtain a single-number measure of the performance of the two-step model. We do this step only for $\lambda = 0$ as we already know that this is better than $\lambda = 1$. We will compare the explained variance of our two-step model with the unmodified single-task learning SVR with $C = 2.3$.

Note that $C \approx 2.3$ is the best when all tasks data is in the validation set, so it can be beneficial to search for optimal C after the clustering step because we could exclude the bad cluster from the validation set and focus on finding optimal C for the good cluster only.

There are 40 tasks in the bad cluster and 99 tasks in the good cluster. Validating the model on the good cluster, the optimal $C \approx 2.9$. The baseline model achieves 33.67% explained variance, while the proposed model achieves 35.5%. Thus there is a substantial increase in the explained variance in the two-step multi-task model.

Because the framework of the two-step multi-task approach is flexible, its high customizability allows the construction of strongly performing models. This example highlights the benefits of the proposed approach in a simplified case.

4.3 Methods development

The purpose of this section is to demonstrate and evaluate multiple applications of the two-step multi-task modeling approach. It allows the creation of multitude of different model configurations, depending on the needs of a practitioner. The first step of the modification considers the task clustering criteria, and the second step considers the choice of models for the clusters and the final performance evaluation. The procedure is encapsulated in Algorithm 1.

For the ILEA schools data, the linear kernel multi-task learning model in Section 2.5 will be regarded as starting point for comparison with all other models. It achieves a test set explained variance of slightly below 34% at $\lambda = 0$ averaged over 10 random train-test splits. I repeat the experiment for 10 random train-test splits and achieve 33.72% explained variance on the test set, using SVR with $C = 1.96$ which was found using the average value of 10-fold cross-validations on the training set over all 10 random train-test splits.

In the following, multiple approaches are considered to increase the test set explained variance for this dataset. As the definition of a two-step multi-task modeling approach is quite general, these special case applications can be seen as separate frameworks on their own. Therefore the approaches are designed in such a way as to be generalizable to other datasets.

These approaches are first and foremost statistical in their nature. In this context, it means that the clustering will be based on the training sets only, and use the test set only to evaluate the final model performance. It should be noted that Evgeniou et al (2005) [30] used the test set to select the most optimal value of the multi-task coupling parameter λ, and not only to measure the final model performance. In other words, the test set was also used to find the most important model parameter. Therefore, also in my approaches, it might sometimes be necessary to use the test set to cross-validate the key parameter.

4.3.1 Explained variance rebalancing approach

In the first approach, I define a per-task *explained variance rebalancing* procedure, which will be used to cluster the tasks. It is defined as the weighted average of explained variance of a model trained on an individual task and then fitted on its own training set and on all other tasks' training sets. $\alpha \times 100\%$ of weight is placed on the explained variance measured on the task's own training set, and $(1-\alpha) \times 100\%$ of weight is placed on the training set performance for average explained variance as fitted on all other tasks' training sets. The parameter α can be decided a priori or can be found through cross-validation. This can be seen as an extension of explained variance EV, which is based on data.

As a starting point, I choose $\alpha = 0.5$, which induces an equal trade-off to performance for the task itself and for all other tasks. The steps in the approach follow the general two-step modeling framework, with a diagnostic phase and remedial measures phase:

1. For each task's training set, perform a 5-fold cross-validation of SVR on its own training set to find the optimal cost parameter in the SVR models. For each task, repeat this process for each of the train-test splits, and average optimal costs across the splits for each task.

2. Train individual task SVR for each task separately using their optimal costs.

3. Fit each model to its own training set and all other tasks' training sets, obtaining a matrix of explained variances. Perform it for all train-test splits, and average the results.

4. Split the tasks into two clusters according to the weighted average of explained variance performance on its own training set and all other tasks' training sets. The tasks with above-median weighted explained variance go to a good cluster, and all other tasks go to the bad cluster.

5. For both clusters, we modify the fitted values according to the table below and measure the explained variance performance of the final model on the test set.

Good cluster model	Bad cluster model	Explained variance
Random Forest single task	SVR single task ($C = 1.96$)	33.6%
SVR single task ($C = 1.96$)	Random Forest single task	34.7%

Table 4.1: Final test set explained variance per the combination of models in both clusters in the explained variance rebalancing approach.

The results of the application of the explained variance rebalancing approach are stated in Table 4.1. Note that the performance has increased compared to using single-task SVR, which achieves explained variance of 33.67% with the optimal cost of $C = 1.96$. This is also the cost that we use in our modified procedure as a simplification.

It appears to be more advantageous to use SVR in the good cluster, and random forest in the bad cluster, likely because SVR was actually used as a model for initially clustering the tasks. Another possible explanation is that the SVR algorithm may be better suited to generalize knowledge between the tasks since the good cluster contains tasks that have great performance on themselves and on other tasks. On the contrary, the bad cluster tasks aren't useful for explaining other tasks, and so here the random forest is able to more accurately capture the variation in the data due to its tree structure, and make predictions more precise.

4.3.2 Best subsets information extraction

In the following approach, I attempt to extract additional information about the tasks to measure their similarity by using the best subsets methods of multiple regression models. The motivation is that similar tasks will have similar variables that are important. The ILEA schools data has 26 predictor variables, and for each task, I will find the optimal choice of predictor variables by using the best subsets algorithm with a multiple regression model. As such, I am only interested in the multiple regression and its best subsets variable selection method in order to extract the information about tasks' similarity. As I won't be fitting the multiple regression to the task clusters, the information about which variables are important for each task will give us information about the way the tasks are related. Moreover, I won't be reducing the variable space when fitting the model, and we are only doing variable selection in order to cluster the tasks. When fitting the final models, all variables will be used for each cluster.

For each task, we find the variables that are the best according to the best subsets algorithm. The metrics to be optimized are BIC and Mallows's Cp, thus we will perform two separate analyses. We define the clusters to include the tasks where the same variables are important according to the variable selection algorithm.

The ILEA schools data has $m = 139$. When splitting the tasks according to the best subsets that minimize BIC, the number of clusters K is between 79 and 87, depending on the train-test split. When minimizing Mallow's Cp, the number of clusters K increases to be in the range between 96 and 107. Most of the clusters have only one task, while few clusters group multiple tasks, with the biggest cluster containing 21 tasks in two of the random train-test splits.

After the tasks have been separated into clusters, we fit a separate model for each cluster. Using support vector regression, we apply 5-fold cross-validation to find an optimal C for that cluster. Then we predict the test set for each cluster, gather the results, and measure the explained variance for the whole test set. We repeat the experiment for the 10 random train-test splits and average the explained variance across the splits to arrive at the final performance. The results of the experiments are illustrated in the Table 4.2.

Best subsets criteria	Cluster models	Explained variance
BIC	SVR (optimal C)	33.6%
Mallow's Cp	SVR (optimal C)	33.3%

Table 4.2: Final test set explained variance per best subsets criteria and models in the best subsets information extraction approach.

The best performance is achieved when clustering according to variables selected with BIC. The benchmark 33.7% of the single-task learning model is only slightly higher than the performance of the proposed method. The number of clusters K is very high for both of the clustering criteria, which means that there are many individual models that are being trained and cross-validated, making this procedure somewhat similar to individual task learning. The primary benefit is the individual cross-validation in each cluster. However, the loss of information from reducing the data for each model has seemingly counteracted the benefit of the clustering. It is an open research question to search for other best subsets criteria and cluster methods to increase the performance on ILEA data.

4.3.3 Other tasks performance approach

In this approach, we consider the second component of the previously discussed explained variance rebalancing approach. It is a sum of two terms, where the first term is the performance of the task predicted on itself, and the second term is the average performance on predicting all other tasks. We focus more closely on the second term, the average performance on all other tasks.

Some tasks will have relatively high performance on other tasks, while some tasks will have very low, or even negative average performance on other tasks. For the former, it means that those tasks are well suited to predict not only themselves, but also other tasks, while for the latter, it means that tasks aren't suitable to predict other tasks, and are only suited to predict themselves.

We define ξ to be the level of average explained variance on other tasks above which we consider a task to be suitable to predict other tasks. If a task has an average explained variance on other tasks which is lower than ξ, then it is not considered suitable for other tasks' prediction. For example, if $\xi = 10\%$, tasks that achieve at least 10% average explained variance when predicting other tasks, then it will be modeled in a single-task learning model, together with other such tasks. And for tasks that are below ξ, we only do independent task learning and predict on this particular task only.

We perform experiments separately with support vector regression and random forest models and record the results. For SVR, each model uses optimal cost parameters, found using 5-fold cross-validation on the training set. For random forest, we use 1000 trees for the single-task learning model, while the independent task learning model uses default parameters. The experiments are repeated across 10 random train-test splits, and the results are averaged out. We use cross-validation to find the level ξ which yields the highest overall explained variance on the test set. The

ξ	**SVR expl.var.**	**RF expl.var.**
-50	34.62	31.99
-40	34.64	31.50
-30	34.74	18.90
-20	34.54	16.84
-10	34.32	11.64

Table 4.3: Final test set explained variance per level ξ for the other tasks performance approach.

results are illustrated in the Table 4.3.

Positive values of ξ didn't yield good results, and it is interesting to note that as ξ gets lower, the performance of SVR improved gradually. At 34.74%, it yields the highest explained variance in this chapter so far, and 1% higher than the baseline performance. The reason that so negative level ξ is relevant, can be that there are some tasks that are extreme outliers, and separating those tasks yields in greater performance improvements. However, it doesn't seem to be the case that the provided approach is useful in random forest models, where the performance flattens out at high 18% as ξ gradually decreases. We also observe that as ξ decreases further, the random forest gradually increases its performance, being slightly above 30% when $\xi = -40$ and lower. With such a low value, most tasks will be considered to be one cluster, and as ξ decreases further, it leads to single-task learning for all tasks.

4.3.4 Best performance subsets combinations

In this approach, we cluster tasks into subsets according to their best performance on every task. For each task, we find the tasks which predict it with the highest explained variance. Usually, it will be a model based on this task itself, but also other tasks can provide models which give high performance. We use distance τ away from the maximum to capture those tasks. Then, we train one learning model on those best tasks in the cluster, and predict only the task considered.

For example, let $\tau = 10\%$, and that for task 2, we find that the maximum explained variance was achieved by its own model, with 35% explained variance. We also know that task 1 yields 27% explained variance when predicting task 2 training data, task 3 - 22%, and task 4 - 31%. Tasks 1 and 3 are within $\tau = 10\%$ distance below the maximum performance of 35%, while task 2 is not. Thus, we consider models based on tasks 1, 2, and 3 to be a cluster of tasks that predicts task 2 very well, we train a single-task model on a combined dataset of those tasks and predict on the test set for task 2. This will be the final performance measure on task 2. This process proceeds for all tasks in the dataset so that every task gets its own cluster of tasks that perform best for this task, and a separate single-task learning model is trained in each cluster to predict that task.

The procedure is summarized in Algorithm 2.

Algorithm 2: Best performance subsets combinations

```
input  : 10 random train-test splits, parameter $\tau$
output: Measurements of explained variance on test set for each split
for each of 10 train-test splits do
    for each task $i$ do
        select training data for task $i$;
        if necessary, perform 5-fold cross-validation to cross-validate the
          parameters of the model;
        record the optimal parameters for this split and task;
average the optimal costs across the splits for each task;
for each of 10 train-test splits do
    for each task $i$ do
        train the model on training data for task $i$ with the optimal
          parameters for that task;
        for each task $j$ do
            predict responses in training data for task $j$ using the model
              trained on task $i$ with its optimal cost;
            measure and record the explained variance;
average the explained variances across the splits for each combination of
  tasks $i$ and $j$ (as defined in the loop above)
for each task $i$ do
    select all performances of explained variance that were achieved when
      predicting on task $i$;
    sort the explained variance values from largest to lowest;
    find the task with maximum explained variance and use cut-off distance
      $\tau$ to select the tasks which have maximum $\tau$ lower explained variance
      than the maximum performance task;
    save these tasks as the cluster for task $i$;
for each of 10 train-test splits do
    for each cluster $i$ do
        select training data from tasks which belong to the cluster for task $i$,
          and merge this data together;
        if necessary, perform 5-fold cross-validation to find optimal
          parameters of the model;
        train the model on training data for this cluster with the optimal
          parameters;
        use this model to predict test set data for task $i$;
        save the predicted response values;
    combine the predicted response values for all tasks;
    measure and record the explained variance by comparing the predicted
      response values with the actual test set response values;
```

In the Algorithm 2, after the first two for-loops, we obtain explained variance matrix. Each element (i, j) contains explained variance achieved when training a model on task i and using it to predict on task j. This is illustrated in Figure 4.1 for linear SVR and random forest models. Note that there is a slight diagonal line, where tasks are trained and predicted on themselves. The universal pattern is that some tasks are good at predicting other tasks, while other tasks have quite low predictive power. This matrix is then used to cluster the tasks, based on the cut-off parameter τ. Its optimal value can be found using a grid search.

τ	**Linear SVR expl.var.**	**RF expl.var.**
10	37.55	33.60
15	38.78	33.75
20	37.58	33.60

Table 4.4: Final test set explained variance per level τ for the best performance subsets aproach.

The performance of the experiment for a range of values of τ is illustrated in Table 4.4. Note that in the case of SVR diagnostics, 10 out of 139 tasks have maximum explained variance by a model trained on a different task than its own, while when using random forest, there are no such tasks.

These results are the strongest performance that is achieved among all customizations of the two-step multi-task procedure in this chapter. The performance of this procedure can be potentially even stronger when considering other models than support vector regression. However, one downside is that this procedure is computationally expensive, and depends a lot on the size of the data. A grid search for parameter τ necessitates redoing the whole procedure over again for each value of τ. An efficient way to work with the best performance subsets combinations approach is to use parallel computing, as was done for evaluating Table 4.4. The Algorithm 2 is suitable for parallelization for each value of τ in the grid search, and for each train-test split.

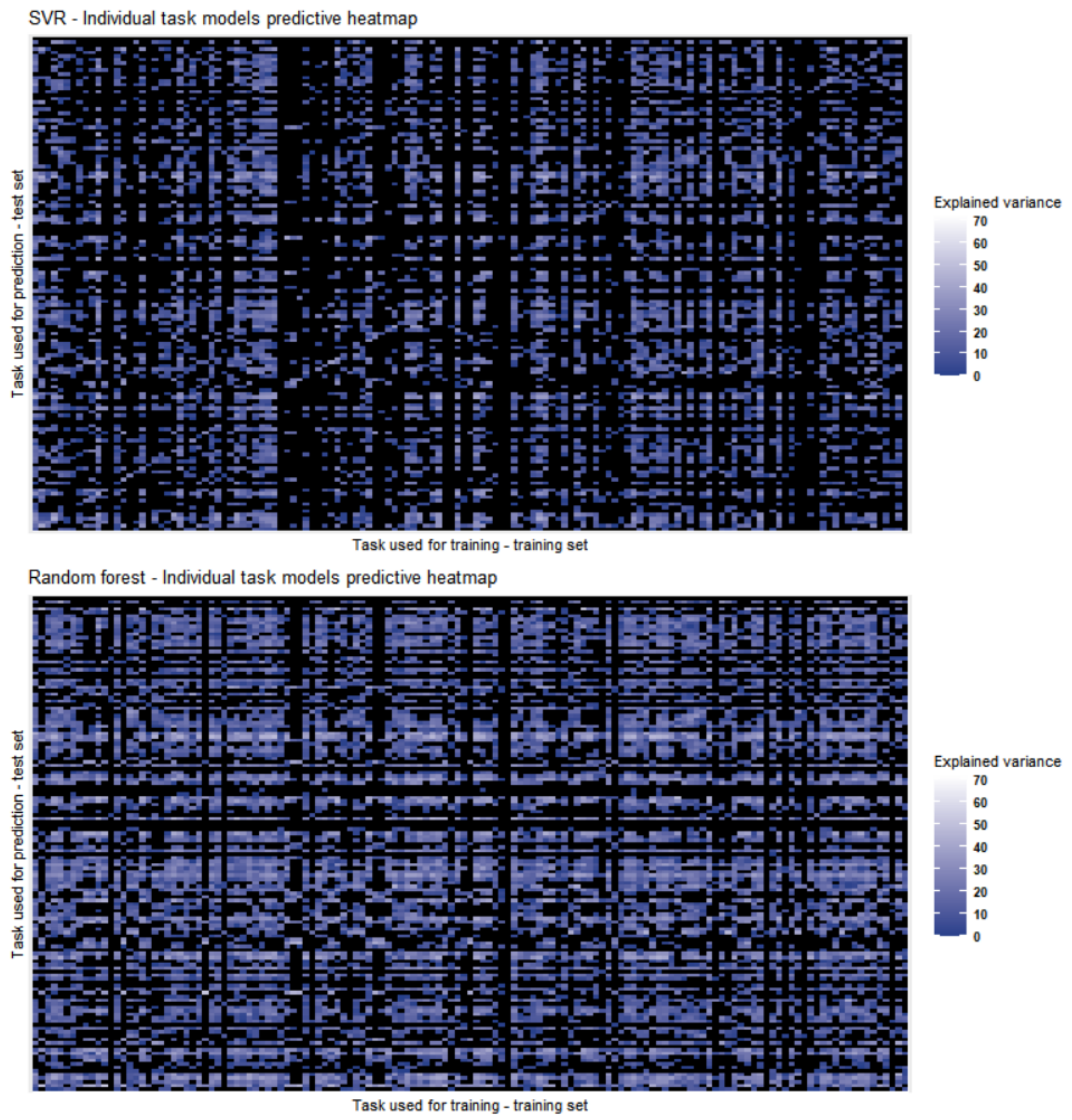

Figure 4.1: Explained variance matrix for SVR and random forest. Columns are tasks used for training, and rows are tasks used for prediction on. Cells are blacked out when explained variance is negative.

