Machine learning network-constrained regression of epigenetic data

Sivo V. Daskalov
Corpus Christi College

UNIVERSITY OF CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: sivodaskalov@gmail.com

June 5, 2017

Declaration

I Sivo V. Daskalov of Corpus Christi College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 7875 *

* As calculated by TeXcount [https://www.ctan.org/pkg/texcount]

Signed:

Date:

Abstract

Computational biology often involves working with high-dimensional data. Penalized regression methods are often used on such data, as they can effectively perform feature selection. Several approaches for network-constrained regression have been suggested in literature over the recent years. They use prior knowledge in the form of a network to exploit known relationships between predictors. Synthetic datasets have been generated to do parameter tuning for the various implemented methods.

We suggest an approach for cooperative parameter tuning in the context of multiple alternative methods that share common input and goals. The aim is to tune the different regression methods iteratively, in a way that increases agreement between their coefficients. Neighboring values on the tuning parameter grid are considered for each method and iteration, selecting the set of values that achieves largest correlation with the averaged coefficients of all other methods for the previous iteration. Given enough iterations and granularity of the tuning grids, this process converges.

We also implement a simple approach to aggregate the coefficients produced by the various regression methods. Each predictor is considered relevant if it corresponds to a non-zero coefficient in a certain fraction of the underlying methods. Once a consensus has been reached through this form of voting, ordinary least squares estimation is used to fit only the relevant predictors to the data.

The traditional way of hyperparameter tuning by minimization of the prediction mean squared error is implemented alongside our suggested approach. Comparison of both tuning approaches is performed on synthetic datasets. Analysis of similarities between the various regression methods is carried out. An optimal set of tuning parameters is assembled for each regression method for use on real data. Gene methylation and expression data has been processed with the implemented algorithms. A map is created that shows methylation of which genes affects the expression levels of each gene.

Contents

1 Introduction **1**

2 Background and Related Work **3**
2.1 Linear regression 3
2.2 Ordinary least squares estimation 4
2.3 Penalized regression 4
2.3.1 Ridge regression 4
2.3.2 Lasso 5
2.3.3 Elastic Net 5
2.4 Network-constrained regularization 5
2.4.1 Grace 6
2.4.2 aGrace 6
2.4.3 GBLasso 7
2.4.4 Linf and aLinf 7
2.4.5 TTLP and LTLP 8

3 Implementation Details **11**

4 Synthetic Dataset Generation **13**
4.1 Predictor network generation 13
4.2 Generation of predictor observations 14
4.3 Response variable generation 14
4.3.1 Primary simulation setups 14
4.3.2 Secondary simulation setups 16

5 Composite Voting Regression **17**

6 Hyperparameter Tuning **19**
6.1 Traditional approaches 19
6.2 Orchestrated parameter tuning 21
6.2.1 Context and motivation 21

6.2.2 Inspiration 21
6.2.3 Methodology 22

7 Simulation studies **27**
7.1 Simulation setup 27
7.2 Hyperparameter search spaces 27
7.3 Model metrics 29
7.3.1 Prediction evaluation metrics 29
7.3.2 Variable selection metrics 29
7.4 CV-MSE tuning 30
7.5 Orchestrated tuning 32
7.6 Comparison of tuning approaches 34
7.7 Optimal hyperparameter value selection 36
7.7.1 Lasso 36
7.7.2 Elastic Net 37
7.7.3 Grace 38
7.7.4 aGrace 39
7.7.5 GBLasso 40
7.7.6 Linf 41
7.7.7 aLinf 42
7.7.8 TTLP and LTLP 43
7.7.9 Composite 43

8 Method Similarity Evaluation **45**
8.1 Cosine Similarity 45
8.2 Similarity for CV-MSE tuning 46
8.3 Similarity for Orchestrated tuning 48
8.4 Similarity comparison by tuning method 50

List of Figures

6.1 Local search grid neighborhood, two tuning hyperparameters; P_{t-1} is the parameter combination selected in iteration $t-1$, parameter values in the P_t neighborhood are considered in iteration t 22
6.2 Orchestrated tuning abstract structure, three regression methods; $M1_{t-1}$ denotes the parameter estimates of method 1 for iteration $t-1$; Each tuning iteration t of a method uses the common input data and the parameter estimates of all other methods for the previous iteration 24

7.1 CV-MSE tuning mean model metrics with standard deviation error bars for 20 synthetic datasets 32
7.2 Orchestrated tuning mean model metrics with standard deviation error bars for 20 synthetic datasets 33
7.3 Comparison of mean model metrics by parameter tuning method 35
7.4 Lasso distribution of selected parameter values 36
7.5 Elastic Net distribution of selected parameter combinations . . 37
7.6 Grace distribution of selected parameter combinations; darker blue and larger marker size indicate a higher frequency of choice 38
7.7 aGrace distribution of selected parameter combinations; darker blue and larger marker size indicate a higher frequency of choice 39
7.8 GBLasso distribution of selected parameter combinations; darker blue and larger marker size indicate a higher frequency of choice 40
7.9 Linf distribution of selected parameter values 41
7.10 aLinf distribution of selected parameter values 42
7.11 Composite method distribution of selected parameter values . 43

8.1 CV-MSE tuning mean regression method similarities 46
8.2 Orchestrated tuning mean regression method similarities . . . 49
8.3 Iterations elapsed before convergence for all synthetic datasets 49

List of Tables

3.1 Regression methods implementation details 12

7.1 Hyperparameter search space for all regression methods 28
7.2 CV-MSE tuning mean model metrics for 20 synthetic datasets 31
7.3 Orchestrated tuning mean model metrics for 20 synthetic datasets 34
7.4 Comparison of mean model metrics by parameter tuning method 34

8.1 CV-MSE tuning regression method similarities 47
8.2 Orchestrated tuning regression method similarities 48
8.3 Comparison of regression method similarities by tuning method 50

Chapter 1

Introduction

Epigenetics [1] studies the heritable traits that cannot be explained by changes in the DNA sequence. Examples of epigenetic mechanisms include DNA methylation and histone modification. These mechanisms adjust the expression level of genes [2], which allows organisms to dynamically adapt to changes in the environment.

Disruption of gene expression levels is related to the development of various diseases [3]. For example, the epigenetic deactivation of certain tumor suppressor genes commonly leads to the development of cancer [4]. The expression levels of certain genes can therefore be used as additional tools in early diagnostics of cancer, as prognosis factors and as predictors of response to treatment.

Good understanding of the relationship between DNA methylation and gene expression is important for both cancer prevention and epigenetic disease treatment. We have used the gene methylation and expression level data discussed in [5] to explore this relationship. One of the goals in this project is to produce a map that shows the methylation of which genes affects the expression levels of each gene.

Several methods [6, 7, 8, 9, 10, 11, 12] have been implemented and considered for use with real data. The hyperparameters for each method have been tuned with the use of synthetic datasets as suggested in [8]. This is done because ground truth remains unknown for the relationship between gene methylation and expression.

A novel method of hyperparameter tuning is developed as an alternative to the widely used method of minimizing the cross-validated mean squared

test error. In our context we have a bundle of regression methods that operate on the same training data and share a common goal - to correctly identify the relationship between predictor and target variables. Instead of tuning the various regression methods independently, our approach performs cooperative hyperparameter tuning on all methods simultaneously. It uses an iterative algorithm to increase the similarity of estimated coefficients for the various methods by tuning their hyperparameters. For each method and iteration, the method's parameter grid neighborhood is searched for a set of parameters that maximizes the correlation between its estimated coefficients and the averaged estimates of all other methods for the previous iteration. When this process converges a set of parameters is defined for each method that maximizes the overall agreement across the whole set of methods.

The various regression methods discussed in this project minimize different cost functions. As a result, each method exhibits specific strengths and weaknesses. The relative performance of the methods depends on the dataset used. For this reason it is impossible to predict their effectiveness on an arbitrary real data set with no access to ground truth. We have developed a simple way to merge the estimation results of the method bundle. It is designed to balance the behavior of any individual method. Each of the predictor variables is considered important if it has non-zero coefficients in a fraction of the methods above a given threshold. This approach for variable selection in practice implements a voting system where each predictor must achieve a certain electoral threshold to be selected. Ordinary least squares estimation is then performed only using the set of selected variables.

The similarities between estimates of the various regression methods have been explored. We also look into the effect that a choice of hyperparameter tuning procedure has on these similarities.

The following chapters discuss the theoretical and implementation details of each linear regression approach. Afterwards, we introduce the synthetic dataset generation process used in our simulations. We then define our estimate merging procedure and our proposed cooperative hyperparameter tuning approach. In the simulations carried out we compare the various regression approaches, as well as the traditional and our proposed cooperative parameter tuning approach. Analysis of similarities between the various regression methods is carried out. The final chapter defines the real dataset used for exploration of the relationship between gene methylation and expression, presents and discussed the obtained results.

Chapter 2

Background and Related Work

This chapter briefly reviews the main concepts of linear regression. We describe in detail the various methods for penalized regression found in literature and used in this project.

2.1 Linear regression

Let us consider an entity with a number of scalar measurable (observable) properties, e.g. temperature, weight, dimensions. We can define a matrix X of n rows and p columns, such that each column contains the observed values of a particular property and each row represents an independent observation of values for all properties. Let us also define a vector y of length n containing the corresponding observed values of an arbitrary property of interest.

Linear regression is a method for modeling the relationships between a scalar dependent (target) variable y and a number of explanatory variables (predictors) $X_1, ..., X_p$. It assumes that this relationship is linear and assigns a regression coefficient β_i to each predictor X_i, as well as a constant (offset) term β_0. The linear regression model takes the form shown in Equation 2.1

$$y_i = \beta_0 1 + \beta_1 X_{i1} + \beta_2 X_{i2} + ... + \beta_p X_{ip} + \epsilon_i, \quad for\ i = 1, 2, ..., n \tag{2.1}$$

where ϵ_i represents noise, capturing all external factors influencing the target values, such as inaccuracy of measurement. The error ϵ_i introduces cannot be predicted or reduced.

2.2 Ordinary least squares estimation

Ordinary least squares (OLS) is a method of estimating the unknown parameters β in a linear regression model. It aims to minimize the sum of squared deviations of the observed values from the model prediction (2.2), also called residual sum of squares (RSS).

$$L(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2 \tag{2.2}$$

The parameter estimate $\hat{\beta}$ for the linear regression model is obtained as shown in equation 2.3 through the minimization of the objective function $S(\beta)$.

$$\hat{\beta} = argmin_{\beta \in R}\ S(\beta) = L(\beta) \tag{2.3}$$

2.3 Penalized regression

Penalized regression methods introduce a penalty $P(\beta)$ to the objective function $S(\beta)$ in addition to the loss function $L(\beta)$. P penalizes values of the unknown parameters that are considered unrealistic in the current context, which is done to obtain a more meaningful estimation. One or more regularization parameters λ_i can be used to balance the effect of any introduced penalties by scaling them. The general form of penalized regression is shown in Equation 2.4.

$$S(\beta) = L(\beta) + P(\beta) \tag{2.4}$$

2.3.1 Ridge regression

Ridge regression [13], also called Tikhonov or L2 regularization, is used to penalize large values in the β estimate. The penalty, shown in Equation 2.5, causes the parameter estimates of the less important predictors to be shrinked, but remain non-zero. As a result, L2 regularization does not perform feature selection.

$$P(\beta) = \lambda \sqrt{\sum_{i=1}^{p} \beta_i^2} \tag{2.5}$$

2.3.2 Lasso

The least absolute shrinkage and selection operator (LASSO) was introduced by Tibshirani in [6]. It produces a sparse coefficient vector, whose remaining non-zero elements define a subset of the most relevant predictors. Model sparsity is especially important in high-dimensional problems, such as those arising when processing epigenetic data. The L1 penalty, shown in Equation 2.6, performs both variable selection and regularization.

$$P(\beta) = \lambda \sum_{i=1}^{p} |\beta_i| \tag{2.6}$$

2.3.3 Elastic Net

The Elastic Net [7], suggested by Zou and Hastie, linearly combines the L1 (2.6) and L2 (2.5) penalties. This approach overcomes the individual limitations of the Lasso and Ridge methods. The elastic net penalty, shown in Equation 2.7, is adjusted by two hyperparameters λ_1 and λ_2, one for each of the two penalty terms.

$$P(\beta) = \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sqrt{\sum_{i=1}^{p} \beta_i^2} \tag{2.7}$$

2.4 Network-constrained regularization

Various approaches for network-constrained regularization have been developed in recent years. They enable the use of prior knowledge in the form of a network in the parameter estimation process. This allows methods to consider known relationships between predictors. In the context of epigenetic research, prior knowledge could be provided as a gene network representing known interactions between genes. Biological knowledge about the predictors should lead to a better understanding of the data and improved (biological) meaningfulness of the results.

