Empirical Analysis of Regularised Multi-task Learning

Joseph Withers

Bachelor of Science in Computer Science with Honours
The University of Bath
May 2018

Empirical Analysis of Regularised Multi-task Learning

Submitted by: Joseph Withers

Abstract

Multi-task learning aims to generalise the learned model by leveraging relationships between multiple related learning tasks. This is often done through structural regularisation. In this dissertation we categorise and discuss different multi-task learning approaches, implement a subset, and analyse the results giving showing the Low-rank approach out-performing all other implemented methods, with the feature selection approaches also performing amicably and the Dirty approach performing poorly.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Kwang In Kim, for his supervision and useful discussion.

Chapter 1

Introduction

In machine learning the aim is to fit a model to data which can be used to predict the output of new, previously unseen, data. A significant problem in machine learning is that data is noisy and the noise in the data can be captured in the learned model. Noise is introduced into data in any number of ways, from inaccurate measurements to heuristic noise found in every day life. Either way, a model that includes noise will perform well on the data it was learned on, but won't generalise to new data. Generalising a model for a data set is in a sense the essence of machine learning. Multi-task learning methods, discussed in this dissertation, aim to improve the generalisation of the learned model.

The high-level concept of multi-task learning is much like how humans learn. Humans learning two or more similar activities at the same time, such as reading and writing, will leverage the learned skills from both activities. This usually results in both activities being learned faster/better.

These activities may have certain relationships, such as a shared subset of actions (or knowledge) used in each activity, or they could just be similar actions (or knowledge) that can be adapted. Where the word 'activities' has been used above, in a multi-task learning domain we talk about tasks, where each task can be considered as a separate learning problem. Multi-task learning learns multiple related tasks together and uses the relationships between these tasks to generalise the models.

1.1 Motivation

Using models to make predictions is useful in a variety of areas, from stock trading to the detection of illnesses. In the multi-task learning domain, the relationships between the tasks aren't fully known, and can often only be speculated. This makes it problematic to decide on the best multi-task learning method to use. With this in mind, an empirical analysis on current multi-task learning methods, hypothesising the best and worst approaches, will cultivate the use of the best multi-task learning methods for a given problem.

1.2 Hypothesis

The hypothesis investigated in this dissertation is as follows:

Multi-task learning methods will out-perform each other for an array of data sets, leading to a ranking of multi-task learning methods.

1.3 Aim

The aim of this dissertation is to investigate and compare the performance of multi-task learning methods on both toy data and real datasets, providing novel advice with respect to the best multi-task learning methods. This should provide a basis for aiding the use of multi-task learning methods with real-world problems and the further development of learning approaches in this area.

1.4 Objectives

1. **Implement working multi-task learning methods that leverage task relationships through regularisation for linear regression problems.** Research into various multi-task learning methods must be conducted to find viable approaches to implement, as well as research into any available libraries including useful code. This should lead to implementation of fully working multi-task learning methods.

2. **Implement a framework for optimising the hyper-parameters.** The optimisation of each method implemented in objective 1 is for a specific set of hyper-parameters. Research into various approaches such as cross validation will be conducted to find and implement an approach for optimising the hyper-parameters in each multi-task learning method's loss function.

3. **Critical analysis of the results gathered.** A critical analysis based off the performance results gathered from the different implementations will be conducted for both toy data sets and real data sets, linking any findings back to the literature reviewed.

4. **Generation of toy data sets.** Generate toy data sets specifically for testing models on data with certain attributes, with respect to task relationships, useful for the analysis of the different multi-task learning implementations in this dissertation.

1.5 Summary

In summary, this dissertation sets out to implement different multi-task learning methods, conducting experiments to determine a ranking for the implemented methods based on real data sets applicable to the multi-task learning problem domain.

1.6 Dissertation Structure

In Chapter 2, existing literature regarding multi-task learning and the different approaches within the field are categorised and reviewed, as well as consideration of current applications and other relevant topics. The reviewed literature is used to provide a comparison and critic of the categorised approaches. This chapter also reviews optimisation techniques for the multi-task learning approaches themselves (gradient descent, accelerated gradient descent, and proximal operators) along with techniques for optimisation of the hyper-parameters (cross validation). Chapter 3 introduced the implemented multi-task learning methods, the generated toy data sets, and the real data sets used. In Chapter 4, the results are presented and discussed, with Chapter 5 concluding the dissertation. Chapter 5 reflects upon the aims and objectives defined in this chapter.

Chapter 2

Overview and Analysis of Multi-task Learning

Machine learning techniques traditionally involve learning a model based on a feature set. Essentially, machine learning uses a feature set, taken from data, to learn a model. The model can be formed by minimising some loss function, i.e., least square; or training a neural network. There are several different machine learning paradigms depending on the type of data available, such as: supervised for labelled data, unsupervised for unlabelled data, and semi-supervised for some labelled data and some unlabelled data. These traditional machine learning techniques, such as SVM for classification and least square regression are used to learn a model for an individual task; referred to throughout as single task learning (STL).

More recently there has been a development in the field of statistics and computer science, into multi-task learning (MTL). Multi-task learning aims to help with the *labelled data deficiency problem*, whereby in real world application there may be limited labelled data for training. Semi-supervised learning also tries to improve performance of models where labelled data is sparse (Prakash and Nithya, 2014), but multi-task learning takes a different approach by considering multiple learning problems that may be related and exploiting relationships between them.

Multi-task learning has useful applications within natural language processing (Wu and Huang, 2015; Zhao et al, 2015; Wu et al, 2015), computer vision (Fourure et al, 2017; Ranjan et al., 2017; Pons and Masip, 2018; Kienzle and Chellapila, 2006; Heisele et al, 2001), bioinformatics (Xu et al, 2011; He et al, 2016) and more.

Zhang and Yang (2017) Gives a definition for multi-task learning as 'Given m learning tasks $\{T_i\}_{i=1}^{m}$ where all the tasks or a subset of them are related, multi-task learning aims to help improve the learning of a model for T_i by using the knowledge contained in the m tasks.

Furthermore, Caruana (1997) describes multitask learning as: 'MTL improves generalisation by leveraging the domain-specific information contained in the training signals of related tasks.

These definitions imply the conditions that define multitask learning:

- Some, or all, tasks must be related.
- Information from related tasks should be used to improve the generalisation of each tasks model.

In this context, the data from other related tasks can be interpreted as an inductive bias, where the inductive bias causes the learner to prefer some hypothesis to another (Caruana, 1997).

Symmetric multitask learning, covered in this paper, extends these conditions to include:

- The method tries to improve the generalisation of all tasks equally.
- All tasks are learned together.

This is to avoid confusion between other related topics not covered in the context of this paper, such as asymmetric multi-task learning, and transfer learning. Asymmetric multi-task learning involves learning a single task leveraging information from other already learned tasks. Similarly, transfer learning, defined by Pan and Yang (2010) as "Given a source domain D_S and learning tasks T_S, a target domain D_T and learning tasks T_T, transfer learning aims to help improve the learning of target predictive functions $f_T(.)$ in D_T using the knowledge in D_S and T_S, where Ds doesn't equal D_T, or T_S doesn't equal T_T", and therefore not trying to improve the generalisation of all tasks equally. From these definitions, asymmetric multitask learning and transfer learning can be thought of as the same problem; they both focus fundamentally on using additional information (taken from other tasks) to improve a single task. By comparison symmetric multitask learning is the process of learning all tasks in combination, not favouring any particular task.

2.1 Multi-task Learning Setting

A multi-task learning problem needs data from multiple related tasks. This data can come in different forms requiring different kinds of multi-task learning methods.

As referred to in (Zhang and Yang, 2017), homogeneous-feature multi-task learning defines a learning problem where each all tasks share the same feature space. Conversely, heterogeneous-feature multi-task learning is where the features differ over the tasks. Not to be confused with homogeneous multi-task learning and heterogeneous multi-task learning, also defined in (Zhang and Yang, 2017), where homogeneous multi-task learning refers to the type of learning problem, for example classification and regression, for all tasks being the same. Heterogeneous multi-task learning methods allow for each task to be different types of learning problems; some tasks may be classification tasks, and some may be regression tasks, for example.

For completeness, the default setting is outlined using concepts defined in the introduction of this subsection for the multi-task learning methods to be discussed in this paper. All tasks are assumed to be homogeneous, with respect to the type of learning problem. Furthermore, this paper only considers regression learning problems. With respect to the data from each task, all tasks feature sets are assumed to consist of labelled homogeneous features. This constrains the learning paradigm to supervised regression multi-task learning problems.

2.2 Notation

So, within the context of this paper, all problems are supervised learning problems, with training data where the feature set for each task is in the form $\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_N, y_N\}\}$ where $x \in R^d$ and $y \in R$, y being x's label, and N is the number of data points for the particular task. Tasks may have a different number of data points to each other. The common notations used in this paper are shown in Table 2.1.

2.3 Motivation

There are many multi-task learning methods and algorithms with different views and approaches on the problem of leveraging the tasks relationships, as will be discussed further throughout this section. As is the case with single task learning, different multi-task learning methods will perform better for different types of problems. This chapter will provide a high-level overview and discussion of current multi-task learning methods, providing a basis for implementation and analysis. To this end we look towards examining existing works on multi-task learning with an emphasis on summarising and comparing current methods.

2.4 Multi-task Learning Methods

Some of the different approaches for multi-task learning are discussed in this subsection. For the approaches covered in this paper relationships are leveraged or enforced in two ways, shared layers in a neural network, or by using regularisation. These can be used in various ways giving many techniques for multi-task learning. These approaches are categorised into three main categories: learning relationships with features, learning relationships with model

Table 2.1: A description of the notation used throughout this paper.

Notation	**Description**
T	Number of Tasks
m	Dimension of Feature Space
w	The model for a single task
W	The models for a multi-task data set
W_t	The model for task t
x	The features for a single task
x_i	The features for the i^{th} data point in a single task
X	The features for a multi-task data set
X_t	The features for all data points in task t
$X_{t,i}$	The features of data point i, for task t
y	The labels for a single task
y_i	The labels for the i^{th} data point in a single task
Y	The labels for a multi-task data set
Y_t	The labels for all data points in task t
$Y_{t,i}$	The label of data point i, for task t
$L(w)$	Referring to a loss function on a single tasks model
$L(W)$	Referring to a loss function on a multi-task model
$L(W_t)$	Referring to a loss function on a task t's model
$L_{lsr}(W)$	Referring to a least square regression function, $Y = W.X$
$\|\|W\|\| or \|\|W\|\|_2$	L_2 norm regularisation
$\|\|W\|\|_{p,q}$	$L_{p,q}$ norm regularisation
$\|\|W\|\|_F$	$L_{2,2}$ norm regularisation
$\|\|W\|\|_1$	L_1 norm regularisation
$\|\|W\|\|_*$	Trace norm regularisation

parameters, and learning multiple task clusters. Each main category is split further giving a total of eight different categories.

2.4.1 Learning Relationships with Features

Some of the earliest work in multi-task learning leveraged information between tasks directly from the features themselves (Caruana, 1997). Techniques for finding relationships between the raw features, such as using a feature selection approach, a shared feature approach, or deep learning, can allow for complex relationships to be modelled.

Feature Selection

$L_{p,q}$ norm regularisation (discussed in more detail later) can be used as a means of feature selection. Using $L_{p,q}$ norms that enforce sparsity means that the model will have zero values for less important features and therefore ignore these features. This can be thought of as decreasing the feature space for the tasks. Intuitively this is useful for datasets where some features are independent or nearly independent of the true model and therefore not useful in determining the label.

One regularisation Lp norm commonly used in STL is the L_1 norm, known as Lasso in least square linear regression (Tibshirani, 1996). This is easily extended to multi-task learning with an approach outlined in (Liu et al, 2009).

$$\min\{L_{lsr}(W) + ||W||_{p,1}\} \tag{2.1}$$

where $p > 1$.

In line with this (Lee et al 2010) introduces this method with $L_{2,1}$ regularisation on the models.

$$\min\{L_{lsr}(W) + ||W||_{2,1}\} \tag{2.2}$$

This approach carries the assumption that all tasks have the same relevant and non-relevant features. This assumption doesn't hold for problems with outlier and/or negatively correlated tasks.

Shared Representations

The shared representation approach is a feature learning approach which assumes all tasks share some common feature representation. The idea of the shared feature representation approach is to transform the original features to the common shared representation, which may enable more expressive power. An example of this approach is (Caruana, 1997), a method which trains multiple tasks on a network with a shared hidden layer.

Caruana (1997) leverages the relationships between the features; it's "an inductive transfer method that uses the domain specific information contained in the training signals of related tasks". Rather than training each task separately on a backpropagation networks, the tasks are learnt in parallel on the network and share a hidden layer of the network. This is simply an extension of the backpropagation network to have multiple outputs, one for each task. The approach outlined in (Caruana, 1997) can be referred to as shallow multi-task learning.

There are also other variants of shallow multi-task learning which include a task specific input and only one output.

Common representations can also be learned under a regularisation framework (Argyriou et al., 2007). The tasks share a common model U, where

$W = A.U$.

$$\min\{L_{lsr}(W) + ||A||_{2,1}\} \tag{2.3}$$

Using $l_{2,1}$ norm on A encourages row sparsity of A, effectively feature selection on the data, X, transformed by the shared transformation, $U(U.X)$. The loss function for this approach seen in Eq 2.3.

Deep Learning

Deep learning uses more layers in the network than shallow learning which is mentioned above. This can find more complex relationships between the data and tasks. For multi-task deep learning a subset of the layers consist are shared between tasks (Wu et al, 2015)

The disadvantage of deep learning in generally is that it requires much more data than a shallow model to be effective. Recently deep learning has become popular and has shown good results (Wu et al, 2015; Ranjan et al, 2017).

2.4.2 Learning Relationships with Model Parameters

Regularised Tasks Parameters

The regularised task parameters approach assumes the relatedness of each task from a distribution. The methods discussed below achieved this by using a regularisation term on some relationship between the tasks.

More specifically in (Evgeniou and Pontil, 2004), each tasks relatedness is quantified by the mean of the task models, whereby the difference of each task model from the mean model is regularised. Evgeniou and Pontil (2004) extend the previously established single task learning methods which use kernel methods, such as SVM or Least squares, into a multitask learning problem, claiming to be the "first generalization of regularization-based methods from single-task to multi-task learning". As this method assumes the kernel function is linear, i.e., $F(X) = W.X$, where $F(X)$ is the kernel function, W is the model, and X the feature set. This restrains the model for each task to be linear. So, for each task $F_t(x) = W_t.X_t$, Evgeniou and Pontil (2004) assumed that if all tasks are related then they will have similar functions W_t. This implies that they tend towards a common function (the mean model), denoted W_0, which could be found by assuming all the tasks are Gaussian distributed and having W_0 as the mean (Evgeniou and Pontil, 2004). This assumption is formulated as $W_t = W_0 + V_t$. This assumption implies that all task models must be from the same distribution; all task must be related to each other. From here, intuitively, the tasks are more closely related if the V_t's are small. This makes sense as the smaller V_t the closer the tasks' model W_t is to the mean of all other tasks w_0.

The problem in (Evgeniou and Pontil, 2004) is then to minimise the cost of the kernel function for the task model, with regularisation on the error of each tasks model and the difference each model is to the mean model of all task. This is formulated in Eq 2.4.

$$\min\{L_{lsr}(W) + ||W||^2 + ||W - W_0||^2\} \tag{2.4}$$

Changing the regularisation parameters changes how related the tasks are assumed to be. This means that the strength of the relationships between the tasks are determined priori and can be optimised with cross validation, for example.

2.4.3 Splitting Model Parameters with Regularisation

Above we have seen that multi-task learning can be achieved with regularisation, however, the assumption in (Evgeniou and Pontil, 2004) was the relationship of the tasks. This could be less than ideal if there are outlier tasks which are less related than others as it will, firstly, negatively skew the mean, as well as the task model for the outlier task being negatively affected by regularising the variance with another task.

The feature selection approach assumes that all tasks share the same subspace of useful features. This isn't robust to multi-task learning problems where there are outlier tasks in which its model is more dependent on different features.

The question posed in (Jalali et al, 2010) is "can we leverage parameter overlap when it exists, but not pay a penalty when it does not".

Dirty Approach

The dirty approach is where each tasks model's parameters are decomposed into two parameters which enables higher complexity relationships to be captured between the tasks. Different $L_{p,q}$ normalisations on the model can exploit different types of relationships between the models.

The first steps towards the dirty approach (Jalali et al, 2010) came from the fact that there is no effective way to regularise dirty data with one $L_{p,q}$ norm. Dirty data in this respect means data that isn't just row sparse or elementwise sparse. (Jalali et al, 2010) considers the case of linear regression, $y_t = w_t.x_t$. Combining all tasks into one matrix we re-write as $Y = W.X$. The issue here is that finding the optimal W with a regularisation on W (in Eq 2.5) may not fully capture the complexity of the relationships between tasks.

$$\min\{L_{lsr}(W) + P(W)\} \tag{2.5}$$

where $P(.)$ is some regularisation.

Decomposing this matrix W into $S + B$ (i.e. $W = S + B$) allows for different $L_{p,q}$ regularisations on S and B, represented in Eq 2.6.

$$\min\{L_{lsr}(W) + P(S) + Q(B)\} \tag{2.6}$$

where $P(.)$ and $Q(.)$ are some regularisations. Proposed in (Jalali et al, 2010), this decomposition allows for block-sparsity regularisation $L_{1,inf}$, which enforces block sparsity on S, as well as elementwise sparsity regularisations $L_{1,1}$, for enforcing elementwise sparsity on B.

$$\min\{L_{lsr}(W) + ||S||_{1,1} + ||B||_{1,\infty}\} \tag{2.7}$$

Equation 2.7 allows for more complex relationships to be captured. Also, in contrast to without decomposition of W, using the dirty approach the enforced elementwise sparse matrix S means that any outlier tasks will have been taken out of the row sparse matrix B and therefore will not induce a bias within the row sparse regularisation. This makes the dirty approach more robust than regularising with a single $L_{p,q}$ norm.

Multi-Level Approach

To encapsulate even more complexity in the relationships between models the dirty approach can be extended to the multi-level approach. This approach generalises decomposing the model into two matrices, into decomposing the model into n matrices, where $n > 2$.

Low-Rank Approach

The low rank approach leverages the relationships between tasks in the form of assuming a shared low-rank matrix of all tasks model. (Chen et al, 2010) finds a similar solution to improve multi-task learning using a similar idea to (Jalali et al, 2010). This low-rank approach decomposes the regression problem $Y = W.X$ into $Y = (S + B).X$, i.e. $W = S + B$, but forces B to be low rank, Eq 2.8.

$$\min\{L_{lsr}(W) + ||S||_{1,1}\}\ s.t.\ ||B||{*}\ \leq\ threshold \tag{2.8}$$

In other works, such as (Ando and Zhang, 2005), the low-rank approach assumes that all tasks share some low rank sub-space, meaning it can be formulated that $Y = (S + O.P).X$, where $B = O.P$, and O is a shared low rank matrix (Eq 2.9). This can be viewed as an extension of the common representation approach (Argyriou et al., 2007).

$$\min\{L_{lsr}(W) + ||S||_{2,1}^{2}\}\ s.t.\ OO' = I \tag{2.9}$$

Effectively, the low rank approach assumes that the model parameters of each task are a product of some shared low rank matrix. This low rank matrix represents the relationship shared between all tasks. Enforcing this matrix to be low-rank limits the hypothesis space (Ando and Zhang, 2005).

2.4.4 Multiple Task Distributions

The methods covered above (finding task relationships through regularisation, the dirty approach, the multi-level approach, and the low-rank approach) presume a supervised learning setting. They primarily consider that all relationships are related to each other, or in other words, all tasks come from the same cluster, although we have seen attempts for outlier task to not negatively bias the other task models using parameter decomposition into sparse matrices. Multi-task learning, however, encompasses problems where only some of the tasks are related, possibly in multiple clusters.

Task Clustering

The task clustering problem extends the data grouping problem in machine learning to tasks instead of data. Task clustering could be used in conjunction with other methods mentioned in this text and then each cluster could be treated separately, however this defeats one of the main purposes of multi-task learning which is to use more data to leverage better results.

The approach taken in (Jacob et al, 2008) is to introduce multiple regularisations, as is the case in the multi-level approach, however, these regularisations effect the constraints on different types of relationships between the data and groups. Three regularisations are considered: the mean - a constraint on the relationship between all tasks, cluster variance - a constraint on the relationships between clusters, and task variance within a cluster - a constraint on the relationships between the tasks within a cluster. The weighted sum of these regularisations forms a regularisation which can leverage the relationships between all task, tasks within clusters, and the clusters themselves. The objective function formed in (Jacob et al, 2008) is like that in (Evgeniou and Pontil, 2004) extended with regularisation on tasks within clusters, and between task clusters. This approach groups up task models, W_t, groups of models $G_{1...g}$ where g is the number of groups and G_i is a group of task models. Here we define $G_{i,0}$ as the mean model for the group G_i. The loss function described here is shown in Eq 2.10.

$$\min\{L_{lsr}(W) + \sum_{i=1}^{g} ||G_{i,0}||^2 + \sum_{i=1}^{g} ||G_i||^2 + ||S||^2\} \tag{2.10}$$

This approach assumes that the task cluster are known prior, which likely won't be the case in many real-world problems.

The paper (Jacob et al, 2008) proposes weighting the importance of relationships within clusters over the relationships between clusters, both of which over the relationships of all tasks. This ensures that the tasks within clusters are more closely related that the tasks in general.