Chapter 5

Additive multi-task model

5.1 Overview

In this section, I introduce the extension of the generalized additive modeling approach to the multi-task learning framework. In the context of machine learning, and specifically for multi-task learning, it can be of great benefit to explicitly separate the linear and nonlinear effects in the model. We might know or reasonably believe that some part of the model can be better expressed as a parametric model, while the other part is better expressed nonparametrically. This setup can make the model more flexible and capable, and in this chapter I introduce a new approach to multi-task learning modeling that addresses these concerns.

The main inspiration for this model is the well-established framework of additive models. For a brief review of the theory of additive models (Hastie, Tibshirani, and Friedman (2001) [38]), consider a regression problem as:

$$y_i = \alpha + \sum_{j=1}^{p} f_j(x_{i,j}) + \epsilon \tag{5.1}$$

with intercept α, $\sum_{i=1}^{n} f_j(x_{i,j}) = 0, \forall j$, and ϵ being an error term with mean 0. The functions $f_1, f_2, ..., f_p$ are additive in their effect on the response variable y, and they can be either parametric for linear relationships or nonparametric for nonlinear relationships. The functions f_j can be fitted with nonparametric methods such as additive cubic smoothing splines, local linear estimators, and polynomial estimators. These functions can be found using the backfitting algorithm, which is an iterative procedure that estimates each of the functions sequentially until their convergence.

In this chapter, I introduce an extension of additive models which applies to multi-task learning. I define such models as additive multi-task learning models. The primary benefit is the possibility to combine parametric and nonparametric effects in an additive way. Therefore, it is a semiparametric model. As a special case, if the parametric part of the model is linear, it becomes a partially linear model.

An additive multi-task model can be customized to fit a particular problem, which makes this approach incredibly useful in the multi-task learning framework. Casting the multi-task model in an additive way allows great flexibility in a range of statistical models that can be applied to a particular problem, and allows the researcher to explicitly specify the parts of the model. From a multi-task learning perspective, it allows clear separation of various task effects and can lead to a great increase in performance with a suitable design.

In the following, I start by generalizing explained variance to cover new important cases and adapting combined estimation to a multi-task framework, additionally proving its upper bound of performance improvement. then, I introduce multi-task combined estimation and investigate its properties. I proceed to introduce and define the additive multi-task model and continue with a further generalization of the fitting algorithm. This is followed by an application of the additive multi-task model to the real data. The chapter is concluded with an illustrative comparison of model components.

5.2 Generalization of explained variance

Recall the definition of explained variance by Bakker et al (2003) [9] in Section 2.4 of this dissertation, which is given by the Equation 2.8:

$$EV(\mathbf{Y}_{test}, \hat{\mathbf{Y}}_{test}) = \frac{\mathrm{Var}(\mathbf{Y}_{test}) - MSE(\mathbf{Y}_{test}, \hat{\mathbf{Y}}_{test})}{\mathrm{Var}(\mathbf{Y}_{test})} \times 100\%$$

In this conventional definition, the statistical model is trained on the training set, and its performance is measured on the test set. The universal interpretation and application of this definition ensure that the results of various models are measured in a consistent manner in the machine learning field.

In practice, there arise situations where it is advantageous to measure explained variance in different ways from the traditional definition. For example, it can be useful in applications where explained variance is used for cross-validation on the training set, or when there is no splitting between train and test sets. The conventional definition of explained variance can't be used in these situations as it would cause inconsistencies in the interpretation of results.

5.2.1 Extensions of the definition

In the following, I extend the explained variance definition to cover the two cases mentioned in the previous section.

Definition 1. *The fraction of variation in the training set that is explained by a model trained on this training set is measured by the general explained variance as:*

$$GEV(\mathbf{Y}_{train}, \hat{\mathbf{Y}}_{train}) = \frac{Var(\mathbf{Y}_{train}) - MSE(\mathbf{Y}_{train}, \hat{\mathbf{Y}}_{train})}{Var(\mathbf{Y}_{train})} \times 100\% \tag{5.2}$$

In the existence of train-test splits, general explained variance GEV measures the variability explained in the training set only. This is helpful in model selection procedures, or in preliminary data analysis for early investigations of model performance.

Definition 2. *The fraction of total variation in the dataset that is explained by a model trained on this dataset is measured by the total explained variance as:*

$$TEV(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{Var(\mathbf{Y}) - MSE(\mathbf{Y}, \hat{\mathbf{Y}})}{Var(\mathbf{Y})} \times 100\% \tag{5.3}$$

The purpose of the total explained variance TEV metric is to cover those cases where no train-test splitting is done. Thus, the model is trained and predicted on the same data, and there are no other sets to consider. This metric is designed to be used in statistical procedures, where there are often no train-test splits, and the whole data is used for analysis.

The differences in the definitions of EV, GEV, and TEV are about which data is used for the model training and which variation is measured. Along with explained variance, both general explained variance and total explained variance will be used to augment the research in this dissertation.

GEV and TEV are measured when the model is trained and predicted on the same data, while EV is measured when the model is trained on the training data, and used for prediction on the test set. Therefore, GEV and TEV are measures of goodness of fit, while EV is a measure of prediction quality.

To understand the total explained variance deeper, consider the definition of the coefficient of multiple determination R^2, given in Kutner, Nachtsheim, Neter and Li (2005) [51]. Let $SSTO = \sum_{i=1}^{n}(y_i - \bar{y})^2$, $SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ and $SSR = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$. Then:

$$R^2 = \frac{SSR}{SSTO} = 1 - \frac{SSE}{SSTO} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} =$$
$$= 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \frac{n}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{MSE(\mathbf{Y}, \hat{\mathbf{Y}})}{\text{Var}(\mathbf{Y})} = TEV(\mathbf{Y}, \hat{\mathbf{Y}})$$

Thus it becomes evident that reformulating explained variance EV for the whole dataset to become total explained variance TEV makes it equivalent to the coefficient of multiple determination R^2, which also has its maximum value at 100%, denotes equal predictive power of the model and response mean at 0% and indicates performance worse than response mean when it is negative.

The statistical meaning of R^2 is that it measures the fraction of variation in the response variable that is explained by its relationship with the predictor variables. R^2 is often used in applications with linear models. In its nonlinear applications, $SSTO$ doesn't necessarily decompose to a sum of SSR and SSE, and there is a debate as to whether R^2 can be used for nonlinear relationships (for example, see Spiess and Neumeyer (2010) [71]). R^2 also assumes that the error terms $\boldsymbol{\epsilon}$ have Normal distribution. The implication is that all statistical properties and interpretation of R^2 transfer directly to TEV in cases of a linear relationship. When the model is nonlinear, we cannot say for certain whether the statistical properties transfer from R^2 to TEV, even though the equality holds regardless.

5.2.2 Decomposition of total explained variance

In this section, I show the decomposition of total explained variance into its individual components. Recall that $n = n_{train} + n_{test}$. Moreover, recall from the definition of explained variance in Section 2.4, that in the formula of EV, $\text{Var}(\mathbf{Y}_{test}) = \frac{\sum_{i=1}^{n_{test}}(y_i - \bar{y})^2}{n_{test}}$, which implies that for TEV, $\text{Var}(\mathbf{Y}) = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n}$.

Theorem 5.2.1. *Total explained variance $TEV(\mathbf{Y}, \hat{\mathbf{Y}})$ depends only on the mean, standard deviation, and MSE measured on training and test sets.*

Proof. First, consider the mean-squared error MSE. It follows from its definition that

$$MSE(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} = \frac{\sum_{i=1}^{n_{train}}(y_i - \hat{y}_i)^2 + \sum_{i=n_{train}+1}^{n}(y_i - \hat{y}_i)^2}{n} =$$
$$= \frac{n_{train}MSE(\mathbf{Y}_{train}, \hat{\mathbf{Y}}_{train}) + n_{test}MSE(\mathbf{Y}_{test}, \hat{\mathbf{Y}}_{test})}{n} \tag{5.4}$$

Thus, MSE of the whole data can be expressed as a weighted average of two groups' individual MSE. The weights sum up to 1, since $n_{train} + n_{test} = n$.

If two separate datasets are combined into one dataset, then, based on the law of total variance, the variance of the combined dataset can be decomposed into components that depend only on the two original datasets (Higgins et al (2020) [40]).This well-known property allows for decomposition of Var($\mathbf{Y}$), since $\mathbf{Y} = \mathbf{Y}_{train} \cup \mathbf{Y}_{test}$. Moreover, the overall mean of $\mathbf{Y}$ depends only on the means of training and test sets. We have that

$$\bar{\mathbf{Y}} = \frac{n_{train}\bar{\mathbf{Y}}_{train} + n_{test}\bar{\mathbf{Y}}_{test}}{n} \tag{5.5}$$
$$\text{Var}(\mathbf{Y}) = \frac{n_{train}(\text{Var}(\mathbf{Y}_{train}) + \bar{\mathbf{Y}}^2_{train}) + n_{test}(\text{Var}(\mathbf{Y}_{test}) + \bar{\mathbf{Y}}^2_{test}) - n\bar{\mathbf{Y}}^2}{n} \tag{5.6}$$

The derivation of Equation 5.6 is presented in Appendix B.

It is clear from the Equations 5.4, 5.5 and 5.6 that Var($\mathbf{Y}$) and $MSE(\mathbf{Y}, \hat{\mathbf{Y}})$ depend only on sample sizes, and mean, variance and MSE measured on the training and test sets. As the $TEV(\mathbf{Y}, \hat{\mathbf{Y}})$ depends only on $\mathrm{Var}(\mathbf{Y})$ and $MSE(\mathbf{Y}, \hat{\mathbf{Y}})$, it becomes evident that total explained explained variance depends only on sample sizes, and mean, variance and MSE measured on the training and test sets.

$\square$

5.3 Multi-task combined estimation

In statistical modeling, and specifically in multi-task learning, there often arise situations where we may find it advantageous to unite the predictive power of two or more models together, yielding a better final model. In order to achieve it, combined estimation is often applied. We seek to extend this notion to application in multi-task learning. The benefits of combined estimation in multi-task learning can be significantly higher than in its conventional application, because we can combine the models differently for different tasks.

Let $\hat{y}_l^{[1]}$ be a fitted value for an observation from task l for the model one, and let $\hat{y}_l^{[2]}$ to be a fitted value for the same observation from task l for the model two. Let $\theta_l \in [0, 1]$ be a parameter that controls trade-off of these fitted values for task l. Then, the multi-task combined estimate is defined as:

$$\hat{y}_l^{comb} = \theta_l \hat{y}_l^{[1]} + (1 - \theta_l)\hat{y}_l^{[2]} \tag{5.7}$$

for task l. The parameter θ_l controls the trade-off between model one and two on the final model output $\hat{y}_l^{comb}$ for task l. Choosing any value between 0 and 1 leads to a fitted value that contains partial information from both models for that task. As a special case, choosing $\theta_l = 0$ for task l makes the fitted value to be $\hat{y}_l^{[2]}$ for that task, and vice versa, choosing $\theta_l = 1$ forces the fitted value to be $\hat{y}_l^{[1]}$. Moreover, another important special case is setting a global trade-off parameter for all tasks, such that $\theta_l := \theta$ for all $l \in \{1, ..., m\}$. Thus, we may either have different trade-off parameters in every task, which are estimated locally within each task, or have one global trade-off parameter which is estimated in the whole data.

The optimal θ_l values can be found using cross-validation on the training set for task l. Assume that we have trained two separate models on the training set, and have obtained $\hat{\mathbf{Y}}^{[1]}_{test}$ and $\hat{\mathbf{Y}}^{[2]}_{test}$ from their prediction on the test set. These vectors contain predictions for all tasks. Then, for each task l, we can find the optimal θ_l.

Algorithm 3: per-task combined estimate search

```
foreach task $l \in \{1, 2, ..., m\}$ do
    foreach $\theta_l \in \{0, 0.01, 0.02, ..., 1\}$ do
        Compute $\hat{y}_l^{comb} = \theta_l \hat{y}_l^{[1]} + (1 - \theta_l)\hat{y}_l^{[2]}$;
        Measure and record $EV(y_l, \hat{y}_l^{comb})$ and $\theta_l$;
    Select and record $\theta_l$ that has yielded the highest $EV(y_l, \hat{y}_l^{comb})$;
```

Algorithm 3 searches for the optimal value of θ_l for each task l in the interval between 0 and 1, including the bounds, however, the granularity of this interval is customizable. $\{0, 0.01, 0.02, ..., 1\}$ is a reasonable starting choice, as used in the Algorithm.

The notion that the trade-off parameter can vary between the tasks allows each task to benefit from the best combination of two models. In the multi-task kernel framework, it can be beneficial for the two models to be trained on the same dataset. For example, training different models such as a combination of support vector regression and random forest models. Moreover, in the context of the multi-task kernel, there arise opportunities to combine two models of the same algorithm. For example, one support vector regression model may be trained on single-task learning data, while another support vector regression is trained on a dataset that is transformed with a mean-regularized multi-task kernel with a positive task coupling parameter

λ.

A generalization of combined estimation is possible, where combined estimates are further weighted together to form a new combined estimate. Thus, $\hat{y}_l^{comb}$ can become a combination of three or more models, without a limit. The output would be controlled by multiple θ parameters. If the number of models is Q, then the number of θ parameters is $Q-1$. As Q grows, a full grid of parameter combinations increases, thus it may not be computationally feasible to do a deterministic search, and a searching algorithm might have to be used.

The analysis of combined estimation benefits from a theoretical property that puts an upper bound on its performance improvement, compared to using the best model alone.

Theorem 5.3.1. *Let $\hat{g}_1$ and $\hat{g}_2$ be estimators of g, such that $\hat{g} = \theta\hat{g}_1 + (1-\theta)\hat{g}_2$, with $\theta \in [0,1]$, is a combined estimator of g. Then, the maximum reduction of MSE from using $\hat{g}$ as the estimate of g is at most 50% compared to estimating g with either $\hat{g}_1$ or $\hat{g}_2$ alone.*

Proof. To show this, consider

$$MSE(\hat{g}) = \theta^2 MSE(\hat{g}_1) + (1-\theta)^2 MSE(\hat{g}_2) + 2\theta(1-\theta)E[(\hat{g}_1 - g)(\hat{g}_2 - g)] \tag{5.8}$$

By minimizing the equation (5.8) with respect to θ, we get the optimal weight θ_0 as

$$\begin{aligned}\theta_0 &= \frac{MSE(\hat{g}_2) - E[(\hat{g}_1 - g)(\hat{g}_2 - g)]}{MSE(\hat{g}_1) + MSE(\hat{g}_2) - 2E[(\hat{g}_1 - g)(\hat{g}_2 - g)]} \\ &\approx \frac{MSE(\hat{g}_2)}{MSE(\hat{g}_1) + MSE(\hat{g}_2)},\end{aligned} \tag{5.9}$$

under the assumption that the covariance term is small compared to mean squared error.

Let $K = \dfrac{MSE(\hat{g}_1)}{MSE(\hat{g}_2)}$. Then, $\theta_0 = \dfrac{1}{1+K}$. It follows that the optimal θ depends only on the ratio of the MSE. Furthermore, the $MSE(\hat{g})$ reduces to

$$MSE(\hat{g})|_{\theta=\theta_0} = \theta_0 MSE(\hat{g}_1) = (1-\theta_0)MSE(\hat{g}_2) \tag{5.10}$$

Without loss of generality, assume that $MSE(\hat{g}_1) < MSE(\hat{g}_2)$. Then, $\theta_0 > 0.5$, since $K < 1$.

The percentage of reduction in MSE by combined estimate is

$$\frac{MSE(\hat{g}_1) - MSE(\hat{g})}{MSE(\hat{g}_1)} = 1 - \frac{\theta_0 MSE(\hat{g}_1)}{MSE(\hat{g}_1)} = 1 - \theta_0 < 0.5 \tag{5.11}$$

Thus, the maximum reduction of MSE will be smaller than 50%. $\square$

The main implication of Theorem 5.3.1 is that there is a rather high upper bound of potential performance improvement when using combined estimation.