For all network-constrained regularization approaches presented in this section, we define the following notation:
Let us consider a network that is represented by a weighted graph $G = (V, E, W)$, where V is the set of vertices corresponding to the p predictors,

E is the set of edges and W contains their corresponding weights. An edge between the vertices u and v is represented as $u \sim v$ and its edge weight is $w(u, v)$. Let us define the degree d_v of a vertex v as $d_v = \sum_{u \sim v} w(u, v)$.

2.4.1 Grace

The first approach for network-constrained regularization was suggested by Li and Li [8]. The alias "Grace" is derived from the method's full name "GRAph Constrained Estimation". The penalty function, shown in Equation 2.8, contains two terms - an $L1$ penalty for variable selection and a second term that performs the network penalization.

$$P(\beta) = \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sum_{u \sim v} \left(\frac{\beta_u}{\sqrt{d_u}} - \frac{\beta_v}{\sqrt{d_v}} \right)^2 w(u, v) \tag{2.8}$$

The penalty is designed to smooth the parameters β over the gene network. This is achieved by penalizing the scaled difference of the coefficients between neighboring vertices in the network. The penalty encourages genes with a higher degree in the network (e.g. hub genes) to have larger coefficients.

2.4.2 aGrace

One drawback of the original Grace approach is that it performs poorly when the coefficients of two linked predictors have different signs. This scenario is feasible because one of the two genes could be negatively correlated with the target, in which case the coefficients of both genes will be penalized.

Li and Li proposed a modification [9] that performs adaptive graph-constrained regularization (aGrace) to solve this issue. It uses an initial coefficient estimate $\tilde{\beta}_v$ obtained through OLSE (2.2) if $p < n$ or Elastic Net (2.3.3) otherwise. The adaptive Grace penalty function is shown in Equation 2.9.

$$P(\beta) = \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sum_{u \sim v} \left(\frac{sign(\tilde{\beta}_u)\beta_u}{\sqrt{d_u}} - \frac{sign(\tilde{\beta}_v)\beta_v}{\sqrt{d_v}} \right)^2 w(u, v), \tag{2.9}$$

where the multiplier $sign(\tilde{\beta}_u) = \begin{cases} -1 & if\ \tilde{\beta}_u < 0 \\ 1 & otherwise \end{cases}$ adjusts the sign of each fraction as suggested by the initial estimate $\tilde{\beta}$.

2.4.3 GBLasso

One concern regarding the adaptive grace (2.4.2) method is the difficulty to estimate the sign adjustment of all β_i, for which $\tilde{\beta}_i = 0$. To discard the need for this estimation, Pan et al. proposed an alternative approach [10]. The authors suggested the penalty function shown in Equation 2.10.

$$P(\beta) = \lambda 2^{1/\gamma'} \sum_{u \sim v} \left(\frac{|b_u|^\gamma}{w_u} + \frac{|b_v|^\gamma}{w_v} \right)^{1/\gamma}, \tag{2.10}$$

where $\gamma > 1$ and $\lambda > 0$ are hyperparameters and γ' satisfies $\frac{1}{\gamma'} + \frac{1}{\gamma} = 1$. The denominator w_i is a weight function attributed to each node. Three types of weight functions, dependent on the node's degree d_i and/or γ, were initially considered by the authors: $w_i = d_i^{(\gamma+1)/2}$, $w_i = d_i$ and $w_i = d_i^\gamma$.

A simplification of the penalty function is presented in [11]. The authors have selected a node weight function of $w_i = d_i^{\gamma/2}$ and the penalty sum multiplier $\lambda 2^{1/\gamma'}$ has been modified to depend exclusively on λ. The simplified penalty function is shown in Equation 2.11 and referred to with the alias GBLasso in this paper.

$$P(\beta) = \lambda \sum_{u \sim v} \left[\left(\frac{|b_u|}{\sqrt{d_u}} \right)^\gamma + \left(\frac{|b_v|}{\sqrt{d_v}} \right)^\gamma \right]^{1/\gamma} \tag{2.11}$$

2.4.4 Linf and aLinf

Linf

Luo et al. [11] continued researching the GBLasso method (2.4.3). They noted that as $\gamma \to \infty$ the GBLasso penalty (2.11) is transformed into Equation 2.12. This penalty is linear and we denote the method as L_∞ (Linf).

$$P(\beta) = \lambda \sum_{u \sim v} \max \left(\frac{|\beta_u|}{\sqrt{d_u}}, \frac{|\beta_v|}{\sqrt{d_v}} \right) \tag{2.12}$$

The authors also suggest an equivalent formulation of the GBLasso-based regression as the following constrained minimization problem:

$$\begin{aligned} S(\beta) = & \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 \\ & \textit{subject to } \sum_{u \sim v} \left[\left(\frac{|b_u|}{\sqrt{d_u}} \right)^\gamma + \left(\frac{|b_v|}{\sqrt{d_v}} \right)^\gamma \right]^{1/\gamma} \leq C \end{aligned} \tag{2.13}$$

Similarly, regression with the L_∞ penalty can be equivalently defined as:

$$\begin{aligned} S(\beta) = & \sum_{i=1}^{n}(y_i - x_i^T\beta)^2 \\ & subject\ to\ \sum_{u\sim v} \max\left(\frac{|\beta_u|}{\sqrt{d_u}}, \frac{|\beta_v|}{\sqrt{d_v}}\right) \leq C \end{aligned} \tag{2.14}$$

aLinf

The authors suggest an additional modification to reduce bias in the parameter estimates of the standard Linf method. Similarly to [9], they propose a two-step approach using an initial parameter estimate $\tilde{\beta}$, obtained with the L_∞ method. The adaptive penalty, denoted as aL_∞ (aLinf), is shown in Equation 2.15.

$$P(\beta) = \lambda \sum_{u\sim v} \left| \frac{sign(\tilde{\beta}_u)\beta_u}{\sqrt{d_u}} - \frac{sign(\tilde{\beta}_v)\beta_v}{\sqrt{d_v}} \right| \tag{2.15}$$

The following constrained minimization problem can be defined to implement the aL_∞ approach:

$$\begin{aligned} S(\beta) = & \sum_{i=1}^{n}(y_i - x_i^T\beta)^2 \\ & subject\ to\ \sum_{u\sim v} \left| \frac{sign(\tilde{\beta}_u)\beta_u}{\sqrt{d_u}} - \frac{sign(\tilde{\beta}_v)\beta_v}{\sqrt{d_v}} \right| \leq E \end{aligned} \tag{2.16}$$

2.4.5 TTLP and LTLP

Kim et al. [12] suggested two alternative network constrained regression methods based on the penalty shown in Equation 2.17. The first subpenalty is the L_0-loss for sparsest variable selection and unbiased parameter estimation proposed by Shen et al [14]. The second subpenalty encourages simultaneous selection or elimination of neighboring predictors in the network. the penalties are defined with the use of indicator functions notation explained in the following subsection.

$$P(\beta) = \lambda_1 \sum_{i=1}^{p} [|\beta_i| \neq 0] + \lambda_2 \sum_{u\sim v} \left| \left[\frac{|\beta_u|}{w_u} \neq 0\right] - \left[\frac{|\beta_v|}{w_v} \neq 0\right] \right| \tag{2.17}$$

Indicator functions

An indicator (characteristic) function is defined on a set X and some subset $A \subset X$. The function indicates membership of elements in the subset A, having value of 1 for all elements of A and value of 0 for all other elements in X. Formally, indicator functions are defined as follows:

$$[x \in A] = 1_A(x) = \begin{cases} 1 & if \quad x \in A \\ 0 & if \quad x \notin A \end{cases} \quad for\ each\ x\ in\ X \tag{2.18}$$

Note that the Iverson bracket notation $[P(x)]$ can be used equivalently to denote the indicator function of elements for which the condition P is true.

TTLP

Because the indicator function is not continuous, Shen et al. [14] proposed a truncated Lasso penalty (TLP) J_τ for a computational substitute. The TLP penalty, defined in Equation 2.19, tends to $[|z| \neq 0]$ as $\tau \to 0^+$. The tuning parameter τ determines the degree of approximation.

$$J_\tau(|z|) = min\left(\frac{|z|}{\tau}, 1\right) \tag{2.19}$$

Applying the TLP substitute to Equation 2.17 produces the $TTLP_I$ penalty, shown in Equation 2.20, which uses TLP for both variable selection and grouping.

$$P(\beta) = \lambda_1 \sum_{i=1}^{p} J_\tau|\beta_i| + \lambda_2 \sum_{u \sim v} \left| J_\tau\left(\frac{|\beta_u|}{w_u}\right) - J_\tau\left(\frac{|\beta_v|}{w_v}\right)\right| \tag{2.20}$$

LTLP

As an alternative to the TTLP, Kim et al. proposed a modification of their penalty using the Lasso for variable selection. The modified penalty, which the authors call $LTLP_I$, is shown in Equation 2.21.

$$P(\beta) = \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sum_{u \sim v} \left| J_\tau\left(\frac{|\beta_u|}{w_u}\right) - J_\tau\left(\frac{|\beta_v|}{w_v}\right)\right| \tag{2.21}$$

Chapter 3

Implementation Details

Python is the main programming language used in developing the system to prepare, execute and evaluate the experiments discussed in this report. However, Matlab has been used to implement most of the network constrained regression methods. This is due to the extensive use of Matlab's CVX package [15][16] for solving the convex optimization problems defined by the various regression methods. The MATLAB Engine API for Python has been used to integrate the two platforms and enable the invocation of Matlab functions from a Python environment.

The various libraries widely used in the implementation of this project are introduced as follows:

Pandas (Python) is used for data wrangling tasks and its data structures

Numpy (Python) is used for its implementation of N-dimensional arrays and the wide range of operations performed on them

Scipy (Python) is used to minimize the non-convex objective function of the GBLasso (2.4.3) method

Scikit-Learn (Python) is used to obtain OLSE, Lasso and Elastic Net estimates, calculate cosine vector similarity and other model metrics

Matlab Engine (Python) is used for Python-Matlab integration and enables invocation of Matlab functions from a Python environment

CVX (Matlab) is used to find a solution to the convex optimization problems defined by the objective functions of the Grace, Linf and TLP

Matplotlib (Python) is used for the plotting of various figures

The various regression methods discussed in Chapter 2 are implemented as described in Table 3.

Table 3.1: Regression methods implementation details

Platform	Library	Class/function/solver used	Regression Method
Python	Scikit-Learn	LinearRegression	OLSE (2.2)
		Lasso, LassoCV	Lasso (2.3.2)*
		ElasticNet, ElasticNetCV	Elastic Net (2.3.3)*
	Scipy	minimize	GBLasso (2.4.3)
Matlab	CVX	SDPT3	Grace (2.4.1)
		SDPT3	aGrace (2.4.2)
		SDPT3	Linf (2.4.4)
		SDPT3	aLinf (2.4.4)
		SDPT3	TTLP (2.4.5)
		SDPT3	LTLP (2.4.5)

* Scikit-Learn's implementation of the Lasso and Elastic Net does not control the $L1$ and $L2$ penalties independently with hyperparameters λ_1 and λ_2 as discussed in Section 2.3 and shown in Equation 3.1.

$$P(\beta) = \lambda_1 L1 + \lambda_2 L2 \tag{3.1}$$

Instead, the hyperparameters $alpha$ and $l1\_ratio$ are used, $alpha > 0$ controlling the magnitude of penalization and $l1\_ratio \in [0, 1]$ defining the level of contribution of each penalty. The two approaches to parametrization could be expressed equivalently through the relationship shown in Equation 3.2. Specifically, $l1\_ratio = 0$ performs Ridge Regression (2.3.1), $l1\_ratio = 1$ performs the Lasso (2.3.2) and $l1\_ratio \in (0, 1)$ performs Elastic Net (2.3.3) regression.

$$\begin{aligned} alpha &= \lambda_1 + \lambda_2 \\ l1\_ratio &= \frac{\lambda_1}{\lambda_1 + \lambda_2} \end{aligned} \tag{3.2}$$

Chapter 4

Synthetic Dataset Generation

Synthetic datasets have been generated for use in the hyperparameter tuning process for the various regression methods. These synthetic datasets have been designed to be very similar to real epigenetic datasets. The assumption is that the various regression methods would continue performing well in the context of real epigenetic data after having been tuned on similar generated datasets.