More recently (Kshirsagar et al, 2017) proposed using $L_{p,q}$ regularisation on separate tasks. This approach also groups up task models, W_t, groups of models $G_{1..g}$ where g is the number of groups and G_i is a group of task models, shown in Eq 2.11.

$$\min\{L_{lsr}(W) + \sum_{i=1}^{g} ||G_i||_{1,2}\} \tag{2.11}$$

2.5 Comparing Multi-Task Learning Approaches

In this paper I have classified several categories of multi-task learning approaches, feature selection, shared features, deep learning, task regularisation, dirty approach, multi-level approach, low rank approach, and task clustering.

In the setting of tasks all related in one cluster, the task regularisation approach is a good solution as it assumes all task come from one distribution and uses other task models to enforce that each task's model tends towards this distribution. The dirty approach and multi-level approach also primarily leverage the relationships of tasks with the assumption of one cluster, however the decomposition of the task model parameters means the task models will be less negatively affected by outlier. It also allows for a greater complexity of relationships to be leveraged, with the multi-level approach allowing for greater complexity than the dirty approach. However, decomposing the model too much could result in model which are less generalised, defeating the main purpose of multi-task learning in the first place. The low rank approach, by assuming that all task model parameter's share a low rank matrix provides similar benefits to the dirty approach, allowing more information to be leveraged about the tasks. Furthermore, the low-rank approach assumes a shared low rank basis for all the models. This comes with similar benefits and negatives as the feature selection approach, as it is doing a similar thing. The clustered approach isn't ideal in a single cluster situation, however, could perform as well as the pairwise regularisation approach provided it determines there is a single cluster. The task regularisation, dirty approach, multi-level approach, low rank approach, clustered approach all encapsulate important information about the task by using leveraging the model's parameters. Using the feature data allows even more complex relationships to be leveraged between tasks, however this approach can be negatively affected by outlier tasks. The feature-based learning approach leverages information gained from the raw feature sets of each task. In the cases mentioned above learning a shallow model with a shared layer may mean there isn't enough leeway for different tasks models to differ. Better than this is to learn a deep model with task specific layers as well as shared layers, however to learn a comprehensive deep model requires a lot of data.

Below is a table (Table 2.2) comparing these methods. Encapsulation applies to if the approach encapsulates important information to leverage. Complexity refers to if the approach can leverage complex relationships between the tasks. Multiple clusters apply to how well the method handles multiple clusters.

Table 2.2: An table rating each multi-task learning category in different attributes

Category	**Encapsulation**	**Complexity**	**Multiple Clusters**	**Outliers**
Mean Regularisation	✓✓			
Dirty Approach	✓✓	✓		✓✓
Multi-level Approach	✓✓	✓✓		✓✓
Low-ranks Approach	✓✓	✓		
Clustered Approach	✓✓		✓✓	✓
Feature-Based Approach		✓✓	✓	
Deep Learning		✓✓	✓✓	✓✓

2.6 Applications

The increase in popularity of multi-task learning has shown it span into many areas. Multi-task learning can be seen in: natural language processing (Wu and Huang, 2015; Zhao et al, 2015; Wu et al, 2015), computer vision (Fourure et al, 2017; Ranjan et al., 2017; Pons and Masip, 2018; Kienzle and Chellapila, 2006; Heisele et al, 2001), bioinformatics (Xu et al, 2011; He et al, 2016). Some examples are briefly discussed below.

2.6.1 Natural Language Processing

Wu and Huang (2015) take ideas from the task relation approach as well as the lasso approach to develop a sparse learner for all tasks with individual task learners. Their approach is implemented to learn the sentiment of natural language.

Zhao et al (2015) use feature selection approach, based off Lasso, for forecasting events from social media. In the multi-task learning problem framework each location (city) is considered as a separate task.

Wu et al (2015) uses a multi-task learning deep neural network which shares hidden layer between two tasks to produce speech synthesis. The paper uses the idea of stacking consecutive frames of bottleneck features as the input to another network.

2.6.2 Face Recognition

Heisele et al (2001) considers different facial components as tasks using single-task learning linear SVM specifically for each component, as well as a combined classifier (multi-task learning classification) for the whole face. This high-level idea of a learner for all tasks as well as individual learners is similar to that seen in (Wu and Huang, 2015).

Ranjan et al (2017) learns face detection, landmarks localisation, pose estimation and gender recognitions simultaneously with multi-task deep learning.

Pons and Masip (2018) propose using multi-task convolution neural networks to learn emotion recognition and facial Action Units together. Their approach extends the classical multi-task deep learning approach to have a single output for all tasks with a sigmoid selective cross-entropy function, used for differentiation between task models.

2.6.3 Image Classification

Kienzle and Chellapila (2006) use SVM with regularisation on the pairwise relationships of each task in the context of handwriting recognition; a similar approach seen in (Evgeniou and Pontil, 2004). For this problem, each character is a separate task.

Fourure et al (2017) uses a similar approach to Ranjan et al (2017) to classify images from utilization of multiple data sets, using multi-task deep learning.

2.6.4 Bioinformatics

Xu et al (2011) implements (Evgeniou and Pontil, 2004) to learn protein subcellular location prediction with sharing model parameters. The paper also looks at using (Argyriou et al, 2006) as a shared representation approach; sharing the latent features.

He et al (2016) uses Lasso and $L_{2,1}$ norm to enforce group sparsity in the models for genetic trait prediction. Zhou et al (2011) also uses Lasso regularisation, in the context of Alzheimer's disease assessment.

2.7 Regularisation

This chapter, so far, has given an overview of related work within the multi-task learning domain. Here we first introduce regularisation and norms in more detail as a basis for moving on to a more detailed discussion of the multi-task learning algorithms implemented in the following chapters.

2.7.1 Overfitting

In non-regularised machine learning, for example, least squares regression, the learning algorithm aims to minimise a loss function Eq 2.12.

$$Y = W.X \tag{2.12}$$

Overfitting occurs when the learning algorithm works "too hard to find the very best fit to the training data" (Dietterich. T, 1995). This means that the model fits the dataset too closely, and the model will contain the residual variance within its structure. The model isn't generalised, therefore will perform worse on data outside of the training dataset.

2.7.2 Regularisation

Regularisation is the addition of information to avoid overfitting. In machine learning this can be thought of as prior information about the model's structure.

The non-regularised machine learning loss function, $L(W)$, and the regularisation function, which we define as $P(W)$, will both have different minimums. The regularised machine learning problem develops into minimizing a new loss function, $L(W) + P(W)$, the sum of the (old) loss function and regularisation function. Regularisation hyper-parameters are often introduced, defined as λ in, $L(W) + \lambda P(W)$, which weight the effect of the regularisation term on the model.

Using regularisation is an attempt for the models to avoid overfitting and generalise the data more. The bigger the hyper-parameters the more weight is given to the regularisation term in the loss function and the simpler the model is and the more it generalises the data. The machine learning methods implemented in this paper use various norm regularisation (for $P(W)$).

Norms

Without going into the mathematical definition of norms her we look at the L_p and $L_{p,q}$ norms needed for the multi-task learning methods used in this paper. L_p norms are defined in Eq 2.13.

$$||w||_p = (\sum_{i=1}^{m} |w_i|^p)^{\frac{1}{p}} \tag{2.13}$$

L_{inf} norm, $||W||_\infty$, is defined as the maximum element in W.

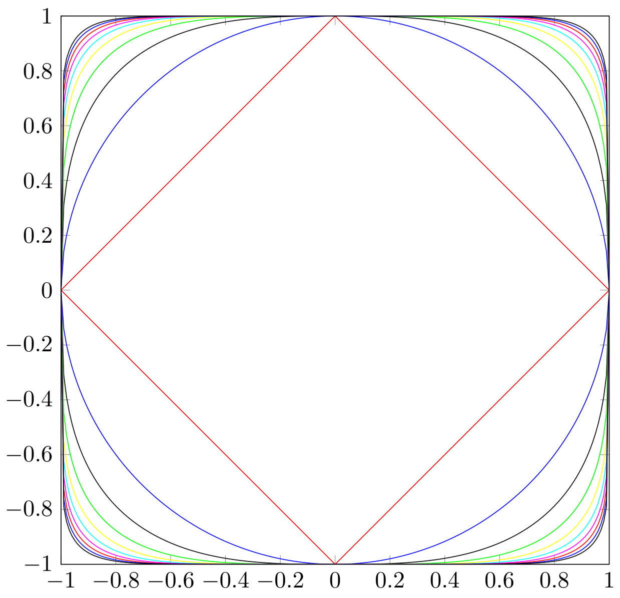


Figure 2.1: visualisation of different L_p norms, where p = 1 to 10, and the L_1 norm in red.

It's apparent from Fig 2.1 that the L_1 norm, which is commonly referred to as Lasso in the context of least square regression, encourages dimensions (elements of W) to be zero. In the context of machine learning, L_1 norms are used to encourage sparsity in the models (Tibshirani, 1996). Conversely to the L_1 norm, the L_∞ norm discourages sparsity. In fact, using any regularisation with L_p norm where $p < 2$, encourages sparsity, where the smaller p the more it encourages sparsity.

Another norm to mention is the absolute value which is the positive value of the same magnitude. This is a special case of the L_1 norm.

L_p norms are used on vectors. In multi-task learning multiple models are being learnt, each model being a vector, forming a matrix. In this case, matrix norms need to be used as matrices are being regularised as apposed to vectors. Relevant is $L_{p,q}$ norms, defined in Eq 2.14.

$$||W||_{p,q} = (\sum_{t=1}^{T}(\sum_{i=1}^{m}|W_{t,i}|^p)^{\frac{q}{p}})^{\frac{1}{q}} \tag{2.14}$$

The $L_{p,q}$ norm, $||W||_{p,q}$, effectively uses L_p norm on the columns of the matrix W and L_q norm on rows. In multi-task learning, this allows methods to encourage row or column sparsity or elementwise sparsity. For example, we have seen that using the $L_{2,1}$ norm encourages row sparsity (Argyriou et al., 2007) and that using $L_{1,\infty}$ norm encourages block sparsity (Jalali et al, 2010).

The Hilbert-Schmidt/Frobenius norm defined in Eq 2.15, is $L_{2,2}$ norm and often denoted $||W||_F$.

$$||W||_F = ||W||_{2,2} = (\sum_{t=1}^{T} \sum_{i=1}^{m} |W_{t,i}|^2)^{\frac{1}{2}} = (trace(W'W))^{\frac{1}{2}} \tag{2.15}$$

The trace of a matrix is the sum of the singular value decomposition of that matrix. The trace norm is a well know way of enforcing sparsity.

In multi-task learning the matrix norms mentioned are used to share information between models and leverage the relationships. This generalises the models, preventing overfitting to the data.

2.7.3 Underfitting

Contrary to overfitting the term underfitting is used to refer to a model which has a high bias, or in other words is too simple. In the case of underfitting the model won't represent trends in the data. Underfitting can be a result of regularising too much. For example, in the feature selection approach, regularising too much will enforce too much sparsity in the model and it won't capture trends in enough features.

2.7.4 Bias vs. Variance

Models that are too complex will suffer from overfitting the data, with high variance, but low bias. Models that are not complex enough will suffer from underfitting the data, with low variance, but high bias. The ideal model has low variance and low bias and in machine learning often a trade-off is found between bias and variance. This is usually found empirically, for example with cross validation which is discussed below.

2.8 Cross Validation

Cross validation is commonly used in machine learning to optimise the regularisation parameters. The accuracy of this hyper-parameter is important in performance. Mentioned above, giving too much weight to the regularisation parameters can result in underfitting, and in contrast, not giving enough weight to the regularisation parameter can result in overfitting. Finding the right hyper-parameters give the best generalisation and therefore a model which is a good generalisation of the data, giving the best prediction results and lowest generalisation error.

The cross-validation extension used to optimise the model's hyper-parameters in this paper is k-fold cross-validation, or more specifically 10-fold cross validation.

2.8.1 K-Fold Cross-Validation

K-fold cross-validation involves splitting the training data set into K equal folds (parts). The model is then learned for $Kminus1$ folds and validated on the held folds. The model learned from the $Kminus1$ folds is validated on the held fold by using the learned model to predict the label for each data point and comparing it to the true label. The error value is calculated and averaged for all data points in the held fold using Eq 2.16. This is repeated (K times) for each part as the validation set and the error for each partition as the held fold is averaged Eq 2.17.

$$Err_k = L(Y - W.X) \tag{2.16}$$

where Y and X are from the held fold.

$$Err = \frac{1}{K}\sum_{l=1}^{K} Err_l \tag{2.17}$$

The loss function, denoted $L(.,.)$ in Eq 2.16 is commonly absolute error Eq 2.18 or sum of squared error Eq 2.19.

$$\frac{1}{T}\sum_{t=1}^{T}\{\frac{1}{N \in t}\sum_{n=1}^{N\in t}|Y_{t,i} - W_t.X_{t,i}|\} \tag{2.18}$$

$$\frac{1}{T}\sum_{t=1}^{T}\{\frac{1}{N \in t}\sum_{n=1}^{N\in t}(Y_{t,i} - W_t.X_{t,i})^2\} \tag{2.19}$$

It's usual to use 5-fold and 10-fold cross validation. Particularly uneven and/or sparse data sets may need larger partitions than 10-fold and get better results from 5-fold validation because this is using a greater percentage of points for the validation set.

K-fold cross validation is done for various chosen regularisation hyper-parameters and the errors are compared to find the best performing hyper-parameters. For each selected hyper-parameter and validation set, in this framework, the corresponding loss function needs to be minimised. This is done using an appropriate optimisation technique.

2.9 Optimization

Optimization techniques are employed in machine learning to find the minimum of a loss function. There exist many optimization methods; this paper discusses gradient descent, accelerated gradient descent and the proximal gradient method. Gradient descent and accelerated gradient descent only work for functions which are convex and differentiable (Nesterov, 1983). The proximal gradient method can be used with functions $L(X) + P(X)$, where $L(X)$ is convex and differentiable and $P(X)$ is convex.

2.9.1 Gradient Descent

In order to be able to use gradient descent to on a loss function $L(W)$, all parts of the loss function must be differentiable, as well as the whole function being convex. The reason the function needs to be convex is because this method finds local minimums, dependent on the starting values, and has no way of determining whether the local minimum is the global minimum. If this is the case, the minimum of the function is found by iteratively moving down the gradient until the gradient is sufficiently small. This is shown algorithmically below in Algorithm 1, where t is chosen empirically.

Algorithm 1 Iterative Gradient Descent

```
1: procedure GRADIENT DESCENT
2:     initialise $W_{k+1}$
3:     $W_k = W_{k+1}$
4:     $W_{k+1} = W_k - t.\nabla L(W_k)$
5:     repeat (from 2) until gradient < threshold
```

2.9.2 Accelerated Gradient Descent

Accelerated gradient descent can be used to minimise a function, fitting the same requirements as gradient decent, with better time complexity. The concept of acceleration is somewhat not yet fully understood, with many interpretations leading to the accelerated gradient descent method. The gradient descent methods outlined above assumes that every new model calculated, line 4 of Algorithm 1, decreases the error of the loss function, whereby $L(W_{k+1})$ ¡ $L(W_{k+1})$ is assumed. The accelerated gradient descent introduced in Nesterov (1983) disregards this assumption allowing for oscillating loss function error, but resulting in a more efficient algorithm. The accelerated gradient descent function is shown in Eq 2.20. With the accelerated gradient decent method shown in Algorithm 2.

$$R = W^k + \tfrac{k-1}{k}(W^k - W^{k-1})$$

$$W^{k+1} = R - t.\nabla L(R) \tag{2.20}$$

2.9.3 Proximal Gradient Descent

Proximal gradient method is an alternating minimization method used when the whole function isn't differentiable. This method used gradient descent or accelerated gradient descent on the differentiable part, and a proximity operator for the non-differentiable parts of the function.

Algorithm 2 Accelerated Gradient Descent

```
1: procedure ACCELERATED GRADIENT DESCENT
2:     initialise $W_{k+1}$
3:     $W_k = W_{k+1}$
4:     $R = W^k + \frac{k-1}{k}(W^k - W^{k-1})$
5:     $W^{k+1} = R - t.\nabla L(R)$
6:     repeat (from 2) until gradient < threshold
```

In the context of multi-task learning with norm regularisation, we consider the case of least square regression with norm regularisation; $L(W) + P(W)$ where $L(W)$ is least square regression and $P(W)$ is the regularisation.

The least square part of the loss function is differentiable. We assume here that the norm regularisation is not differentiable, i.e. not L_2 norm. The proximal gradient method performs an iteration of gradient descent or accelerated gradient descent at W_k to find Z, Algorithm 3 or Algorithm 4. Z is then used in a proximity operator, depending on the norm regularisation, to find W_{k+1}.

Algorithm 3 Gradient Descent with Proximal Operators

```
1: procedure PROXIMAL GRADIENT DESCENT
2:     initialise $W_{k+1}$
3:     $W_k = W_{k+1}$
4:     $Z = W_k - t.\nabla L(W_k)$
5:     $W_{k+1} = prox(Z)$
6:     repeat (from 2) until gradient < threshold
```

Algorithm 4 Accelerated Gradient Descent with Proximal Operators

```
1: procedure PROXIMAL ACCELERATED GRADIENT DESCENT
2:     initialise $W_{k+1}$
3:     $W_k = W_{k+1}$
4:     $R = W^k + \frac{k-1}{k}(W^k - W^{k-1})$
5:     $Z = R - t.\nabla L(R)$
6:     $W_{k+1} = prox(Z)$
7:     repeat (from 2) until gradient < threshold
```

This paper doesn't cover the specifics of proximity operators, but the concept of the proximity operator is to ensure that the solution from the gradient descent or accelerated gradient decent iteration also minimizes the regularisation function.

Chapter 3

Empirical Analysis

3.1 Learning Approach Implementations

To conduct an analysis of different multi-task learning approaches categorised in Chapter 2, five multi-task learning approaches are implemented: regularised-task parameters approach, feature selection approach using Lasso, feature selection approach using $L_{2,1}$ norm, dirty approach, and low ranks approach. As well as implementing multi-task learning, single-task learning is implemented for comparison between multi-task learning and single-task learning.

For some of the implementations discussed below the author makes use of a subset of the methods implemented in the MALSAR package (Zhou et al, 2012). MALSAR, short for Multi-Task Learning via Structural Regularisation, is a package including various multi-task learning methods which utilise structural regularisation to leverage the relationships between tasks. Any use of this library/package is mentioned in the relevant implementation section.

3.1.1 Single-Task Learning - Regularised Least Square Regression

The single-task learning method implemented in this paper is least squares linear regression. To avoid overfitting this was implemented with ridge regularisation (L_2 regularisation), giving the loss function Eq 3.1.

$$E(w) = \frac{1}{2}\sum_{j=1}^{m}(y_j, (w)'x_j + b) + \lambda_1 ||w||^2 \tag{3.1}$$

This loss function has a well-known closed form solution to finding the minimum shown in Eq 3.2. This is found by setting the gradient of $E(w)$, in Eq 3.1, to zero and rearranging for w.

$$w = (x'x + \lambda_1.I)^{-1}x'y \tag{3.2}$$

Extending to a multi-task learning problem domain, Eq 3.2 can be re-written as Eq 3.3, where t is the task being learned.

$$W_t = (X_t'X_t + \lambda_1.I)^{-1}X_t'Y_t \tag{3.3}$$

The solution from Eq 3.3 is for a chosen hyper-parameter, λ_1. As mentioned in Chapter 2, cross validation can be used to choose the best hyper-parameter. In this implementation 10-Fold Cross validation is used to optimise λ_1 in the cost function.

For all combinations of $\lambda_1 = [1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}1e^{0}, 1e^{1}, 1e^{2}, 1e^{3}, 1e^{4}, 1e^{5}]$ the cost function is minimised using Eq 3.3 for each of the 10 folds, and the mean squared error is calculated for each λ_1. Figure 3.1(a) shows the mean squared error found from 10-fold cross validation at each of the λ value defined above, using the school dataset as an example.

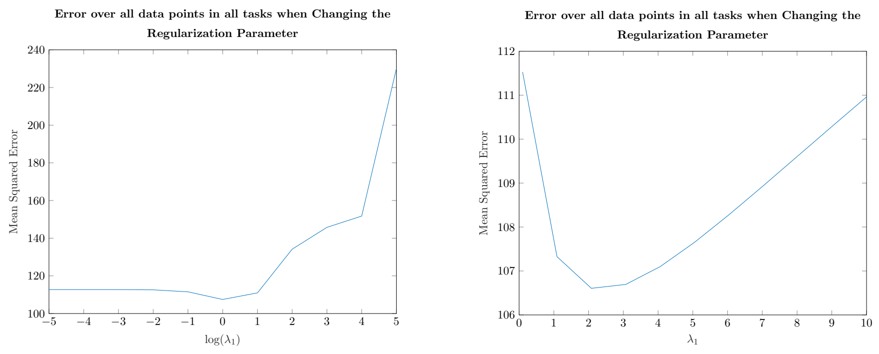


Figure 3.1: Mean squared error from cross validation for different hyper-parameters

The standard 10-fold cross validation framework is extended in this paper to find greater precision λ_1 values. The process above is repeated with a new search range of λ_1 values around the minimum of the previously searched values. For example, if in the first iteration, the optimal λ_1 value was $1e^{-3}$ then the new λ search space would be $\lambda_1 = [1e^{-4}...1e^{-2}]$. In this implementation the λ search space is further refined once, with 10 lambda's equally distributed throughout the new search space, shown in Fig 3.1(b). The whole process is shown in Algorithm 5.

Algorithm 5 is a generalised framework for solving any multi-task learning problem with a loss function that has one hyper-parameter. The loss function on Line 5 in Algorithm 5 is dependent on the method being used. In the case of least square regression with ridge regularisation, the closed form solution is used.