It can be seen from the definition of explained variance in Equation 2.8 that regardless of the value of $\text{Var}(\mathbf{Y}_{test})$, MSE and explained variance are connected in a one-to-one relationship. Therefore, the implications of Theorem 5.3.1 transfer directly to applications with explained variance metric, which is used extensively in this dissertation.

5.4 Additive multi-task model

Recall that $y_{i,j}$ is an observation j in task i, where $i = 1, 2, ..., m$ and $j = 1, 2, ..., n_i$. Let $\mathbf{y} = [y_{1,1}, y_{1,2}, ..., y_{1,n_1}, y_{2,1}, ..., y_{m,n_m}]'$, $\boldsymbol{\epsilon} = [\epsilon_{1,1}, \epsilon_{1,2}, ..., \epsilon_{1,n_1}, \epsilon_{2,1}, ..., \epsilon_{m,n_m}]'$, $\mathbf{f} = [f_1(\mathbf{z}), f_2(\mathbf{z}), ..., f_m(\mathbf{z})]'$; let I_m be an $m \times m$ identity matrix, $\mathbf{1}_{n_i}$ be the vector of n_i 1s, $\mathbf{Z} = I_m \bigotimes [\mathbf{1}'_{n_1}, \mathbf{1}'_{n_2}, ..., \mathbf{1}'_{n_m}]'$ where $\bigotimes$ is Kronecker product.

I define the additive multi-task model as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{f} + \boldsymbol{\epsilon}, \tag{5.12}$$

where $\mathbf{X}$ is a certain design matrix related to the linear term structure, and error terms $\boldsymbol{\epsilon}$ are uncorrelated with mean zero, and each task has distinct variance, i.e. $\mathrm{Var}(\epsilon_{i,j}) = \sigma_i^2$ for all i. The structure of matrix $\mathbf{Z}$ is inspired by the model matrix in ANOVA with a block design. This matrix is fixed, and is designed to separate the multi-task models. The matrix $\mathbf{X}$ is a design matrix for the linear term, and vectors $\mathbf{z}$ denote datapoints. These two matrices can be either the same, or different, depending on the model design. The $\mathbf{X}\boldsymbol{\beta}$ part is parametric, while the individual functions $f_i(\mathbf{z})$ are nonparametric in the term $\mathbf{Z}\mathbf{f}$.

Using a backfitting procedure (Hastie et al (2001) [38]) as a motivation, I propose Algorithm 4 to estimate $y_{i,j}$:

Algorithm 4: fitting the additive multi-task model

```
Initialize $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}$ based on a reduced model: $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$;
repeat
    Consider the residual model $\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{Z}\mathbf{f} + \boldsymbol{\epsilon}$, where $\hat{\mathbf{f}}$ can be solved by
      regularization criterion such as (2.10) using SVR;
    Update $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}(\mathbf{y} - \mathbf{Z}\hat{\mathbf{f}})$ ;
until $\hat{\boldsymbol{\beta}}$ converges;
```

where $\boldsymbol{\beta}$ are fitted with weighted least squares in order to adjust for different variances in different tasks, and the weights in weight matrix $\mathbf{W}$ of the WLS model are estimated as reciprocals of task sample variances, inspired by the approach for constant values of response variables described in Montgomery, Peck, and Vining (2012) [26].

The flexibility of this definition will be shown in the experiment in Section 5.6, where the matrix $\mathbf{X}$ is designed to capture multi-task effects through a one-way ANOVA model, while vectors $\mathbf{z}$ are defined as the original predictor variables. In connection to this example, note that the parametric part can be any model, so any experimental design can be applied here, including three-way ANOVA, however one-way ANOVA was performed in that example due to its simplicity and effectiveness for ILEA schools data.

The iterative algorithm of fitting the model is inspired by the backfitting algorithm, but it also differs in a major way. The original backfitting algorithm has p components (usually the number of predictors), while the proposed algorithm has only 2 components: the linear and nonlinear parts, as $\mathbf{X}\boldsymbol{\beta}$ and $\mathbf{Zf}$ are additive to each other. The first iteration fits the linear estimator to the design matrix $\mathbf{X}$. The initial fitted linear part gets deducted from the response variable in the second step, and the nonparametric model is applied. Then, the parametric coefficients get updated with the response adjusted for the nonlinear part, and the algorithm runs until the linear coefficients converge, which is also the point where the nonlinear part converges. As an extension, an alternative stopping criterion can be defined, for example, the maximization of explained variance or minimization of a loss function.

Alternatively, a fit of $\boldsymbol{\beta}$ can be performed by using the ordinary least squares algorithm, which is a special case of weighted least squares where all observation weights are the same. However, OLS fit would not accommodate the assumption of distinct error variances in each task. A further generalization of the additive multi-task model is needed and will be performed in Section 5.7.

Note that the form of the additive multi-task model in Equation 5.12 is highly reminiscent of a partially linear model. There are, however, a few notable differences. In the additive multi-task model, there is a different nonparametric function for every task, and the error variances are different in every task. Note that the latter assumption is not strictly necessary, and it will be relaxed in Section 5.7.

Algorithm 4 can be seen as a form of continuous bias correction. At first, some variability in predictions is removed by the linear part of the model, then the remaining variability is captured by the nonlinear part. This, in turn, allows the linear part to reduce the variability further, and the process continues until the convergence criteria are satisfied. Thus, linear and nonlinear models are continually correcting each other's biases. However, because of the bias-variance trade-off, correcting the bias can increase variance. This can lead to overfitting, making the model's prediction less generalizable to new, unseen data. Therefore, particular care should be taken when choosing the stopping criterion.

5.5 Algorithmic model selection

In the following, I seek to extend the principle of Algorithm 4. Note that there are two distinct steps in the model fitting procedure. First, the parametric component $\boldsymbol{\beta}$ is fitted, and in the second step, the nonparametric component $\mathbf{f}$ is fitted on a residual model from the first step. This is designed to be one loop of the algorithm. After every loop, the stopping criterion is checked to see whether the performance has improved or worsened. If the performance keeps improving, it is reasonable to run further loops; otherwise, the algorithm should be stopped.

In Algorithm 4, no consideration is taken for the intermediate model, inside the loop. However, there may arise circumstances where the intermediate model performs better than the model of one full loop. Thus, in order to improve this procedure, I generalize this algorithm to cover these intermediate cases.

In Algorithm 5, stopping criterion and metric are flexible. Denote this metric as M. Then,

Algorithm 5: algorithmic model selection for additive multi-task model

```
Initialize $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}$ based on a reduced model: $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$;
Measure $M$ and denote as $M_0$;
repeat
    Solve for $\hat{\mathbf{f}}$ in the residual model $\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{Z}\mathbf{f} + \boldsymbol{\epsilon}$;
    Measure $M$ and denote as $M_1$;
    if $M_0$ is better than $M_1$ then
        Keep the previous model and break the loop
    else
        Set $M_1 \rightarrow M_0$
    Update $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}(\mathbf{y} - \mathbf{Z}\hat{\mathbf{f}})$ ;
    Measure $M$ and denote as $M_1$;
    if $M_0$ is better than $M_1$ then
        Keep the previous model and break the loop
    else
        Set $M_1 \rightarrow M_0$
until loop breaks;
```

Algorithm 5 is designed to run the model fitting procedure on the additive multi-task model until the performance metric is not improving anymore.

Let $K + 1$ be the total number of loops until the stopping criterion is triggered. At the loop $K + 1$ it would be decided that the new model is not as good as the previous one, thus the new model is discarded and the previous model from K loops is chosen as the final model.

It is assumed that the performance metric values can be ranked against each other quantitatively, such that it is possible to select one with the highest performance. Thus, M_0 is better than M_1 implies that M_0 and M_1 can be ranked, and depending on the numerical scale, one is higher/lower than or equal to the other. For example, if general explained variance GEV is the metric for the stopping criterion, the algorithm stops when GEV starts to decrease. Other examples of performance metrics include mean-squared error MSE, Akaike information criterion AIC, Bayesian information criterion BIC, and deviance information criterion DIC.

It can be seen that the Algorithm 4 is a special case of a more general Algorithm 5. The decision of stopping the loop is done at even multiples of K. Thus the first K value where the loop can stop is at $K = 2$, then after the second loop it can stop at $K = 4$, and so on.

5.6 Model construction and application

5.6.1 Motivation

In the following, I demonstrate the effectiveness of the additive multi-task learning model by applying it to the real-world dataset, ILEA schools data. The motivation for this model construction comes from an observation of different distributions of the response variables in each task.

In Figure 5.1 it becomes apparent that the means and standard deviations for each task have quite strong deviations from their estimates for all tasks together. These can dramatically affect the model performance, and the framework of additive multi-task models can be customized to address this specifically.

Moreover, in Figure 5.2 we note that high task response variable means tend to occur with high variances, and vice versa. Thus there is a positive correlation between the tasks' response variable means with their standard deviations, and the correlation coefficient is 0.56. Bringing various task distributions to a standardized scale has the potential to improve the model performance. Moving forward, I make use of the high customizability of the additive multi-task model to address and solve

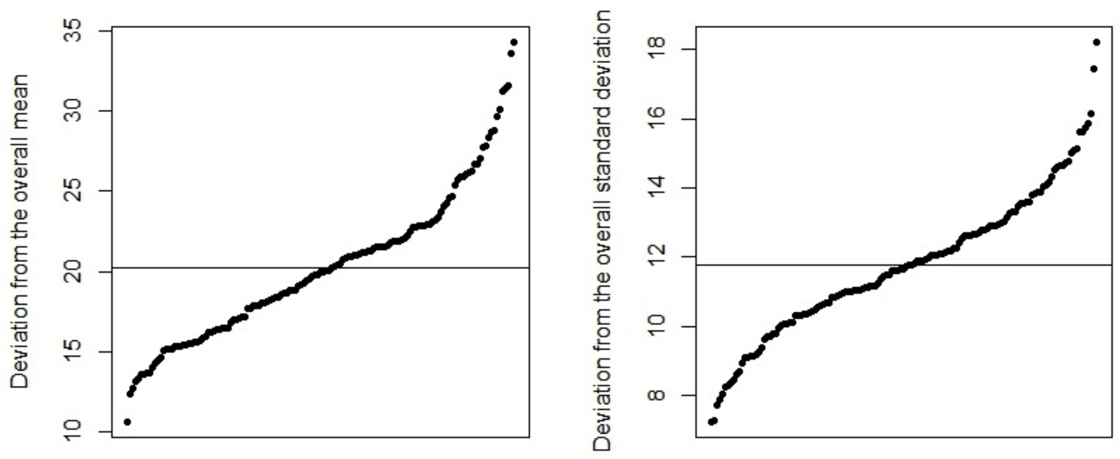

Figure 5.1: Tasks are ranked separately according to their deviation from the overall mean score and standard deviation, which are marked by horizontal lines.

this concern.

5.6.2 Customizing the additive multi-task model

Overview

I apply the additive multi-task model (5.12) in the following way. The matrix $\mathbf{X}$ is defined to be a design matrix of one-way ANOVA to estimate each task's training set mean. This captures each task's group effect, and after decentering, the residual model with the response variable $\mathbf{y}-\mathbf{X}\boldsymbol{\beta}$ allows the distributions of tasks to be closer to each other. This can be seen as a variation of a block design where each task is a block. By applying a simple one-way ANOVA, the block effect is estimated and is removed from the analysis by calculating the residual for the further estimation with the residual model in the next step. Moreover, I adjust for different variances in each task by fitting $\boldsymbol{\beta}$ by weighted least squares with the reciprocals of task sample variances as weights in the weight matrix $\mathbf{W}$. In combination with centering, it has

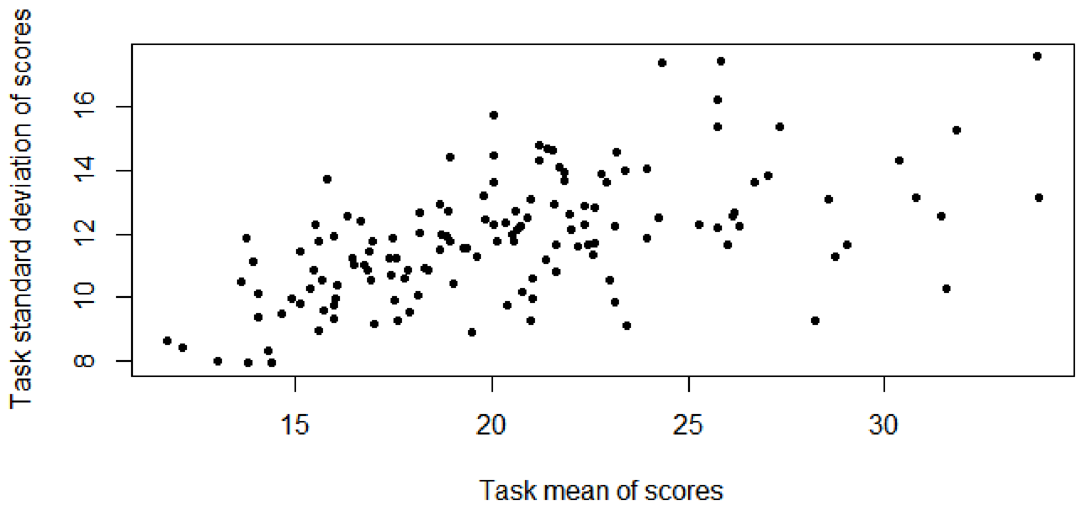

Figure 5.2: Scatterplot of the task means against task standard deviations

the effect of standardization of all observations in each task by using estimates of each task's mean and standard deviation.

As mentioned above, the purpose of the one-way ANOVA is to decenter the tasks separately. We can regard each task as a separate level of factor so that the ANOVA design matrix can be created to find the group (task) means after fitting the parametric part of the model. Defined in Neter et al (2005) [51] and Christensen (2016) [22], the design matrix $\mathbf{X}$ for one-way ANOVA is $n \times m$ matrix, where n is the total number of observations and m is the number of tasks (factor levels). The column j indices observations for task j with integers 1 in rows that belong to task j, and 0 in all other rows. The parameters $\boldsymbol{\beta}$ are encoded in a $m \times 1$ column vector of unknown task (group) means:

$$\boldsymbol{\beta} = [\mu_1, \mu_2, ..., \mu_m]^T \tag{5.13}$$

where μ_j is the unknown mean of task j.

Decentering the tasks in this way addresses the implicit assumption that every task has its own unknown mean of the response variable. Moreover, fitting the parameters with weighted least squares addresses the assumption in 5.4 that every task j has its own variance, different from other tasks.

In the next step, I solve for $\hat{\mathbf{f}}$ in $\mathbf{Zf}$ by applying SVR with Gaussian kernel on a dataset which is transformed with a mean-regularized multi-task kernel described in Section 2.5.

After the above two steps, we may consider that the model training is finished, completing one full loop of Algorithm 4. Note that this is equivalent to running $K = 2$ repetitions in the Algorithm 5.

As the dataset has the highest performance with low lambdas, for demonstration purposes I choose two low values of lambdas and unite their fitted values using the multi-task combined estimate as described in Section 5.3, such that the best value of θ_l is chosen for each task using cross-validation on the test set. The combinations of λ values to be considered are 0 together with 0.01 and 0.1.

Definitions

Let's first consider how this configuration can be stated in a form of a statistical model. Recall the definition of multi-task additive model in Equation 5.12: $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Zf} + \boldsymbol{\epsilon}$. As already mentioned, I define the matrix $\mathbf{X}$ to be a $n \times m$ one-way ANOVA matrix, with weighted least squares fit on $\boldsymbol{\beta} = [\mu_1, \mu_2, ..., \mu_m]^T$ that together facilitates standardization of response variable in each task. Recall that $\mathbf{f} = [f_1(\mathbf{z}), f_2(\mathbf{z}), ..., f_m(\mathbf{z})]'$, where $\mathbf{z}$ are inputs to the functions, and that matrix $\mathbf{Z}$ facilitates functions $f_j(\mathbf{z})$ to link with their respective tasks.

Define $\mathbf{f}^{[q]} = [f_1(\mathbf{z})^{[q]}, f_2(\mathbf{z})^{[q]}, ..., f_m(\mathbf{z})^{[q]}]'$, $\mathbf{X}^{[q]}$ and $\boldsymbol{\beta}^{[q]}$ to be nonparametric functions, parametric design matrix, and parametric component vector, respectively, for model q. Then, the output of model q is:

$$\mathbf{y}^{[q]} = \mathbf{X}^{[q]}\boldsymbol{\beta}^{[q]} + \mathbf{Z}\mathbf{f}^{[q]} + \boldsymbol{\epsilon} \tag{5.14}$$

where the matrix $\mathbf{Z}$ is constant between the models since its function is unaffected by a model choice.

The combined estimation, covered in Section 5.3, was limited to two model outputs. In the following, assume that we operate with Q model outputs, and combine them together in a general form of combined estimate. For task l, we have that:

$$\hat{y}_l^{comb} = \sum_{q=1}^{Q} \theta_{l,q}\hat{y}_l^{[q]} \tag{5.15}$$

where $\theta_{l,q}$ is a weight parameter for task l trained from model q.