The benefit of using synthetic data for parameter tuning is the existence of ground truth about the relationship between predictors and the target variable. This ground truth enables comparing the various regression methods not only in terms of prediction error, but also with regard to the sensitivity, specificity and precision of their variable selection.

The sections of this chapter describe in detail the synthetic dataset generation process. The gene network, simulated expression levels of all genes and the primary simulation setups of the response variable are implemented as suggested by Li and Li [8]. Four secondary simulation setups are derived from each of the four primary setups, resulting in a total of 20 different simulation setups.

4.1 Predictor network generation

All simulation setups share the following common predictor network. Consider a setup, for which 50 transcription factors regulate 10 independent genes each. In the gene network corresponding to this scenario, there would be edges between all transcription factors (TFs) and their 10 regulated genes.

The resulting network contains 550 nodes and 500 edges, all edge weights set to 1. This graph consists of 50 star-shaped connected components of 11 nodes, the central node of each representing the corresponding TF.

4.2 Generation of predictor observations

As a consequence of using the shared predictor network described in the previous section, all synthetic datasets contain 550 predictors. The expression levels for each of the 50 transcription factors follow a standard normal distribution $X_{TF_j} \sim N(\mu = 0, \sigma = 1)$.

The expression level of the regulated genes (RG) is dependent on the expression level of their corresponding TF_j and follows the normal distribution $X_{RG} \sim N(\mu = 0.7 * X_{TF_j}, \sigma = 0.71)$. This means that the expression levels of a TF and each of its RG are jointly distributed as a bivariate normal with a correlation of 0.7.

The predictor order in the expression matrix X is shown in Equation 4.1.

$$[TF_1, RG_{1,1}, ..., RG_{1,10}, ..., TF_{50}, RG_{50,1}, ..., RG_{50,10}] \tag{4.1}$$

4.3 Response variable generation

Values of the response variable y are generated according to a linear model $y = X\beta + \epsilon$, where ϵ is added noise and the coefficient vector β is specified by the current simulation setup.

The added noise follows a normal distribution $\epsilon \sim N(\mu = 0, \sigma = F(\beta)))$, whose shape is calculated from the coefficient vector β for the current setup according to equation 4.2.

$$\sigma_\epsilon = F(\beta) = \sqrt{\frac{\sum_{j=1}^{p} \beta_j^2}{4}} \tag{4.2}$$

4.3.1 Primary simulation setups

The four primary simulation setups assume that four transcription factors and their regulated genes are related to the response variable y. Therefore, only the first 44 coefficients of the coefficient vector β are non-zero.

Setup 1

The first model, shown in Equation 4.3, assumes that the regulated genes of each transcript factor affect the response variable in the same way as the TF itself, either positively or negatively.

$$\begin{aligned}\beta = (5,\ & \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}},\\ -5,\ & \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}},\\ 3,\ & \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}},\\ -3,\ & \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}},\\ 0,\ & \ldots,\ 0)\end{aligned} \tag{4.3}$$

Setup 2

The second model, shown in Equation 4.4, assumes that the regulated genes of each transcript factor can have both positive and negative effects on the response variable.

$$\begin{aligned}\beta = (5,\ & \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}},\\ -5,\ & \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}}, \tfrac{-5}{\sqrt{10}},\\ 3,\ & \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}},\\ -3,\ & \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}}, \tfrac{-3}{\sqrt{10}},\\ 0,\ & \ldots,\ 0)\end{aligned} \tag{4.4}$$

Setup 3

The third setup, shown in Equation 4.5, is similar to the first setup, but with reduced effect magnitude of the regulated genes.

$$\begin{aligned}\beta = (5,\ & \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10},\\ -5,\ & \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10},\\ 3,\ & \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10},\\ -3,\ & \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10},\\ 0,\ & \ldots, 0)\end{aligned} \tag{4.5}$$

Setup 4

The fourth setup, shown in Equation 4.6, is similar to the second setup, but with reduced effect magnitude of the regulated genes.

$$\begin{aligned}\beta = (5,\ & \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10},\\ -5,\ & \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10}, \tfrac{-5}{10},\\ 3,\ & \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10},\\ -3,\ & \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10}, \tfrac{-3}{10},\\ 0,\ & \ldots, 0)\end{aligned} \tag{4.6}$$

4.3.2 Secondary simulation setups

Four secondary setups have been derived from each primary simulation setup. As discussed in Section 4.3.1, the primary setups define a linear model relating four transcript factors and their regulated genes to the response variable. Instead, the derived secondary simulation setups distribute the same effect evenly over 8, 12, 16, and 20 transcript factors and their regulated genes. This enables to tune and compare the various regression methods in scenarios with a varying number of relevant predictors.

Chapter 5

Composite Voting Regression

Several different methods for linear regression are present in the context of this research. They all operate on the same input data and share a similar goal. However, the solutions produced by the different methods can vary greatly due to differences in the objective functions. Each method exhibits its strengths and weaknesses in performance when processing different input data and no method is better than the others in all considered scenarios. For this reason it is not possible to claim that any of the methods would be superior for an arbitrary real dataset. As a way to reduce inconsistencies in performance, our solution is to independently process the data with a selected set of regression methods and merge their outputs.

All regression methods considered in this report perform variable selection while building their linear models. However, the sets of predictors selected by the different methods are rarely identical. We propose a voting scheme to merge the results of variable selection in the context of multiple methods for multiple linear regression.

Let there be k different regression methods processing a dataset with p predictors. We compose the matrix B of size k-by-p, where each element $B_{i,j}$ is the coefficient corresponding to the predictor j estimated by the method i as shown in 5.1.

	Predictor 1	*Predictor* 2	...	*Predictor* p
Method 1	$M_1(\beta_1)$	$M_1(\beta_2)$	...	$M_1(\beta_p)$
Method 2	$M_2(\beta_1)$	$M_2(\beta_2)$	...	$M_2(\beta_p)$
...	...	...	...	...
Method k	$M_k(\beta_1)$	$M_k(\beta_2)$	...	$M_k(\beta_p)$

(5.1)

For each predictor we count the number of non-zero coefficients in its corresponding column. This is equivalent to calculating how many of the methods consider the current predictor relevant to the target variable. We then calculate the fraction of votes FV for the predictor's importance by dividing that number by k as shown in Equation 5.2. FV would then be in the range $[0, 1]$, 0 meaning that all regression methods consider the predictor unimportant, while 1 indicates complete agreement on the predictor's relevance to the response.

$$FV_j = \frac{\sum_{i=1}^{k} B_{i,j}}{k}, \text{ where j is the index of the current predictor} \tag{5.2}$$

The FV statistic can be used to perform variable selection by retaining only the predictors with score above a set threshold. Different predictor selection strategies can be implemented through modifying the vote fraction threshold, for example selecting a predictor if *all* /*any* / *at least* 50% of the predictors consider it relevant to the target variable.

After variable selection is performed, predictor coefficients can be estimated through any regression approach using the selected set of predictors. We use the OLSE (2.2) for this estimation to avoid using a hyperparameterized approach.

The proposed composite voting regression approach, denoted as Composite in the following chapters, is designed to balance the benefits and drawbacks of its underlying regression methods when processing arbitrary datasets. It is primarily a method for output fusion of the contained regression approaches. As a result, its performance depends on both the selection of underlying regression approaches and the choice of hyperparameter values for each of them.

Chapter 6

Hyperparameter Tuning

Many machine learning algorithms have one or more hyperparameters, whose role is to modify different aspects of the learning process. Their values are not estimated through use of the training data, but must be chosen prior to the start of the learning process. Choosing a set of suitable hyperparameter values for a learning algorithm, also called model selection, is an important step to ensure that the algorithm performs well and does not overfit the training data.

6.1 Traditional approaches

Grid search

Grid search, also called parameter sweep, is commonly used to perform hyperparameter optimization. A predefined set of values is selected for each of the tuning parameters used by the learning algorithm. Models are then trained with each possible combination of tuning parameter values. All models are evaluated according to some performance metric. The combination of parameter values that produces the model performing best (minimizing/maximizing the performance metric) is chosen as optimal. The performance metric used typically in regression problems is the mean squared prediction error (MSE), calculated as shown in Equation 6.1.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \tag{6.1}$$

where $\hat{y}_i$ is the model's predicted value and y_i is the true value of the target.

The validation set approach

Estimating a model's MSE using the same data it was trained on is not truly indicative of the model's performance due to the possibility of overfitting. One way to measure a model's accuracy of prediction on unseen data is to calculate generalization error, also known as out-of-sample error.

This could be done through the validation set approach by partitioning the available data into two mutually exclusive subsets called training and test (validation) sets. Models are then trained on the training subset of observations and their prediction MSE is calculated using the test set, also called holdout. However, this method has the following drawbacks [17]:

- The validation estimate of the test error rate can vary greatly depending on the training-validation set partitioning
- The method makes inefficient use of the data as a significant part of the observations are never used for training

K-fold cross validation

Cross validation [17, 18] is a resampling method closely related to and addressing the drawbacks of the validation set approach. The often used k-fold cross-validation involves partitioning the data into k subsets (folds). One of the folds is treated as a validation set and the model is trained on the remaining $k-1$ folds. The mean squared error for the fold MSE_i is computed for the observations in the held-out fold i. The process is repeated k times, each of the folds being held out once, and the k-fold CV estimate is obtained by averaging the estimates for the different folds as shown in Equation 6.2.

$$MSE_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{6.2}$$

Grid search minimizing the cross-validated MSE

One traditional approach of performing model selection uses a grid search with cross-validated mean squared prediction error for a metric of model performance. The final model is then trained on the whole dataset using the optimal hyperparameter values that minimize the cross-validated MSE. This model selection approach is denoted as "CV-MSE" tuning in the latter chapters of this report.

6.2 Orchestrated parameter tuning

6.2.1 Context and motivation

In the context of this project multiple methods of performing regression are defined, each of them minimizing a different objective function. All approaches operate on the same training data and share a common goal to correctly identify the relationship between predictors and the target variable.

Traditionally, we would tune the hyperparameters for each regression method independently before processing the desired data with the optimal parameter combinations for each method. Such an approach would not benefit from having an ensemble of regression approaches as they would all function completely independently. Our aim was to develop a hyperparameter optimization approach that would use this presence of multiple alternative approaches to perform simultaneous and cooperative parameter tuning.

6.2.2 Inspiration

Our novel hyperparameter tuning approach presented in Section 6.2.3 is inspired by a different kind of ensemble, that of a philharmonic orchestra. Much like the different regression methods, every musical instrument produces its own distinguishing sound even when playing the same melody. All of the various instruments complement each other and contribute in their own way to the skillful masterpiece that is a symphony.

During a performance every musician needs to ensure that they are in synchronization with the rest of the orchestra. They need to be constantly aware of what the current general tempo and state of the melody across the whole ensemble is. In case of a mismatch, the performer needs to adjust their own way of playing their instrument to bring it back to synchronization with the other instruments.

The behavior of each regression method in our proposed hyperparameter tuning closely resembles these synchronizing adjustments made by orchestra members during a performance.

6.2.3 Methodology

We propose an alternative approach for simultaneous cooperative hyperparameter tuning that uses an ensemble of regression methods sharing a similar goal and input data. Our tuning method features an iterative algorithm that attempts to increase the similarity between parameter estimates of the various regression approaches at each step.

Iterative procedure

Initially, each of the methods is trained using some hyperparameter combination in their respective search grids, forming the method outputs for the "zero" iteration. The choice of this initial starting point is discussed in a subsection below.