Algorithm 5 Cross Validation for a single hyper-parameter

```
 1: procedure Cross Validation
 2:     $\lambda = [1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^{0}, 1e^{1}, 1e^{2}, 1e^{3}, 1e^{4}, 1e^{5}]$
 3:     for i = $\lambda$ do
 4:         Split up training data into 10 folds
 5:         for k = 1 to number of folds do
 6:             $W$ = Optimisation on 9 non-held folds
 7:             $Err(k)$ = Least Square Error on held fold using $W$ as the model
 8:         $Err_\lambda = Err(k)$
 9:     Find new $\lambda$ search space around $min(Error)$
10:     Repeat from 3 with refined $\lambda$ search space
```

3.1.2 Regularised-Task Parameter Approach (Implementation 1)

The regularised-task parameter approach implemented is using the loss function defined in (Evgeniou and Pontil, 2004). Evgeniou and Pontil (2004) presents a multi-task learning method using regularisation on the variance of each tasks model from the mean model. Extending the high-level discussion of this method in chapter 2, finer detailed of this methods and the implementation is discussed. The cost function being minimised is shown in Eq 3.4.

$$\min\{L_{lsr}(W) + ||W||^2 + ||W - W_0||^2\} \tag{3.4}$$

All parts of Eq 3.4 are differentiable and it is a convex function meaning a gradient step function can be used. The implementation in this dissertation uses gradient descent.

Gradient descent, introduced in chapter 2, uses the derivative of the cost function to find the minimum. The workings for the gradient function of Eq 3.4 are shown below, Equation 3.5 giving the derivative.

$$\nabla\{L_{lsr}(W) + ||W||^2 + ||W - W_0||^2\}$$

$$= 2(W.X - Y).X + 2W + \frac{-2}{T}(\sum_{t=1}^{T}(W_t - W_0)) + \sum_{t=1}^{T}(W_t - W_0) \tag{3.5}$$

The model W is initialised using the single-task least square regression closed form solution to guess the starting model for each task. This improves the convergence speed of this method, than simply starting with zero filled or random models. The gradient is found by inputting the current model values into Eq 3.5. A fraction of the gradient, determined by alpha (the convergence rate), is subtracted from the model, thereby stepping the model towards the optimal model to minimise Eq 3.4. This step is repeated until the gradient is sufficiently small, which indicates the cost function is (close to) minimised. This is shown algorithmically in Algorithm 1.

Algorithm 1 optimises the model for the chosen hyper-parameter λ_1 and λ_2 in the cost function Eq 3.4. To choose the hyper-parameters the approach described in Algorithm 5 is extended to optimise for two lambda values. For all combinations of $\lambda_1 = [1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}1e^{0}, 1e^{1}, 1e^{2}, 1e^{3}, 1e^{4}, 1e^{5}]$ and $\lambda_2 = [1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}1e^{0}, 1e^{1}, 1e^{2}, 1e^{3}, 1e^{4}, 1e^{5}]$ the cost function is minimised using Algorithm 1 for each of the 10 folds, and the mean squared error is calculated for each combination of λ_1 and λ_2. Figure 3.2(a) shows the mean squared error found from 10-fold cross validation at each of the lambda value combinations defined above.

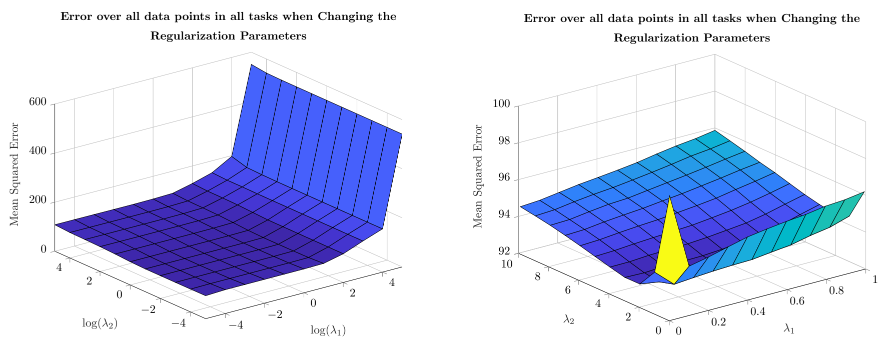


Figure 3.2: Mean squared error in cross validation for different hyper-parameters

Again, the standard 10-fold cross validation framework is extended to find greater precision λ values. The process above is repeated with a new search range of lambda values around the best λ values found. For example, if for the first iteration, the best λ_1 value was $1e^{-3}$ and the best λ_2 value was $1e^{2}$ the new λ search space would be $\lambda_1 = [1e^{-4} \ldots 1e^{-2}]$ and $\lambda_2 = [1e^{1} \ldots 1e^{3}]$.In this implementation the λ search space is further refined once, with 10 by 10 grid of lambda's equally distributed throughout the new search space, shown in Fig 3.2(b). This process is outlined in Algorithm 6.

3.1.3 Regularised-Task Parameter Approach (Implementation 2)

The mean regularised method presented by Evgeniou and Pontil (2004) is implemented again but using MALSAR's Least_SRMTL.m function. This implementation uses accelerated gradient descent, which will give different performance results to the authors implementation. This implementation is used in the empirical analysis for better consistency with respect to the other implementations in this paper. MALSAR's implementation minimises the cost

Algorithm 6 10-Fold Cross Validation for 2 hyper-parameters

```
 1: procedure 10-Fold Cross Validation
 2:     $\lambda_1 = [1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^{0}, 1e^{1}, 1e^{2}, 1e^{3}, 1e^{4}, 1e^{5}]$
 3:     $\lambda_2 = [1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^{0}, 1e^{1}, 1e^{2}, 1e^{3}, 1e^{4}, 1e^{5}]$
 4:     for $j = \lambda_1$ do
 5:         for $j = \lambda_2$ do
 6:             Split up training data into 10 folds
 7:             for k = 1 to number of folds do
 8:                 $W$ = Optimisation on 9 non-held folds
 9:                 $Err(k)$ = Least Square Error on held fold using $W$ as the model
10:             $Error_{\lambda_1,\lambda_2} = Err(k)$
11:     Find new $\lambda_1$ and $\lambda_2$ search space around $min(Error)$
12:     Repeat from 3 with refined $\lambda$ search space
```

function shown in Eq 3.6.

$$\min_W \{ \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_F^2 + \lambda_1 ||W.R||_F^2 + \lambda_2 ||W||_1 \tag{3.6}$$

This is a generalisation of Evgeniou and Pontil (2004) approach, which can fit the task relation structure to any graph R, in Eq 3.6. Defining R as shown in Eq 3.7 instantiates Eq 3.6 to the cost function defined in (Evgeniou and Pontil, 2004), Eq 3.8, where t is the number of tasks and both matrices in Eq 3.7 are size t by t.

$$R = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} - \frac{1}{t} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{3.7}$$

$$\min_W \{ L_{lsr}(W) + \lambda \sum_{i=1}^{t} ||W_i - \frac{1}{t} \sum_{s=1}^{t} W_s|| \tag{3.8}$$

The regularisation parameters λ_1 and λ_2 are optimised with 10-fold cross validation using the same method outlined in Algorithm 6.

3.1.4 Feature Selection Implementation (Lasso)

To encourage sparsity in the model the feature selection implementation in this paper uses the L_1 norm as regularisation on the models. The L_1 norm was introduced as regularisation in single-task linear regression in (Tibshirani, 1996), known as the Lasso. Described in the feature selection category in chapter 2, the Lasso can be extended to multi-task learning (Liu et al, 2009), and encourage shared sparsity in the models. Encouraging block sparsity in the models leads to feature selection, whereby less important features are ignored by the models. The cost function for this approach is shown in Eq 3.9.

$$\min_{W} \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_1 + \lambda_1 ||W||_{2,1} + \lambda_2 ||W||_F^2 \tag{3.9}$$

The regularisation parameters for this function are not differentiable, so proximal operators are used alongside accelerated gradient descent to optimise this function. This paper uses MALSAR's *Least_Lasso.m* function for optimisation of this method.

The regularisation parameters λ_1 and λ_2 are optimised with 10-fold cross validation using the same method outlined in Algorithm 6.

3.1.5 Feature Selection Implementation ($L_{2,1}$)

As discussed in Chapter 2, some $L_{p,q}$ norms can be used for joint feature learning in the multi-task learning framework. This feature selection implementation instantiates using $L_{p,1}$ regularisations on the model W, where $p > 1$ (Liu et al, 2009). The cost function optimised uses $L_{2,1}$ regularisation on the models for join feature selection (Lee et al, 2010; Argyriou et al., 2007), shown in Eq 3.10.

$$\min_{W} \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_F^2 + \lambda_1 ||W||_{2,1} + \lambda_2 ||W||_F^2 \tag{3.10}$$

This cost function, for a given set of hyper-parameters, is optimised using MALSAR's *Least_L21.m* function, which uses accelerated gradient descent and proximal operators.

The hyper-parameters λ_1 and λ_2 are optimised with 10-fold cross validation using the same method outlined in Algorithm 6.

3.1.6 Dirty Approach Implementation

The dirty approach uses the decomposition of the model to allow for two regularisations on the model itself. Jalali et al (2010) uses the $L_{1,1}$ norm and $L_{1,\infty}$ regularisations giving the cost function Eq 3.11.

$$\min_{W,P,Q} \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_F^2 + \lambda_1 ||P||_{1,\infty} + \lambda_2 ||Q||_1, s.t W = P + Q \tag{3.11}$$

Neither $L_{1,1}$ or $L_{1,\infty}$ norms are differentiable, so accelerated gradient descent with proximal operators is implemented for optimisation. MALSAR's *Least_Dirty.m* function is used for optimisation of this method.

Again, the hyper-parameters λ_1 and λ_2 are optimised with 10-fold cross validation using the same method outlined in Algorithm 6.

3.1.7 Low Rank Implementation

The low rank approach assumes all models share a low rank subspace. The low rank approach implemented uses trace norm regularisation. The trace norm is the sum of the singular value decomposition of the model matrix and is well known to enforce low rank matrices. Ji and Ye (2000) present the accelerated gradient descent with proximal operator's method for solving this problem, Eq 3.12.

$$\min_{W} \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_F^2 + \lambda_1 ||W||_* \tag{3.12}$$

For a given hyper-parameter MALSAR's *Least_trace.m* function is used for implementation of this method's optimisation.

The implementation, again, chooses the hyper-parameter λ_1 using 10-fold cross validation outlined in Algorithm 5.

3.2 Toy Experiments

Before testing the multi-task learning implementations on real datasets the methods presented above are first tested on generated toy datasets, where the true models are known. This is done for proof of concept and preliminary analysis of methods.

Using toy datasets allows for the learned models to be directly compared to the true model which was used to generate the data. This can give an insight into the strengths and weaknesses of the methods. In this section the generated datasets with different properties are presented and the results lead to some hypotheses of which methods work best for each type of datasets. However, for real data sets, often the user/programmer will not necessarily know these properties, so this section is mainly to show proof of concept of the multi-task learning methods. Also looking at how different learning methods perform with different types of data gives a basis for using these multi-task learning techniques to infer possible attributes of real data sets, which would be useful information for tailoring a new multi-task learning method for a particular type of data set.

The outline of this section is as follows. First 2D models used to generate data sets are discussed and the learned models' accuracy is compared for each learning method implementation. The methods are then tested to see how well they deal with anomalies, and finally, how they perform when the tasks are negatively correlated. The models are extended to a multivariate feature space (10 features). A dataset is generated where the task models are from a distribution. Next, different model with tasks having shared sparsity and mixed sparsity are discussed.

3.2.1 2D Linear regression data sets

Similar Tasks

A toy data set of four similar tasks is generated from the following four regression functions $y = 2.5x + 3$, $y = 3x + 3$, $y = 3.5x + 3$ and $y = 3x + 3.5$. For each task's regression function, x is randomly sampled between 0 and 20 and y is calculated. This is done 100 times for each task to form 100 data points per task, with no noise. All real data has noise, so to emulate a real data set gaussian noise with mean 0 and variance 2 is added to the data points. An example of the resultant data set is shown in Fig 3.3.

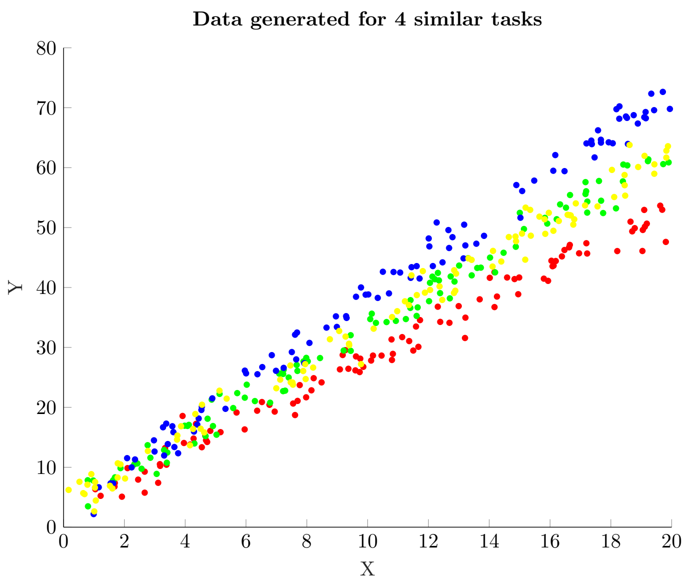


Figure 3.3: Plot of data points generated using the models $y = 2.5x + 3$, $y = 3x + 3$, $y = 3.5x + 3$ and $y = 3x + 3.5$, with noise added

This is a very simplistic multi-task learning regression case where all multi-task learning algorithms should perform well on.

The multi-task learning methods are each trained on the 100 data points generated for each task. The closeness of the learned model from each method with respect to the true model used to generate the data is shown in Table 3.1.

Without giving much insight, these results simply shows that all methods performing similarly, including single-task linear regression. This does however provide some confidence that the implemented methods work as intended and is useful for proof of concept.

Table 3.1: A table showing the results of the multi-task learning methods on the generated data from the similar task model

Similar Data	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Average**
Dirty	0.23	0.2477	0.2437	0.2733	0.248675
$L_{2,1}$	0.229	0.2443	0.2399	0.2724	0.2464
Lasso	0.2288	0.2443	0.24	0.2724	0.246375
Low Rank	0.2296	0.2451	0.2409	0.2732	0.2472
Reg Mean	0.2287	0.2442	0.24	0.2723	0.2463
STL	0.2289	0.2448	0.2401	0.2723	0.246525

Adding Anomalies

Outliers and anomalies can occur in real world data. This can be due to a variety of reasons, from a piece of measuring equipment failing, all the way to deliberate tampering of data. The data set above is amended to create a new toy data set by randomly adding 5 anomaly data points to each task's feature set with x taken from a random distribution and the label, y, set to zero, resulting in the data set shown in Fig 3.4.

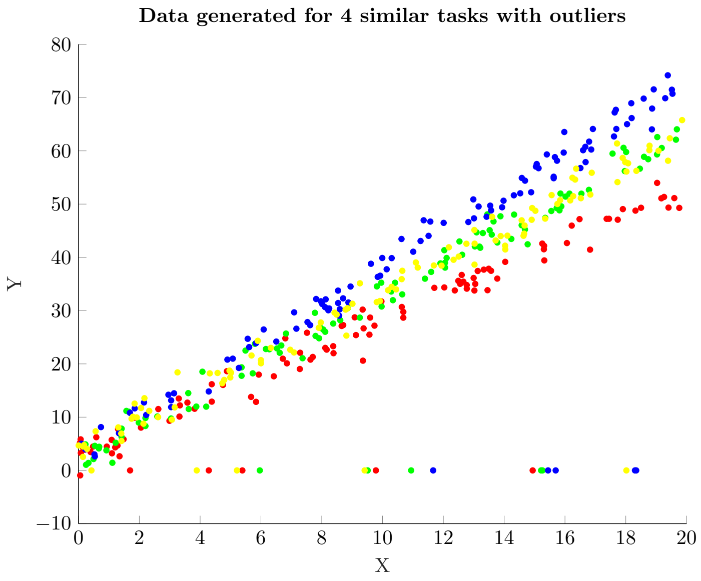


Figure 3.4: Plot of data points generated using the models $y = 2.5x + 3$, $y = 3x + 3$, $y = 3.5x + 3$ and $y = 3x + 3.5$, with noise and outliers added

The multi-task learning methods are each trained on the data points generated. The closeness of the models learned by each method in comparison to the true model which was used to generate the data points is shown in

Table 3.2. This shows all methods performing amicably with the introduction of anomalies, giving proof of concept that all methods can handle anomalies to some extent.

Table 3.2: A table showing the results of the multi-task learning methods on the generated data from the similar task model, with the addition of outlier data points

Anomalies	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Average**
dirty	0.008	0.0935	0.0943	0.1085	0.076075
$L_{2,1}$	0.0183	0.0813	0.082	0.0986	0.07005
Lasso	0.0183	0.0813	0.082	0.0986	0.07005
Low rank	0.0183	0.0813	0.082	0.0986	0.07005
Reg mean	0.0187	0.0813	0.0816	0.0986	0.07005
STL	0.0183	0.0788	0.0807	0.0986	0.0691

Negative Correlation

A toy data set of three tasks, one of which having a negative correlation, is created to asses the methods performance with negatively correlated task model's. The three regression tasks are defined by $y = 5x+2$, $y = -5x-2$ and $y = 4x - 2$. Again, Gaussian noise is added to the output from the regression functions giving the dataset shown in Fig 3.5.

The multi-task learning methods are each trained on the data points generated. The closeness of the models from each method to the true model is shown in Table 3.3.

Table 3.3: A table showing the results of the multi-task learning methods on the generated data from the negatively correlated task model

Neg Corr	**Task 1**	**Task 2**	**Task 3**	**Average**
dirty	0.0482	0.0816	0.1208	0.083533
$L_{2,1}$	0.0409	0.1015	0.1276	0.09
Lasso	0.0411	0.1014	0.1278	0.0901
Low rank	0.0412	0.1012	0.128	0.090133
Reg mean	0.0421	0.1004	0.1279	0.090133
STL	0.0425	0.1016	0.1258	0.089967

No method was particularly impeded by the introduction of a negatively correlated model. The dirty approach preforming the best. The dirty approach implemented, by decomposing the models, can better leverage negative correlation relationships between tasks, and should be able to leverage

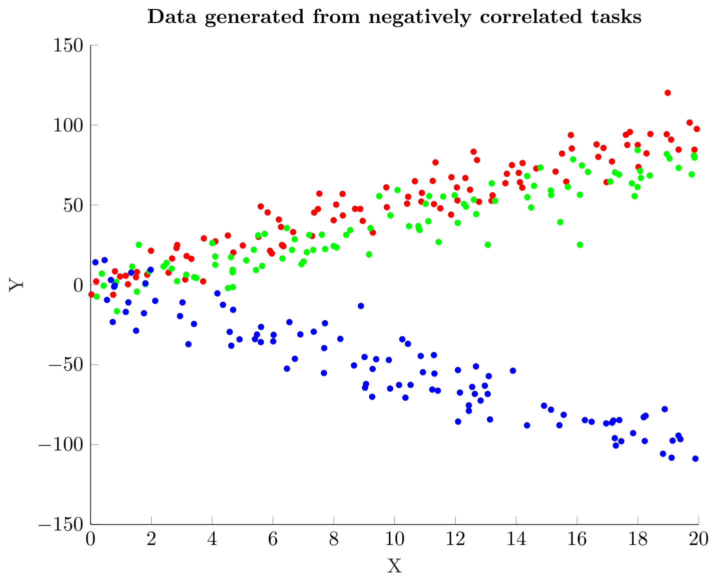


Figure 3.5: Plot of data points generated using the models $y = 5x + 2$, $y = -5x - 2$ and $y = 4x - 2$, with noise

more complex relationships between tasks than some of the other implemented methods.

3.2.2 Multivariate Linear Regression Data Sets

Extending the toy data sets above to multiple features we generalise the single-task regression function to $y = w.x$ where y is the label w is an n-dimensional vector, the model, and x is an n-dimensional vector, the features. This is the same as $Y_t = W_t.X_t$ where t is the task, and can be written in a multi-task learning domain as $Y = W.X$.

Similar Tasks

A dataset of related tasks is constructed from first randomly sampling a single task model, w, and then using this model with gaussian noise for each task, giving a set of similar models, W. W is used as the true models for the tasks. This is done for 6 tasks and the true model W is visually represented in Fig 3.6. 100 features are randomly sampled from a uniform distribution for each task and the tasks model's used to find the label, Y_t, i for each sampled feature, X_t, i, where t is the task and i is the data point for that task. Again, Gaussian noise is added to the data points as to emulate real data. This gives a toy data set for 6 similar 10-dimensional models.

The multi-task learning methods are each trained on this data set and the learned models are compared to the true models using absolute error. This

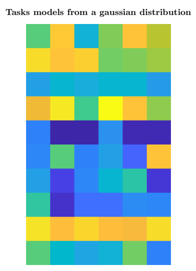


Figure 3.6: Visual representation of task's models from a Gaussian distribution - *with the Parula colour scale where dark blue is the minimum value and yellow the maximum* - each column represents a tasks model

can be seen in Table 3.4.