For all the trade-off parameters, we have conditions that $\sum_{q=1}^{Q} \theta_{l,q} = 1$, and $0 \leq \theta_{l,q} \leq 1 \; \forall q \in \{1, ..., Q\}$. Therefore, as $\theta_{l,Q} = 1 - \sum_{q=1}^{Q-1} \theta_{l,q}$, it is only necessary to search among the other $Q-1$ trade-off parameters for every task l. It follows that for task l:

$$\hat{y}_l^{comb} = \sum_{q=1}^{Q-1} \theta_{l,q}\hat{y}_l^{[q]} + (1 - \sum_{q=1}^{Q-1} \theta_{l,q})\hat{y}_l^{[Q]} \tag{5.16}$$

Let Θ be a $m \times (Q-1)$ matrix of all parameters, where (l,q)-th element is $\theta_{l,q}$. This matrix omits a column for the Q-th model since its weight is constrained by all other $Q-1$ weights.

Let $\odot$ be a Hadamard product, i.e. element-wise product of two vectors or matrices of the same dimensions. Let n_l be the number of observations in task l, such that $\sum_{l=1}^{m} n_l = n$ is the total number of observations in the whole data (without loss of generality). Let Θ_q be a $n \times 1$ column vector where elements 1 to n_1 are $\theta_{1,q}$ for $l = 1$, and $(\sum_{i=1}^{l} n_i)$ to $(\sum_{i=1}^{l+1} n_i - 1)$ are $\theta_{l,q}$ for $l \in \{2, ..., m\}$. Then the general version of combined estimate is:

$$\mathbf{y}^{[comb]} = \sum_{q=1}^{Q-1} \Theta_q \odot \mathbf{y}^{[q]} + (\mathbf{1}_n - \sum_{q=1}^{Q-1} \Theta_q) \odot \mathbf{y}^{[Q]} \tag{5.17}$$

In this section's demonstration on ILEA schools data, I apply a case of $Q = 2$, which is the standard version of the combined estimate in Section 5.3.

In the following, I describe the algorithm for this example to combine the model outputs, combining two models that use a mean-regularized multi-task kernel with λ_1 and λ_2. Notationwise, $\hat{\mathbf{Y}}^{[j]}_{test}$ denotes prediction of model j on the test set.

Algorithm 6: combined estimation model training for the mean-regularized kernel with λ_1 and λ_2

```
Select $\mathbf{Y}_{train}$ and $\mathbf{X}_{train}$;
Train the models 1 and 2 with mean-regularization parameters $\lambda_1$ and $\lambda_2$,
  respectively;
Predict on $\mathbf{X}_{test}$ with models 1 and 2, obtaining $\hat{\mathbf{Y}}^{[1]}_{test}$ and $\hat{\mathbf{Y}}^{[2]}_{test}$;
foreach task $l \in \{1, 2, ..., m\}$ do
    Select observations of task $l$ in $\mathbf{X}_{test}$, $\mathbf{Y}_{test}$, $\hat{\mathbf{Y}}^{[1]}_{test}$ and $\hat{\mathbf{Y}}^{[2]}_{test}$;
    foreach $\theta \in \{0, 0.01, 0.02, ..., 1\}$ do
        For all task observations, compute combined estimate
          $\hat{y}^{comb}_l = \theta \hat{y}^{[1]}_l + (1-\theta)\hat{y}^{[2]}_l$, obtaining $\hat{\mathbf{Y}}^{[comb]}_{l,test}$;
        Compute and record $EV(\mathbf{Y}_{l,test}, \hat{\mathbf{Y}}^{[comb]}_{l,test})$ that is associated with $\theta$;
    Choose $\theta_l = \arg\max_\theta EV(\mathbf{Y}_{l,test}, \hat{\mathbf{Y}}^{[comb]}_{l,test})$;
    The final predictions for the task $l$ are to be computed as
      $\hat{y}^{comb}_l = \theta_l \hat{y}^{[1]}_l + (1-\theta_l)\hat{y}^{[2]}_l$;
```

To summarize, the additive MTL models in this experiment implement the setup of ANOVA and SVR, together with the combined estimation of models trained on datasets transformed with mean-regularized MTL kernel with $\lambda = 0$ and another small value of λ. For each task i, the optimal θ_i is found in the test set of that task. The optimal C and γ parameters of SVR with Gaussian kernel are found using cross-validation for each λ.

5.6.3 Experimental results

Method	mean EV $\pm$ s.d.
Regularized MTL (Evgeniou et al (2004) [31])	34.8 ± 0.5
MTL-FEAT (Gaussian kernel) (Argyriou (2008) [6])	37.6 ± 1.0
Multi-boost—unweighted (Chapelle (2011) [19])	37.7 ± 1.2
Additive MTL - combined estimate of $\lambda \in \{0, 0.01\}$	38.86 ± 1.07
Additive MTL - combined estimate of $\lambda \in \{0, 0.1\}$	39.07 ± 1.06

Table 5.1: Results of the mean and standard deviation of explained variance on the test set for the ILEA schools data. The results for regularized MTL, MTL-FEAT, and multi-boost are as in Chapelle (2011) [19] and averaged over 10 train-test splits. Additive multi-task learning model results are averaged over 100 train-test splits.

The results of the experiments are stated in Table 5.1, along with the results of state-of-the-art methods in the literature. The additive multi-task model outperforms the best performance by Chapelle (2011) [19] by about 1.3%, indicating a significant improvement over the existing methods. The model with $\lambda = 0.1$ allows a wider trade-off between single-task learning and mean-regularization and achieves better performance than the model with $\lambda = 0.01$, where this trade-off is not so wide. A deeper examination of other values of λ, running further loops in the Algorithm 4, and customizing the linear and nonlinear components for this particular dataset are the directions that are some possible ways to further improve the performance, and are left for further research.

5.7 Full and reduced model generalization

In this section I consider a further generalization of the additive multi-task model and propose model testing procedures. Recall the Equation 5.12 in Section 5.4, which defines the additive multi-task model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{f} + \boldsymbol{\epsilon},$$

The model makes assumptions of uncorrelated error terms, and different variances in each task, and in the Algorithm 4 the model fit of the parametric part $\boldsymbol{\beta}$ is performed using a weighted least-squares algorithm to implement that assumption.

Relaxing these assumptions, consider the model of Equation 5.12 to have independent and identically distributed error terms $\boldsymbol{\epsilon}$ with mean 0, and assume that the parametric part $\boldsymbol{\beta}$ can be estimated by any regression approach. It can be seen that these conditions make the model more general, as the purpose of this generalization is to open for further customizability. Note that with these assumptions, the only difference between the additive multi-task model and partially linear model is the separation of nonparametric components into tasks and the fitting algorithm which can go in multiple loops, compared to only one loop for the partially linear model (see for example Engle, Granger, Rice and Weiss (1986) [28]).

Under these assumptions, the model is very general, and it now includes most of the other previously covered models as special cases. In particular, the additive multi-task model of Section 5.4 is a special case with error terms having different variances within each task, and parametric fit with WLS. However, this is a full model, as both parametric and nonparametric components are present in the model.

Usually, we can't be sure of the correctness of any model until we test all the assumptions. Therefore, the important class of special cases is reduced models. Both the parametric part and the nonparametric part can be evaluated for their suitability to be included in the model.

To test the appropriateness of the parametric part, the decision is to be made on the null hypothesis:

$$H_0 : \boldsymbol{\beta} = \mathbf{0} \tag{5.18}$$

and to test the nonparametric part, a decision is to be made on

$$H_0 : \mathbf{f} = \mathbf{0} \tag{5.19}$$

The decisions on these hypotheses can be done with model selection and diagnostics tools. It can be seen that the set of techniques that can be used to test these hypotheses can potentially include a large part of the statistical inference field. For example, both tests in Equations 5.18 and 5.19 can be separately tested using model complexity penalty measures such as Akaike Information Criterion, Bayesian Information Criterion, or minimum description length. Test for the parametric part $\boldsymbol{\beta}$ of Equation 5.18 can be performed using general linear test (F-test). Some notable nonparametric model selection procedures include those proposed by Yang (1999) [81] and Wegkamp (2003) [77].

In particular, the parametric part of the model in Equation 5.18 can be tested by applying model testing procedures on $\mathbf{y} - \mathbf{Z}\hat{\mathbf{f}}$. In the Algorithms 4 and 5, the model fitting happens on the parametric part first, and then the residual model is trained nonparametrically. In order to facilitate testing on $\mathbf{y} - \mathbf{Z}\hat{\mathbf{f}}$, the algorithms would need to fit the models in a reverse order: first nonparametrically, and then the residual model can be tested with $H_0 : \boldsymbol{\beta} = \mathbf{0}$. For example, a decision on this H_0 can be done with a general linear test. Also, note that since the parametric part has individual components, these can be tested as separately as $H_0 : \beta_j = \mathbf{0}$ for element j in $\boldsymbol{\beta}$. Variable selection procedures can be well-suited, for example, to test of model coefficients in OLS regression.

For partially linear models, Eubank (1999) [29] states the conditions for the test of the parametric part of the model in a special case when nonparametric smoothing is used for estimating the nonparametric part $\mathbf{f}$. The estimator of $\boldsymbol{\beta}$ becomes:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T(\mathbf{I}-\mathbf{S})^T(\mathbf{I}-\mathbf{S})\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{I}-\mathbf{S})^T(\mathbf{I}-\mathbf{S})\mathbf{Y} \tag{5.20}$$

where $\mathbf{S}$ is a smoother matrix of a kernel type estimator for the nonparametric fit of $\mathbf{f}$. It can be shown that $\hat{\boldsymbol{\beta}}$ achieves asymptotic Normal distribution, and the decision on $H_0 : \boldsymbol{\beta} = \mathbf{0}$ can be done on the basis of the t-statistic:

$$t = \frac{\hat{\boldsymbol{\beta}}}{\hat{s}\sqrt{\mathbf{X}^T(\mathbf{I}-\mathbf{S})^T(\mathbf{I}-\mathbf{S})\mathbf{X}}} \tag{5.21}$$

where $\hat{s} = \sqrt{\frac{\sum(y_i - \bar{\mathbf{Y}})^2}{n-1}}$ is a sample standard deviation of $\mathbf{Y}$.

For the two-sided alternative $H_A : \boldsymbol{\beta} \neq \mathbf{0}$, we reject H_0 at α confidence level if $|t| > t_{\alpha/2}$, where $t_{\alpha/2,(n-1)}$ is $\alpha/2$-th percentile of the t-distribution with $n-1$ degrees of freedom.

Reduced versions of the additive multi-task model include conventional parametric and nonparametric models as special cases. As the model fit is done with Algorithm 4, or its general version in Algorithm 5, the model performance can be further strengthened even in cases of reduced models. Note that the initial definition of the additive multi-task model in Equation 5.12 is also a special case of this generalization, although it is not reduced in the parameter space. In the subsequent real dataset example, an analysis of the full version and two restricted versions of the model is performed.

5.8 Additive multi-task model components comparison

As an additional evaluation of model performance for the additive multi-task model, I perform a comparison of the performance of the full additive multi-task model under general assumptions in Section 5.7 and its reduced versions with $\boldsymbol{\beta} = \mathbf{0}$ and $\mathbf{f} = \mathbf{0}$. The purpose of this demonstration is to illustrate the performance benefits of the full model compared to its reduced models, which consist of two separate components of the full model.

For this experiment, the ILEA schools data is transformed with a mean-regularized multi-task kernel with $\lambda = 0.1$. For the full model, the parametric component $\boldsymbol{\beta}$ in the additive multi-task model is fitted with an ordinary least squares linear model, and the matrix $\mathbf{X}$ is defined to contain all predictor variables. The residual component is then fitted on $\mathbf{f}$ with support vector regression with Gaussian kernel, and it has its parameters cross-validated on the residuals $\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$.

The two reduced models consist of the two components of the full model. The first model has $\mathbf{f} = \mathbf{0}$, which means that only the parametric component is included. The second model has $\boldsymbol{\beta} = \mathbf{0}$, thus it is the nonparametric-only model.

The whole ILEA schools data is used and only one fitting loop is performed. Therefore, the performance will be measured by using the total explained variance TEV, as defined in Definition 2.

The results are summarized in Table 5.8. Note that for the full model, TEV values are generally higher than the previously measured EV; this is due to the fact that all the data was used for training and testing, compared to using train-test splits in the previous experiments.

Support vector regression reduced model achieved higher explained variance than

Method	TEV / R^2
Reduced: OLS only	46.5%
Reduced: SVR Gaussian	52.6%
Full: Additive MTL with OLS + SVR Gaussian	53.5%

Table 5.2: Overall explained variance on ILEA schools data with mean-regularized multi-task kernel and $\lambda = 0.1$

the parametric ordinary least squares model alone. However, the additive multi-task model achieves an even higher performance when both of these models are combined. Especially, there is a significant benefit in applying a nonlinear model to the residuals of the linear model fit. This result demonstrates that linear and nonlinear models can increase the mean-regularization effects of the multi-task kernel when they are applied together in the framework of the additive multi-task model. The combination of the two components provides an important boost in performance when maximum performance is required. Applying multiple fitting loops and tuning the values of λ in the multi-task kernel can potentially provide additional performance benefits.

Chapter 6

Statistical task diagnostics

6.1 Overview

In this chapter, I develop and research new statistical task diagnostic algorithms. In multi-task learning, it is assumed that all tasks are somehow related to each other, and it is the objective of the multi-task learning model to learn these task relationships. Tasks can be similarly or differently distributed or have clusters of similarly distributed tasks. There may also arise situations where most tasks are similar to each other, while some few tasks differ in their distribution. We denote the latter as outlier tasks. The vast distributional differences may affect the predictive power of the model fit, such that the model may predict poorly on the majority class tasks and on outlier tasks. It is therefore desirable to identify such tasks and to address them specifically to improve the model performance.

6.2 Response means diagnostics

The purpose of this section is to demonstrate preliminary task diagnostic procedures applied to ILEA schools data. In the real world applications, when the data are sufficiently large, and we know that training and test sets have similar distributions, then it is reasonable to assume that:

$$\bar{\mathbf{Y}}_{train} \approx \bar{\mathbf{Y}}_{test} \tag{6.1}$$

The implication of Equation 6.1 is that the training set and test set means are approximately equal, thus being approximately equal to the whole data mean. This has significance, as if the condition in Equation 6.1 is satisfied, then tasks are distributionally closer to each other among the splits, improving the model's generalization ability from from training to test set. Thus, in the following I investigate whether this assumption can be satisfied.

In the ILEA schools data, the exam scores $y_i \in [1, 70]$. Across 10 splits of ILEA schools data, I find that the average difference $\bar{\mathbf{Y}}_{test} - \bar{\mathbf{Y}}_{train}$ equals -0.017 with standard deviation of 0.228. This indicates that, on average, we expect that the difference between the test and train set to be close to zero somewhat precisely.

The difference between train and test set average responses can vary a lot for different tasks. Figure 6.1 illustrates the density of deviation of test and train set task means $\bar{y}_i$, averaged over 10 train-test splits. Even though the spread of values is attributable to the sampling variation only, nonetheless some tasks are more variable than others. Note that Figure 5.2 demonstrates that there is a positive correlation between tasks' response variable mean and standard deviation, showing that tasks vary a lot in their characteristics.

Figure 6.2 illustrates estimated densities of differences between test and train set

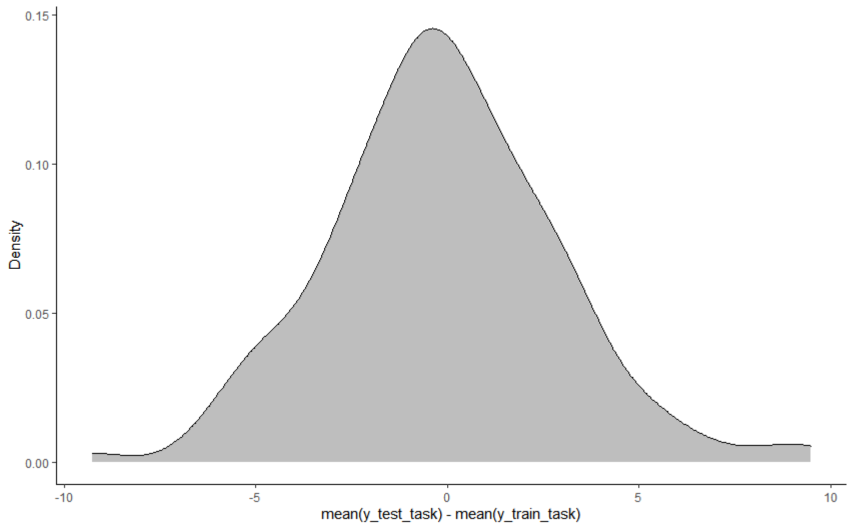

Figure 6.1: Estimated density of difference of test and train set $\bar{y}_i$ for each task, averaged over 10 train-test splits.

response variable means of 9 tasks over 10 train-test splits. The tasks are chosen purely for illustration purposes; these are tasks $1, 2, ..., 9$. If the density is peaked near zero, the response variable tends to the same center in the training and test sets. When the density is flat, it signifies that the difference varies significantly between the splits for that task, meaning that the spread of values is high. Thus, the figure shows that the variability of the difference can vary dramatically between the splits and tasks. It signifies that tasks vary significantly in their distribution of the response variable.