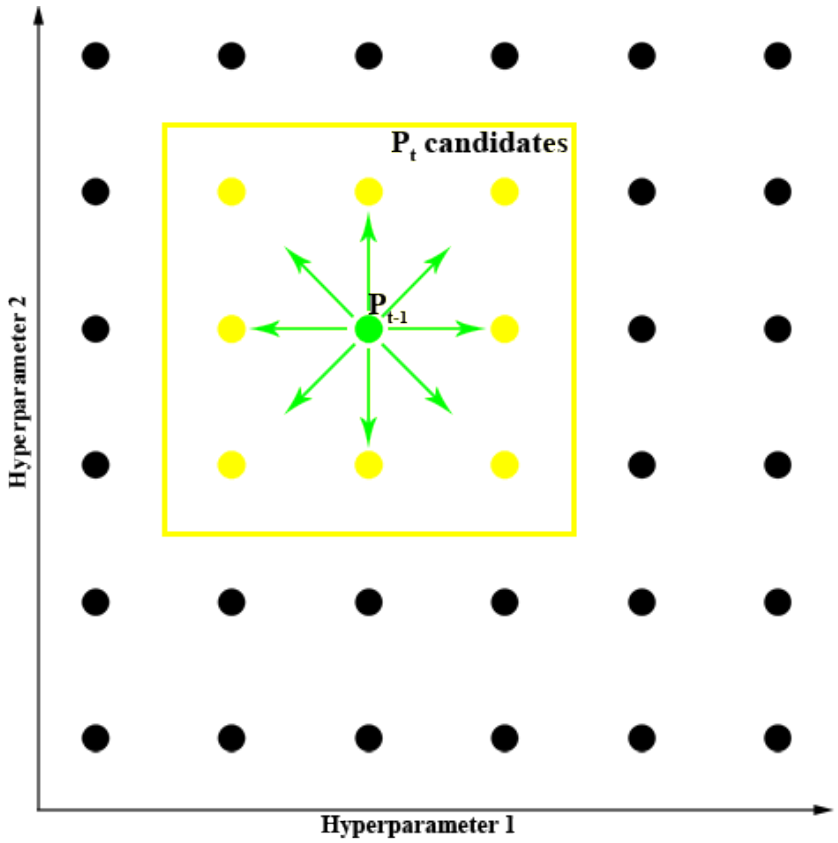

Figure 6.1: Local search grid neighborhood, two tuning hyperparameters; P_{t-1} is the parameter combination selected in iteration $t-1$, parameter values in the P_t neighborhood are considered in iteration t

The orchestrated tuning procedure of k regression methods is defined below. The following sequence of steps is performed for each method $M_{i \in 1..k}$ in an arbitrary iteration t.

1. Consider the coefficient vectors $\beta_{j,t-1}$ estimated by all other methods $M_{j \neq i}$ for the previous iteration $t-1$. A target coefficient vector $\beta_{tar,t}$ is created by calculating the mean coefficient value for each predictor estimated by the regression methods $M_{j \neq i}$.

2. Consider the candidate hyperparameter value combinations located in proximity (in the search grid) to the combination selected by the previous iteration P_{t-1}. An example of a set of candidate combinations is shown on Figure 6.1 for a regression method with two tuning parameters. We train the current method M_i using each of the candidate hyperparameter combinations.

3. Consider the estimated coefficient vectors resulting from each of the candidate hyperparameter combinations defined in 2. The combination P_t selected for the current iteration t is the one that maximizes correlation between its estimated coefficient vector and the target vector $\beta_{tar,t}$ defined in 1. The method's estimate $\beta_{i,t}$ for the current iteration is the one produced by the chosen candidate combination.

This iterative procedure converges when any movement along the parameter grids is settled for all methods. The tuning stops when the selected hyperparameter combinations for the current iteration are identical to those of the previous one for all methods. The combinations of hyperparameter values selected in the last iteration are considered optimal for their respective regression methods.

If the parameter search grids for the various methods are coarse, fluctuation between two parameter combinations can occur, preventing convergence. A maximum number of iterations can be defined to force the completion of the tuning process, resulting in an approximate solution.

The temporal relationships that occur between the various regression methods during the orchestrated hyperparameter tuning can be illustrated with the graph shown in Figure 6.2. This abstract structure of method interactions resembles a recurrent neural network.

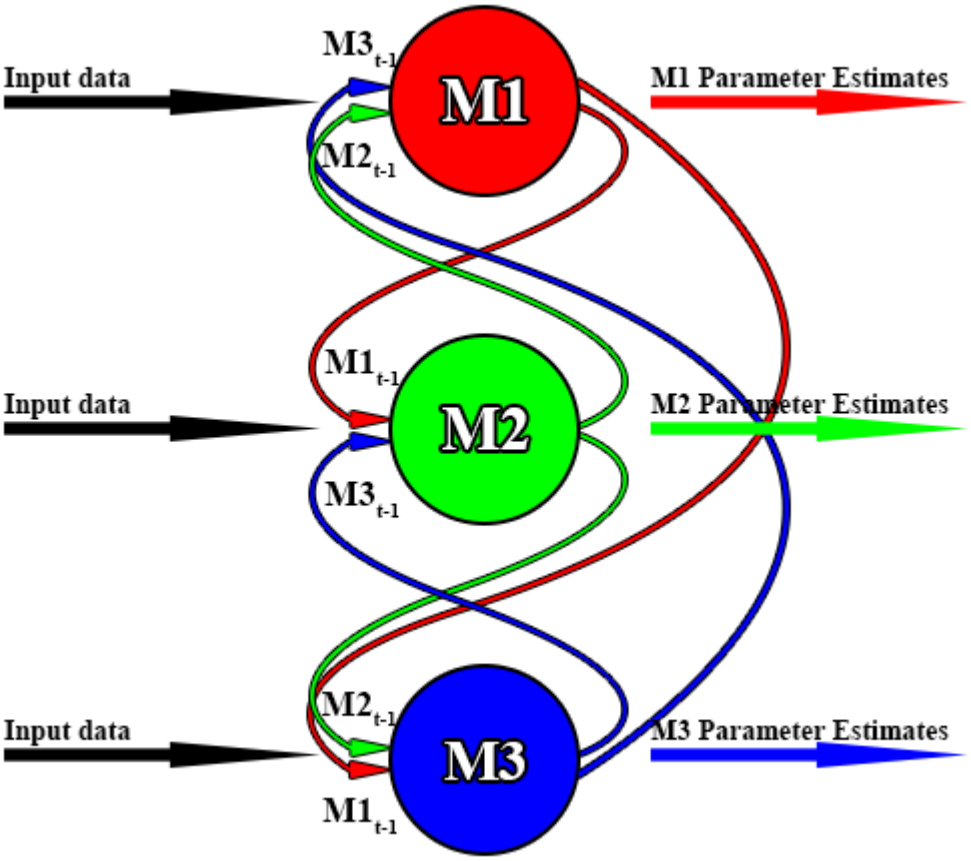

Figure 6.2: Orchestrated tuning abstract structure, three regression methods; $M1_{t-1}$ denotes the parameter estimates of method 1 for iteration $t-1$; Each tuning iteration t of a method uses the common input data and the parameter estimates of all other methods for the previous iteration

Choice of starting points for the search

The choice of hyperparameter starting points in the search grids for the ensemble of regression methods directly affects the convergence state of the tuning process. This initial state of the system can be chosen in one of the following ways:

1. One approach could be to initialize the tuning process with hyperparameter values located in the centers of the search grids of all methods. Such initialization would allow the tuning procedure to decide which orthant of the search space to converge in.

2. An alternative approach would be to start the tuning procedure from specific corners of the search grids. Depending on whether we wish to promote simpler or more complete models, the appropriate grid corner should be chosen, maximizing or minimizing the severity of imposed penalties.

3. The proposed orchestrated hyperparameter tuning could be combined with the traditional tuning approach discussed in Section 6.1. This can be done by obtaining the initial parameter values for the orchestrated tuning from a previously performed CV-MSE tuning. Such a combined tuning procedure could potentially benefit from the strengths of both tuning approaches - selection of models with good generalized prediction capabilities, but also coordinated for better agreement between regression methods.

4. A randomized orchestrated tuning procedure could be constructed by repeatedly performing the search with randomized starting points and collecting statistics from each convergence. The optimal hyperparameter combinations would be selected after analysis of the convergence statistics for all tuning executions.

Discussion

The orchestrated hyperparameter tuning approach presented in this chapter operates in the context of an ensemble of alternative methods sharing a common input and goals. In the case of regression, the approach aims to find combinations of hyperparameter values for the various methods that result in increased similarity of their estimated coefficients.

As previously discussed, the optimization focus is on the estimated predictor coefficients, not on the prediction error. This discards the need to perform cross validation, which could be valuable when working with computationally intensive regression methods. What is more, not optimizing the hyperparameters in terms of prediction error could result in reduction of overfitting.

The proposed hyperparameter tuning approach introduces cooperation between otherwise completely independent regression methods. This collective simultaneous tuning promotes consensus and agreement between the different methods. The resulting increased similarity of estimates simplifies the results interpretation because fewer conflicts between method outputs occur.

Chapter 7

Simulation studies

Suitable search spaces have been selected for the hyperparameters of the various regression methods discussed in Chapter 2. Hyperparameter tuning has been performed using both approaches discussed in Chapter 6 on synthetic datasets generated according to the specification defined in Chapter 4. Various model metrics have been calculated to evaluate the performance of the different regression methods, as well as compare the two parameter tuning approaches.

7.1 Simulation setup

A bundle of 20 synthetic datasets has been generated according to the specification discussed in Chapter 4. Each of the datasets contains 400 observations and is split into a training dataset of 300 and test dataset of 100 observations. Both hyperparameter tuning and fitting of the final models for each approach have been performed on the training dataset. All model metrics have been calculated for the final models on the test dataset.

7.2 Hyperparameter search spaces

Suitable hyperparameter values have been considered in the tuning of the various regression methods. The hyperparameter search spaces for each regression method are shown in Table 7.2. Values for the TTLP and LTLP methods are selected as suggested by [12].

Table 7.1: Hyperparameter search space for all regression methods

Method name	Parameter	Search space
Lasso (2.3.2)	Alpha (α)	100 values chosen by Scikit-Learn's implementation of the Lasso
ENet (2.3.3)	Alpha (α)	100 values chosen by Scikit-Learn's implementation of the Elastic Net
	L1 ratio	[0.1, 0.25, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, .95, .99]
Grace (2.4.1)	Lambda 1 (λ_1)	[0.01, 0.1, 1, 10, 100, 1000, 10000]
	Lambda 2 (λ_2)	[0.01, 0.1, 1, 10, 100, 1000, 10000]
aGrace (2.4.2)	Lambda 1 (λ_1)	[0.01, 0.1, 1, 10, 100, 1000, 10000]
	Lambda 2 (λ_2)	[0.01, 0.1, 1, 10, 100, 1000, 10000]
GBLasso (2.4.3)	Gamma (γ)	[2, 3, 4]
	Lambda (λ)	[0.01, 0.1, 1, 10, 100, 1000, 10000]
Linf (2.4.4)	C	[5, 10, 15, ..., 100]
aLinf (2.4.4)	E	[5, 10, 15, ..., 100]
TTLP (2.4.5)	Delta 1 (δ_1)*	3 evenly spaced values in the range $[t, \frac{pt}{4}]$**
	Delta 2 (δ_2)*	3 evenly spaced values in the range $[t, tg]$**
	Tau (τ)	3 evenly spaced values in the range $[10^{-6}, \frac{t}{2}]$**
LTLP (2.4.5)	Delta 1 (δ_1)*	3 evenly spaced values in the range $[\frac{\hat{\lambda_{lasso}}}{1.5}, 1.5\hat{\lambda_{lasso}}]$**
	Delta 2*, Tau	Same as TTLP
*Where Lambda 1 (λ_1) $= f(\delta_1, \tau) = \frac{\delta_1}{\tau}$ and Lambda 2 (λ_2) $= f(\delta_2, \tau) = \frac{\delta_2}{\tau}$		
**Where t is the maximum absolute value of the coefficients estimated by the Lasso, p is the number of predictors, g is the number of edges in the network and $\hat{\lambda_{lasso}}$ is the tuning parameter value selected by the Lasso		
Composite (5)	Vote Threshold	[0.1, 0.2, 0.3, ... 1]

7.3 Model metrics

A combination of prediction evaluation and variable selection metrics has been used to compare the regression and hyperparameter tuning methods discussed in the previous chapters.

7.3.1 Prediction evaluation metrics

The mean squared error (MSE) is calculated as defined in Section 6.1 on the independent test datasets. / Lower is better /

7.3.2 Variable selection metrics

Ground truth about the true relationships between the predictors and the target variable is available resulting from the use of synthetic datasets for model selection and evaluation. Knowing the true predictor coefficients, we can define a number of metrics to evaluate the variable selection of each model. Note the use of Iverson bracket notation to express conditional counting in Equations 7.2, 7.3 and 7.4.