Table 3.4: A table showing each multi-task learning methods learned model's absolute error from the true model

Toy Data	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**	**Average**
dirty	0.019212	0.028845	0.036978	0.038714	0.039383	0.029309	0.032073
$L_{2,1}$	0.016091	0.023843	0.037763	0.035926	0.040259	0.024999	0.029814
Lasso	0.016078	0.023831	0.037763	0.035909	0.040259	0.024986	0.029804
Low rank	0.016098	0.023844	0.037763	0.035933	0.040259	0.025014	0.029819
Reg mean	0.05994	0.061487	0.040289	0.059465	0.047032	0.086301	0.059086
STL	0.023143	0.023398	0.040523	0.036307	0.040431	0.021485	0.030881

This multi-task leaning problem can be thought of as the tasks being from a gaussian distribution. Surprisingly, the regularised mean approach, virtually designed for this situation, performed the worst. The dirty approach also performed worst than the single task learning approach as it seemed to optimise particular tasks at the cost of others. The feature selection and low rank approaches all performed better than the single-task learning method.

Extending this experiment, each method is run on toy data sets generated in the same way described above, but with variable noise added to the data. The methods are run on datasets with noise variances of $5, 10, 15, \ldots, 50$, with the corresponding model's error plotted in Fig 3.7. Fig 3.7 shows a strong correlation of error with respect to the different methods, showing that all methods have similar performances with respect to varying gaussian noise in the data. The expected trend of performance decreasing as the noise of the data increases is also seen, without any method performing consistently better at higher noise data.

It is also interesting to compare the methods with varying relatedness of each tasks model. Each method is run on data generated using the ap-

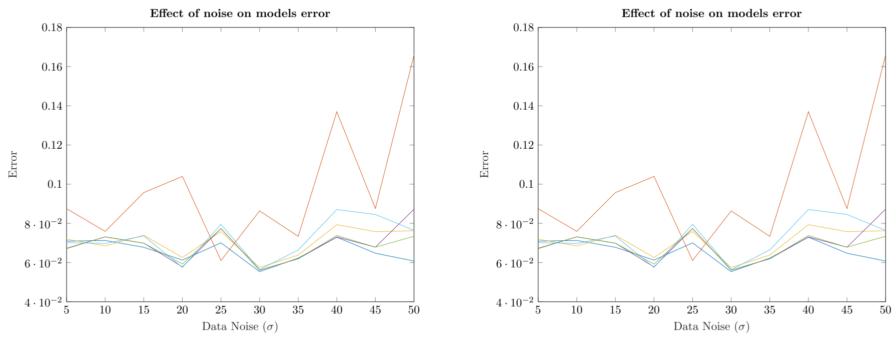


Figure 3.7: Change of model's error compared to the true model for each learning method as the noise introduced into the data varies

proach outlined above, changing the variance of the noise added to the generated model to form each task's true model. The values of variance used are $5, 10, 15, \ldots, 50$. This is shown in Fig 3.8. The dirty model's performance is most affected by the variance of the gaussian distribution the tasks are from, with occasional peaks, seemingly happening at random times. This indicates the implemented dirty approach is more volatile than the other methods.

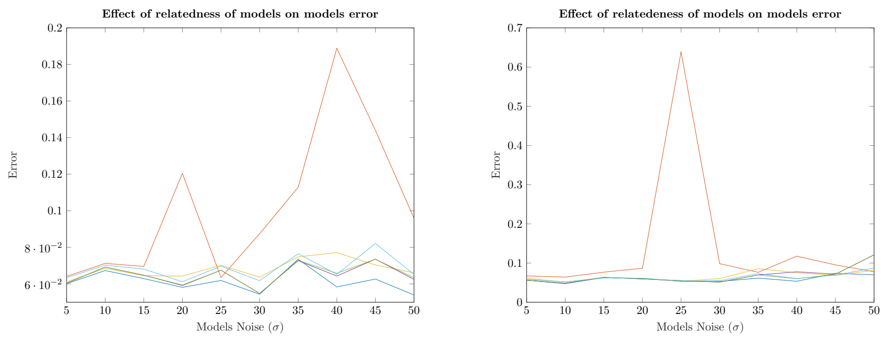
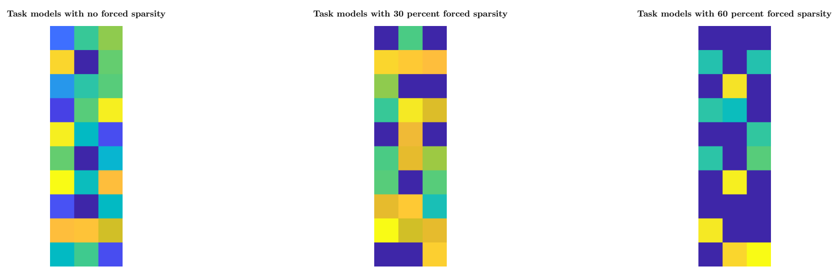


Figure 3.8: Change of model's error compared to the true model for each learning method as the variance of the distribution of models varies

3.2.3 Block Sparse Correlation

Methods, such as the feature select approaches and low rank approach implemented assume some shared sparsity between task models' and leverage this relationship. A toy dataset is generated where the true models share sparsity. To do this, each element of a model w is generated randomly between 5 and 10. This gives a model with 0% enforced sparsity. To form two model's where the 30% and 60% sparsity is enforced, 30% and 60% of the elements in w are set to zero respectively. This gives an approach for forming models of different sparsity. For each sparsity (0%, 30% and 60%), N models are generated this way, where N is the number of tasks, giving a model for all tasks W where all task's models share the same sparsity. An example of each of the models generated, with 0%, 30% and 60% enforced sparsity, are shown in Fig 3.9.



(a) No sparsity correlation in models (b) 30% sparsity correlation in models (c) 60% sparsity correlation in models

Figure 3.9: Visual representation of task models with all tasks sharing the same sparsity - *with the parula colour scale where dark blue is the minimum value and yellow the maximum* - each column represents a tasks model

For of the models in Fig 3.9, 100 data points per task are formed by taking random samples for X_t and using the task models to find the labels. Each data point is introduced to Gaussian noise. This gives three multivariate multi-task regression data sets, with the true models used to generate each dataset in Fig 3.9.

The multi-task learning implementations are each trained on the data sets generated and the closeness of the learned models compared to the true model is given in Table 3.5.

Table 3.5 shows different tasks performing better for different model sparsity's, however all models perform similarly overall, up until 60% sparsity where the dirty approach, Lasso approach and regularised mean approach perform much better than the others. At high sparsity, again, we see the dirty approach optimising a particular tasks model (task 2) at the cost of other tasks (task 1).

To look for any trends that occurring the approach above is implemented for all sparsity's 0%, 10%, 20%, ..., 90%. The results are shown in Fig 3.10 which again shows the dirty approach acting more unstable than the other learning methods. While the other methods perform similarly in comparison.

3.2.4 Mixed Sparsity

Using a similar method to that outlined above a mixed sparsity model is generated, with one task having 0% enforced sparsity, another task with 30% enforced sparsity, and the final task with 60% enforced sparsity. An example of this model is shown in Fig 3.11. This is used to see how the methods perform on data sets where the tasks have different sparsity.

Table 3.5: A set of tables showing the absolute error of each multi-task learning methods learned model compared to the true model, for three different sparsity's

0% Sparse	**Task 1**	**Task 2**	**Task 3**	**Averages**
dirty	2.027969	1.32702	1.656584	1.670524
$L_{2,1}$	1.99281	1.25074	1.52437	1.589307
Lasso	1.962745	1.266854	1.627966	1.619188
Low rank	1.994666	1.255504	1.526203	1.592124
Reg mean	1.964212	1.27212	1.634098	1.623477
STL	1.955549	1.196303	1.529747	1.560533

30% Sparse	**Task 1**	**Task 2**	**Task 3**	**Averages**
dirty	0.377784	0.3171	0.236542	0.310475
$L_{2,1}$	0.302258	0.289622	0.31157	0.30115
Lasso	0.302258	0.289621	0.31157	0.30115
Low rank	0.327354	0.320996	0.251121	0.299824
Reg mean	0.295416	0.260664	0.2788	0.278293
STL	0.329214	0.312529	0.249132	0.296958

60% Sparse	**Task 1**	**Task 2**	**Task 3**	**Averages**
dirty	0.271186	0.074347	0.152877	0.166137
L_{21}	0.346404	0.237526	0.28483	0.289587
Lasso	0.070295	0.15822	0.17776	0.135425
Low rank	0.355181	0.233099	0.281008	0.289763
Reg mean	0.073714	0.219144	0.199578	0.164145
STL	0.293309	0.266593	0.295694	0.285199

Again, this model is used to generate labels for 100 randomly sampled feature sets for each task, generating a multi-task data set.

The multi-task learning methods are each trained on the data points generated. The methods that enforce block sparsity, the feature selection approach using $L_{2,1}$ norm regularisation and dirty approach will likely encourage sparsity which isn't there in the low-sparsity task, and/or not encourage enough sparsity in the high-sparsity task. The results showing closeness of the models from each method to the true model is shown in Table 3.6.

This shows the dirty and $L_{2,1}$ norm approach performing the worst, as predicted above. The regularised mean approach performs the best, which makes sense as the regularised mean approach doesn't enforce any sparsity but just assumes the task are from a gaussian distribution. All multi-task learning methods, excluding the dirty approach, outperform the single-task learning method, showing that they are still able to leverage useful relation-

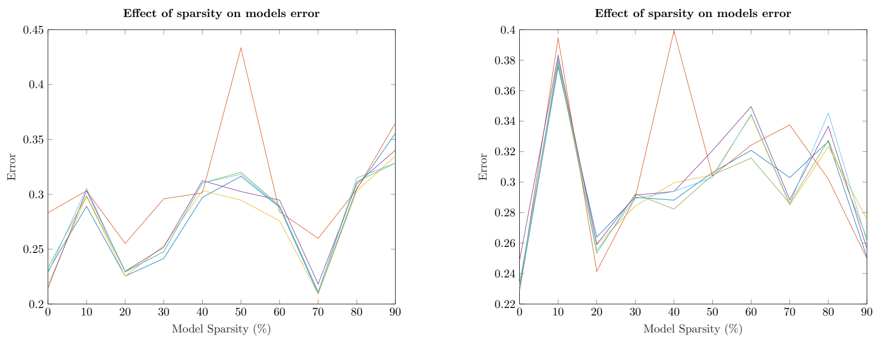


Figure 3.10: Change of model's error compared to the true model for each learning method as sparsity varies

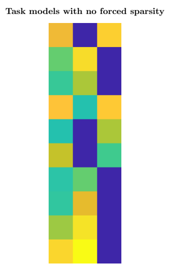


Figure 3.11: Visual representation of task models with each tasks having different sparsity - *with the parula colour scale where dark blue is the minimum value and yellow the maximum* - each column represents a tasks model

ships between the tasks with this data set.

3.3 Data sets

The toy datasets discussed above are useful for gaining insights into how different types of data effects the performance of the implemented multi-task learning methods. However, usually the properties of the data are unknown and the relationships between data sets can be much more complex than the relationships of the toy data models. In this dissertation eight real datasets are used for an empirical analysis of the performance for each multi-task learning implementation. This section introduces and describes the datasets used for the analysis. The data sets described in this section are as follows: the Chinese Air Pollution data set, the Chinese Rainfall data set, the Parkinson's Telemonitoring data set, the SARCOS data set, the Student Performance data set, and the School data set.

Table 3.6: A table showing the absolute error of each multi-task learning methods learned model compared to the true model, for a model where each task has different sparsity

0%, 30%, 60% Sparse Mix	**Task 1**	**Task 2**	**Task 3**	**Averages**
dirty	0.280704	0.496404	0.259961	0.34569
L_{21}	0.349465	0.352941	0.176137	0.292848
Lasso	0.360594	0.34824	0.11047	0.273101
Low rank	0.360595	0.348241	0.110471	0.273102
Reg mean	0.345232	0.266298	0.10121	0.23758
STL	0.35759	0.354117	0.202814	0.304841

3.3.1 Chinese Air Pollution

The first datasets used for the empirical analysis discussed, which is referred to in this dissertation as the Chinese Air Pollution dataset (Liang et al, 2016), contains meteorological data from five cities in china, Beijing, Chengdu, Guangzhou, Shanghai and Shenyang. For each city in this dataset there also contains multiple measurements of PM2.5 from various locations and organisations. Specifically, for each city there exists PM2.5 measurements from the local US Post, as well as measurements from different regions within the city. The findings in(Liang et al, 2016) show that the measurements of PM2.5 from the Chinese and US are consistent. This consistency in the data means it can form multiple tasks effectively. This dataset can fit into a multi-task learning framework in two ways.

The first way is to determine each city as a separate task and use the US Post measurements of PM2.5 for that city as the label for the meteorological data. Clearly each city will have different environmental factors that may affect the data, but logically all cities (tasks) models will share some relationships which can be leveraged. The dataset in this form will be referred to as the PM2.5 data of cities.

Another way is to consider the PM2.5 measurement for each region within a city and conjoin each regions PM2.5 measurement as the label for the relevant city's meteorological data. This approach will have clusters of very similar tasks for the regions in each city, as well as presumably less strong relationships between. The dataset in this format will be referred to as the PM2.5 data of regions. The features and labels for this dataset are shown in Table 3.7.

For this dataset any data points with a NaN (unmeasured) value is disregarded. This means all data points in this dataset are complete, i.e. without any missing features in any of the data points. After removal of any unwanted data points, the number of data points for each city in the PM2.5 data of cities dataset is shown in Table 3.8 and the number of data points for each region

Table 3.7: A table showing the features and labels in the Chinese Air Pollution data-set

Features
year: year of data in this row
month: month of data in this row
day: day of data in this row
hour: hour of data in this row
season: season of data in this row
DEWP: Dew Point (Celsius Degree)
TEMP: Temperature (Celsius Degree)
HUMI: Humidity (%)
PRES: Pressure (hPa)
cbwd: Combined wind direction
Iws: Cumulated wind speed (m/s)
Iprec: Cumulated precipitation (mm)
precipitation: hourly precipitation (mm)
Label
PM2.5 measurement for tasks city/region

in the PM2.5 data of regions dataset is shown in Table 3.9.

Table 3.8: A table showing the data points for each task in the PM2.5 data of cities

Beijing	**Chengdu**	**Guangzhou**	**Shanghai**	**Shenyang**
49579	23389	24931	19335	20505

For linear regression in 8-dimensional feature space, each task has a large number of data points. The benefits of multi-task learning will be more visible on data sets where each task doesn't have so many data points, as to heighten the impact of leveraging the data points from other tasks.

Logically, we can predict some attributes of the relationships between tasks. Different cities, implies similar models, however, not necessarily that the tasks share any common feature. The task would logically be from some distribution, but not necessarily gaussian.

3.3.2 Chinese Rainfall

The Chinese Air Pollution dataset above (ignoring the PM2.5 measurements) can be used to model the hourly precipitation from meteorological data. For

Table 3.9: A table showing the data points for each task in the PM2.5 data of regions

Dongsi	**Dongsihuan**	**Nongzhanguan**	**Caotangsi**	**Shahepu**
24237	20166	24137	22997	23142

Guangzhou station	**Guangzhou school**	**Jingan**	**Xuhui**	**Taiyuanjie**	**Xiaoheyan**
21095	32351	22687	23128	24384	24623

use in the multi-task learning setting, the data from each city is considered as separate tasks, thereby forming a dataset of five tasks. This modified version of the Chinese Air Pollution dataset is referred to in this dissertation as the Chinese Rainfall dataset. The features and labels for this dataset are defined in Table 3.10.

Table 3.10: A table showing the features and labels in the Chinese Weather data-set

Features
year: year of data in this row
month: month of data in this row
day: day of data in this row
hour: hour of data in this row
season: season of data in this row
DEWP: Dew Point (Celsius Degree)
TEMP: Temperature (Celsius Degree)
HUMI: Humidity (%)
PRES: Pressure (hPa)
cbwd: Combined wind direction
Iws: Cumulated wind speed (m/s)
Iprec: Cumulated precipitation (mm)
Label
precipitation: hourly precipitation (mm)

Again, any data points with a NaN (unmeasured) value for any of the features are disregarded meaning all data points are complete without any missing measurements.

Similar inferences can be made about this dataset as made about the Chinese Air Pollution data set, with the addition that not all features will really have any impact the rainfall. This would infer that a method which encourages a block sparse model will perform better for this data set.

3.3.3 Parkinson's Telemonitoring

The Parkinson's telemonitoring dataset (Tsanas, 2009) contains data about persons with Parkinson's, with the features and labels shown in Table 3.11. In a machine learning context, measurements about the persons voice, such as jitter and shimmer, are used to model the persons UPDRS score. The dataset contains 5875 data points (around 200 per patient) over time for 42 people with Parkinson's. In the multi-task learning framework, the data for each person can be considered as a separate task, giving a 42-task learning problem.

Table 3.11: A table showing the features and labels in the Parkinson's telemonitoring data-set

Features
Persons age
Persons gender (0/1)
Time since recruitment
Jitter(%)
Jitter(Abs)
Jitter:RAP
Jitter:PPQ5
Jitter:DDP
Shimmer(dB)
Shimmer:APQ3
Shimmer:APQ5
Shimmer:APQ11
Shimmer:DDA
NHR: A measures of ratio of noise to tonal components in the voice
HNR: A measures of ratio of noise to tonal components in the voice
RPDE: A nonlinear dynamical complexity measure
DFA: Signal fractal scaling exponent
PPE: A nonlinear measure of fundamental frequency variation
label
total UPDRS: Clinician's total UPDRS score, linearly interpolated

Here, voice shimmer refers to several measures of variation of amplitude, and voice jitter attributes to several measures of variation in the fundamental frequency. The data label, UPRDS score, is the persons unified Parkinson's disease rating scale. To obtain this score, the patient needs to have an evaluation of behaviour's and perform a self-evaluation of daily activities, as well as clinician-scored monitored motor evaluation. A good set of models able to obtain UPRDS scores from voice measurements alone would lessen the need for the current method of scoring UPDRS, which is time consuming.

The 42 tasks all have a similar number of data points (between 101 to 168). For a 16-dimensional feature space there is a lack of data points per task, which is where multi-task learning is particularly useful. However the 16-dimensional feature space is made up of features which will be very highly correlated, i.e. all the shimmer measurements will be highly correlated as well as all the jitter measurements.

3.3.4 SARCOS

SARCOS refers to a robotic arm with 7 degrees of freedom. The SARCOS dataset can be used to model inverse dynamics of the robot arm (Vijayakumar and Schaal, 2000; Vijayakumar et al, 2002; Vijayakumar et al, 2005) , that is, to find the torque values for each of the 7 joints from the position, acceleration and velocity at each joint. In a multi-task learning domain, learning the torque for each joint can be considered as a separate task, giving a 7-task regression problem. The features and labels are shown in Table 3.12.

Table 3.12: A table showing the features and labels in the SARCOS data-set

Features
Joint 1 position
⋮
Joint 7 position
⋮
Joint 7 velocity
Joint 1 acceleration
⋮
Joint 7 acceleration
Labels
Joint 1 torque
⋮
Joint 7 torque

For this multi-task learning problem the feature set for each task is the same but with different labels. The SARCOS dataset has a 21-dimensional feature space with each task having the the number of data points shown in Table 3.13. This is a relatively large number of data points for a 21-dimensional feature space.

Table 3.13: A table showing the data points for each task in the SARCOS data set

Joint 1 Torque	**Joint 2 Torque**	**Joint 3 Torque**	**Joint 4 Torque**	**Joint 5 Torque**	**Joint 6 Torque**	**Joint 7 Torque**
48933	48933	48933	48933	48933	48933	48933

3.3.5 Student Performance

The student performance dataset contains demography attributes about students at two different Portuguese schools, as well as three exam grades (first year, second year, and third year). the features and labels for this data set is shown in Table 3.14.

This data set is split into a multi-task learning problem in two ways, described below.

Firstly, the data is be separated with each school being a separate task and simply using the *third year grade* as the label with *first year grade* and *second year grade* as part of the features. The dataset in this format will be referred to as the Performance per schools dataset. This is a 2-task regression leaning problem. Logically, there will be key factors in every students demography that will particularly effect their grade. For example, students that perform well in the first two exams will likely perform well in the final exam, regardless of their school. This implies a shared low rank representation between the schools.

The other way to modify the data into a multi-task learning problem is for the two schools' data to be combined and each of the three exams to be considered as a separate task. For this structure each label (exam results) uses the same feature set. This gives a 3-task regression problem, with each exam as a task, which will be referred to as Performance per Exam database. This dataset, unlike it's counterpart, doesn't necessary imply a shared low rank representation. We can logically assume however that demography factors that don't effect the results of one exam won't effect the results of the other exams. This indicates a share block sparsity in each tasks model. Methods that leverage this block sparsity will likely perform the best for this data set.

3.3.6 School Data-set

Like the student performance dataset, the school dataset contains demography attributes of 15362 students from 139 schools which are labelled with exam score. The aim being to use these attributes to predict exam results of students. Each school is a separate task, 139 tasks. The mean number of data points (students) per school is 111 for a 28-dimension feature space dataset.

Table 3.14: A table showing the features and labels in the Student performance data-set

Features
Student's school
student's sex
student's age (numeric: from 15 to 22)
student's home address type
family size
parent's cohabitation status
mother's education level
father's education level
mother's job type
father's job type
reason to choose this school
student's guardian
home to school travel time
weekly study time
number of past class failures
extra educational support status
family educational support status
extra paid classes
extra-curricular activities
attended nursery school
wants to take higher education
Internet access at home
In a romantic relationship
quality of family relationships
free time after school
going out with friends
workday alcohol consumption
weekend alcohol consumption
current health status
number of school absences
Labels
First year grade
Second year grade
Third year grade

3.4 Experimental Set-up

To test each method on the various datasets an experimental setting is as follows. The data set, method name, and number of fold for k-fold cross validation is input into Algorithm 5 or Algorithm 6 depending on the number of hyper-parameters, with the addition that for every iteration of this algorithm, the previously learned model is used as a starting point. This optimises the hyper-parameters which are then used to learn the model for the whole training set. The model is then used with the test data to find the mean squared error for each model, using Eq 2.19. This multi-task learning experimental framework is shown diagrammatically in Fig 3.12.