In Figure 6.3 we can observe the residual plots for multiple linear regression models for tasks $1, 2, ..., 9$ in ILEA data. Recall that the assumptions of the MLR model are constant error variance, Normally distributed error terms, linearity, and independence of observations. From the residual plots, we can clearly see that error term variance is not constant in tasks 1, 5 and 9, while for some tasks the assumption is satisfied.

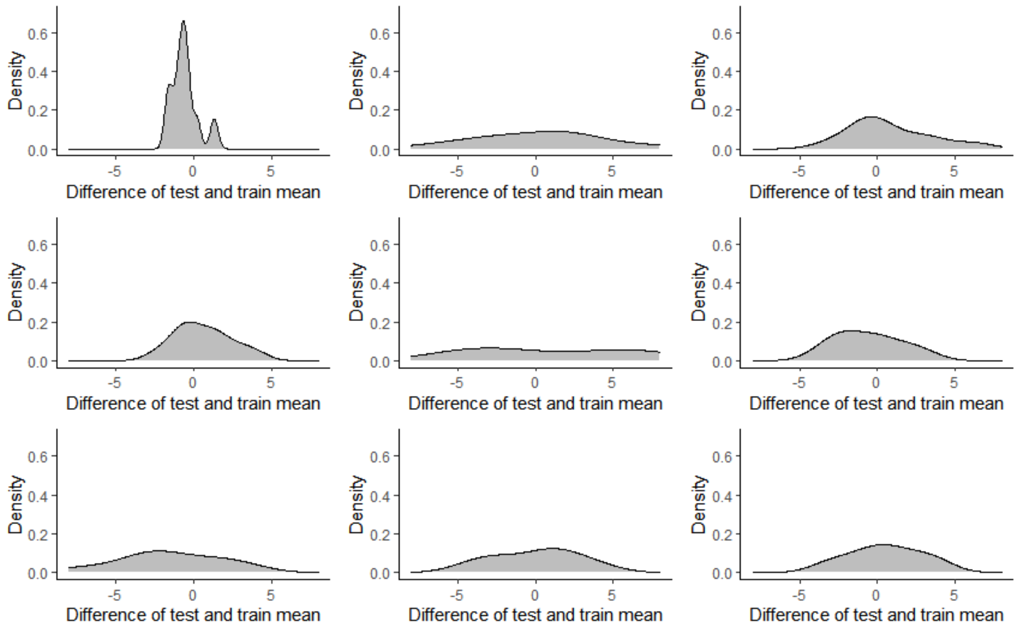

Figure 6.2: For tasks 1, 2, ..., 9 (from right to left, line by line), the estimated densities of difference of test and train set response variable means from 10 train-test splits.

The influence of task data on the satisfaction of model assumption is an important distinction between the tasks, for example as illustrated in Figure 6.3. The methodology of task diagnostics with regard to model assumptions will be further developed in Section 6.6.

As already shown, residuals' conditions differ from task to task. We may use this knowledge to identify outlier observations. Assume that each task has its own OLS fit. To make the maximum absolute residuals in each task to be comparable between the tasks, it is necessary to standardize them. One way is to divide each task's maximum absolute residual by the square root of MSE achieved in that task, which would correct the maximum residual's influence on the overall error rate in a task. Note that the square root of task MSE is the same as task residuals' standard error SE, i.e. estimate of the standard deviation of their sampling distribution. Since the

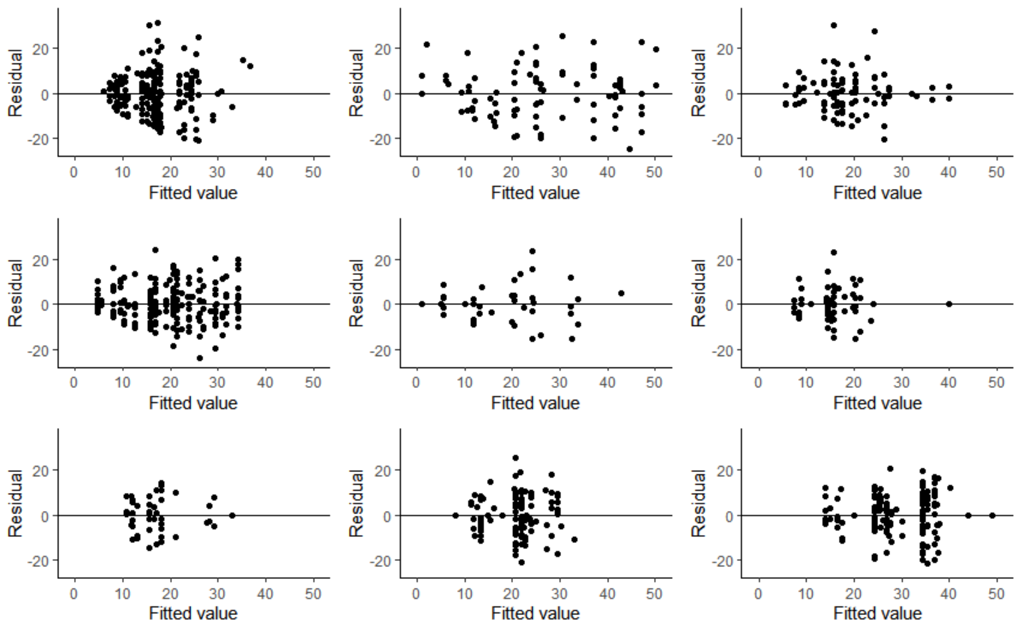

Figure 6.3: For tasks 1, 2, ..., 9 (from right to left, line by line), residual plots of multiple linear regression fits.

mean of residuals is 0 in multiple regression fit, we are now looking at standardized residuals.

Task	Max absolute residual / $\sqrt{MSE}$	Max absolute residual	MSE
37	4.135095	44.80827	117.42088
118	4.075685	33.34340	66.92968
59	4.016647	36.20701	81.25651
82	3.950368	35.47398	80.63892
106	3.825195	26.65971	48.57398

Table 6.1: Extreme observations identification for the top 5 tasks with the highest ratio of maximum absolute residual and square root of MSE.

The results of this experiment are stated in Figure 6.4 and Table 6.1. Note that for most of the tasks, the maximum adjusted absolute residuals are on the low side. There are some tasks where they are high, which is a probable cause to consider these observations as outliers in general. Standardized absolute residuals correlate

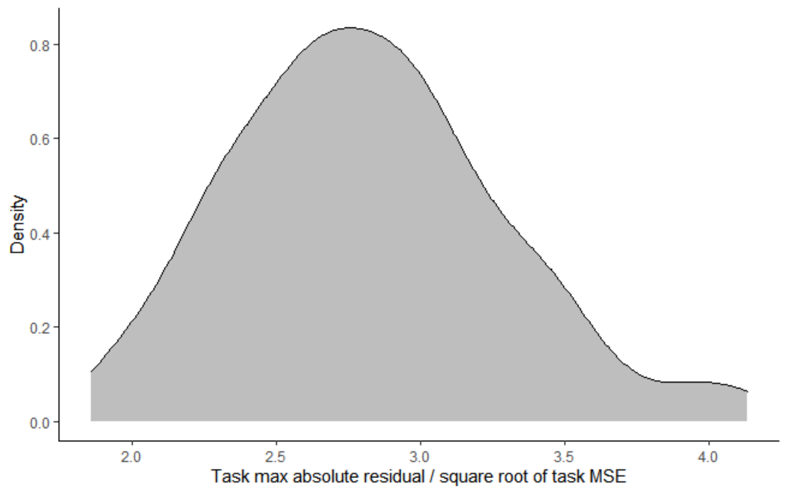

Figure 6.4: Estimated density of tasks' maximum absolute residuals divided by the square root of task MSE.

well with maximum absolute residuals, but there exist tasks with extraordinarily high MSE, which lowers the ratio despite having a high residual. For example, task 102 has a quite high maximum absolute residual of 43.14963, but its MSE is extraordinarily high at 153.53695, which makes the ratio be 3.482336.

In the following, I limit my view to the identification of task outliers. It is an open research question to extend the multi-task learning procedures to the identification of observation outliers in the fashion considered above.

6.3 Task influence test

In this section, I propose an approach to identify outlier tasks inspired by Cook's distance procedure in regression analysis in Cook (1997) [24] and Cook (1979) [25]. For each observation in a regression model, Cook's distance measures the influence of deleting that observation on the predictive performance of that model. The observations with large Cook's distance values are potential outliers and/or highly influential, and merit closer examination.

For multi-task learning, we seek to find outlier tasks instead, as it may not be suitable to consider each observation individually as in Cook's distance procedure. Thus, we can consider the influence of a task by deleting this task from the training data, refitting the model, and checking its influence on the predictive power of the training set. We also seek to keep the deleted task observations in the training set for the purposes of measuring the performance of a model with that task deleted for training. That way, all the models with the deleted tasks can have their performance measured on the same dataset, which allows the easier identification of the outlier tasks.

Using this procedure we obtain *deleted explained variance* (DEV) for each task. For task i, DEV_i is defined as

$$DEV_i(\mathbf{Y}_{train}, \hat{\mathbf{Y}}^{(i)}_{train}) = \frac{\mathrm{Var}(\mathbf{Y}_{train}) - MSE(\mathbf{Y}_{train}, \hat{\mathbf{Y}}^{(i)}_{train})}{\mathrm{Var}(\mathbf{Y}_{train})} \tag{6.2}$$

where $\hat{\mathbf{Y}}^{(i)}_{train}$ are fitted values of all tasks' training set, based on a model which is trained with task i deleted from the training set.

If the DEV of a task is higher than the general explained variance GEV, which is also measured on the training set, then deleting this task has improved the model fit, and this task can be considered to be deleted. If the DEV is lower than the overall explained variance, then deleting this task worsens the overall model fit, therefore such a task should be kept in the model. Thus we seek to remove tasks with high DEV. A more intuitive way of doing that is to compute the difference of the general explained variance GEV, when all tasks are present in the model, with the DEV for every task. This defines the *deleted explained variance test statistic* for task i as D_i:

$$D_i = GEV(\mathbf{Y}_{train}, \hat{\mathbf{Y}}_{train}) - DEV_i(\mathbf{Y}_{train}, \hat{\mathbf{Y}}^{(i)}_{train}) \tag{6.3}$$

where GEV is the general explained variance with all tasks' training sets used both for training and testing. Note that in the presence of train and test splits, the general explained variance GEV should be used in the formula of D_i. When no such splitting is performed, GEV is replaced by total explained variance TEV, which is the proper metric due to the absence of splits.

When D_i is high, then the DEV is low, and vice versa. Using GEV in this way gives us a natural threshold level to identify outlier tasks. Even though only tasks with negative D_i are candidates for being outlier tasks, it may not be the best strategy to delete all tasks with negative D_i. A good strategy is to start with the investigation by deleting the task with the lowest D_i and move forward in a sequential procedure, removing tasks one-by-one until we find a task combination that results in the highest explained variance in the final model (or the lowest error rate).

Algorithm 7: task influence testing procedure

```
train the model on $\mathbf{Y}_{train}$ and $\mathbf{X}_{train}$;
use the model to predict on $\mathbf{X}_{train}$, attain $\hat{\mathbf{Y}}_{train}$;
measure the general explained variance $GEV(\mathbf{Y}_{train}, \hat{\mathbf{Y}}_{train})$;
for each task $i$ do
    select the subset of $\mathbf{Y}_{train}$ and $\mathbf{X}_{train}$ where the task $i$ is excluded;
    train the model;
    use the model to predict on $\mathbf{X}_{train}$, thus attain $\hat{\mathbf{Y}}^{(i)}_{train}$;
    measure $DEV_i(\mathbf{Y}_{train}, \hat{\mathbf{Y}}^{(i)}_{train})$;
    compute $D_i$;
sort $D_i$ from lowest to largest;
remove the tasks with $D_i$ lower than some threshold value $\tau$;
use the data of the remaining tasks to train the final model;
```

The task influence testing procedure is summarized in Algorithm 7. For illustration, this example uses threshold value τ, such that tasks with lower D_i than τ are regarded as outlier tasks. The value of τ can be chosen using cross-validation for the best performance.

6.4 t-distribution cut-off values for task outliers

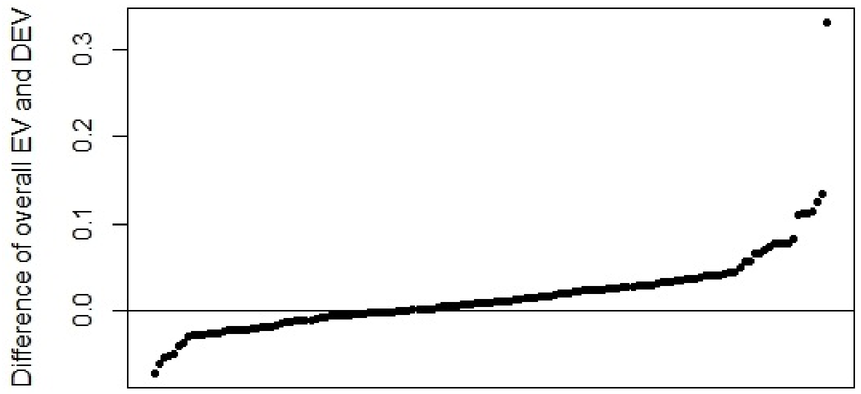

Figure 6.5: Tasks are ranked according to their D_i values. Most differences are positive, meaning that deleting these tasks worsens the model performance. The task furthest to the right has a particularly positive DEV difference, indicating its high importance.

In this section, I establish an empirical approach to choosing the outlier tasks. Figure 6.5 displays each task's deleted explained variance test statistic D_i and ranks them from lowest to highest. Note that this plot has characteristics of a QQ plot and that D_i values look similar to a sample from a normal distribution, although with rather fat tails. This means that the cut-off values can be devised in a manner of a hypothesis test.

Recall that D_i is defined as $D_i = GEV(\mathbf{Y}_{train}, \hat{\mathbf{Y}}_{train}) - DEV_i(\mathbf{Y}_{train}, \hat{\mathbf{Y}}^{(i)}_{train})$, and that the total number of tasks is m. A high value of D_i implies that the task is important to keep in the model, while its low value means that the task is possibly an outlier. As the D_i have approximately normal distribution, we assume that the standardized values of D_i are drawn from a t-distribution with $m-1$ degrees of freedom.

In this setup, it is natural to consider a cut-off value that separates the t-distribution to acceptance and rejection regions. The acceptance region is defined as an area in the t-distribution with $m-1$ degrees of freedom where tasks are not considered outliers. The rejection region is an area where tasks are considered to be outliers and are deleted from the training set. Since tasks with very low D_i are considered outlier tasks, their standardized values of D_i are far in the left tail of the t-distribution with $m-1$ degrees of freedom. It means that we should seek to define the rejection region to be an area of certain probability in the left tail of t-distribution.

In other words, let α be a chosen confidence level. Let $t(\alpha, m-1)$ be a α-quantile of t-distribution with $m-1$ degrees of freedom. Define the acceptance region as

$$\Theta_0 = \{t(\alpha, m-1), +\infty\} \tag{6.4}$$

and the rejection region as

$$\Theta_1 = \{-\infty, t(\alpha, m-1)\} \tag{6.5}$$

Let $\bar{D} = \frac{1}{L}\sum_{i=1}^{m} D_i$ be a sample average and $SD(D)$ be a standard deviation of D_i for all m tasks. The implication of acceptance region is that $P\left(\dfrac{D_i - \bar{D}}{SD(D)} \in \Theta_0\right) = 1-\alpha$, such that we can conclude with $1-\alpha$ confidence that if standardized D_i for task i is in the acceptance region, then the task is drawn from the same distribution as other tasks. Otherwise, if the standardized D_i of task i is in the rejection region, such that $\dfrac{D_i - \bar{D}}{SD(D)} \in \Theta_1$, then it is an outlier task, and it is deleted from the training set.

This task outlier testing procedure is inspired by traditional statistical hypothesis testing methods. Usually, such inferential testing is done to make a conclusion about a particular statistic that has a sampling distribution and its properties often depend on the sample and its size. However, the task outlier procedure is quite different. Every task has its individual D_i value, and every D_i can be considered a statistic as it is based on a sample for task i. And these can have either the same or different distributions across the tasks in the data. The initial assumption in the cut-off values approach is that all D_i have the same distribution, thus they come from the same population. Then, for each D_i a test is performed, to measure whether this task is consistent with the overall distribution of all tasks' D_i values. Then if the task deviates substantially from the overall tasks distribution, it means that it is potentially an outlier task. However, it should only be considered for deletion if it influences the overall model performance in a negative way, i.e. if D_i is lower than for other tasks. If a task has unusually high D_i compared to other tasks, then it influences the model performance in a positive way, so it should not be deleted, despite it being highly influential.