Correlation

We define the Correlation metric of a model M as the Pearson correlation coefficient between the estimated coefficient vector β_M and the true coefficient vector β_{true} as shown in Equation 7.1. / Higher is better /

$$Correlation(M) = \rho_{\beta_M, \beta_{true}} = \frac{cov(\beta_M, \beta_{true})}{\sigma_{\beta_M} \sigma_{\beta_{true}}} \tag{7.1}$$

Sensitivity

We define the variable selection Sensitivity of a model M as the fraction of correctly identified relevant predictors. Formally, this is the fraction of predictors with non-zero coefficients in the true coefficient vector β_{true} that correctly have non-zero coefficients in the estimated coefficient vector β_M as shown in Equation 7.2. / Higher is better /

$$Sensitivity(M) = \frac{\sum_{i=1}^{p} [\beta_{true_i} \neq 0, \beta_{M_i} \neq 0]}{\sum_{i=1}^{p} [\beta_{true_i} \neq 0]} \tag{7.2}$$

Specificity

We define the variable selection Specificity of a model M as the fraction of correctly identified irrelevant predictors. Formally, this is the fraction of predictors with zero coefficients in the true coefficient vector β_{true} that correctly have zero coefficients in the estimated coefficient vector β_M as shown in Equation 7.3. / Higher is better /

$$Specificity(M) = \frac{\sum_{i=1}^{p}[\beta_{true_i} = 0, \beta_{M_i} = 0]}{\sum_{i=1}^{p}[\beta_{true_i} = 0]} \tag{7.3}$$

Precision

We define the variable selection Precision of a model M as the fraction of identified predictors that are truly relevant. Formally, this is the fraction of predictors with non-zero coefficients in the estimated coefficient vector β_M that have non-zero coefficients in the true coefficient vector β_{true} as shown in Equation 7.4. / Higher is better /

$$Precision(M) = \frac{\sum_{i=1}^{p}[\beta_{M_i} \neq 0, \beta_{true_i} \neq 0]}{\sum_{i=1}^{p}[\beta_{M_i} \neq 0]} \tag{7.4}$$

7.4 CV-MSE tuning

The traditional hyperparameter optimization approach, discussed in Section 6.1, was used with 5-fold cross-validation to tune all 10 regression methods. The mean model metrics and their corresponding standard deviations for all synthetic datasets are shown on Table 7.2 and Figure 7.1. The following observations can be made regarding the model metric data:

- The relatively large obtained standard deviations for the various model metrics are immediately evident. It must be noted that this standard deviation does not result from multiple executions of the same scenario. It is obtained from evaluation of the metrics for significantly different synthetic datasets, which explains the large standard deviation values.
- All 10 regression methods achieve similar values for MSE and its standard deviation. Because these mean errors are small, their corresponding standard deviations appear especially large.

- The GBLasso regression method consistently performs poorly in terms of variable selection Specificity and Precision. It tends to select a larger subset of the predictors, which could be a characteristic of the method itself or could be due to the differences in implementation. As discussed in Chapter 3, it is the only regression method whose implementation is based on Scipy's minimize function.
- The Linf and aLinf methods perform best in terms of the variable selection metrics Sensitivity, Specificity and Precision. Furthermore, the Linf method also achieves the second lowest mean squared test error out of all regression methods.

Table 7.2: CV-MSE tuning mean model metrics for 20 synthetic datasets

Method	MSE (σ_{MSE})	Correlation ($\sigma_{Correlation}$)	Sensitivity ($\sigma_{Sensitivity}$)	Specificity ($\sigma_{Specificity}$)	Precision ($\sigma_{Precision}$)
Lasso	19.55 (13.15)	0.83 (0.11)	0.56 (0.15)	0.92 (0.05)	0.68 (0.12)
ENet	18.60 (12.65)	0.85 (0.10)	0.67 (0.21)	0.81 (0.22)	0.62 (0.15)
Grace	15.47 (11.33)	0.86 (0.09)	0.95 (0.09)	0.43 (0.27)	0.35 (0.13)
aGrace	20.67 (13.75)	0.71 (0.24)	0.76 (0.23)	0.63 (0.33)	0.46 (0.21)
GBLasso	21.37 (12.23)	0.64 (0.16)	0.99 (0.03)	0.24 (0.19)	0.31 (0.17)
Linf	16.44 (10.02)	0.78 (0.15)	0.99 (0.02)	0.92 (0.04)	0.78 (0.13)
aLinf	20.16 (10.63)	0.57 (0.25)	0.99 (0.02)	0.92 (0.04)	0.78 (0.13)
TTLP	23.64 (16.87)	0.78 (0.19)	0.74 (0.34)	0.43 (0.46)	0.48 (0.29)
LTLP	20.99 (15.28)	0.80 (0.15)	0.88 (0.11)	0.69 (0.26)	0.51 (0.15)
Composite	21.32 (13.86)	0.75 (0.14)	0.58 (0.17)	0.94 (0.10)	0.78 (0.14)

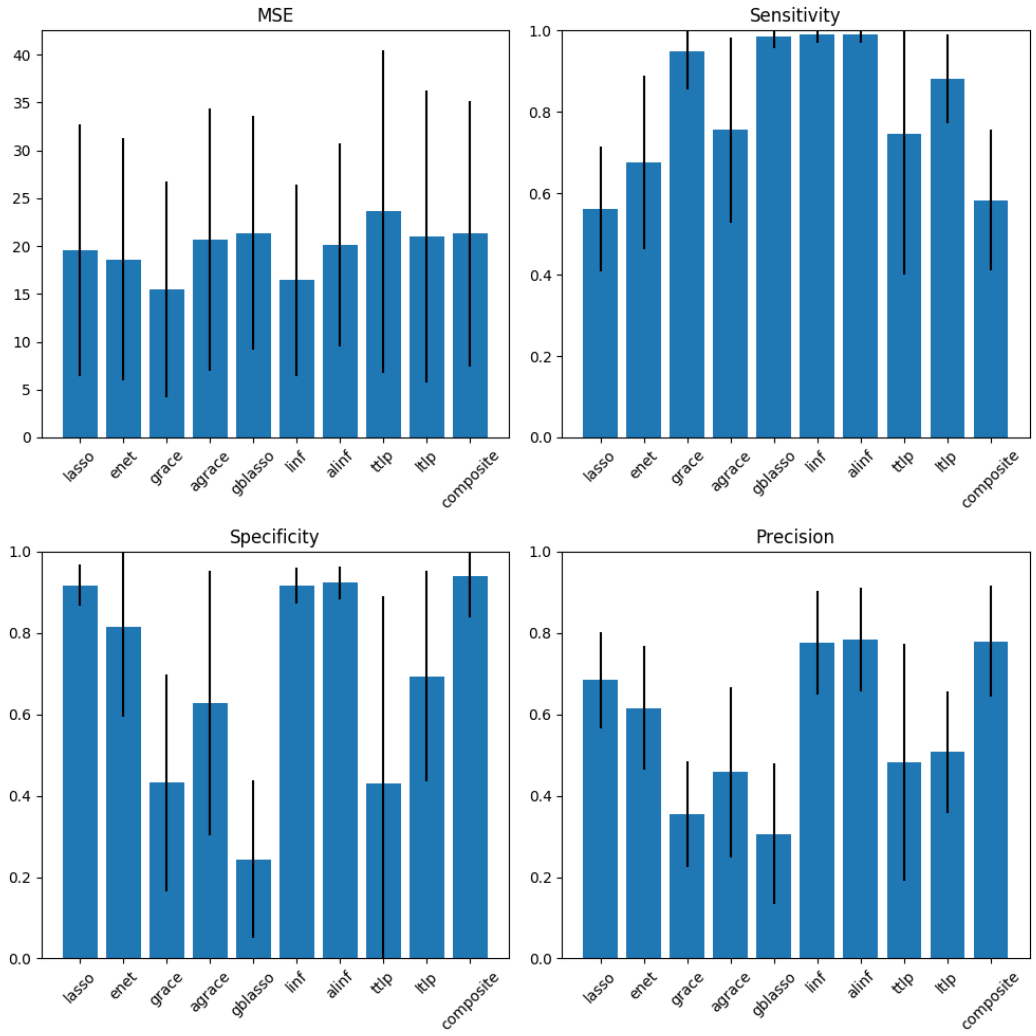

Figure 7.1: CV-MSE tuning mean model metrics with standard deviation error bars for 20 synthetic datasets

7.5 Orchestrated tuning

The orchestrated hyperparameter tuning approach, discussed in Section 6.2, was used to tune a subset of all regression methods. We excluded those approaches that inherently rely on previous estimates by other regression methods, namely the aGrace, aLinf, TTLP, LTLP and Composite. As a result, the ensemble of regression methods used for orchestrated tuning includes the Lasso, Elastic Net, Grace, GBLasso and Linf. Hyperparameter combina-

tions selected by the CV-MSE tuning have been chosen for starting points in initialization of the orchestrated tuning procedure. The mean model metrics and their corresponding standard deviations for all synthetic datasets are shown on Table 7.3 and Figure 7.2. All of the observations made in Section 7.4 continue to be true for the results of this parameter tuning method.

The GBLasso method continues to perform poor variable selection. However, in the context of orchestrated tuning its poor performance has a direct effect on the parameter tuning of all other methods in the ensemble. For some cases it might be suitable to discard such a method from the ensemble in order to avoid possible distortions in the tuning process. The discarded method can either be tuned separately or not considered at all in the experiments.

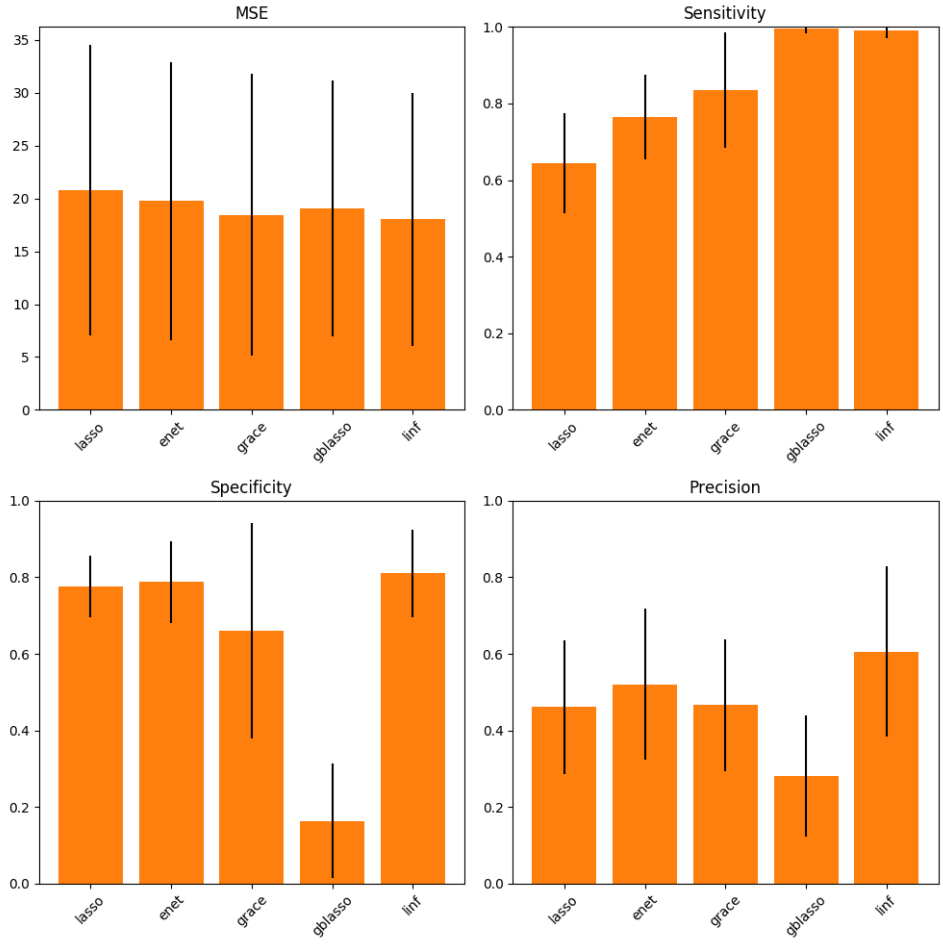

Figure 7.2: Orchestrated tuning mean model metrics with standard deviation error bars for 20 synthetic datasets

Table 7.3: Orchestrated tuning mean model metrics for 20 synthetic datasets

Method	MSE (σ_{MSE})	Correlation ($\sigma_{Correlation}$)	Sensitivity ($\sigma_{Sensitivity}$)	Specificity ($\sigma_{Specificity}$)	Precision ($\sigma_{Precision}$)
Lasso	20.78 (13.74)	0.79 (0.13)	0.64 (0.13)	0.78 (0.08)	0.46 (0.18)
ENet	19.78 (13.15)	0.77 (0.12)	0.76 (0.11)	0.79 (0.11)	0.52 (0.20)
Grace	18.46 (13.34)	0.82 (0.12)	0.84 (0.15)	0.66 (0.28)	0.47 (0.17)
GBLasso	19.06 (12.13)	0.67 (0.18)	0.99 (0.01)	0.16 (0.15)	0.28 (0.16)
Linf	18.04 (11.99)	0.74 (0.15)	0.99 (0.02)	0.81 (0.11)	0.61 (0.22)

7.6 Comparison of tuning approaches

The model metrics results from Sections 7.4 and 7.5 have been merged for easier visual comparison of the different parameter tuning approaches. The combined results are shown in Table 7.4 and Figure 7.3.