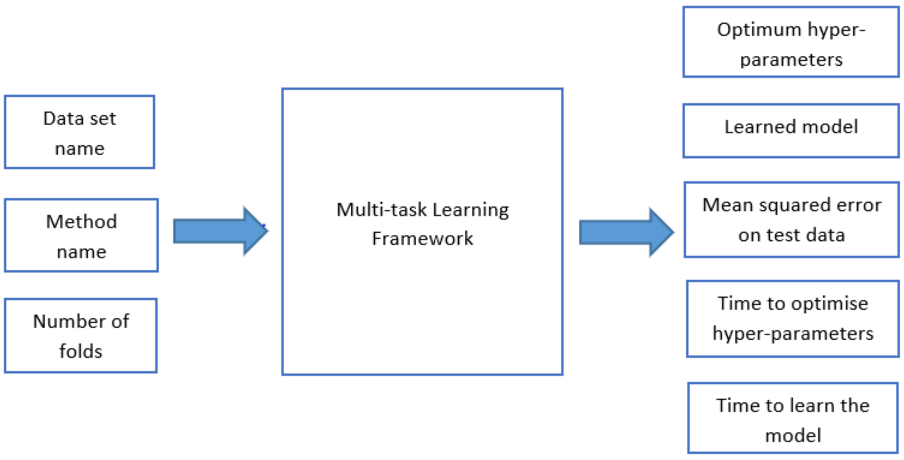


Figure 3.12: high-level black box diagram of the multi-task learning framework implemented to conduct the experiments in this dissertation

This is done for each multi-task learning method with each data set and the results are presented in the results section.

Chapter 4

Results

Multi-task learning sets out to generalise each task's model by using data from other related tasks. This is useful in situations where all the tasks are related, and particularly when each task or a sub-set of tasks are limited in the amount of labelled data available. Multi-task learning, generalises the models, using data from other tasks. This is, of course, only useful when there exists noise in the data, else single task learning can learn the exact models without any regularisation. All real-world data used in machine learning, however, does include some noise.

This chapter presents and discusses the results of the multi-task learning methods performances on real multi-task data sets, as introduced in Chapter 3. As well as comparing multi-task learning methods, it's important to determine multi-task learnings usefulness over single-task learning. For this reason, single-task least squared regression with ridge regularisation is included in the comparisons and discussion in this chapter.

For each multi-task learning implementation presented in Chapter 3, the hyper-parameters are optimised with 10-fold cross validation on a training set, using Algorithm 5 or Algorithm 6. The chosen hyper-parameters from cross-validation are used to learn the model for the whole training set and the means squared error is calculated on the test set. The error relates to how well the leaned models perform on new (unseen) data. This is discussed in the first section of this chapter. The time complexity of learning the model can also be a key factor in deciding which method to employ. This chapter further moves on to show and discuss the time taken to optimise the hyper-parameters, as well as the time taken to learn the model with the chosen hyper-parameters. Finally the effect of changing the hyper-parameters on the error for each function is analysed.

4.1 Error

This section first looks at the mean squared error (Eq 4.1) of each multi-task learning implementation, trained on a training set of 90% of the data, and tested on a testing set of the other 10% of the data. Moving on to look at a 50% / 50% split of training and test data.

Table 4.1 shows the mean squared error for the implementations on each data set. Table 4.2 re-represents this data in terms of percentage worse with respect to the best method for the respective data set.

$$Err = \sum_{t=1}^{T} \sum_{i=1}^{m \in t} Y_{t,i} - W_t.X_{t,i} \tag{4.1}$$

Table 4.1: A table showing mean squared error for learned models on a 90% / 10% split of training / testing data

Error	Dirty	L21	Lasso	Low Rank	regularised	STL
City PM	7783.23	7458.31	7458.3	7455.24	7457.51	7462.6
Region PM	3820.8	3796.43	3796.45	3795.32	3796.12	3765.2
China Weather	2.79	0.84	0.84	0.85	0.85	0.84
Parkinson's	6.76	6.3	6.4	6.41	6.64	6.4
SARCOS	32.51	11.9	11.9	11.9	11.9	11.85
School Data	110.44	106.17	105.33	106.24	104.36	105.77
Performace Exams	14.65	14.6	14.35	14.57	14.7	14.55
Performance School	7.35	7.21	7.91	6.72	8.86	7.68

No method is a stand out performer for all data sets. The Lasso approach, the Low rank approach and the single-task learning approach all performed the best for 2 data sets. The L21 and Mean regularised approach both performing best for 1 data set each. The Dirty approach not performing the best for any data set and in fact performing the worst for 6 of the 8 data sets.

Table 4.2 shows that the Low rank approach performed consistently well and the *average* column shows that the Low Rank approach performed the best overall. The Lasso approach performed consistently well for 7 of the data sets, however performed considerably worse than the Low Rank approach for the performance schools data set. The L21 norm approach, performed similar to the other feature selection approach, the Lasso approach, whereby its performance was consistently good, apart from the performance schools. The L21 approach has a better average performance that the Lasso approach. The regularised approach, more inconsistent in performance, gave good results for half of the data sets, with an exceptionally bad performance on the performance schools data set in comparison to the low rank approach. The dirty

Table 4.2: A table showing percentage difference of mean squared error for learned models on a 90% / 10% split of training / testing data, in each case respective to the best performed method for each data set

Error %	Dirty	L21	Lasso	Low Rank	regularised	STL
City PM	4.40	0.04	0.04	0.00	0.03	0.10
Region PM	1.48	0.83	0.83	0.80	0.82	0.00
China Weather	233.39	0.99	0.00	1.62	2.29	0.85
Parkinsons	7.51	0.00	1.68	1.78	5.43	1.56
SARCOS	174.46	0.49	0.49	0.49	0.49	0.00
School Data	5.83	1.74	0.93	1.80	0.00	1.35
Performace Exams	2.07	1.69	0.00	1.49	2.43	1.37
Performance School	9.29	7.27	17.63	0.00	31.78	14.29
Average	54.80	1.63	2.70	1.00	5.41	2.44

approach performed consistently bad for all data sets, with the worst average performance.

Based off the results from this experimental set-up the best to worst ranking of methods is shown in Table 4.3.

Table 4.3: A table showing the ranking of implemented methods

Best
Low Rank Approach
$L_{2,1}$ norm Approach
Single-task learning Approach
Lasso Approach
Mean regularised Approach
Dirty Approach
Worst

As previously stated, the aim of multi-task learning isn't to fit a model more precisely to data, but to generalise the model to perform better for new data. The above experiment is done with a 90% / 10% split of training / test data. If more new data was to be added we may find a shift in which model performs the best, which will be the model that has generalised the data the best. Table 4.4 shows the results for the experiment above repeated, but with a 50% / 50% split of training and testing data.

Table 4.5 shows that the dirty approach, relative to the other methods, has improved its performance for 5 of the data sets, with the best performance of all methods for the performance schools data set. The dirty approach is

Table 4.4: A table showing mean squared error for learned models on a 50% / 50% split of training / testing data

Error	Dirty	L21	Lasso	Low Rank	regularised	STL
City PM	8286.71	7753.14	7753.15	7751.52	7752.69	7755.67
Region PM	3800.87	3790.27	3790.32	3821.55	3817.31	3821.53
China Weather	7.68	1.02	1.07	1.03	1.20	1.04
Parkinsons	6.47	6.03	6.04	6.02	6.32	6.00
SARCOS	35.66	13.01	13.07	13.23	13.23	13.17
School Data	128.35	137.57	133.82	127.43	127.57	131.35
Performace Exams	14.49	14.29	14.29	14.37	14.41	14.27
Performance School	4.70	11.02	11.28	9.27	10.75	6.10

Table 4.5: A table showing percentage difference of mean squared error for learned models on a 50% / 50% split of training / testing data, in each case respective to the best performed method for each data set

Error %	Dirty	L21	Lasso	Low Rank	regularised	STL
City PM	6.90	0.02	0.02	0.00	0.02	0.05
Region PM	0.28	0.00	0.00	0.83	0.71	0.82
China Weather	654.64	0.00	5.35	1.05	17.58	2.58
Parkinsons	7.80	0.40	0.57	0.21	5.34	0.00
SARCOS	174.06	0.00	0.43	1.69	1.66	1.18
School Data	0.72	7.96	5.01	0.00	0.11	3.08
Performace Exams	1.57	0.17	0.17	0.69	1.01	0.00
Performance School	0.00	134.66	140.29	97.48	128.86	30.00
Average	105.75	17.90	18.98	12.74	19.41	4.71

still, however, the clear overall loser, performing the worst for 5 data sets. The single task learning method showed the best performance in 2 data sets, with the best average performance. The Low rank approach still performing well and the best in 2 data sets again. The L21 approach performing the best in 3 datasets and one of the best in 3 other data sets. The Lasso and mean regularised approach performing similarly. The gap between the better performing methods and the mean regularised approach has closed, which shows that the mean regularised method generalises a lot. The ranking of these methods for this experiment is shown in Table 4.6.

The ranking of the multi-task methods (disregarding the single-task learning method) is consistent in the 90%/10% (Table 4.2) and 50%/50% (Table 4.5) training/testing data split.

Table 4.6: A table showing the ranking of implemented methods

Best
Single-task learning Approach
Low Rank Approach
$L_{2,1}$ norm Approach
Lasso Approach
Mean regularised Approach
Dirty Approach
Worst

4.1.1 Summary

In this section the performance based on mean squared error for each method has been analysed, leading to a ranking of the implementations. This has only, however, viewed the overall performance of each method on all the tasks. This means that some of the methods implemented that performed badly above may be optimising the models for certain tasks while performing particularly badly few, leading to an overall bad performance. In the case of some easy tasks to learn, with one particularly hard task, we will see the single-task learning approach performing better overall. This could explain the poor performance of the dirty approach, as well as the good average performance of the single-task learning approach.

4.2 Time performance

Moving on to present the results for time complexity of each implementation with respect to optimising the model for a given set of hyper-parameters, as well as for optimising the hyper-parameters using the 10-fold cross validation algorithm outlined in Algorithm 5 and 5. Every time the model is optimised for a set of hyper-parameters, the last learned model is fed in as a starting point. This is done in an attempt to decrease the time complexity of hyper-parameter optimisation.

Table 4.7 shows the results of optimising the model with given hyper- parameters for each implementation. Single task learning uses a closed form solution which is much faster than using a gradient step method and therefore is consistently the fastest. Disregarding the single-task learning approach (Table 4.8), the Lasso approach was the fasted for half the data sets as well as the fastest averaged over all data sets. The regularised mean approach is the second fastest to optimise the model averaged over all datasets, consistently fast for all data sets. The Low Rank approach and L21 approach are also relatively quick with slightly more variance across data sets than the mean regularised approach. The dirty approach being one of the slowest for all apart from 2 where it was the fastest.

Table 4.7: A table showing the time to optimise for a set of defined hyper-parameters

Time to learn x100	Dirty	L21	Lasso	Low Rank	regularised	STL
City PM	46.60	5.44	4.97	5.70	5.00	0.23
Region PM	106.59	7.16	6.47	6.76	6.69	0.31
China Weather	64.35	5.12	4.93	6.74	5.24	0.36
Parkinsons	7.47	0.60	0.66	0.94	0.77	0.21
SARCOS	167.72	74.63	73.16	76.16	73.53	1.24
School Data	60.97	436.83	115.95	530.97	227.87	1.40
Performace Exams	2.02	5.65	2.83	2.53	4.77	0.07
Performance School	0.75	0.66	0.72	1.13	1.92	0.03

Table 4.8: A table showing the percentage slower each multi-task learning method is to the fastest method for each data set

Time to learn %	Dirty	L21	Lasso	Low Rank	regularised
City PM	838.08	9.56	0.00	14.74	0.73
Region PM	1547.07	10.64	0.00	4.43	3.30
China Weather	1206.30	4.00	0.00	36.84	6.46
Parkinsons	1151.11	0.00	0.10	56.74	28.61
SARCOS	129.26	2.01	0.00	4.10	0.51
School Data	0.00	616.51	90.18	770.92	273.75
Performace Exams	0.00	179.96	40.09	25.35	136.66
Performance School	13.86	0.00	9.92	71.60	191.20
Average	610.71	102.83	17.54	123.09	80.15

From these tables, the multi-task learning implementations are ranked, shown in Table 4.9. The methods are rated based on their average shown in Table 4.8.

Table 4.9: A table showing the ranking of implemented methods

Best
Lasso Approach
Mean regularised Approach
$L_{2,1}$ norm Approach
Low Rank Approach
Dirty Approach
Worst

If the hyper-parameters are known then Table 4.7 gives a representation of which method will likely learn the model fastest. Usually the hyper-parameter needs to be optimised. Table 4.10 shows the time taken to optimise these hyper-parameters using the 10-fold cross validation algorithms Algorithm 5 and Algorithm 6 depending on the number of hyper-parameters to optimise.

Table 4.10: A table showing the time taken to optimise the hyper-parameters using cross validation

Time to optimise	Dirty	L21	Lasso	Low Rank	regularised	STL
City PM	870.12	133.56	130.40	12.83	133.21	2.86
Region PM	25417.45	186.91	184.65	17.18	185.42	4.11
China Weather	1813.15	139.74	190.32	14.18	140.29	2.86
Parkinsons	158.18	27.68	48.49	4.94	35.76	0.77
SARCOS	10276.19	2770.52	2776.24	292.26	2827.11	18.34
School Data	1278.23	3742.82	3639.50	421.57	3442.36	4.64
Performace Exams	43.30	45.05	27.46	2.89	35.01	0.22
Performance School	24.25	33.31	18.27	3.53	53.16	0.35

The clear winner is the Low Rank Approach which runs in the least time for all data sets. The L21, Lasso and mean regularised approaches all run in similar times for each of the data sets. The dirty approach is the clear loser, taking the most time to run in the case of 5 data sets and averaging 22,000% more running time than the Low Rank Approach.

The Low Rank only needs to optimise one hyper parameter, which explains why its much faster than the other approaches, as the other methods all have to optimise for two hyper-parameters. The dirty approach has to optimise for two hyper parameters, but two of the regularisation terms are non-differentiable meaning the optimisation function needs to use proximal operators for two terms, where as all other methods only have one non-differentiable term. This goes some way to explain why the dirty approach is so much slower in com-

Table 4.11: A table showing the percentage slower each multi-task learning method is to the fastest method for each data set

Time to optimise %	Dirty	L21	Lasso	Low Rank	regularised
City PM	6680.70	940.83	916.16	0.00	938.12
Region PM	147850.50	987.97	974.79	0.00	979.28
China Weather	12684.31	885.28	1241.96	0.00	889.19
Parkinsons	3104.03	460.77	882.24	0.00	624.32
SARCOS	3416.17	847.98	849.94	0.00	867.34
School Data	203.21	787.83	763.32	0.00	716.56
Performace Exams	1397.60	1457.98	849.82	0.00	1110.80
Performance School	587.50	844.33	417.84	0.00	1406.96
Average	21990.50	901.62	862.01	0.00	941.57

Table 4.12: A table showing the ranking of implemented methods

Best
Low Rank Approach
Lasso Approach
$L_{2,1}$ norm Approach
Mean regularised Approach
Dirty Approach
Worst

parison.

4.3 Hyper-Parameters Effect on Error

In this section, the effect of changing the hyper-parameters in each multi-task learning method's loss function, with respect to the mean squared error calculated with 10-fold cross validation, is analysed. Each method is discussed individually for the different data sets, with a summary section at the end linking the inferences made from these results to the performance results at the beginning of this chapter. The figures referred to in this section mainly reside in the appendix, and will be referenced as such. In each case, the each figure consists of two sub-figures. The sub-figure on the left shows the results for the first iteration (hyper-parameter search space) in Algorithm 5/ 6, and the sub-figure on the right shows the results of the refined search space, more precisely around the minimum, i.e., the second iteration in Algorithm 5/ 6.

4.3.1 Low Rank

$$\min_{W} \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_F^2 + \lambda_1 ||W||_* \tag{4.2}$$

The hyper-parameter λ_1 has virtually no effect for the city PM2.5, region PM2.5 and the Chinese weather data sets, Figures A.25, A.26 and A.27, respectively. This could indicate that there is no shared low rank relationship to be exploited, or that the optimum hyper-parameters lie outside the search space used ($[1e^{-5}...1e^{5}]$).

The other data sets are consistent that the hyper-parameter, λ_1, performs better for values less than $1e^3$. The optimal λ_1 values range from around $1e^{-4}$ to 100 (Figures A.28, A.29, A.30, A.31, A.32).The minimum is quite pronounced in the refined search space for each of these data sets, which means the regularisation parameter is having a strong impact on the error.

4.3.2 $L_{2,1}$

$$\min_{W} \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_F^2 + \lambda_1 ||W||_{2,1} + \lambda_2 ||W||_F^2 \tag{4.3}$$

Like with the low-rank method discussed above, the hyper-parameters in the L21 approach barely effects the error for the data sets in Figures A.17 and A.18, the city PM2.5 and regional PM2.5 data sets. This in conjuncture with the results of the low rank approach would indicate there is no shared low rank representation or even shared sparsity between the tasks in these two data sets.

In Figure A.19 (the Chinese weather dataset), λ_2 shows little effect on the error, while λ_1 shows optimal performance at lower values ($1e^{-5}$).

For the pakinson's, SARCOS, school data, performance exams and performance schools data sets shown respectfully in Figures A.20, A.21, A.22, A.23, A.24, the error is relatively static for λ_1 below $1e^2$ and λ_2 below $1e^2$. At lower hyper-parameter values λ_1 has more of an effect on error than λ_1, which can be seen in the refined hyper-parameter search space surface plots in the aforementioned figures.

4.3.3 Lasso Approach

$$\min_W \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_1 + \lambda_1 ||W||_{2,1} + \lambda_2 ||W||_F^2 \tag{4.4}$$

For the lasso approach, again, for the city PM2.5 and region PM2.5 datasets, seen in Figures A.9 and A.10 respectively, shows the hyper-parameters have little effect on the error.

All other data sets, the Chinese weather, parkinson's, SARCOS, school data, performance exams and performance schools data sets perform worse with one or both the hyper parameters above $1e^2$. This can be seen in Figures A.11, A.12, A.13, A.14 , A.15 and A.16, respectively. We can see from the refined hyper-parameter search space for the parkinson's, school data, performance exams and performance schools data sets that λ_2 has the dominant effect on error. The SARCOS dataset shows neither λ_1 or λ_2 having much effect on the error around the optimum hyper-parameter values.

4.3.4 Mean Regularised Approach

$$\min_W \{ \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_F^2 + \lambda_1 ||W.R||_F^2 + \lambda_2 ||W||_1 \tag{4.5}$$

λ_1 in the mean regularised approaches loss function, Eq 4.5, is responsible for the weight of the term that regularises the variance of tasks means. The bigger λ_1, the smaller the difference between the learned models for each task. λ_2, the hyper-parameter weighting the $L_{1,1}$ norm, encourages sparsity in the model.

Figures A.33 and A.34, referring to the error for hyper-parameter values on the city PM2.5 and region PM2.5 data sets, respectively. These show the change in λ_1 and/or λ_2 has very little impact on the error, with the smallest hyper-parameters giving slightly better performance. It can be inferred from this that the task are unlikely from a Gaussian distribution.

For the Chinese weather data set, shown in Figure A.35, the error is unaffected by any change in hyper-parameters, up until $\lambda_1 = 1e^5$, where the error increases.

Figures A.36, A.38, A.39 and A.40, the parkinson's, school data, performance exams and performance schools data sets, show the error relatively static for λ_1 below $1e^4$, for all λ_2 values.

The effect of each hyper parameter at the more refined search space around the optimal hyper-parameters varies from each of these data sets. The parkinson's and performance schools data sets show a similar weight of effect from change in both hyper-parameters, while the performance exams data sets error is mainly only effected by change in λ_1, and the school data mainly effected by change in λ_2.

4.3.5 Dirty Approach

$$\min_{W,P,Q} \sum_{i=1}^{t} ||W_i'.X_i - Y_i||_F^2 + \lambda_1 ||P||_{1,\infty} + \lambda_2 ||Q||_1, s.tW = P + Q \tag{4.6}$$

The hyper-parameters, λ_1 and λ_2, in the dirty approaches loss function Eq 4.6, leverage different relationships in the data. The $L_{1,1}$ norm regularisation parameter (λ_2) encourages element-wise sparsity between the tasks, while the $L_{1,\infty}$ norm regularisation parameter (λ_2) encourages block-wise sparsity.

On the City PM2.5 and Chinese weather datasets, Figure A.1 and Figure A.3 respectively, the best performance is when one or both of λ_1 and λ_2 are greater than $1e^4$.

Figure A.2, the dirty approach performs best at high values λ_1, with λ_2 showing little effect within the hyper-parameter search space used.

On the contrary, lower hyper-parameters give better performance on the parkinson's, school data, performance exams and performance schools data sets, Figures A.4, A.6, A.7 and A.8, respectively. Both λ_1, and λ_2 have an effect on the error for these data sets as only when both values are high does the error rapidly increase. However, the refined hyper-parameter search space around the minimum shows the main effect of error to be from change in λ_2. This implies theses data sets models are element-wise sparse, but not particularly block-wise sparse.

The error on the SARCOS dataset, Figure A.5, is seen to be worse when any one of λ_1 or λ_2 is high, however, it performs best when both hyper-parameters are high, i.e. $1e^5$. This is where the balance of hyper-parameters works well.