In order to evaluate the procedure, I choose to use the whole dataset, without splitting it into training and test sets, just as is often done in statistical practice. I will use a conventional value of $\alpha = 0.05$. $m = 139$ in ILEA schools data. Therefore, the cut-off value in t-distribution is $t(0.05, 138) = -1.656$. It implies the acceptance region of $\Theta_0 = \{-1.656, +\infty\}$ and the rejection region of $\Theta_1 = \{-\infty, -1.656\}$. All tasks with a standardized D_i in the rejection region are to be considered outlier tasks and deleted from training the final model, and all tasks in the acceptance region are to be kept for training the final model.

I model the data with mean-regularized multi-task kernel support vector regression with Gaussian kernel. The optimal parameters γ and C were found using cross-validation on the whole dataset. The task coupling parameter $\lambda = 0.1$. Under these parameters for ILEA schools data, $TEV(\mathbf{Y}, \hat{\mathbf{Y}}) = 52.605\%$, $\bar{D} = 0.38$ and $SD(D) = 0.24$. Plot of estimated density for the standardized D_i is illustrated in

Figure 6.6.

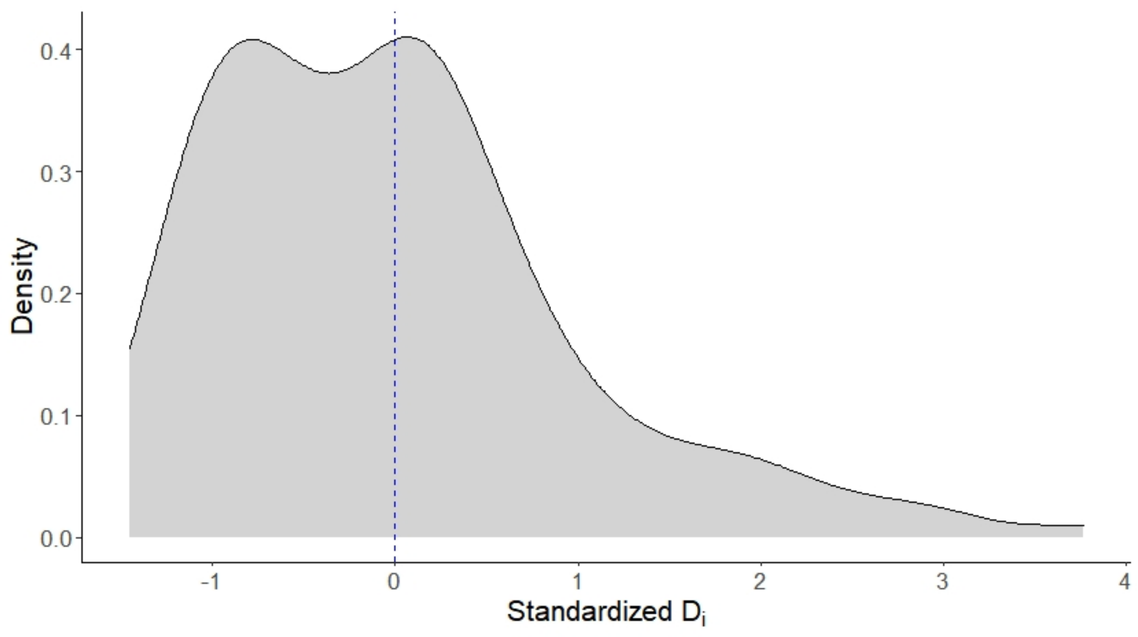

Figure 6.6: Estimated density of standardized D_i

Note that this estimated density curve shows that the distribution of D_i has two peaks, the right tail is rather far out, while there is no left tail. Thus, ILEA schools data has an obvious violation of the assumption that D_i have a normal distribution. In order to address this, I will extend this procedure to cover the cases where the distribution is not normal in the following sections.

There are some tasks with unusually high D_i which perform exceptionally well compared to other tasks, thus making the distribution right-skewed. As discussed previously, deleting these tasks would diminish the overall model performance.

The lowest standardized D_i for this dataset is -1.45, which does not fall into the rejection region of $\Theta_1 = \{-\infty, -1.656\}$. Therefore, with at least 95% confidence, we conclude that there are no outlier tasks, and all tasks need to be included for training the final model.

ILEA data seems to be strongly affected by a few tasks with extraordinarily great performance. For example, the highest standardized D_i is 3.77. It is a known fact that right-skewed distributions have their mean skewed to higher values.

Nonetheless, even though not applicable to the ILEA schools data, the possible benefits of using the t-distribution cut-off values have potential of great use. An open research question is whether the positive task outliers can also be considered to be outside of the overall task distribution, and deleted from the standardization procedure for the purposes of this test only. In this way, the standardized D_i values of other tasks can become more stable, which can potentially ease the identification of the detrimental task outliers with cut-off values from t-distribution.

6.5 Kernel density estimation of D_i

It was made obvious in Section 6.4 and Figure 6.6 that Normal distribution doesn't describe the distribution of standardized D_i values very well. The empirical density shows that the standardized D_i have two peaks, a long right tail, and is quite asymmetric. As a result, the proposed t-distribution test hasn't yielded benefits for task diagnostics.

In order to address this challenge, I propose to apply kernel density estimation for D_i and to use low percentile values from this density as cut-off criteria for the task rejection regions. The theory of kernel density estimation is covered in detail by Scott (1992) [68] and Eubank (1999) [29]. To implement it, I use R's built-in package "stats" [63] and its function "*density*".

For the purposes of this procedure, assume that deleted explained variance statistics $(D_1, D_2, ..., D_m)$ are independent and identically distributed from a univariate distribution given by probability density function s at any given point D. We are interested in estimating this probability density function with the kernel density estimation method, given by:

$$\hat{s}(D) = \frac{1}{mh} \sum_{i=1}^{m} K\Big(\frac{D - D_i}{h}\Big), \tag{6.6}$$

where $h > 0$ is smoothing bandwidth, K is a non-negative kernel function and m is the number of tasks in the data.

The function "*density*" has an option for 7 different kernels, which will all be attempted for the experiment on the ILEA schools data: Gaussian, Epanechnikov, rectangular, triangular, biweight, cosine, and optcosine. For the smoothing bandwidth rule, which decides how the parameter h is estimated, I choose to use Silverman's "rule of thumb" approach (Silverman (1986) [69]), which is built-in to function "*density*" as parameter "*nrd0*" for smoothing bandwidth.

Negative values of D_i indicate that deleting the task i from the training data has increased the explained variance. Thus, the cut-off value is defined in terms of low percentiles of estimated density. Let $\hat{s}_\alpha$ denote its $\alpha \times 100\%$-th percentile. Thus the rejection region is $\Theta_1 = \{-\infty, \hat{s}_\alpha\}$ and the acceptance region is $\Theta_0 = \{\hat{s}_\alpha, +\infty\}$. The tasks in the rejection region are to be deleted from the final model's training data, while tasks in the acceptance region are kept for the model to be trained on.

The results of task identification are stated in Table 6.2. Increasing the significance level includes more tasks that are identified as outlier groups. Notably, the choice of the kernel doesn't change the tasks that are identified as outliers at every significance level.

Figure 6.7 displays estimated kernel density $\hat{s}$. Note the difference from Figure 6.6 where D_i were standardized, while in the current procedure they aren't. The analysis shows that in ILEA schools data, no task has $D_i < 0$, and the reason the estimated

Kernel / Significance Level	5%	10%
Gaussian	1	7
Epanechnikov	1	7
Rectangular	1	7
Triangular	1	7
Biweight	1	7
Cosine	1	7
Optcosine	1	7

Table 6.2: The number of tasks identified as outliers per different combinations of kernel and significance level α. The identified tasks are the same for all kernels at each significance level.

density in Figure 6.7 extends to negative values is the estimation algorithm. As no tasks have negative D_i, all tasks that are identified to be deleted in Table 6.2 have $D_i > 0$.

Recall that in Section 6.4, it was measured that when all tasks are included in the model, $TEV(\mathbf{Y}, \hat{\mathbf{Y}}) = 52.605\%$. At $\alpha = 5\%$, task 99 is identified as outlier. Deleting this task from training data, training the model, and predicting on all tasks' training data is equivalent to its deleted explained variance, with $DEV_{99} = 52.024\%$. At $\alpha = 10\%$, tasks 14, 78, 83, 99, 109, 119 and 138 are identified as outliers. Deleting these tasks from the training set ultimately leads to $TEV(\mathbf{Y}, \hat{\mathbf{Y}}^{(i)}) = 50.428\%$. We find that $TEV(\mathbf{Y}, \hat{\mathbf{Y}})$ is larger than performance both at $\alpha = 5\%$ and $\alpha = 10\%$. Therefore the procedure to use kernel density estimation for task outlier detection hasn't yielded improvement of performance on the ILEA schools data. Nonetheless, the usefulness of the procedure can potentially be high in other multi-task learning applications.

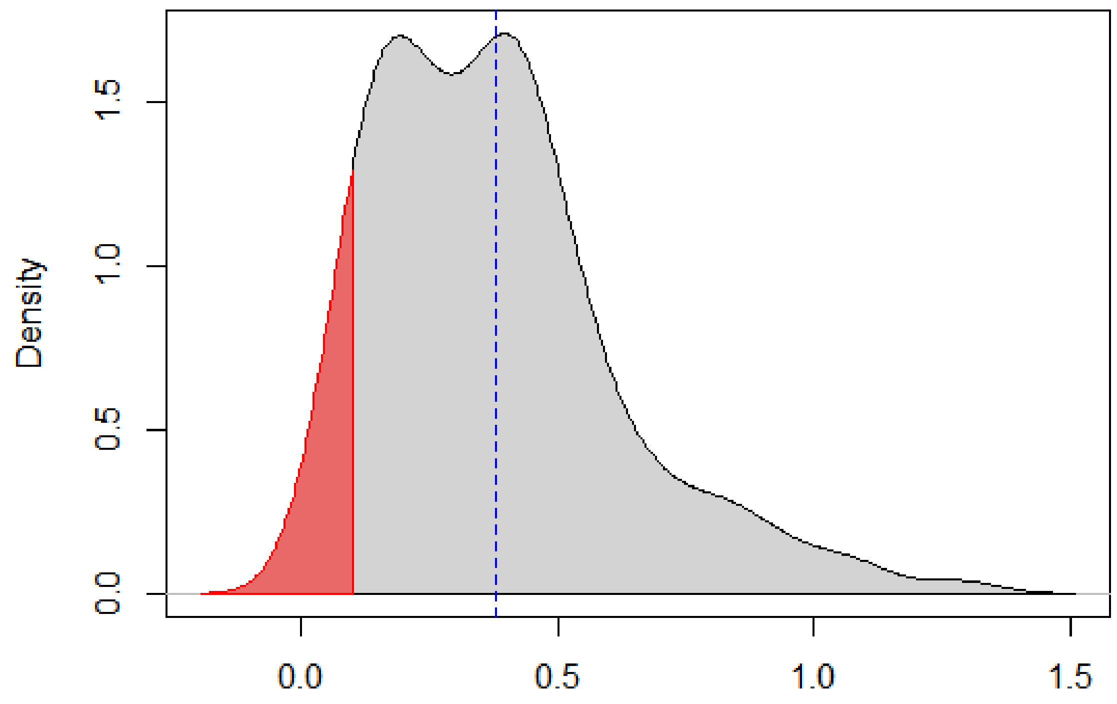

Figure 6.7: Kernel density estimation for $\hat{s}$ with Gaussian kernel. The blue line is mean, and the shaded red area is the rejection region $\Theta_1 = \{-\infty, \hat{s}_{0.10}\}$

6.6 Task importance for model assumptions

Statistical models often come with assumptions about the data that they are applied on. When training the model, the assumptions need to be checked to ensure that the theoretical requirements of the model are satisfied. For example, a multiple regression model has assumptions about independence of observations, constant variance and normal distribution of error terms, and linear relationships between response and predictor variables. Usually, assumptions can be examined in either graphical ways, e.g. residual or QQ plots, or with the help of statistical tests.

In this section, I propose a procedure to identify the outlier tasks with regard to their influence on the satisfaction of model assumptions. The procedure is quantitative, thus I propose to use statistical test statistics instead of visual graph examination, which would have to be done manually. For every task, delete this task from the training data, train the model, and measure the model's conformity with a particular assumption with a statistical test. The obtained test statistic tells about the task's influence on the model assumptions. If a task statistic has moved to a value that shows that the model is in better compliance with an assumption, then deleting the task can be beneficial. Vice versa, if deleting a task made the model less able to satisfy a certain assumption, then this task needs to be kept in the model.

Assume that a model has assumption A, and it can be checked with a statistical test T with H_0 : A is true. The test T is evaluated by producing a test statistic t and comparing it to a distribution of the statistic under the H_0. If such a statistic t is obtained from a model where task i is deleted from the training set, denote this statistic value as t_i and its associated p-value as p_i. Without loss of generality, assume that higher values of t_i are further away from its distribution under H_0. This procedure is encapsulated in Algorithm 8.

Algorithm 8: task importance for a model assumption

```
for each task $i$ do
    select the subset of $\mathbf{Y}_{train}$ and $\mathbf{X}_{train}$ where the task $i$ is excluded;
    train the model;
    use the model to predict on $\mathbf{X}_{train}$, thus attain $\hat{\mathbf{Y}}^{(i)}_{train}$;
    compute $t_i$, the test statistic of $T$ for $H_0$ : A is true;
    obtain the $p_i$, the $p$-value associated with $t_i$;
sort $t_i$ and $p_i$ from lowest to largest;
```

The outcome of the Algorithm 8 is a ranked table of test statistics t_i and their associated p-values p_i for every task $i \in 1, ..., m$. It allows us to measure the influence of every task on the model assumption A. A low value of t_i indicates that deleting the task i has a positive effect on the satisfaction of the model assumption, while a high value of t_i indicates that deleting this task has a detrimental effect on compliance with this assumption.

The course of action depends on the size of the test statistics t_i, and on the range of the spread between the highest and lowest t_i values. Either all, none, or a part of them can be in the rejection region of the test T. If it is obvious that only a few of the t_i's are in the rejection region, then the corresponding tasks are prime candidates for being deleted in the final training dataset. If no t_i values are in the rejection region, it can mean that all tasks are consistent with the model. On the other hand, if all t_i are in the rejection region, it can mean that this model is generally a bad fit to the data, and the task diagnostic methods cannot easily improve the situation.

6.6.1 Application to a real dataset

In this section, I will apply the procedures of Section 6.6 to the ILEA schools data. This process can be applied in a multi-task setting to any statistical model where assumptions need to be checked. For demonstration purposes, I use a multiple linear regression model. The model itself may or may not be a good fit for the data, but it is well-suited for demonstration purposes as it has multiple assumptions which can be checked with numerous statistical tests in a straightforward way. Thus, the Algorithm 8 will be applied to the assumptions of multiple linear regression. Note that this model is a special case of the additive multi-task model in Section 5.7. It is a reduced model with:

- $\mathbf{f} = \mathbf{0}$
- $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$
- $K = 1$ in Algorithm 5

Multiple regression models are operating under a set of assumptions:

- Linear relationship between response variable $\mathbf{y}$ and predictor variables $\mathbf{X}$
- $\boldsymbol{\epsilon} \overset{\text{iid}}{\sim} N(0, \sigma^2)$, i.e. independent and identically Normally distributed with constant variance.

The latter point in the above list consists of three separate assumptions: independence, Normality, and constant variance.

The independence assumption cannot be tested directly but can be partially verified with the Durbin-Watson test for autocorrelated errors (Fox (2008) [35]). The null hypothesis is that the errors are not autocorrelated with each other at lag 1, i.e. are serially uncorrelated. High values of its test statistic indicate that the residuals are autocorrelated. The results of this test are illustrated in Figure 6.8 as an empirical density curve of test statistics for each of 139 tasks deleted. All of the p-values are very close to 0, indicating the rejection of H_0 for each task, and that the residuals are autocorrelated regardless of which task is deleted from the data. The test statistics density is close to the Normal distribution. However, there is some variability with a heavy left tail, which includes the tasks that improve the fit when they are deleted from training.

The constant variance assumption is conventionally tested with the Breusch-Pagan test for heteroskedasticity (Neter (2005) [51]). The null hypothesis is that variances of error terms are constant and equal, and high values of test statistic are evidence against H_0. The results of the test are given in Figure 6.9. All the test statistics are very high, and all p-values are close to zero, indicating that the error term variance is not constant, regardless of which task is deleted from the model. However, note that deleting some tasks results in about $5-10\%$ improvement in the test statistic.