Table 7.4: Comparison of mean model metrics by parameter tuning method

Method	Tuning	MSE	Correlation	Sens	Spec	Prec
Lasso	CV-MSE	19.55	0.83	0.56	0.92	0.68
	Orchestrated	20.78	0.79	0.64	0.78	0.46
ENet	CV-MSE	18.6	0.85	0.67	0.81	0.62
	Orchestrated	19.78	0.77	0.76	0.79	0.52
Grace	CV-MSE	15.47	0.86	0.95	0.43	0.35
	Orchestrated	18.46	0.82	0.84	0.66	0.47
GBLasso	CV-MSE	21.37	0.64	0.99	0.24	0.31
	Orchestrated	19.06	0.67	0.99	0.16	0.28
Linf	CV-MSE	16.44	0.78	0.99	0.92	0.78
	Orchestrated	18.04	0.74	0.99	0.81	0.61

The difference between model metrics is immediately evident - in some cases the use of orchestrated tuning has improved certain model metrics for a subset of the regression methods, while the performance of other methods has been flawed. For example, the Specificity and Precision of the Grace method have been significantly improved at the cost of slightly reduced Sensitivity. Conversely, Sensitivity of the Lasso and Elastic Net has been increased at the expense of their Specificity and Precision. A number of factors could potentially affect the performance of the proposed orchestrated tuning approach:

- The choice of alternative methods forming the orchestrated ensemble
- The hyperparameter search spaces selected for each ensemble method
- The starting points used to initialize the orchestrated tuning procedure

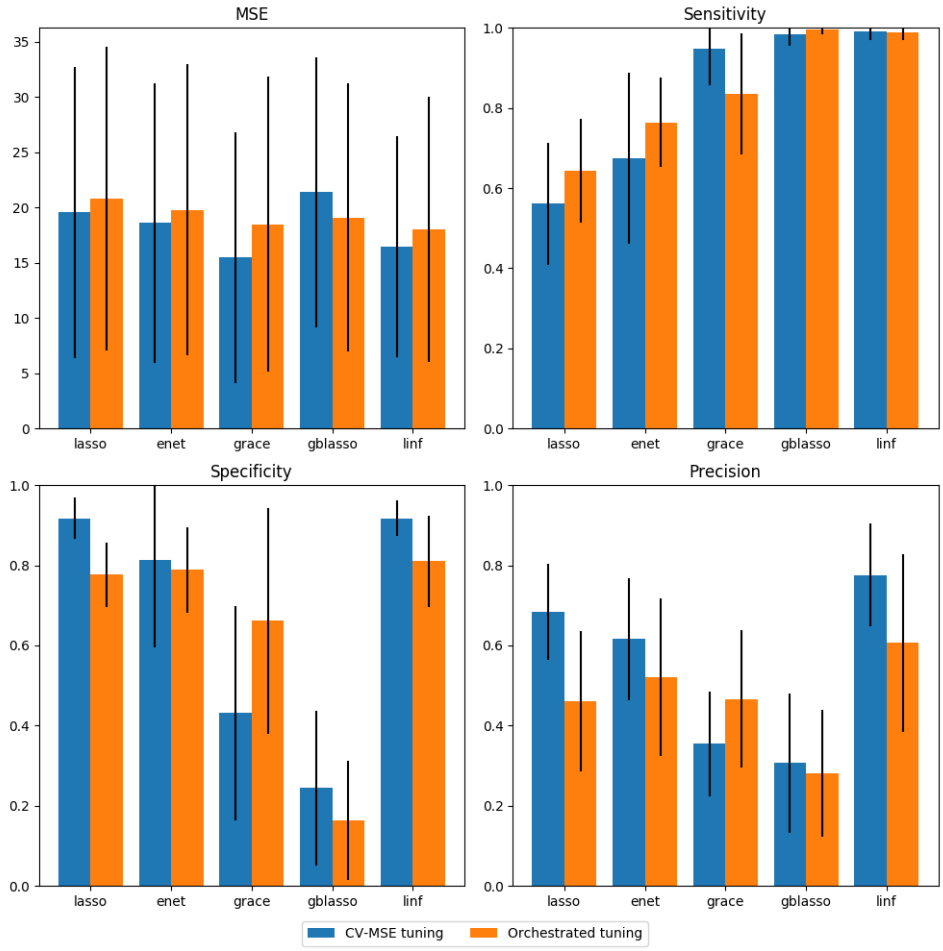

Figure 7.3: Comparison of mean model metrics by parameter tuning method

7.7 Optimal hyperparameter value selection

Data about the optimal hyperparameter values selected when performing both kinds of parameter tuning on all synthetic datasets has been collected. It is used to select overall best combination of hyperparameter values for each regression method for use on a real dataset similar to the synthetic datasets.

7.7.1 Lasso

The distribution of tuning choices for the alpha hyperparameter is shown in Figure 7.4. Multiple values in the range $[0.05, 0.2]$ have been selected most times, of which $alpha = 0.1$ is chosen for use on the real dataset.

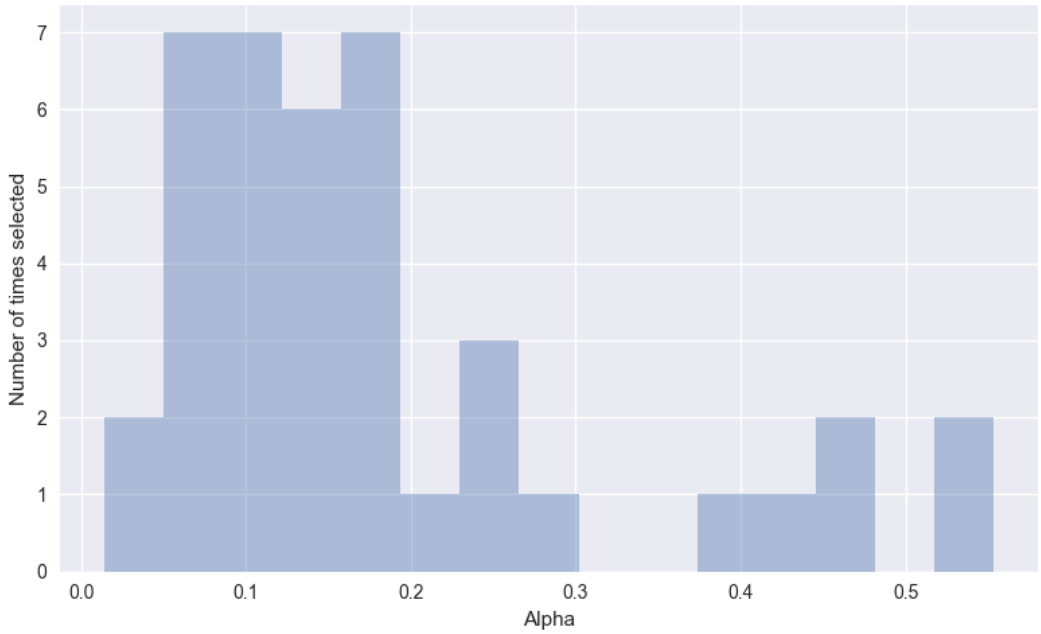

Figure 7.4: Lasso distribution of selected parameter values

7.7.2 Elastic Net

The distribution of tuning choices for combinations of the alpha and L1 ratio hyperparameters is shown in Figure 7.5. The evident choice of alpha value for use on the real dataset is 0.2, but selected values for the L1 ratio parameter are more diversely spread. Although its most commonly selected value is 0.99, we choose a value of $L1\ ratio = 0.8$ for use on the real dataset in order to differentiate the behavior of Elastic Net from that of the pure Lasso.

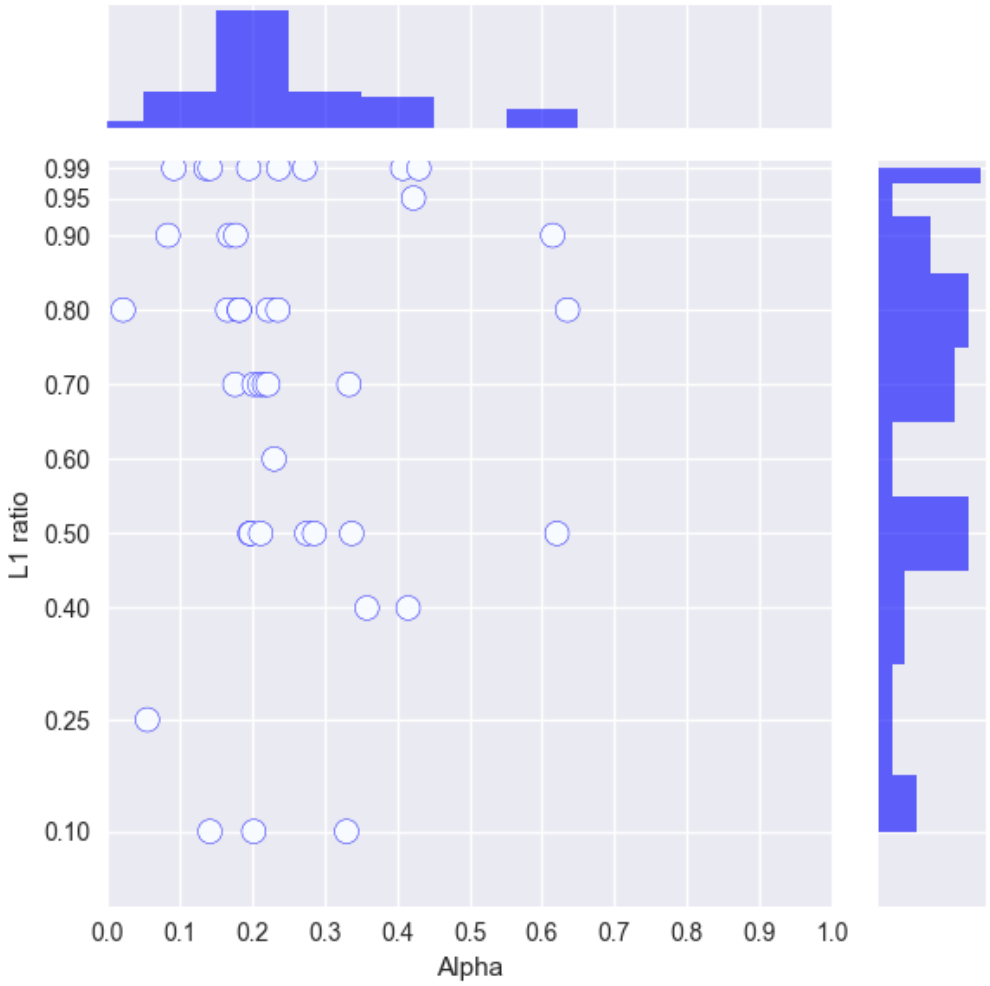

Figure 7.5: Elastic Net distribution of selected parameter combinations

7.7.3 Grace

The distribution of tuning choices for combinations of the lambda 1 and lambda 2 hyperparameters is shown in Figure 7.6. As the results suggest, the most commonly selected tuning parameter combination is ($\lambda_1 = 100, \lambda_2 = 10000$) and it is chosen for use on the real dataset.