The figures discussed in this section lead to the realisation, particularly based off Figures A.1, A.2 and A.3, that the optimum hyper-parameters may have fallen outside the hyper-parameter search space used in this experimental setting. This could account for the poor performance of the dirty approach compared to the other methods seen earlier in this chapter.

4.4 Optimum Hyper-parameter values effect on Performance

Each multi-task learning method aims to leverage different types of relationships between task than each other, as discussed in Chapter 2. In the sections above the effect of the hyper-parameter in each method on the error has been discussed for each data set. The higher the optimum hyper-parameter learned for a method the more weight given to that regularisation term. This means if the optimum hyper-parameter is high, then it's likely that the tasks in the relevant data set share the relationship leveraged by that regularisation term. For example, if the low rank approaches optimum λ_1 value (Eq 4.2) was large, then it is finding a strong shared low rank matrix between the tasks that it then leverage's. In this case you would expect the low rank approach to perform well. This is of course not always going to hold true, as the training data may share a relationship between tasks that the test data doesn't. The optimum hyper-parameters, for these methods and data sets, can be seen in Table 4.13. This table, along with Table 4.2 is used for a discussion on the effect of the value od the optimum hyper-parameters on the performance of the corresponding method and data set.

Table 4.13: A table showing the optimal hyper-parameters used for each method on the data sets

	Parameter	**Dirty**	**Low rank**	**L21**	**Lasso**	**Mean Reg**	**STL**
city PM	lambda 1	1.00E+05	1.00E-05	1.00E-05	1.00E-05	1.00E-05	10900
	lambda 2	1.00E+05	NA	1.00E-05	1.00E-05	1.00E-05	NA
region PM	lambda 1	73000	1.00E-05	1.00E-05	1.00E-05	1.00E-05	1.00E+05
	lambda 2	1.00E+05	NA	1.00E-05	1.00E-05	1.00E+05	NA
chinese weather	lambda 1	80200	0.0001	0.0001	0.0001	0.0001	55000
	lambda 2	46000	NA	0.0001	0.0001	0.0001	NA
parkinsons	lambda 1	50.5	10	7.03	1.09	0.307	0.0307
	lambda 2	8.20E-05	NA	0.001	8.20E-05	9.10E-05	NA
sarcos	lambda 1	64000	9.10E-05	0.0001	0.0001	0.0001	10.9
	lambda 2	10900	NA	0.0001	0.0001	0.0001	NA
School data	lambda 1	1090	20.8	20.8	5.05	0.208	2.08
	lambda 2	0.0802	NA	0.0109	0.0109	1.09	NA
performance exams	lambda 1	10900	109	20.8	20.8	1.00E+05	30.7
	lambda 2	9.01	NA	1.00E-05	0.0307	10.9	NA
performance schools	lambda 1	604	109	109	109	0.0208	109
	lambda 2	46000	NA	0.0307	1.00E-05	1.00E-05	NA

The dirty approach, which performs its best on the region PM2.5, the performance exams and the city PM2.5 data sets (seen in Table 4.2), has large hyper-parameters for both the city PM2.5 and region PM2.5 data sets. The performance exams data set with smaller hyper-parameters. The SARCOS and Chinese weather data sets, with the worst performance from the dirty approach, seen in Table 4.2, also has high hyper-parameters. It's hard to find any definitive trend between size of hyper-parameters and relative performance with the dirty approach results.

The $L_{2,1}$ approach which in Table 4.2 can be seen performing better on the parkinson's, city PM2.5, region PM2.5, Chinese weather and SARCOS data sets, is shown to in fact have the lowest hyper-parameter values for these data sets. Whereas the data sets it performed worse on have higher optimum hyper-parameters. This is unexpected, but could be due to over generalisation, which in the case of this approach (a feature selection approach) results in not enough features being used by the models.

The Lasso approach performs best for the Chinese weather and performance exams data sets. However, again, little correlation can be found between optimum learned hyper-parameter size and performance, as λ_1 (responsible for the models encouraged sparsity) is high for the performance schools data set, but low for the Chinese weather data set.

Further more, little correlation can be found between optimum learned hyper-parameter size and performance for the low-rank approach or the mean regularised approach.

4.5 Summary

The Single-task learning approach performing better than expected. It seems from the results that the multi-task learning methods are more unstable and can give a poor performance for an incompatible data set. This furthers the argument for this empirical analysis as it is crucial the best multi-task learning method is chosen for any given multi-task learning problem.

The dirty approach performing the worst in both performance on test data as well as learning time complexity. The mean regularised approach coming second worst for both. The feature selection approaches both performed similarly, with the Low-rank approach performing the best and having the best time complexity (due to only having a single hyper-parameter).

Chapter 5

Conclusions

The aim of this dissertation was to investigate and compare performance of multi-task learning methods on both toy data and real datasets, providing novel advice with respect to the best multi-task learning methods. Six multi-task learning implementations have been tested with the mean square error, time complexity, and effect of hyper-parameters discussed. The implementations have been ranked with respect to the various aspects tested providing a basis for choice of multi-task learning method, with insights that provide starting point for an implementation of a new multi-task learning method. The objective outlined in Chapter 1 are reiterated below.

1. Implement working multi-task learning methods that leverage task relationships through regularisation for linear regression problems. Research into various multi-task learning methods must be conducted to find viable approaches to implement, as well as research into any available libraries including useful code. This should lead to implementation of fully working multi-task learning methods.

2. Implement a framework for optimising the hyper-parameters. The optimisation of each method implemented in objective 1 is for a specific set of hyper-parameters. Research into various approaches such as cross validation will be conducted to find and implement an approach for optimising the hyper-parameters in each methods loss function.

3. Critical analysis of the results gathered. A critical analysis based off the performance results gathered from the different implementations will be conducted for both toy data sets and real data sets, linking any findings back to the literature reviewed about the methods.

4. Generation of toy data sets. Generate toy data sets specifically for testing models on data with certain attributes, with respect to task relationships, useful for the analysis of different multi-task learning implementations.

All objectives have been completed.

Future work with respect to this dissertation would include the implementation of more multi-task learning methods. Touched upon in Chapter 2 is the Multi-level approach, the Clustered Approach and the Deep learning Approach, all of which would be interesting to include within this experimental setting. As far as results go, the mean squared error results would have provided more useful information to have included the error per task for each method on each task. This would have given greater insight into the downfall of some of the methods, for example, whether most task where learnt well but one task was particularly hard and learnt badly. Also, the time complexity measurements could have been more informative cross referenced with plots of the error over time for the optimisation. This would give more insight to how the error changes over time during the optimisation process for each method, for example, one method may learn an average model very quickly but take a long time to optimise for a good model, and the contrary being a method that optimises a good model quicker but an average model slower. This would be useful information for a time dependent learning problem, where the optimisation method cut out after a certain time.

Bibliography

Lamport, L. (1986), *LaTeX : A Document Preparation System*, Addison-Wesley.

Patashnik, O. (1988), BibTeX ing. Documentation for general BibTeX users.

P. Austin and E. Steyerberg (2015), The number of subjects per variable required in linear regression analyses, *Journal of Clinical Epidemiology*, Vol. 68, pp(627-636)

T. Dietterich (1995), Overfitting and undercomputing in machine learning, *ACM Computing Surveys (CSUR)*, Vol. 27, pp(326-327)

Y. Nesterov (1983), A method of solving a convex programming problem with convergence rate $O(1/k^2)$, *Soviet Math. Dokl*, Vol 27, pp(372-376).

R. Caruana (1997), Multitask learning, *Machine Learning*, Vol 28, pp(41-75).

Jun Liu and Jieping Ye (2010), Efficient L_1/L_q Norm Regularization.

J. Zhou, J. Chen, and J. Ye (2012), MALSAR: Multi-tAsk Learning via StructurAl Regularization, *Arizona State University*, (http://www.MALSAR.org).

Yu Zhang and Qiang Yang (2017), A Survey on Multi-Task Learning.

S. Pan and Q. Yang (2010), A survey on transfer learning, *IEEE Transactions on Knowledge and Data Engineering*, Vol 22, pp(1345-1359).

R. Tibshirani (1996), Regression Shrinkage and Selection via the Lasso, *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol 58, pp(267-288).

Han Liu, Mark Palatucci and Jian Zhang (2009), Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery, *ICML '09 Proceedings of the 26th Annual International Con-*

ference on Machine Learning.

A. Argyriou, T. Evgeniou, and M. Pontil (2007), Multi-Task Feature Learning, *Proc. 19th Ann. Conf. Neural Information Processing Systems.*

A. Jalali, P. Ravikumar, S. Sanghavi and C. Ruan (2010), A dirty model for multi-task learning, *NIPS'10 Proceedings of the 23rd International Conference on Neural Information Processing Systems*, Vol 1, pp(964-972).

J. Chen, J. Liu, and J. Ye (2010), Learning incoherent sparse and low-rank patterns from multiple tasks, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

R. K. Ando and T. Zhang (2005), A framework for learning predictive structures from multiple tasks and unlabeled data, *Journal of Machine Learning Research*, Vol 6, pp(1817-1853).

J. Baxter (2000), A model of inductive bias learning, *Journal of Artifical Intelligence Research*, Vol 12, pp(149-198).

S. Ji, J. Ye (2009), An accelerated gradient method for trace norm minimization, *ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning.*

T. Evgeniou and M. Pontil (2004), Regularized multi-task learning, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

J. Zhou, L. Yuan, J. Liu and J. Ye (2011), A multi-task learning formulation for predicting disease progression, *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.*

L. Jacob, F. Bach, and J. Vert (2008), Clustered multi-task learning: A convex formulation, *Advances in Neural Information Processing Systems*, Vol 21, pp(745-752).

L. Jacob, F. Bach, and J. Vert (2017), HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition, *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, Vol XX.

G. Pons and D. Masip (2018), Multi-task, multi-label and multi-domain learning with residual convolutional networks for emotion recognition.

D. Fourure, R. Emonet, E. Fromont, D. Muselet, N. Neverova, et al (2017), Multi-task, Multi-domain Learning: application to semantic segmentation and pose regression.

L. Jacob, F. Bach, and J. Vert (2006), Personalized handwriting recognition via biased regularization, *ICML '06 Proceedings of the 23rd international conference on Machine learning*, pp(457-464).

B. Heisele, T. Serre, M. Pontil,T. Vetter, T. Poggio (2001), Categorization by Learning and Combining Object Parts.

Qian Xu, Sinno Jialin Pan, Hannah Hong Xue, and Qiang Yang (2011), Multitask Learning for Protein Subcellular Location Prediction, *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, Vol 8, pp(748-759).

Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King (2015), Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis, *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp(4460-4464).

F. Wu and Y. Huang (2015), Collaborative multi-domain sentiment classification, *Proceedings of the 2015 IEEE International Conference on Data Mining*, pp(459-468).

L. Zhao, Q. Sun, J. Ye, F. Chen, C. Lu, and N. Ramakrishnan (2015), Multi-task learning for spatio-temporal event forecasting, *Proceedings 20 of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp(1503-1512).

J. Zhou, L. Yuan, J. Liu, and J. Ye (2011), A multi-task learning formulation for predicting disease progression, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp(814-822).

D. He, D. Kuhn, and L. Parida (2016), Novel applications of multitask learning and multiple output regression to multiple genetic trait prediction, *Bioinformatics*, Vol 32, pp(37-43).

V. Jothi Prakash, Dr. L.M. Nithya (2014), A Survey on Semi-Supervised Learning Techniques, *International Journal of Computer Trends and Technology (IJCTT)*, Vol 8, pp(25-29).

Liang, X., S. Li, S. Zhang, H. Huang, and S. X. Chen (2016), PM2.5 data reliability, consistency, and air quality assessment in five Chinese cities, J. Geophys. Res. Atmos.

S. Vijayakumar and S. SchaalLWPR (2000), An O(n) Algorithm for Incremental Real Time Learning in High Dimensional Space, *Proc ICML 2000*, pp(1079-1086).

S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt, S. Schaal (2002), Statistical Learning for Humanoid Robots, *Autonomous Robot*, Vol 12, pp(55-69).

S. Vijayakumar, A. D'Souza, S. Schaal (2005), Incremental Online Learning in High Dimensions, *Neural Computation 17*, Vol 12, pp(2602-2634).

Appendix A

Plots

A.1 Dirty Approach

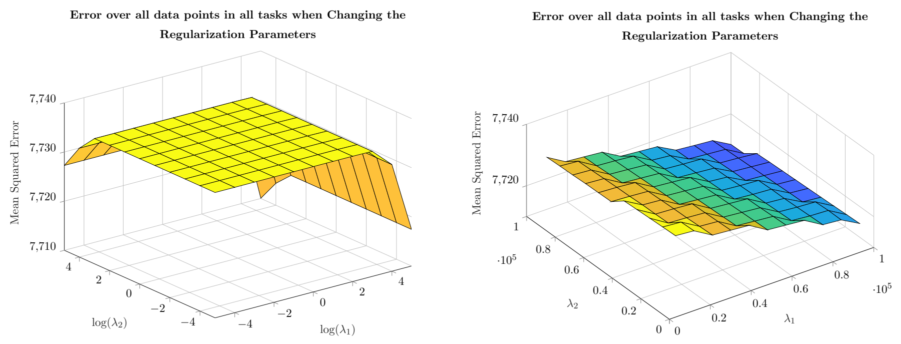


Figure A.1: mean squared error of dirty approach on the city PM2.5 dataset

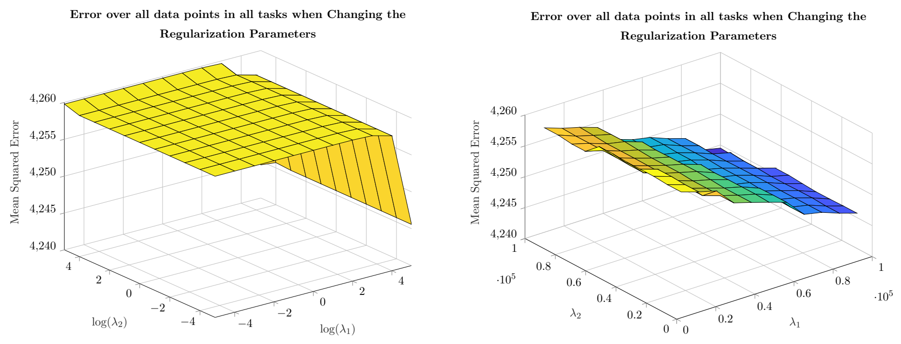


Figure A.2: mean squared error of dirty approach on the regional PM2.5 dataset

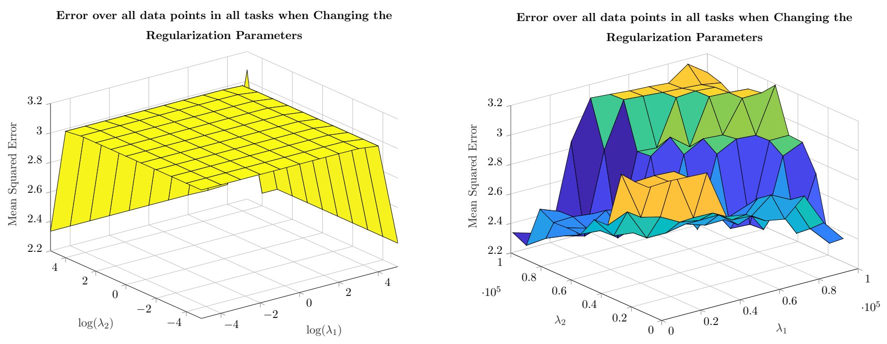


Figure A.3: mean squared error of dirty approach on the Chinese Weather dataset

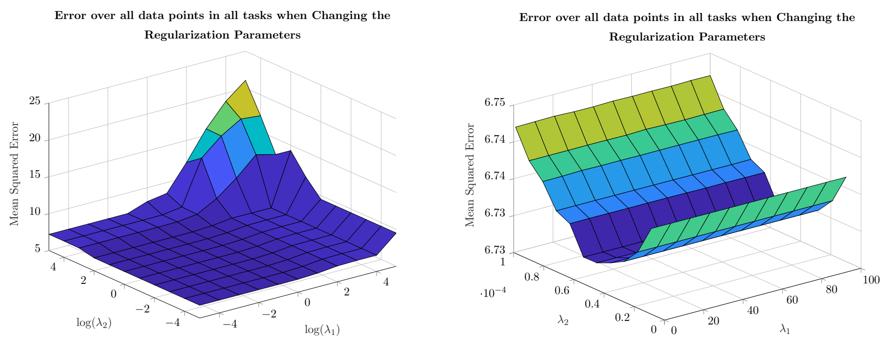


Figure A.4: mean squared error of dirty approach on the Parkinson's dataset

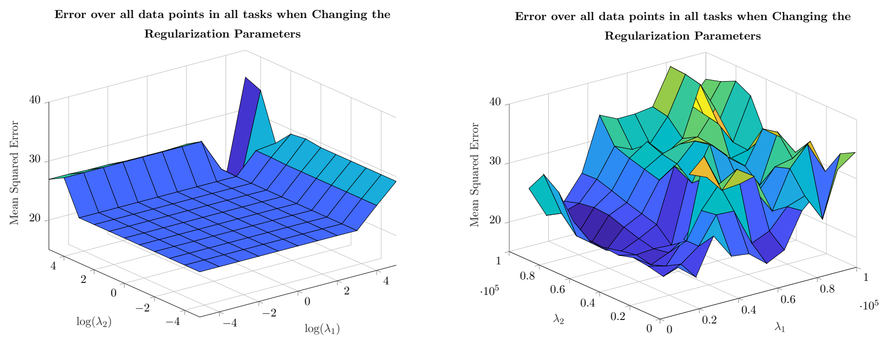


Figure A.5: mean squared error of dirty approach on the SARCOS dataset

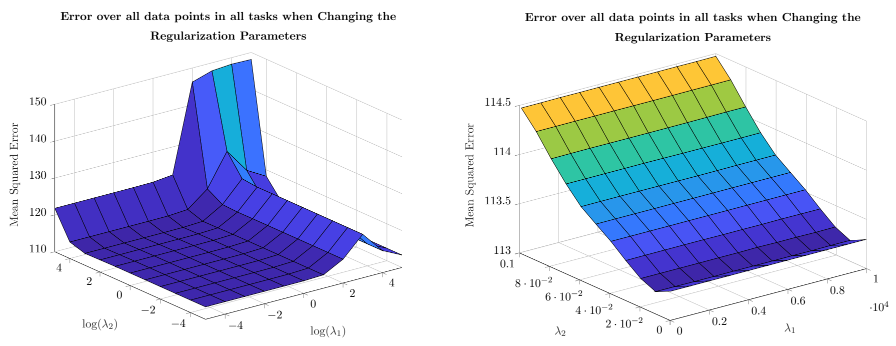


Figure A.6: mean squared error of dirty approach on the School dataset

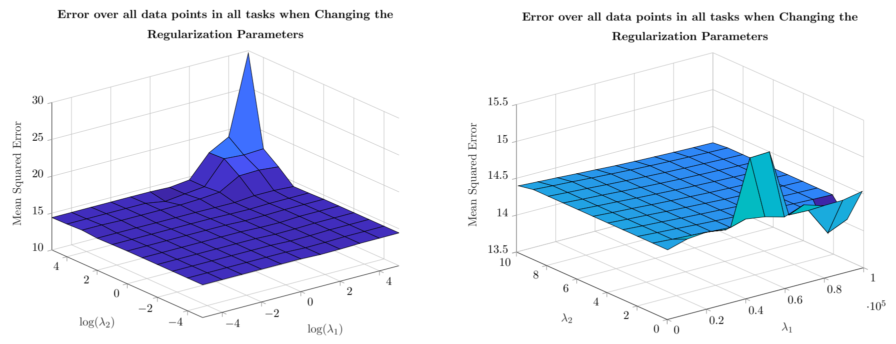


Figure A.7: mean squared error of dirty approach on the Performance exams dataset

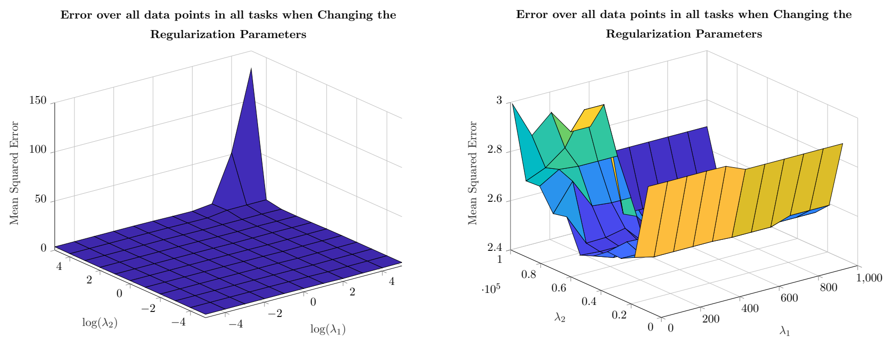


Figure A.8: mean squared error of dirty approach on the Performance schools dataset