I test the error term Normality assumption with by Anderson-Darling test for Normality and a one-sample Kolmogorov-Smirnov test. In both of these tests, H_0 :

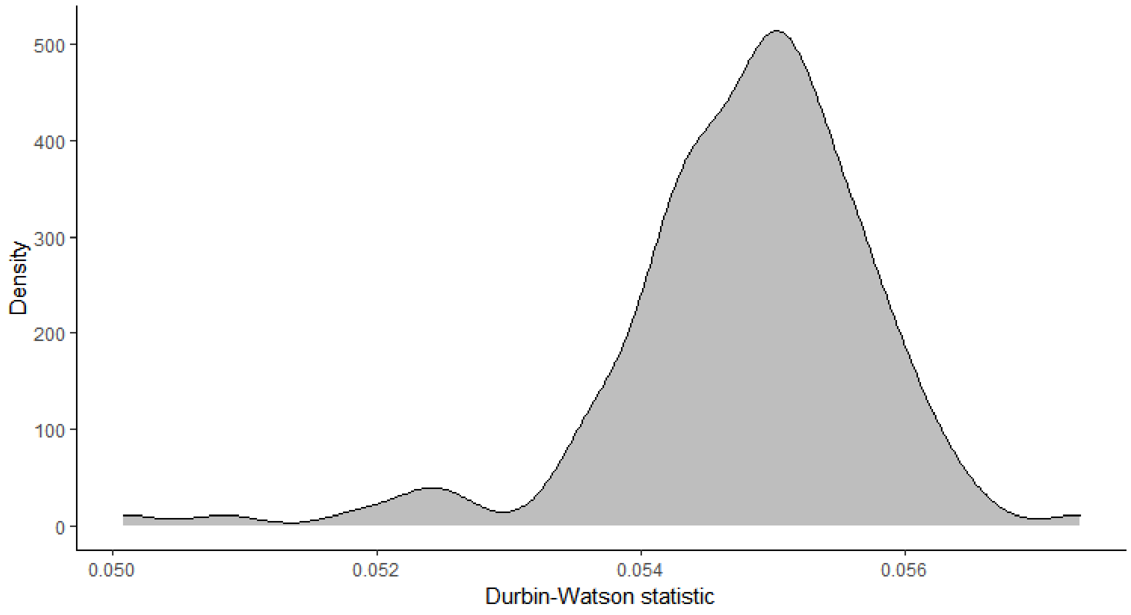

Figure 6.8: Durbin-Watson test statistics density for 139 tasks

ϵ is Normally distributed. For both of these tests, high values of the test statistic are evidence against the null hypothesis. The results for the Anderson-Darling test are illustrated in Figure 6.10, and for the Kolmogorov-Smirnov test in Figure 6.11. The p-values for all tasks are close to zero in both tests, indicating the rejection of null hypotheses that the error term is Normally distributed. In both tests, regardless of which tasks are deleted, the test statistics are closely grouped together in a single peak except for some minor outlier tasks.

The conclusion of the procedure is that deleting any task would not significantly improve the assumptions applicability of the multiple regression model on ILEA schools data. In this case, it indicates that the model is a generally poor fit to the data, and other data or model modifications should be attempted to remedy the assumptions instead.

It can be seen that the usefulness of the procedure is highest when most of the

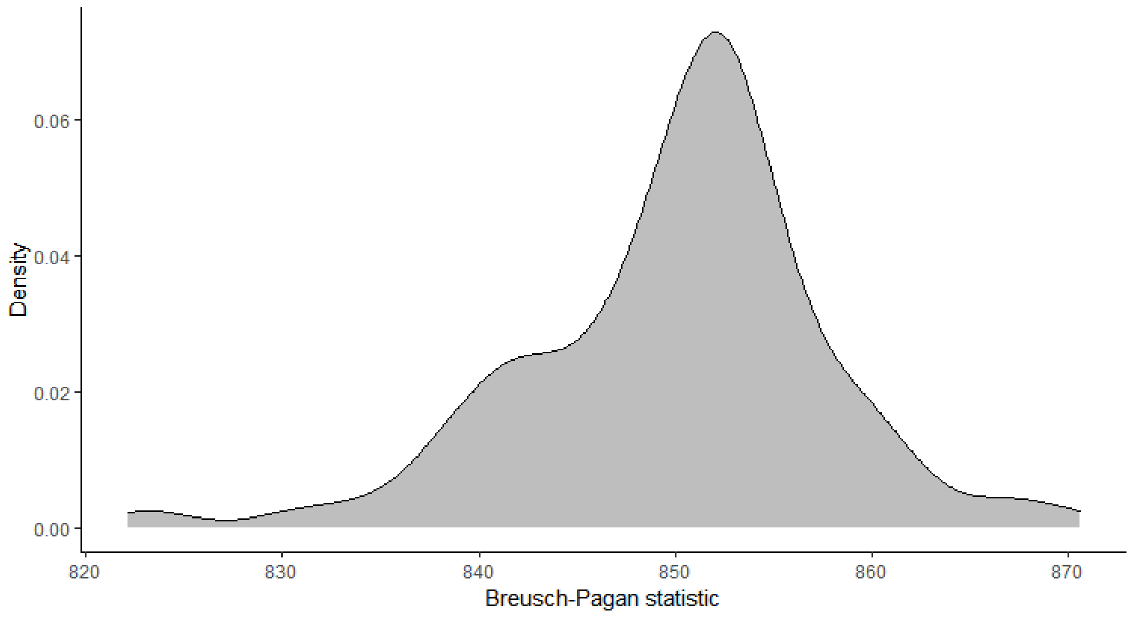

Figure 6.9: Breusch-Pagan test statistics density for 139 tasks. There is a slight variability of statistics between the models with deleted tasks, but all indicate that the constant variance assumption is violated.

statistics t_i are close to the critical value of the rejection region. In that case, the tasks with t_i inside the rejection region are candidates for deletion, which would make it easier to satisfy the model assumptions. This is not the case for the multiple regression model applied to the ILEA schools data, as it can be seen that this model has a generally bad fit to this data. Nonetheless, the usefulness of the proposed procedure can be substantially high in modeling other multi-task datasets.

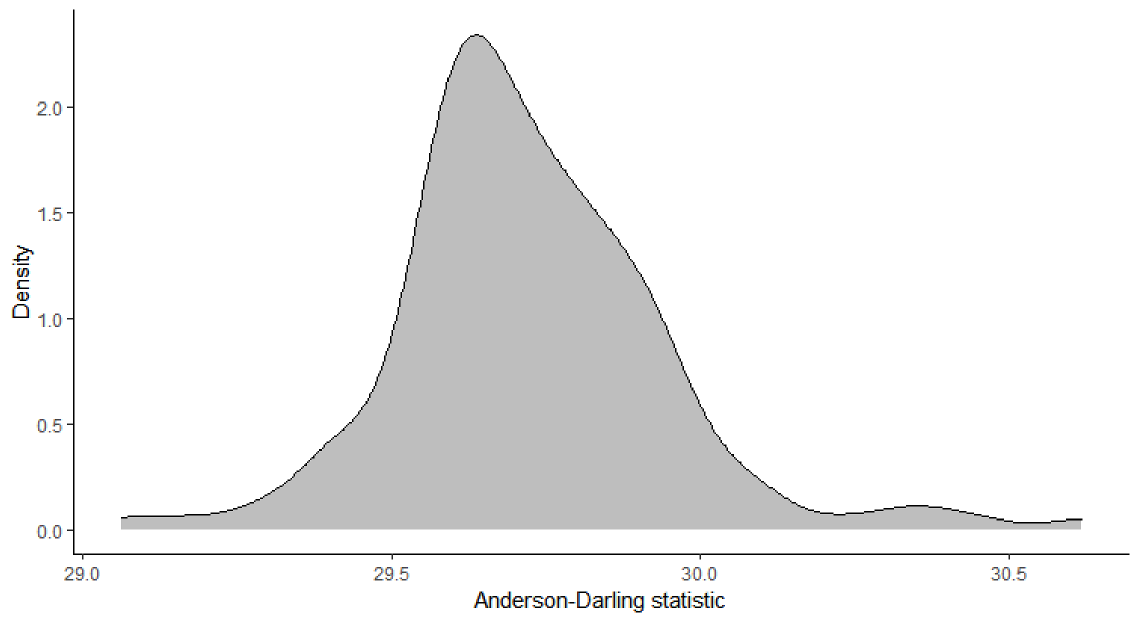

Figure 6.10: Anderson-Darling normality test statistics density for 139 tasks.

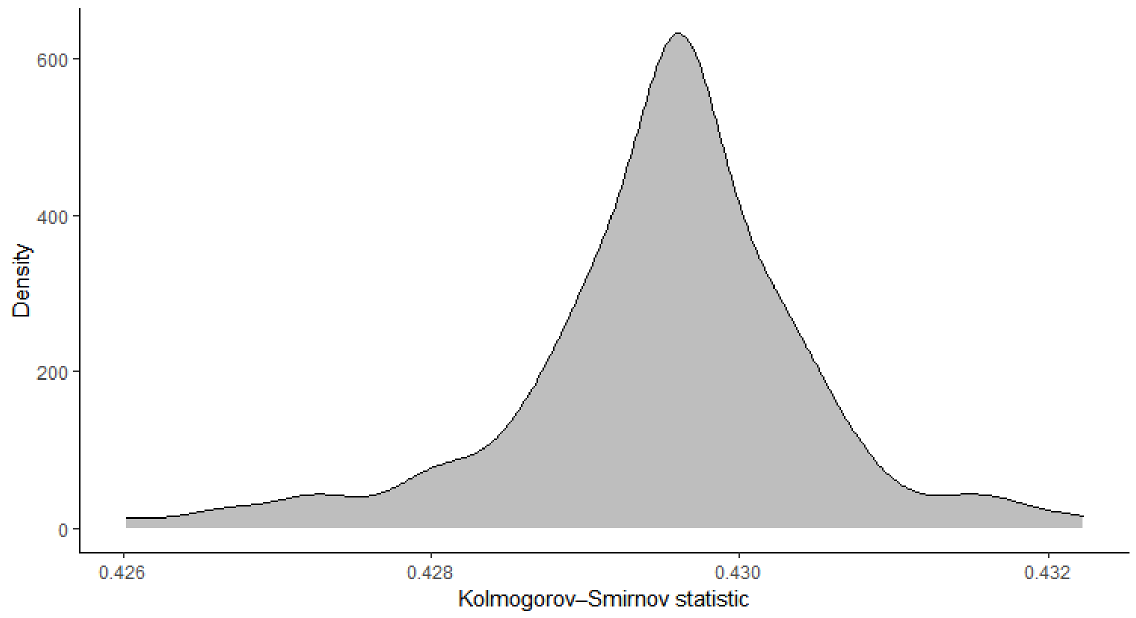

Figure 6.11: One-sample Kolmogorov-Smirnov test statistics density for 139 tasks. Regardless of the deleted task, all models violate the Normality assumption, with all p-values being close to zero.

Chapter 7

Conclusion and open problems

7.1 Conclusion and overview of completed work

Multi-task learning is a very large subfield of machine learning. It is in constant development due to the advancements in modeling and increases in the size and velocity of the data. It is also a field that is inherently connected to statistical modeling, and it is at this intersection of these fields that this dissertation has its main contributions.

First, I contribute to the existing works in multi-task kernel methods by examining the mean-regularization framework both theoretically and experimentally in Chapter 3. I establish the equivalency of two versions of the mean-regularized multi-task kernel as a general case, and for ridge regression in a simplified manner. I examine the influence of the Gaussian kernel on model performance and examine the edge cases of single-task and independent task learning.

Second, I introduce and study the two-step multi-task modeling approach in Chapter 4, which is a new framework that is deeply based on task clustering. I explain the procedure in detail and customize it to present several examples of its application. I also measure their performance, and find that the best performance subsets procedure in Section 4.3.4 is the most powerful of these. More work is needed to find even more beneficial two-step multi-task models.

In the third major contribution, I propose an additive multi-task model in Chapter 5, which is deeply rooted in statistical theory. I further extend this model to general cases of its formulation and fitting algorithm. The model allows intuitive combinations of parametric and nonparametric models for multi-task modeling. Real-world data experiments have shown that this approach achieves a significant performance boost compared to the existing published results of similar multi-task learning models.

Lastly, I propose task diagnostics methods to identify task outliers by their influence on model output in a leave-task-out fashion in Chapter 6. This framework is inspired by statistical testing procedures. I examine cases of task influence on model assumptions and performance metrics, which show strong potential to be useful in real-world data applications. Tracking the sampling distributions of the proposed task statistics is an open question.

To summarize the results achieved by the models in this dissertation, note that two-step multi-task model's best performance subsets algorithm and my customization of additive multi-task model achieve results that are just a few percent explained variance higher than in the literature. Nonetheless, the other published works differ similarly in their achieved performance. The main reason is that Inner London Education Authority data is a rather challenging dataset to improve performance on.

It is a multi-task dataset, where the tasks are schools, and the number of students per school is substantial. Some schools much larger than others, but nonetheless none of the schools have too few students. From the multi-task learning perspective, it may imply that we have enough data from each task, and thus the dataset is modeled best as being close to single-task learning data, which we have observed. It also implies that the multi-task learning procedures are more beneficial in cases where tasks have a stronger need to borrow strength from other tasks. For example, in small area estimation (Rao and Molina (2005) [64]), emphasis is done in particular to model the small areas or domains, thus benefitting the performance of the overall model and the small areas themselves.

7.2 Open problems and future work

In the following, I give an overview of the open questions after the research of this dissertation. This list is not exhaustive. In future work, I intend to use these points to extend and improve my research in multi-task learning.

- Can λ and c versions of mean-regularized multi-task kernel be made equivalent for support vector machines without transforming the datasets, similar to the case of ridge regression as shown in Lemma 3.3.2?
- Can mean-regularized multi-task kernel be learned from the data directly, without setting it a priori?
- The two-step model configuration of best performance subsets (Section 4.3.4) was shown to be the best among all the example customizations. However, it is rather computationally complex. Can its Algorithm 2 be optimized to make it more efficient?
- Further exploration of experimental designs and their model fits for the parametric component in additive multi-task models of Chapter 5.

- Is the (hypothetical) sampling distribution of D_i connected to the model assumptions used for fitted values estimation?
- Which other task statistics can be measured in a leave-task-out fashion for task diagnostics, similar to the methods introduced in Chapter 6?

Appendix A

Data information

The school effectiveness data by Inner London Education Authority has become a popular benchmark in the multi-task learning literature. Some of the analyses of this dataset were performed in Evgeniou et al (2004) [31], Evgeniou et al (2005) [30], Liao and Carin (2005) [53], Argyriou, Evgeniou and Pontil (2008) [6], Zacharia (2009) [82], Agarwal, Daume, and Gerber (2010) [1], Romera-Paredes and Pontil (2013) [66], Fang and Tao (2015) [33] and Kim and Mowakeaa (2019) [48]. The metric that is often considered in the literature when analyzing this dataset is explained variance, which is defined in Bakker and Heskes (2003) [9] as the percentage of test set variance minus the sum of squared errors of the model on the test set, taken as a percentage over the test set variance. Explained variance metric is covered in deeper detail in the Section 2.4 of this dissertation.

The dataset contains records of 15362 students' information in 139 schools. The primary response variable is exam score, and predictors are years, percent of students eligible for free school meals in the school, gender, VR band of the student, percent of students in VR band in the school, student ethnicity, school gender, and school denomination. The school number plays a role of a predictor variable, but is not considered a separate column in the data matrix; rather it is a task indicator that informs the multi-task learning model.

Some data cleaning procedures were performed to improve the quality of the dataset. There are 15 observations with a VR band equal to 0, which is a level that is not mentioned in the dataset description, so I make an assumption that these students belong to the VR band 1. Therefore, after one-hot encoding, there are a total of 26 predictor variables. Moreover, there is one female student in an all-male school 44, which I assume is a typo and edit the gender variable to male.

I follow the procedure of Evgeniou et al (2005) [30] and split the data with a 75%-25% train-test split ratio within each task. The literature on this dataset often considers only 10 train-test splits. The results are then averaged out across the train-test splits and are finally reported as a mean value of explained variance. Results of most experiments are generally stable across different 10 train-test splits, but there are some experiments where more splits are needed. Through trial and error I've found that, depending on the experiment, the results can be unstable between different sets of 10 train-test splits. Thus, in order to give the results more robustness and stability, I have created 100 train-test splits and fixed those for all the experiments performed in this dissertation. For most of the experiments, only the first 10 splits of these 100 splits are used, while there are some experiments where all the 100 splits are used. And for every experiment, I have stated whether the 10 or the 100 splits are used.

Appendix B

Combined variance derivation

This section presents the derivation of Equation 5.6 for a combined variance of two disjoint datasets.

Let the two disjoint datasets be $\mathbf{Y}_{train}$ and $\mathbf{Y}_{test}$. Their sizes are n_{train} and n_{test}, respectively. Let the combined dataset be $\mathbf{Y} = \mathbf{Y}_{train} \cup \mathbf{Y}_{test}$, and let $n = n_{train} + n_{test}$.