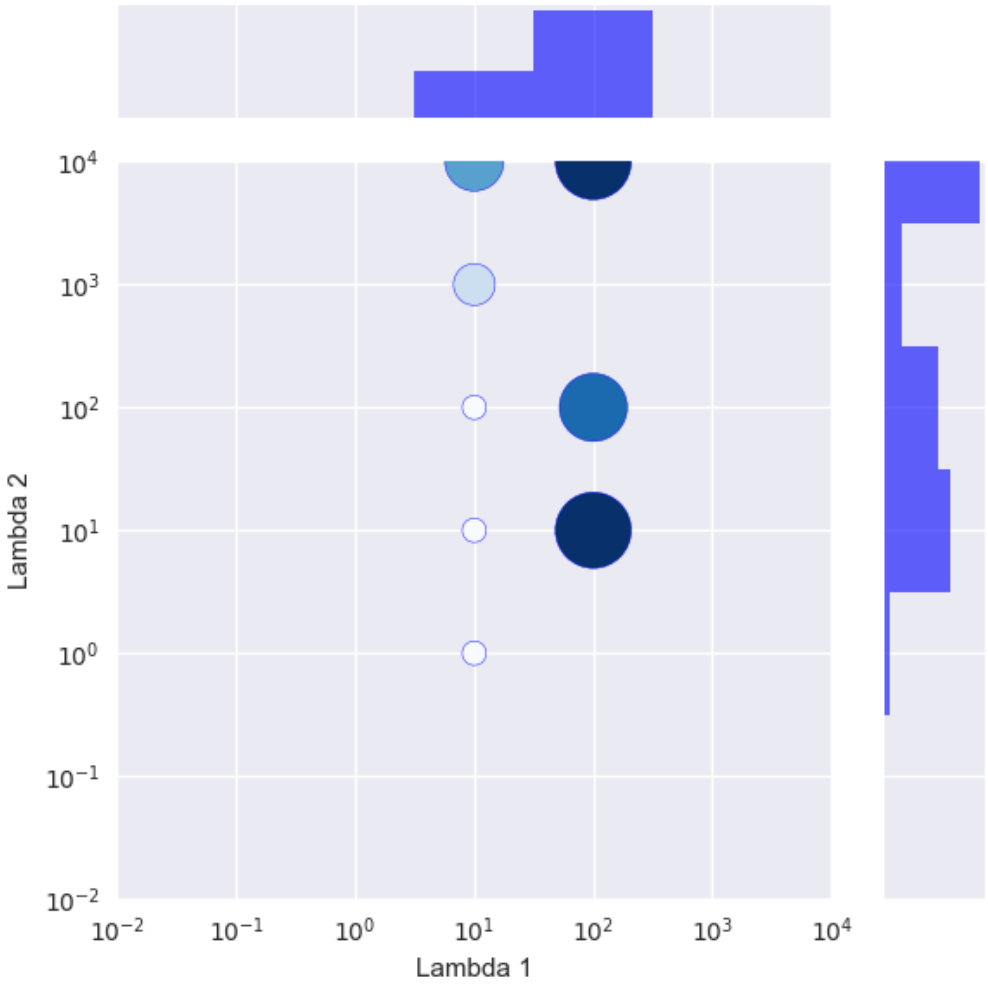

Figure 7.6: Grace distribution of selected parameter combinations; darker blue and larger marker size indicate a higher frequency of choice

7.7.4 aGrace

The distribution of tuning choices for combinations of the lambda 1 and lambda 2 hyperparameters is shown in Figure 7.7. The tuning parameter combination ($\lambda_1 = 100, \lambda_2 = 0.01$) is selected for the largest fraction of the synthetic datasets and therefore is chosen for use on the real dataset.

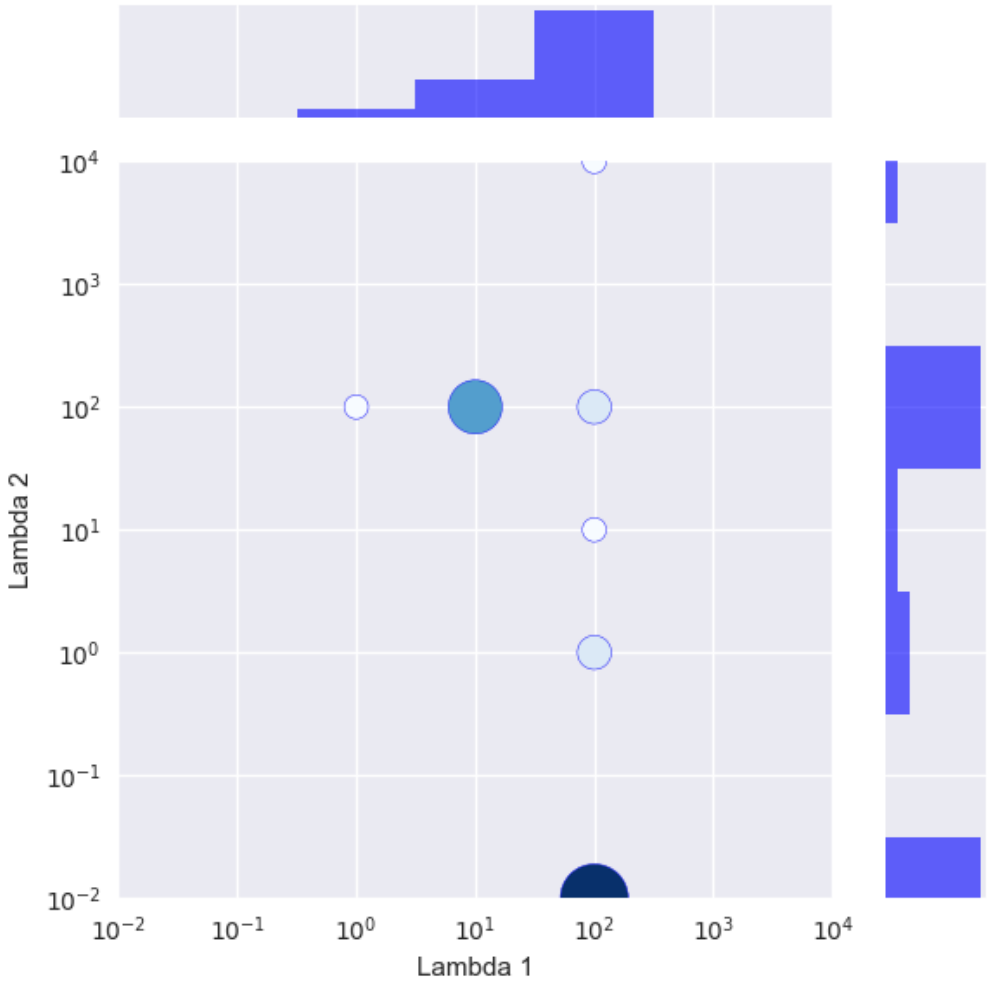

Figure 7.7: aGrace distribution of selected parameter combinations; darker blue and larger marker size indicate a higher frequency of choice

7.7.5 GBLasso

The distribution of tuning choices for combinations of the gamma and lambda hyperparameters is shown in Figure 7.8. The combination of parameter values selected most often is clearly ($\gamma = 4, \lambda = 100$) and it is chosen for use on the real dataset.

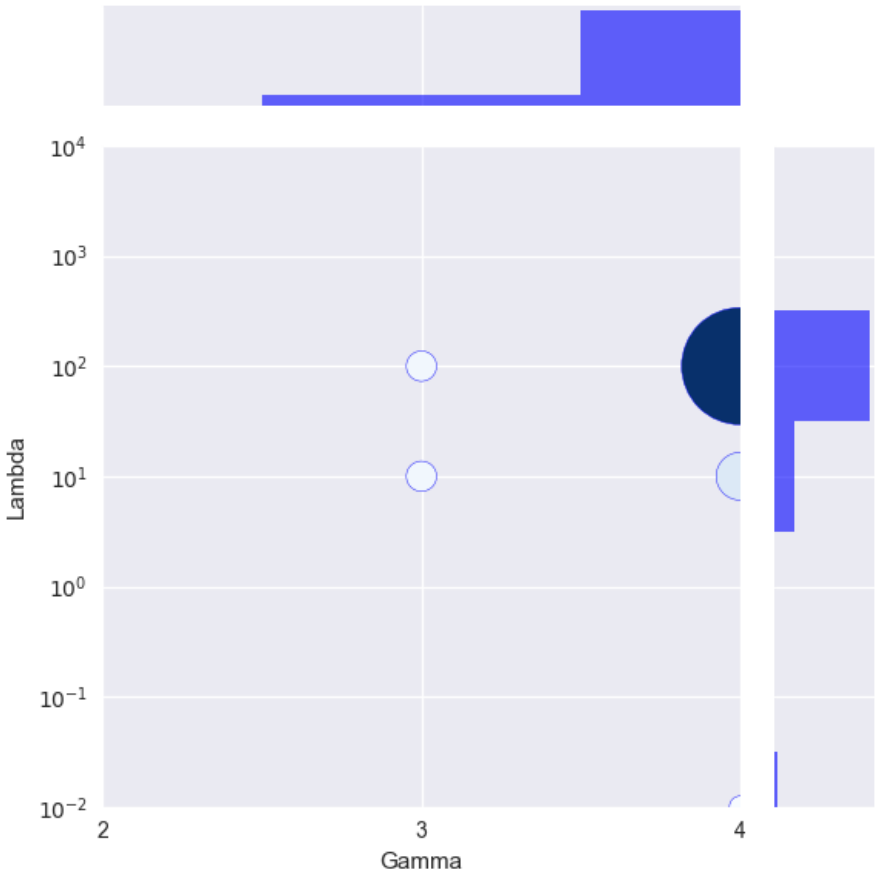

Figure 7.8: GBLasso distribution of selected parameter combinations; darker blue and larger marker size indicate a higher frequency of choice

7.7.6 Linf

The distribution of tuning choices for the C hyperparameter is shown in Figure 7.9. The value selected for most datasets is $C = 25$ and it is chosen for use on the real dataset.

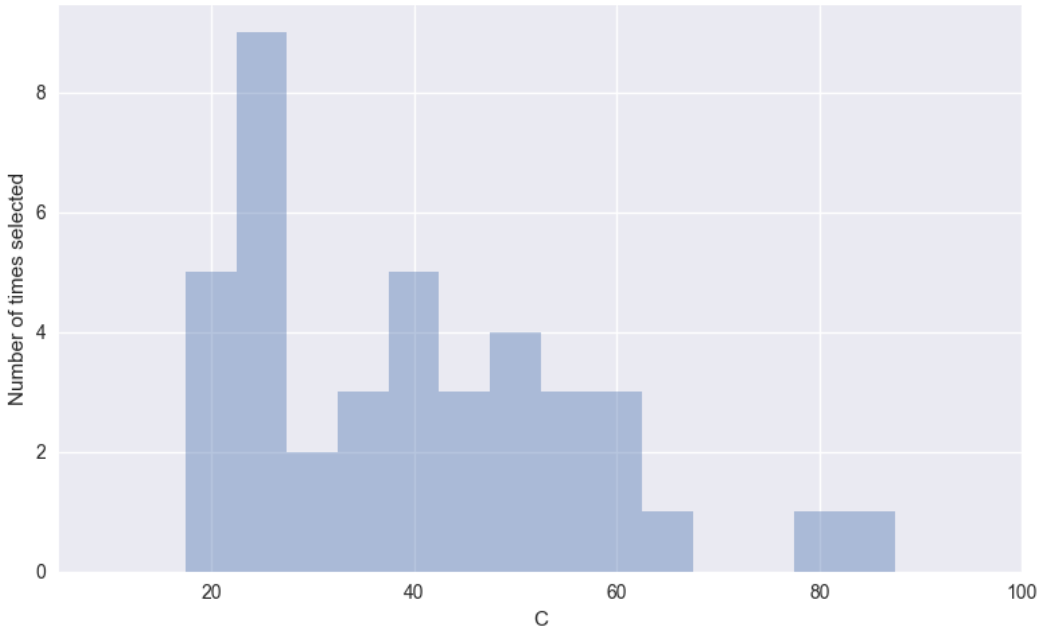

Figure 7.9: Linf distribution of selected parameter values

7.7.7 aLinf

The distribution of tuning choices for the E hyperparameter is shown in Figure 7.10. The values selected most often are 25 and 35, of which $E = 25$ is chosen for use on the real dataset.