A.2 Lasso Approach

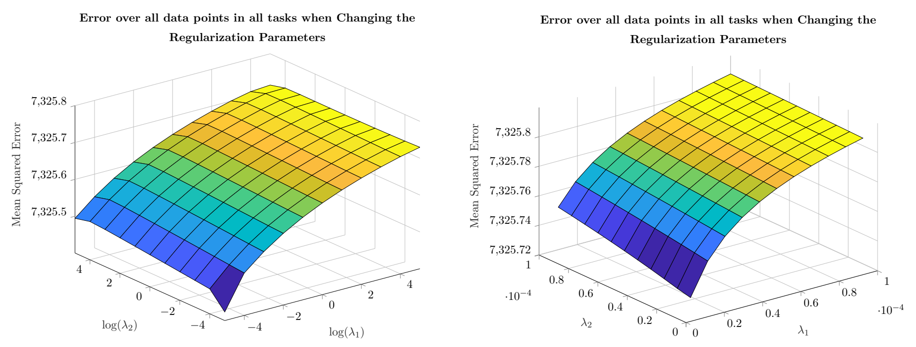


Figure A.9: mean squared error of lasso approach on the city PM2.5 dataset

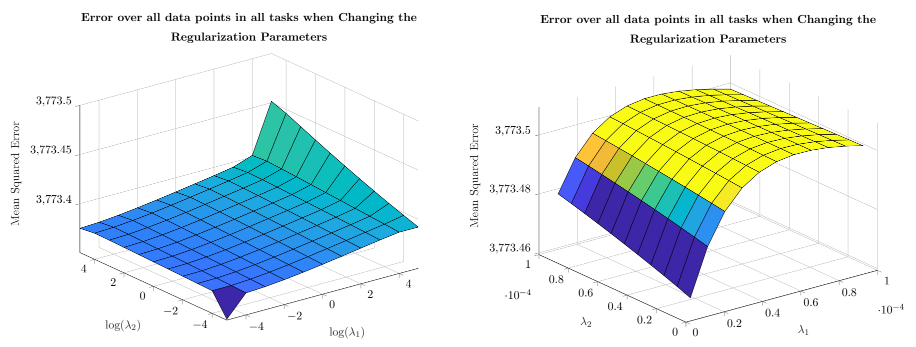


Figure A.10: mean squared error of lasso approach on the regional PM2.5 dataset

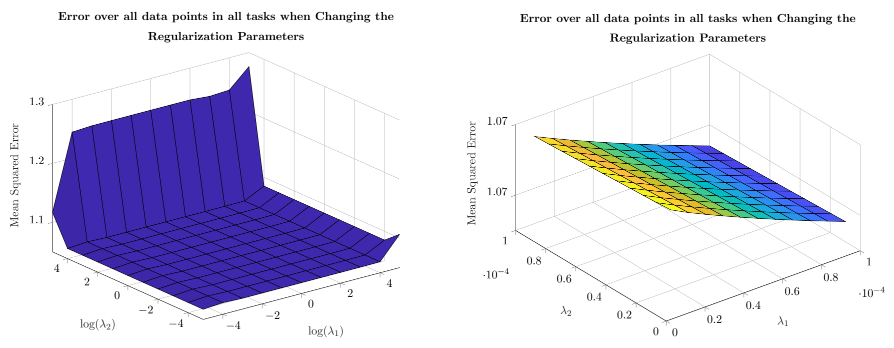


Figure A.11: mean squared error of lasso approach on the Chinese Weather dataset

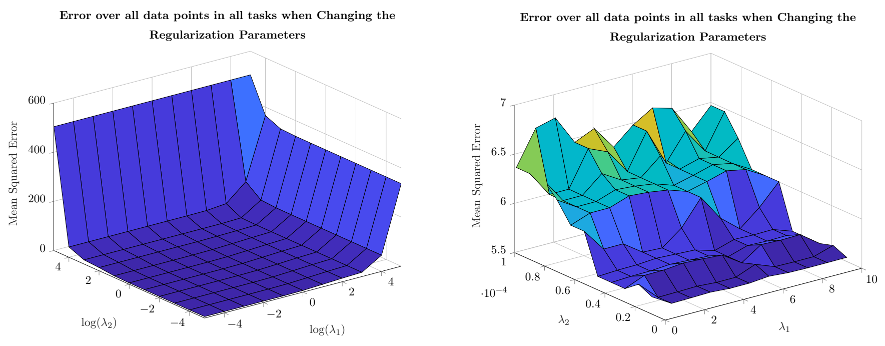


Figure A.12: mean squared error of lasso approach on the Parkinson's dataset

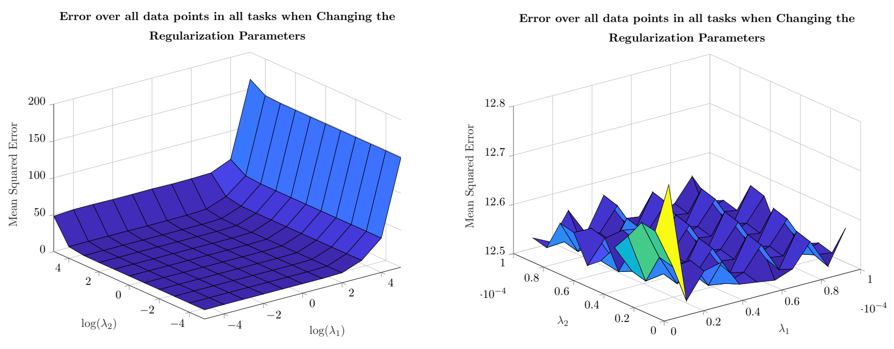


Figure A.13: mean squared error of lasso approach on the SARCOS dataset

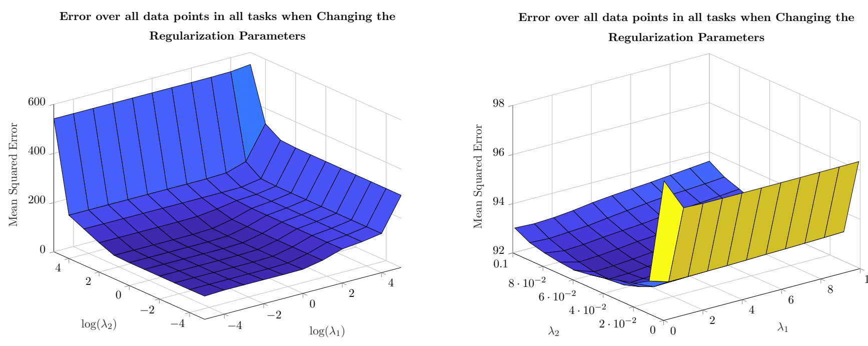


Figure A.14: mean squared error of lasso approach on the School dataset

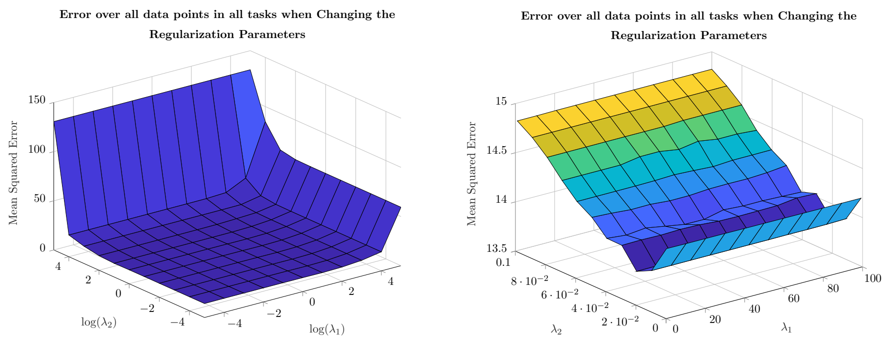


Figure A.15: mean squared error of lasso approach on the Performance exams dataset

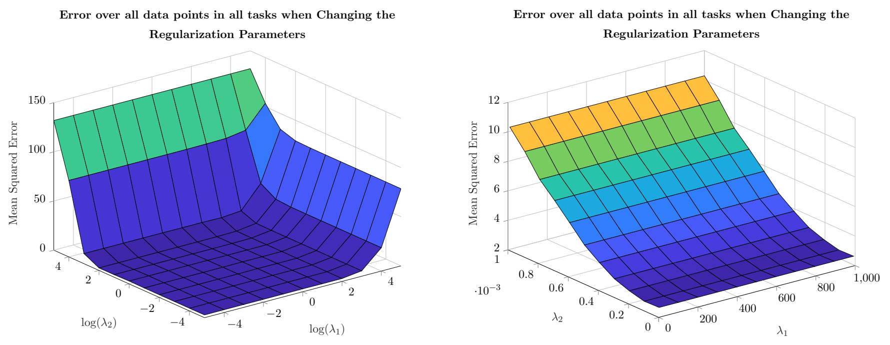


Figure A.16: mean squared error of lasso approach on the Performance schools dataset

A.3 $L_{2,1}$ Approach

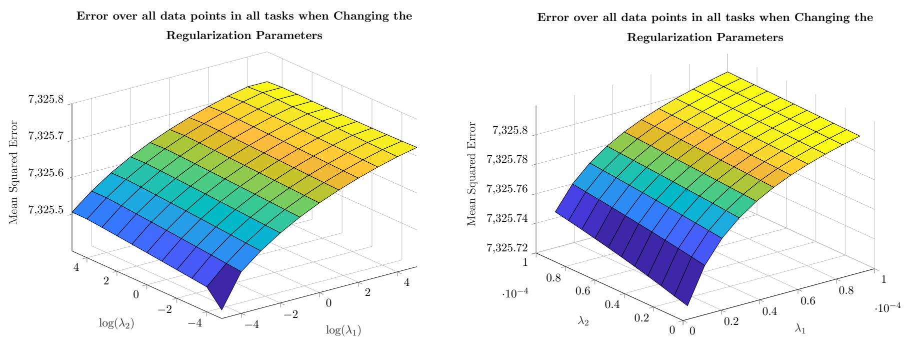


Figure A.17: mean squared error of $L_{2,1}$ norm regularized approach on the city PM2.5 dataset

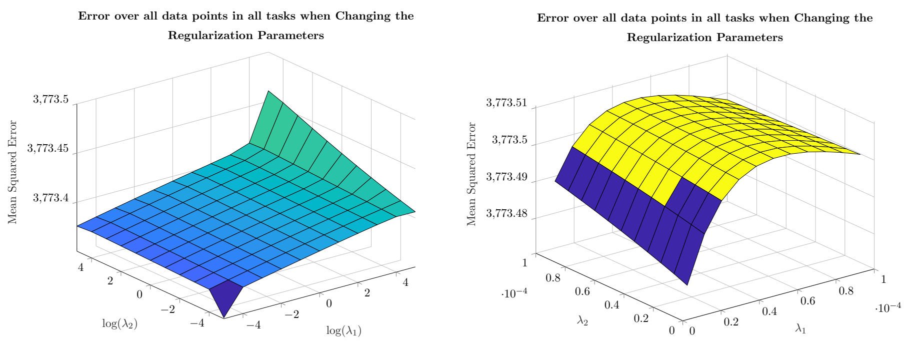


Figure A.18: mean squared error of $L_{2,1}$ norm regularized approach on the regional PM2.5 dataset

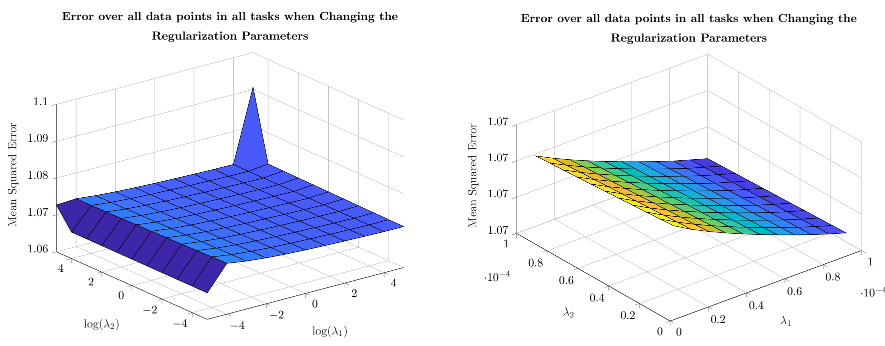


Figure A.19: mean squared error of $L_{2,1}$ norm regularized approach on the Chinese Weather dataset

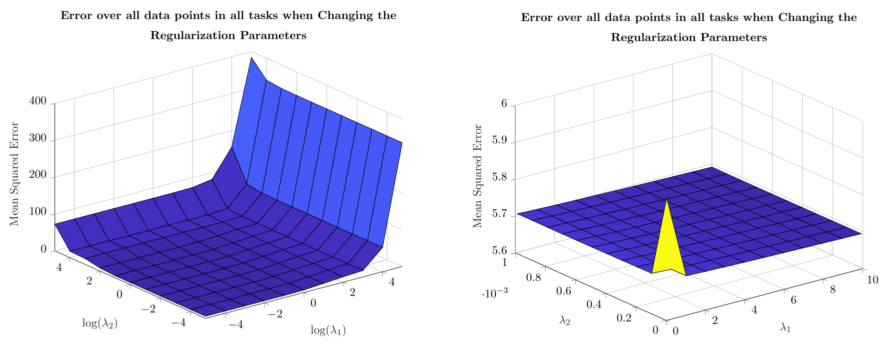


Figure A.20: mean squared error of $L_{2,1}$ norm regularized approach on the Parkinson's dataset

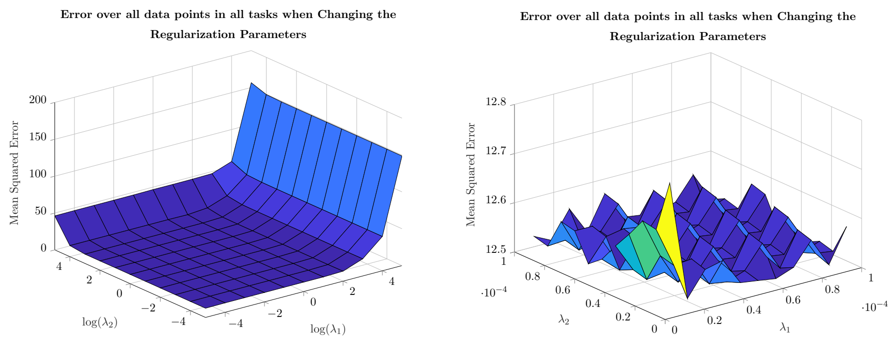


Figure A.21: mean squared error of $L_{2,1}$ norm regularized approach on the SARCOS dataset

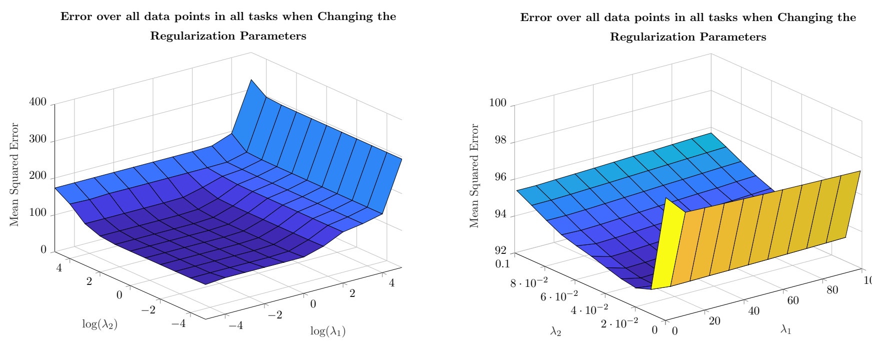


Figure A.22: mean squared error of $L_{2,1}$ norm regularized approach on the School dataset

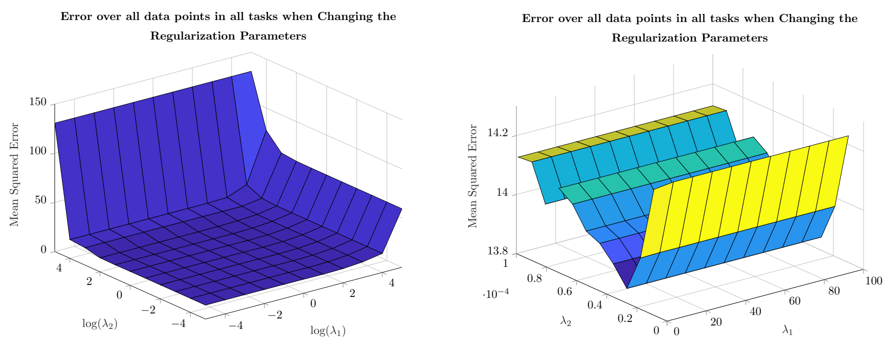


Figure A.23: mean squared error of $L_{2,1}$ norm regularized approach on the Performance exams dataset

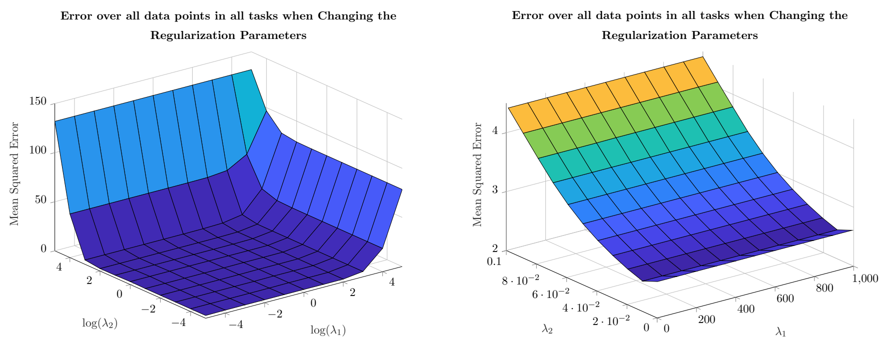


Figure A.24: mean squared error of $L_{2,1}$ norm regularized approach on the Performance schools dataset

A.4 Low Rank Approach

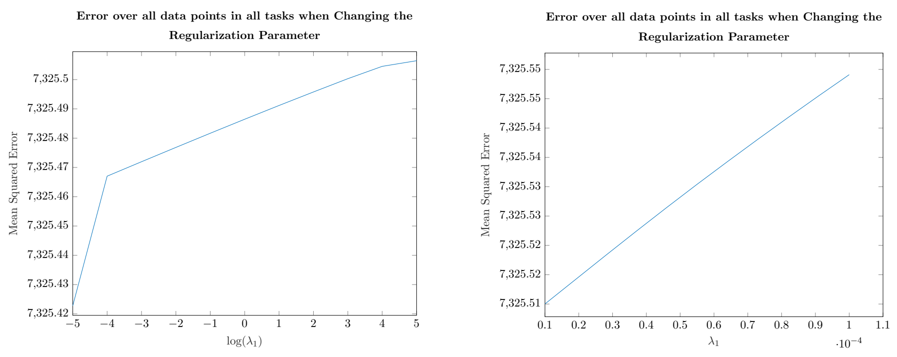


Figure A.25: mean squared error of low rank approach on the city PM2.5 dataset

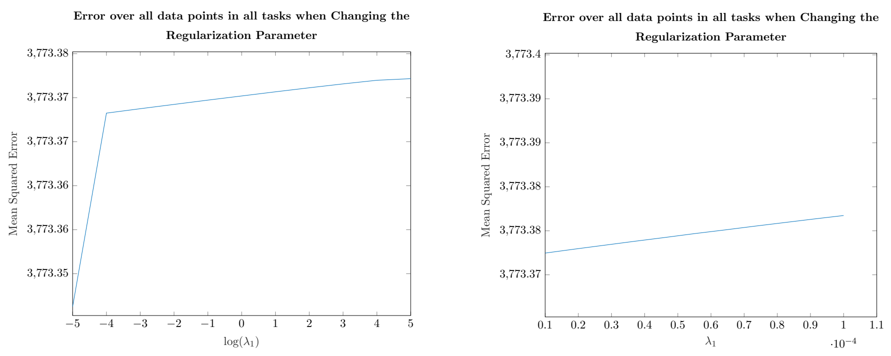


Figure A.26: mean squared error of low rank approach on the region PM2.5 dataset

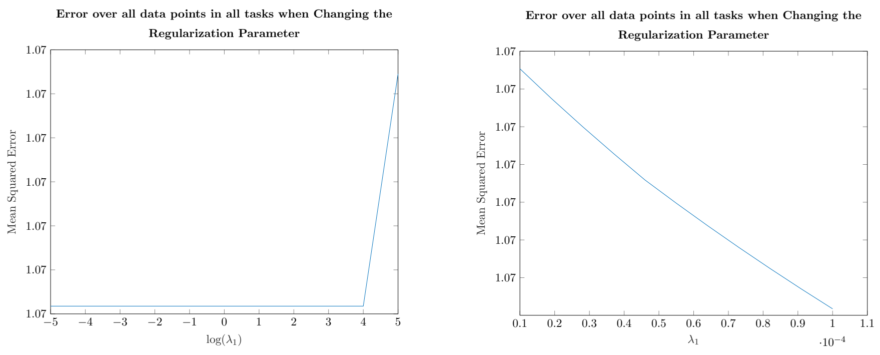


Figure A.27: mean squared error of low rank approach on the Chinese weather dataset