Let $\bar{\mathbf{Y}} = \frac{\sum_{i=1}^{n} y_i}{n}$ be a sample mean. Variance is defined in Section 2.4 as $\text{Var}(\mathbf{Y}) = \frac{\sum_{i=1}^{n}(y_i - \bar{\mathbf{Y}})^2}{n}$, which is conventially called a population variance definition. We have that:

$$\text{Var}(\mathbf{Y}) = \frac{\sum_{i=1}^{n}(y_i - \bar{\mathbf{Y}})^2}{n} = \frac{\sum_{i=1}^{n}(y_i^2 + \bar{\mathbf{Y}}^2 - 2y_i\bar{\mathbf{Y}})}{n} =$$
$$= \frac{\sum_{i=1}^{n} y_i^2}{n} + \frac{\sum_{i=1}^{n} \bar{\mathbf{Y}}^2}{n} - \frac{\sum_{i=1}^{n} 2y_i\bar{\mathbf{Y}}}{n} = \frac{\sum_{i=1}^{n} y_i^2}{n} + \bar{\mathbf{Y}}^2 - \frac{2\bar{\mathbf{Y}}\sum_{i=1}^{n} y_i}{n} =$$
$$= \frac{\sum_{i=1}^{n} y_i^2}{n} + \bar{\mathbf{Y}}^2 - 2\bar{\mathbf{Y}}^2 = \frac{\sum_{i=1}^{n} y_i^2}{n} - \bar{\mathbf{Y}}^2$$

and:

$$n\mathrm{Var}(\mathbf{Y}) = \sum_{i=1}^{n} y_i^2 - n\bar{\mathbf{Y}}^2$$

$$n\bar{\mathbf{Y}} = \sum_{i=1}^{n} y_i$$

Then,

$$n(\mathrm{Var}(\mathbf{Y}) + \bar{\mathbf{Y}}^2) = \sum_{i=1}^{n} y_i^2 = \sum_{i=1}^{n_{train}} y_i^2 + \sum_{j=n_{train}+1}^{n} y_j^2 =$$

$$= n_{train}(\mathrm{Var}(\mathbf{Y}_{train}) + \bar{\mathbf{Y}}^2_{train}) + n_{test}(\mathrm{Var}(\mathbf{Y}_{test}) + \bar{\mathbf{Y}}^2_{test})$$

Solving for the variance of the combined dataset **Y**:

$$\mathrm{Var}(\mathbf{Y}) = \frac{n_{train}(\mathrm{Var}(\mathbf{Y}_{train}) + \bar{\mathbf{Y}}^2_{train}) + n_{test}(\mathrm{Var}(\mathbf{Y}_{test}) + \bar{\mathbf{Y}}^2_{test}) - n\bar{\mathbf{Y}}^2}{n}$$

which completes the derivation of Equation 5.6.

Bibliography

[1] Arvind Agarwal, Hal Daumé III, and Samuel Gerber. “Learning Multiple Tasks using Manifold Regularization.” In: *NIPS*. Ed. by John D. Lafferty et al. Curran Associates, Inc., 2010, pp. 46–54.

[2] Shivani Agarwal. *Lecture Notes in Support Vector Machines for Classification and Regression.* 2020.

[3] E. Alpaydin. *Introduction to Machine Learning, third edition.* Adaptive Computation and Machine Learning series. MIT Press, 2014.

[4] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. “Kernels for Vector-Valued Functions: A Review.” In: *Foundations and Trends in Machine Learning* 4.3 (2012), pp. 195–266.

[5] L. Amodei. “Reproducing kernels of vector-valued function spaces”. In: *Proc. of chamonix, a. le meehaute et al. eds* (1997), pp. 1–9.

[6] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. “Convex multi-task feature learning”. In: *Machine Learning* 73.3 (Dec. 1, 2008), pp. 243–272.

[7] Andreas Argyriou, Charles A. Micchelli, and Massimiliano Pontil. “When Is There a Representer Theorem? Vector Versus Matrix Regularizers.” In: *J. Mach. Learn. Res.* 10 (2009), pp. 2507–2529.

[8] Jing Bai et al. "Multi-Task Learning for Learning to Rank in Web Search". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM '09. Hong Kong, China: Association for Computing Machinery, 2009, pp. 1549–1552.

[9] Bart Bakker and Tom Heskes. "Task Clustering and Gating for Bayesian Multitask Learning." In: *J. Mach. Learn. Res.* 4 (2003), pp. 83–99.

[10] Zsolt Bitvai and Trevor Cohn. "Day trading profit maximization with multitask learning and technical analysis." In: *Mach. Learn.* 101.1-3 (2015), pp. 187–209.

[11] Edwin V Bonilla, Felix V Agakov, and Christopher KI Williams. "Kernel multitask learning using task-specific features". In: *Artificial Intelligence and Statistics*. PMLR. 2007, pp. 43–50.

[12] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[13] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[14] Leo Breiman et al. *Classification and regression trees*. Routledge, 2017.

[15] A. Caponnetto and E. De Vito. "Optimal Rates for the Regularized Least-Squares Algorithm". In: *Foundations of Computational Mathematics* (2007).

[16] Andrea Caponnetto et al. "Universal multi-task kernels". In: *The Journal of Machine Learning Research* 9 (2008), pp. 1615–1646.

[17] Claudio Carmeli, Ernesto de Vito, and Alessandro Toigo. "Vector valued reproducing kernel hilbert spaces of integrable functions and mercer theorem". In: *Analysis and Applications* 04 (2006), pp. 377–408.

[18] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A Library for Support Vector Machines". In: *ACM Trans. Intell. Syst. Technol.* 2.3 (May 2011).

[19] Olivier Chapelle et al. "Boosted multi-task learning." In: *Mach. Learn.* 85.1-2 (2011), pp. 149–173.

[20] Jianhui Chen, Ji Liu, and Jieping Ye. "Learning Incoherent Sparse and Low-Rank Patterns from Multiple Tasks." In: *ACM Trans. Knowl. Discov. Data* 5.4 (2012), 22:1–22:31.

[21] Jianhui Chen, Jiayu Zhou, and Jieping Ye. "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*. 2011, pp. 42–50.

[22] Ronald Christensen. *Analysis of Variance, Design, and Regression - Linear Modeling for Unbalanced Data, Second Edition.* 2016.

[23] Ronald Christensen. *Plane Answers to Complex Questions: Theory of Linear Models, Fifth Edition.* Springer, 2020.

[24] R. Dennis Cook. "Detection of Influential Observation in Linear Regression". In: *Technometrics* 19.1 (1977), pp. 15–18.

[25] R. Dennis Cook. "Influential Observations in Linear Regression". In: *Journal of the American Statistical Association* 74.365 (1979), pp. 169–174.

[26] Montgomery D.C., Peck E.A., and Vining G.G. *Introduction to Linear Regression Analysis, Fifth Edition.* Wiley, 2012.

[27] Harris Drucker et al. "Support Vector Regression Machines". In: *NIPS*. Ed. by Michael Mozer, Michael I. Jordan, and Thomas Petsche. MIT Press, 1996, pp. 155–161.

[28] Robert Engle et al. "Semiparametric estimates of the relation between weather and electricity sales". In: *Journal of the American statistical Association* 81.394 (1986), pp. 310–320.

[29] Randall L. Eubank. *Nonparametric regression and spline smoothing.* 2. ed. Statistics: textbooks and monographs 157. New York, NY: Dekker, 1999.

[30] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. "Learning Multiple Tasks with Kernel Methods". In: *Journal of Machine Learning Research* 6.21 (2005), pp. 615–637.

[31] Theodoros Evgeniou and Massimiliano Pontil. "Regularized Multi–Task Learning". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '04. Seattle, WA, USA: Association for Computing Machinery, 2004, pp. 109–117.

[32] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso A. Poggio. "Regularization Networks and Support Vector Machines." In: *Adv. Comput. Math.* 13.1 (2000), pp. 1–50.

[33] Meng Fang and Dacheng Tao. "Active Multi-task Learning via Bandits." In: *SDM.* SIAM, 2015, pp. 505–513.

[34] Jean-Baptiste Fiot and Francesco Dinuzzo. "Electricity Demand Forecasting by Multi-Task Learning." In: *CoRR* abs/1512.08178 (2015).

[35] John Fox. *Applied Regression Analysis and Generalized Linear Models.* SAGE Publications, 2008.

[36] Joumana Ghosn and Yoshua Bengio. "Multi-Task Learning for Stock Selection." In: *NIPS.* Ed. by Michael Mozer, Michael I. Jordan, and Thomas Petsche. MIT Press, 1996, pp. 946–952.

[37] Pinghua Gong, Jieping Ye, and Changshui Zhang. "Robust multi-task feature learning." In: *KDD.* Ed. by Qiang Yang, Deepak Agarwal, and Jian Pei. ACM, 2012, pp. 895–903.

[38] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[39] et al Hernández-Lobato. "A probabilistic model for dirty multi-task feature selection". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1073–1082.

[40] J. P. T. Higgins et al. *Cochrane Handbook for Systematic Reviews of Interventions*. Version 6.1, 2020. Cochrane Training. 2020.

[41] Laurent Jacob, Francis R. Bach, and Jean-Philippe Vert. "Clustered Multi-Task Learning: A Convex Formulation." In: *NIPS*. Ed. by Daphne Koller et al. Curran Associates, Inc., 2008, pp. 745–752.

[42] Laurent Jacob, Jean-philippe Vert, and Francis Bach. "Clustered Multi-Task Learning: A Convex Formulation". In: *Advances in Neural Information Processing Systems*. Ed. by D. Koller et al. Vol. 21. Curran Associates, Inc., 2009.

[43] Pratik Kumar Jawanpuria et al. "Efficient Output Kernel Learning for Multiple Tasks". In: *Advances in Neural Information Processing Systems* 28 (2015).

[44] Tony Jebara. "Multi-task feature and kernel selection for SVMs". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 55.

[45] Jun-Yong Jeong and Chi-Hyuck Jun. "Variable Selection and Task Grouping for Multi-Task Learning." In: *KDD*. Ed. by Yike Guo and Faisal Farooq. ACM, 2018, pp. 1589–1598.

[46] Zhuoliang Kang, Kristen Grauman, and Fei Sha. "Learning with Whom to Share in Multi-task Feature Learning." In: *ICML*. Ed. by Lise Getoor and Tobias Scheffer. Omnipress, 2011, pp. 521–528.

[47] Tsuyoshi Kato et al. "Multi-Task Learning via Conic Programming." In: *NIPS*. Ed. by John C. Platt et al. Curran Associates, Inc., 2007, pp. 737–744.

[48] Seung-Jun Kim and Rami Mowakeaa. "Kernel-Based Efficient Lifelong Learning Algorithm." In: *DSW*. IEEE, 2019, pp. 175–179.

[49] George Kimeldorf and Grace Wahba. "Some results on Tchebycheffian spline functions". In: *Journal of mathematical analysis and applications* 33.1 (1971), pp. 82–95.

[50] Abhishek Kumar and Hal Daumé III. "Learning Task Grouping and Overlap in Multi-task Learning." In: *ICML*. icml.cc / Omnipress, 2012.

[51] M. H. Kutner et al. *Applied Linear Statistical Models*. Chicago: Irwin, 2005.

[52] Roger J Lewis. "An introduction to classification and regression tree (CART) analysis". In: *Annual meeting of the society for academic emergency medicine in San Francisco, California*. Vol. 14. Citeseer. 2000.

[53] Xuejun Liao and Lawrence Carin. "Radial Basis Function Network for Multi-task Learning." In: *NIPS*. 2005, pp. 792–802.

[54] Cheng Liu et al. "Multitask Feature Selection by Graph-Clustered Feature Sharing". In: *IEEE Transactions on Cybernetics* 50.1 (2020), pp. 74–86.

[55] Karim Lounici et al. "Taking Advantage of Sparsity in Multi-Task Learning". In: *Proceedings of the 22nd Conference on Information Theory*. June 2009, pp. 73–82.

[56] Jonathan H. Manton and Pierre-Olivier Amblard. *A Primer on Reproducing Kernel Hilbert Spaces*. cite arxiv:1408.0952Comment: Revised version submitted to Foundations and Trends in Signal Processing. 2014.

[57] Pesi Masani and Jacob Burbea. *Banach and hilbert spaces of vector-valued functions*. Pitman Research Notes in Mathematics Series, 90, 1984.

[58] David Meyer et al. *e1071 — Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien.* R package version 1.7-9. 2021.

[59] Charles A. Micchelli and Massimiliano Pontil. “Kernels for Multi-task Learning.” In: *NIPS.* 2004, pp. 921–928.

[60] Charles A. Micchelli and Massimiliano Pontil. “On Learning Vector-Valued Functions.” In: *Neural Comput.* 17.1 (2005), pp. 177–204.

[61] Shibin Parameswaran and Kilian Q. Weinberger. “Large Margin Multi-Task Metric Learning.” In: *NIPS.* Ed. by John D. Lafferty et al. Curran Associates, Inc., 2010, pp. 1867–1875.

[62] Jian Pu et al. “Multiple Task Learning Using Iteratively Reweighted Least Square.” In: *IJCAI.* Ed. by Francesca Rossi. IJCAI/AAAI, 2013, pp. 1607–1613.

[63] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria, 2022.

[64] John NK Rao and Isabel Molina. *Small area estimation.* John Wiley & Sons, 2015.

[65] Marco Reisert and Hans Burkhardt. “Learning Equivariant Functions with Matrix Valued Kernels”. In: *Journal of Machine Learning Research* 8.15 (2007), pp. 385–408.

[66] Bernardino Romera-Paredes and Massimiliano Pontil. “A New Convex Relaxation for Tensor Completion.” In: *NIPS.* Ed. by Christopher J. C. Burges et al. 2013, pp. 2967–2975.

[67] Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks.* cite arxiv:1706.05098Comment: 14 pages, 8 figures. 2017.

[68] David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization.* Wiley Series in Probability and Statistics. Wiley, 1992, pp. 1–317.

[69] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* London: Chapman & Hall, 1986.

[70] Alex J Smola and Bernhard Schölkopf. "A tutorial on support vector regression". In: *Statistics and computing* 14.3 (2004), pp. 199–222.

[71] Andrej-Nikolai Spiess and Natalie Neumeyer. "An evaluation of R2 as an inadequate measure for nonlinear models in pharmacological and biochemical research: a Monte Carlo approach". In: *BMC pharmacology* 10.1 (2010), pp. 1–11.

[72] Sebastian Thrun and Joseph O'Sullivan. "Discovering Structure in Multiple Learning Tasks: The TC Algorithm." In: *ICML.* Ed. by Lorenza Saitta. Morgan Kaufmann, 1996, pp. 489–497.

[73] Kim-Han Thung and Chong-Yaw Wee. "A brief review on multi-task learning." In: *Multimedia Tools Appl.* 77.22 (2018), pp. 29705–29725.

[74] V. Vapnik. "Principles of Risk Minimization for Learning Theory". In: *Advances in Neural Information Processing Systems.* Ed. by J. Moody, S. Hanson, and R. P. Lippmann. Vol. 4. Morgan-Kaufmann, 1992, pp. 831–838.

[75] V. N. Vapnik and A. Ya. Chervonenkis. "On the Uniform Convergence of Relative Frequencies of Events to their Probabilities". In: *Theory of Probability and its Applications* 16.2 (1971), pp. 264–280.

[76] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Berlin, Heidelberg: Springer-Verlag, 1995.

[77] Marten Wegkamp. "Model selection in nonparametric regression". In: *The Annals of Statistics* 31.1 (2003), pp. 252–273.

[78] Tao Xiong et al. “Probabilistic Joint Feature Selection for Multi-task Learning.” In: *SDM*. SIAM, 2007, pp. 332–342.

[79] Ya Xue et al. “Multi-Task Learning for Classification with Dirichlet Process Priors.” In: *J. Mach. Learn. Res.* 8 (2007), pp. 35–63.

[80] Ikko Yamane, Hiroaki Sasaki, and Masashi Sugiyama. “Regularized multi-task learning for multidimensional log-density gradient estimation”. In: *Neural Computation* 28.7 (2016), pp. 1388–1410.

[81] Yuhong Yang. “Model selection for nonparametric regression”. In: *Statistica Sinica* (1999), pp. 475–499.

[82] Giorgos Zacharia. “Regularized algorithms for ranking, and manifold learning for related tasks.” PhD thesis. Massachusetts Institute of Technology, Cambridge, MA, USA, 2009.

[83] Xiaotong Zhang et al. “Partially Related Multi-Task Clustering.” In: *IEEE Trans. Knowl. Data Eng.* 30.12 (2018), pp. 2367–2380.

[84] Yu Zhang and Qiang Yang. “An overview of multi-task learning”. In: *National Science Review* 5.1 (2018), pp. 30–43.

[85] Yu Zhang and Qiang Yang. “A Survey on Multi-Task Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* (2021), pp. 1–1.

[86] Shi Zhong et al. “Flexible multi-task learning with latent task grouping.” In: *Neurocomputing* 189 (2016), pp. 179–188.

[87] Yue Zhou et al. “Multi-task learning for segmentation and classification of tumors in 3D automated breast ultrasound images”. In: *Medical Image Analysis* 70 (2021), p. 101918.