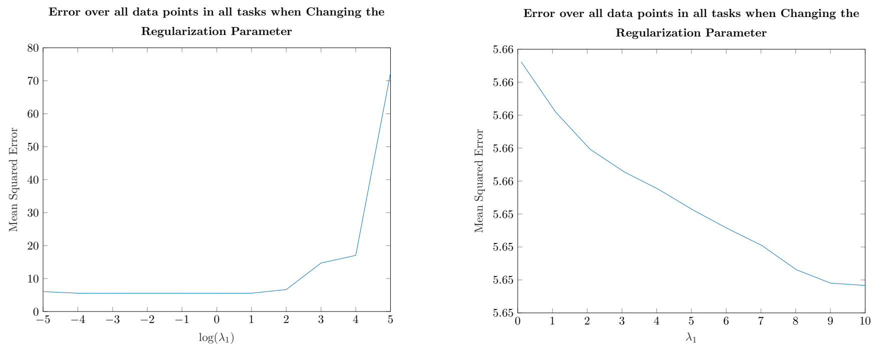


Figure A.28: mean squared error of low rank approach on the Parkinson's dataset

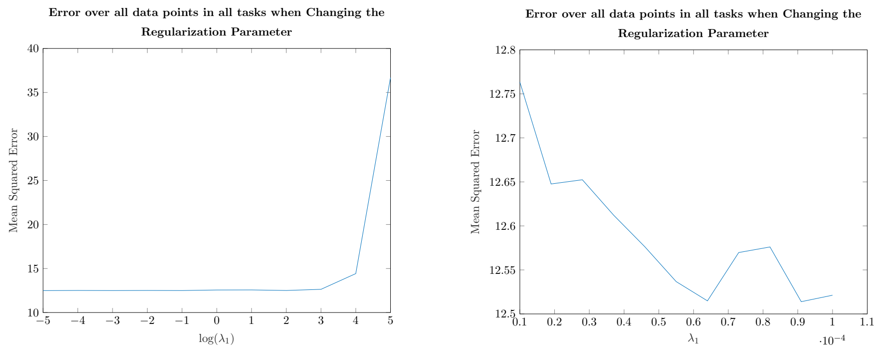


Figure A.29: mean squared error of low rank approach on the SARCOS dataset

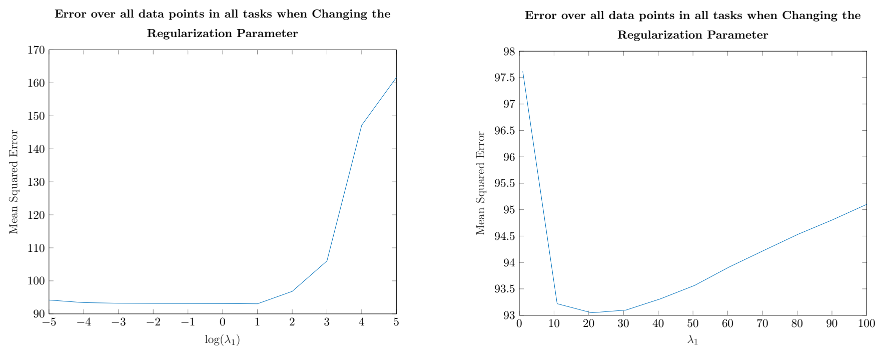


Figure A.30: mean squared error of low rank approach on the School dataset

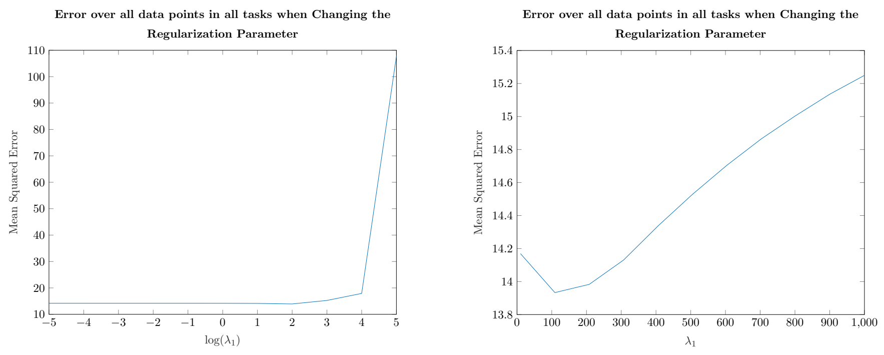


Figure A.31: mean squared error of low rank approach on the Performance exams dataset

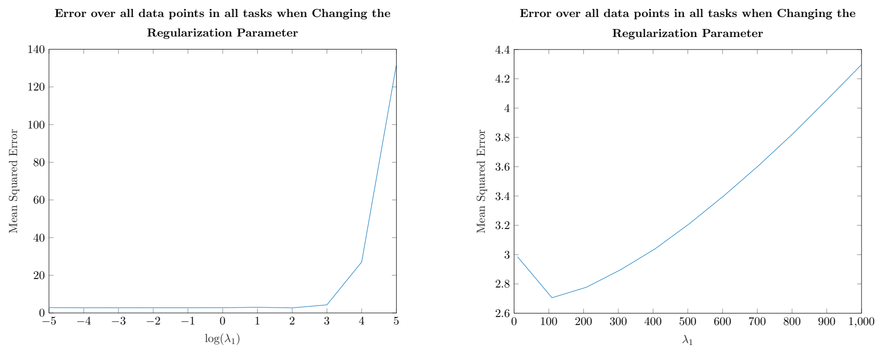


Figure A.32: mean squared error of low rank approach on the Performance schools dataset

A.5 mean regularized Approach

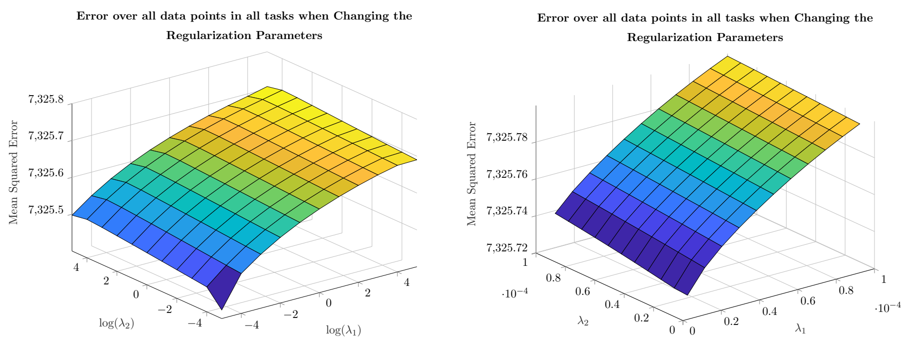


Figure A.33: mean squared error of mean regularized approach on the city PM2.5 dataset

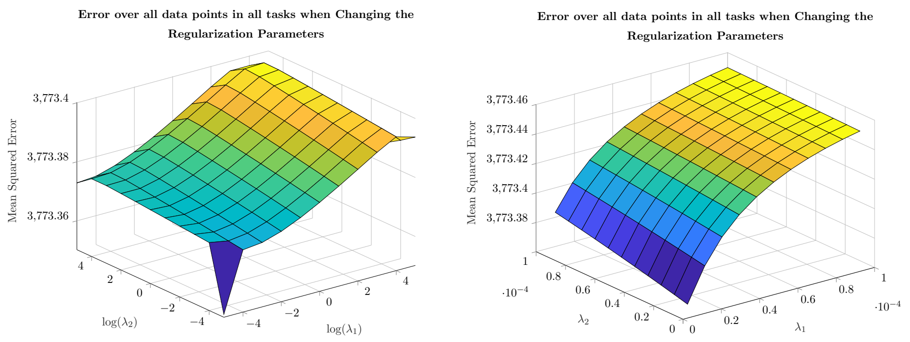


Figure A.34: mean squared error of mean regularized approach on the region PM2.5 dataset

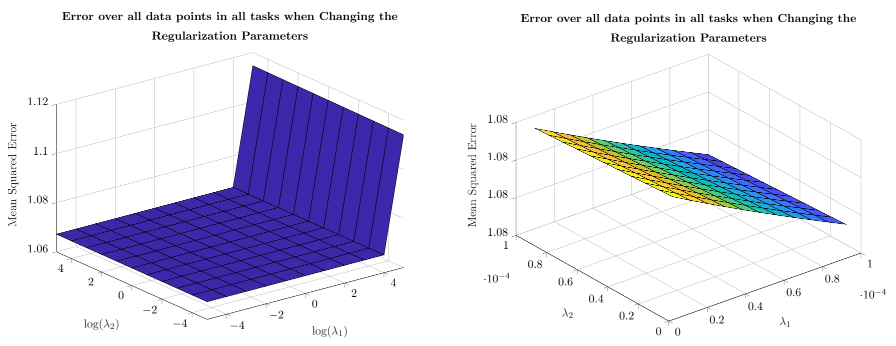


Figure A.35: mean squared error of mean regularized approach on the Chinese weather dataset

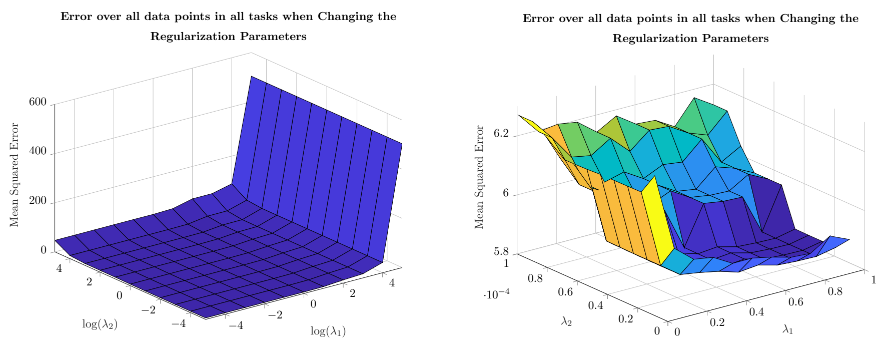


Figure A.36: mean squared error of mean regularized approach on the Parkinson's dataset

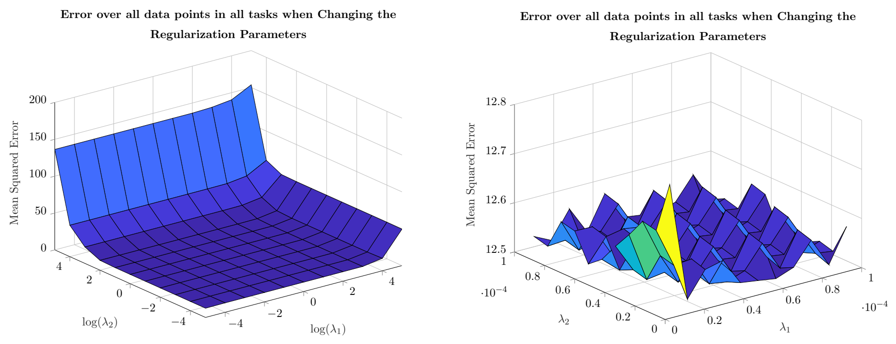


Figure A.37: mean squared error of mean regularized approach on the SARCOS dataset

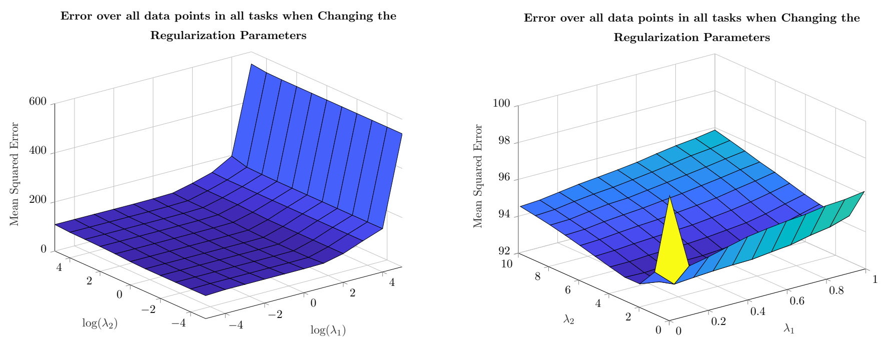


Figure A.38: mean squared error of mean regularized approach on the School dataset

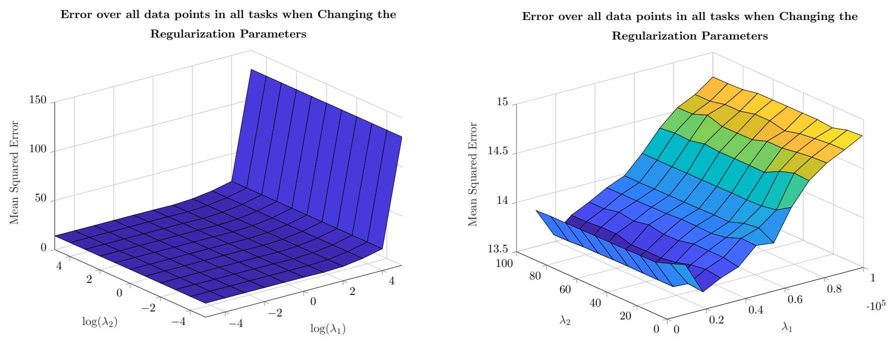


Figure A.39: mean squared error of mean regularized approach on the Performance exams dataset

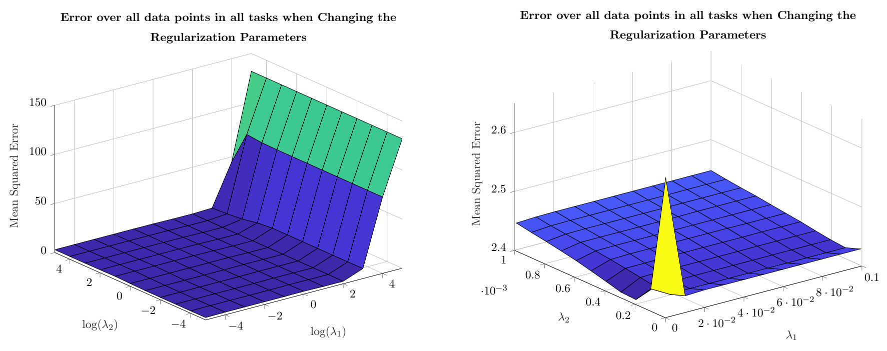


Figure A.40: mean squared error of mean regularized approach on the Performance schools dataset

A.6 Single-task learning Approach

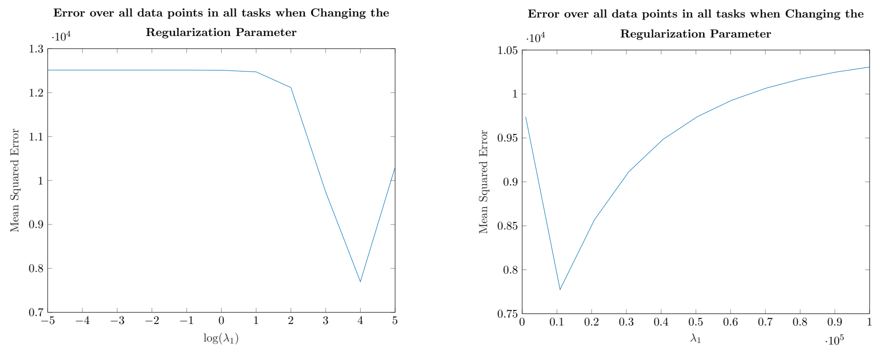


Figure A.41: mean squared error of STL approach on the city PM2.5 dataset

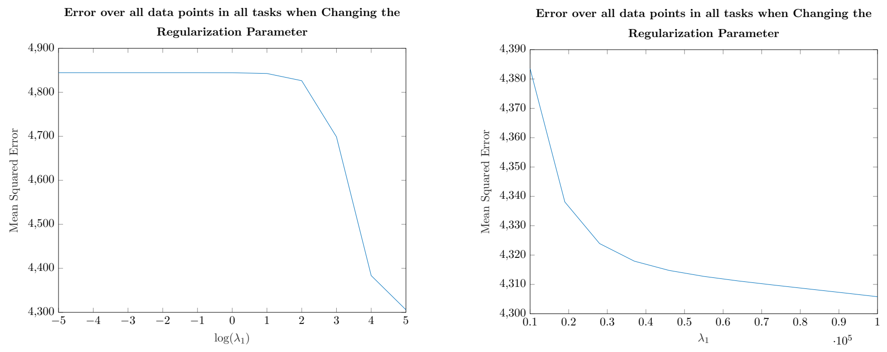


Figure A.42: mean squared error of STL approach on the region PM2.5 dataset

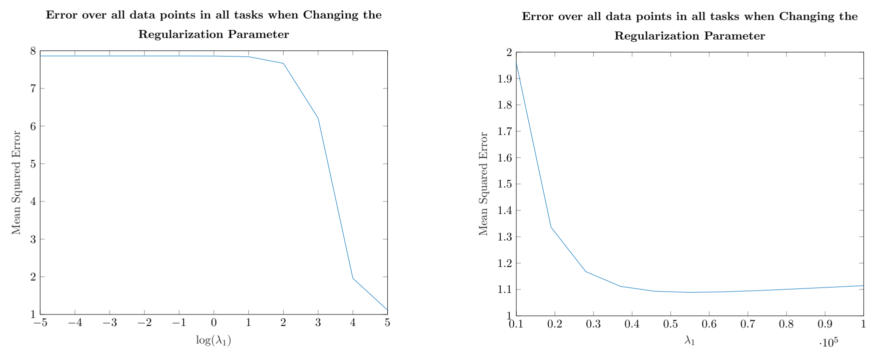


Figure A.43: mean squared error of STL approach on the Chinese weather dataset

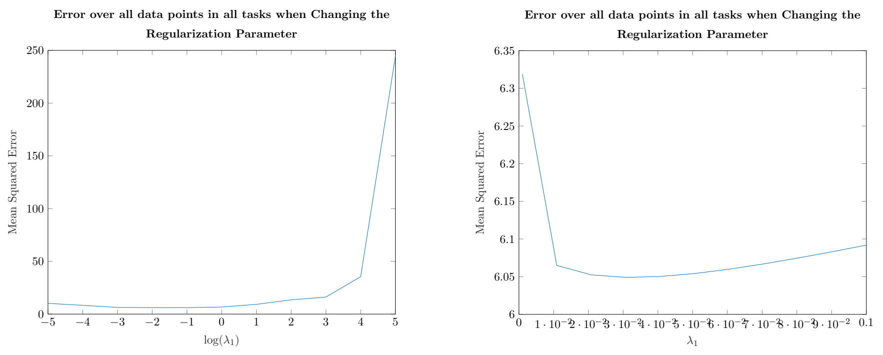


Figure A.44: mean squared error of STL approach on the Parkinson's dataset

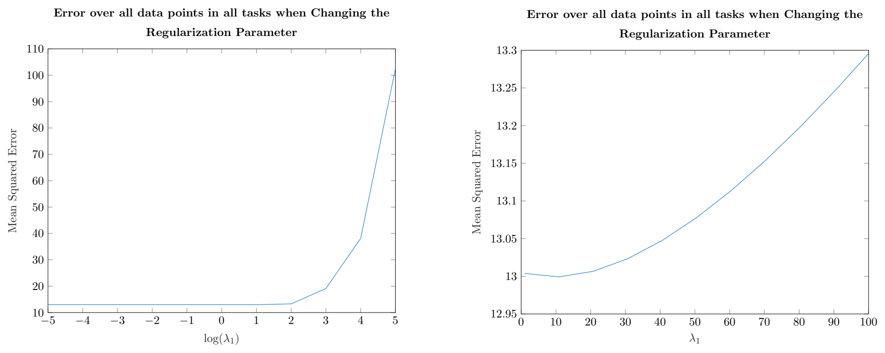


Figure A.45: mean squared error of STL approach on the SARCOS dataset

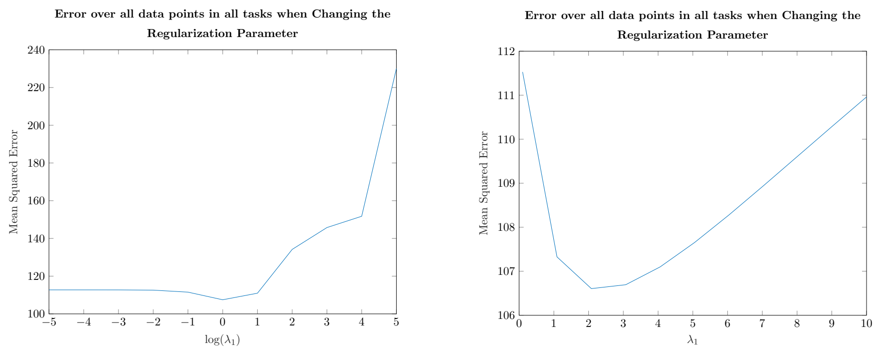


Figure A.46: mean squared error of STL approach on the School dataset

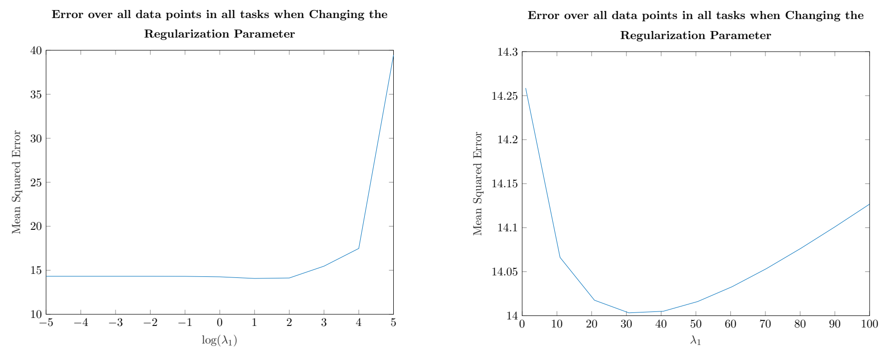


Figure A.47: mean squared error of STL approach on the Performance exams dataset

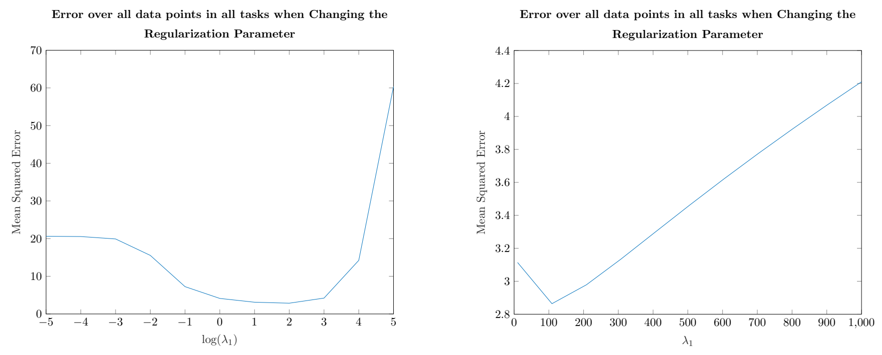


Figure A.48: mean squared error of STL approach on the Performance schools dataset