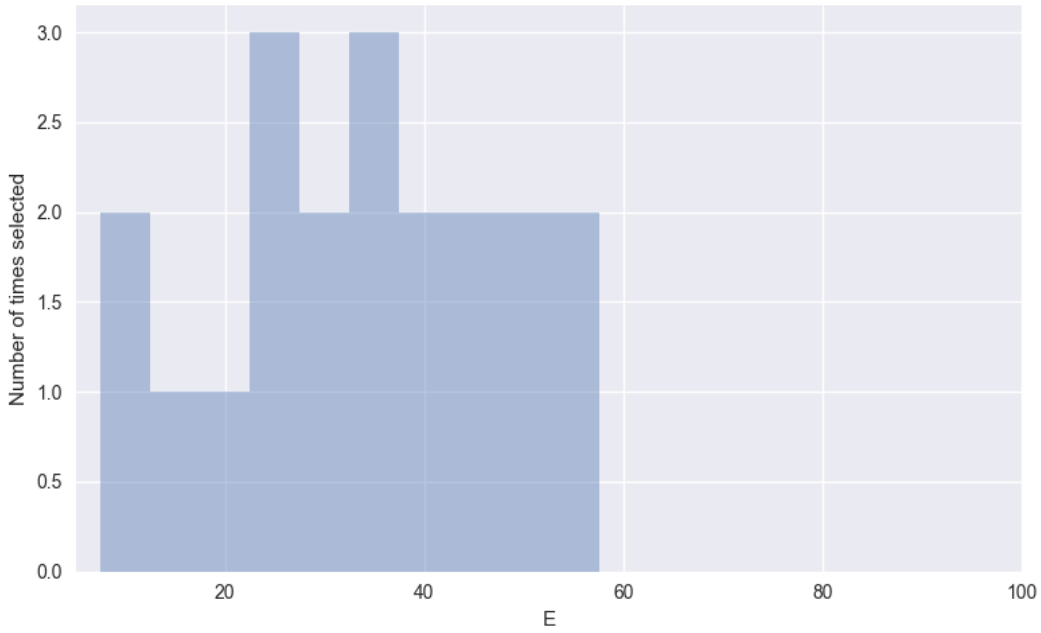

Figure 7.10: aLinf distribution of selected parameter values

7.7.8 TTLP and LTLP

The TTLP and LTLP methods use search spaces for their tuning parameters derived from dataset properties and the Lasso estimate. For this reason attempting to extract an optimal combination of hyperparameter values from tuning with external independent datasets is not feasible.

7.7.9 Composite

The distribution of tuning choices for the vote threshold hyperparameter is shown in Figure 7.11. For most synthetic datasets a value of 0.9 is chosen. Because the methods in the ensemble are less than 10, in practice this means that all underlying regression methods need to have selected a given predictor for it to be selected by the composite method. The vote threshold value of 0.9 is chosen for use on the real dataset.

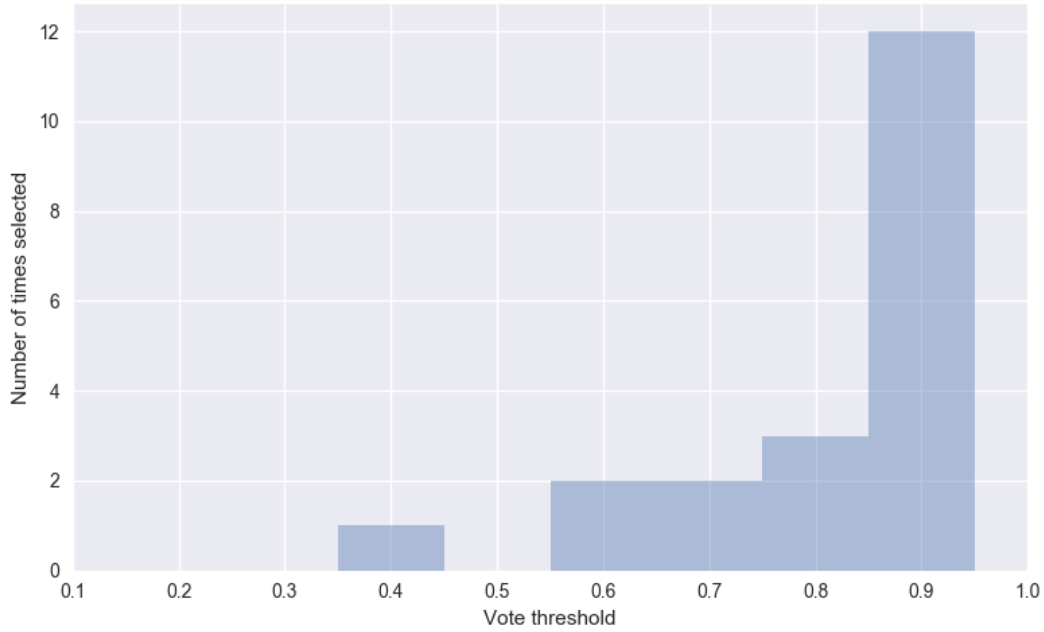

Figure 7.11: Composite method distribution of selected parameter values

Chapter 8

Method Similarity Evaluation

As discussed in previous chapters, in the context of our research multiple different regression methods operate on common input data. The work shown in this chapter aims to determine to what extent the final models produced by the different regression approaches are similar to one another. We also look into whether the orchestrated hyperparameter tuning approach, discussed in Section 6.2, successfully increases this similarity as is intended by design.

The similarity between two linear models is measured with the use of cosine similarity [19], briefly described in the following section. To determine the overall similarity between two regression methods, we calculate the similarity between their final coefficient vectors for each of the synthetic datasets. Final refers to the models produced by fitting the given regression method on the complete training dataset with the optimal hyperparameter combination obtained from the tuning procedure.

8.1 Cosine Similarity

Cosine similarity measures the similarity between two vectors A and B of equal length n. It is defined as shown in Equation 8.1 and produces values in the range $[-1, 1]$. We used the implementation provided by Scikit-Learn through the cosine_similarity function.

$$similarity = cos(\theta) = \frac{A \cdot B}{||A||_2||B||_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{8.1}$$

8.2 Similarity for CV-MSE tuning

Results from the regression method similarity analysis for all synthetic datasets and use of CV-MSE hyperparameter tuning is shown in Table 8.1 and Figure 8.1. The following observations can be made from the obtained results:

- The Lasso and Elastic Net methods produce very similar models. This is to be expected given that the parameter tuning of the Elastic Net commonly selects high values for the L1 ratio(Figure 7.5). As a result, the L1 penalty is a major component of the total Elastic Net penalty.
- The GBLasso and Linf methods also show high similarity, which is unsurprising as the Linf method is derived from the GBLasso (see 2.4.4).
- Our initial expectation was for the TTLP and LTLP methods to be very similar. Surprisingly, the LTLP method bears a stronger similarity to the Lasso and Elastic Net due to its use of the L1 penalty.

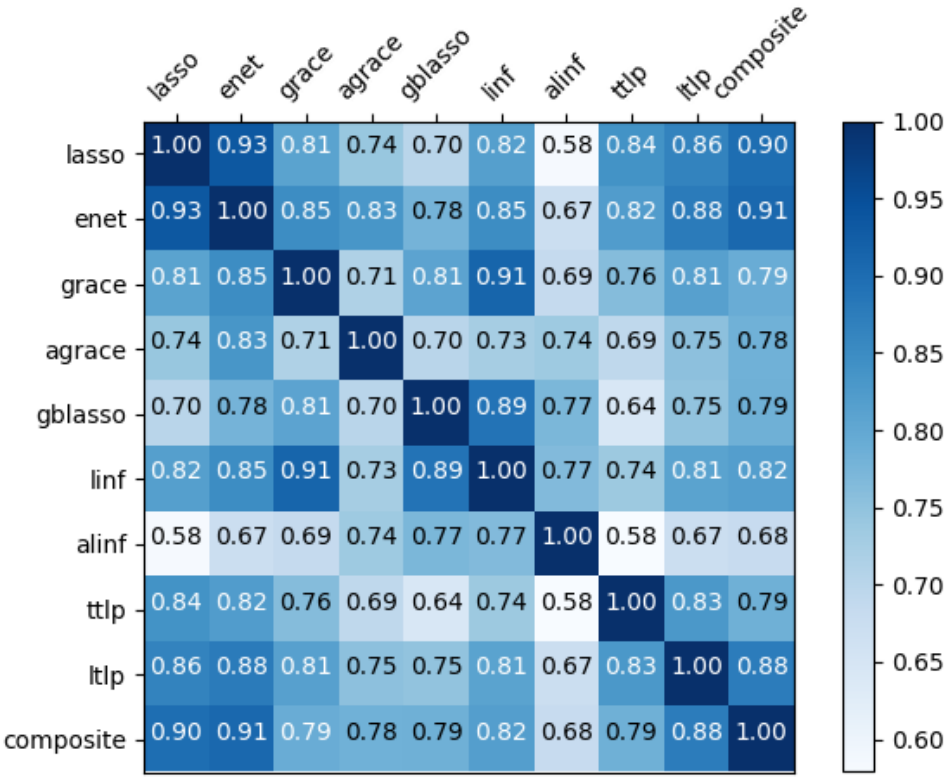

Figure 8.1: CV-MSE tuning mean regression method similarities

Table 8.1: CV-MSE tuning regression method similarities

Method	Lasso (σ_{Lasso})	ENet (σ_{ENet})	Grace (σ_{Grace})	aGrace (σ_{aGrace})	GBLasso ($\sigma_{GBLasso}$)	Linf (σ_{Linf})	aLinf (σ_{aLinf})	TTLP (σ_{TTLP})	LTLP (σ_{LTLP})	Composite ($\sigma_{Composite}$)
Lasso	1.00 (0.00)	0.94 (0.08)	0.81 (0.06)	0.74 (0.28)	0.70 (0.09)	0.82 (0.06)	0.58 (0.21)	0.84 (0.12)	0.86 (0.09)	0.90 (0.10)
ENet	0.94 (0.08)	1.00 (0.00)	0.85 (0.08)	0.83 (0.18)	0.78 (0.12)	0.85 (0.08)	0.67 (0.21)	0.82 (0.13)	0.88 (0.08)	0.91 (0.07)
Grace	0.81 (0.06)	0.85 (0.08)	1.00 (0.00)	0.72 (0.15)	0.81 (0.10)	0.91 (0.06)	0.69 (0.16)	0.76 (0.17)	0.81 (0.11)	0.79 (0.06)
aGrace	0.74 (0.28)	0.83 (0.18)	0.72 (0.15)	1.00 (0.00)	0.70 (0.13)	0.73 (0.13)	0.74 (0.23)	0.69 (0.24)	0.76 (0.21)	0.78 (0.23)
GBLasso	0.70 (0.09)	0.78 (0.12)	0.81 (0.10)	0.70 (0.13)	1.00 (0.00)	0.89 (0.09)	0.77 (0.13)	0.64 (0.16)	0.75 (0.09)	0.79 (0.07)
Linf	0.82 (0.06)	0.85 (0.08)	0.91 (0.06)	0.73 (0.13)	0.89 (0.09)	1.00 (0.00)	0.77 (0.13)	0.74 (0.16)	0.81 (0.08)	0.82 (0.06)
aLinf	0.58 (0.21)	0.67 (0.21)	0.69 (0.16)	0.74 (0.23)	0.77 (0.13)	0.77 (0.13)	1.00 (0.00)	0.58 (0.22)	0.67 (0.19)	0.68 (0.21)
TTLP	0.84 (0.12)	0.82 (0.13)	0.76 (0.17)	0.69 (0.24)	0.64 (0.16)	0.74 (0.16)	0.58 (0.22)	1.00 (0.00)	0.83 (0.16)	0.79 (0.15)
LTLP	0.86 (0.09)	0.88 (0.08)	0.81 (0.11)	0.76 (0.21)	0.75 (0.09)	0.81 (0.08)	0.67 (0.19)	0.83 (0.16)	1.00 (0.00)	0.88 (0.09)
Composite	0.90 (0.10)	0.91 (0.07)	0.79 (0.06)	0.78 (0.23)	0.79 (0.07)	0.82 (0.06)	0.68 (0.21)	0.79 (0.15)	0.88 (0.09)	1.00 (0.00)

8.3 Similarity for Orchestrated tuning

Orchestrated tuning was performed with a subset of the regression methods, the selection of which is discussed in Section 7.5. The ensemble contained the Lasso, Elastic Net, Grace, GBLasso and Linf methods. The orchestrated tuning approach is initialized with parameter starting points obtained from the CV-MSE tuning approach. Results from the regression method similarity analysis for all synthetic datasets is shown in Table 8.2 and Figure 8.2.

The observations made in Section 8.2 regarding the methods contained in the orchestrated ensemble remain true. The Lasso and Elastic Net, as well as the GBLasso and Linf methods continue to produce similar coefficient estimates.

The Elastic Net method shows consistent high similarity to all other regression methods in the ensemble. This could be due to the high number of hyperparameter combinations in its search grid providing a higher flexibility in comparison to the other methods.

Table 8.2: Orchestrated tuning regression method similarities

Method	Lasso (σ_{Lasso})	ENet (σ_{ENet})	Grace (σ_{Grace})	GBLasso ($\sigma_{GBLasso}$)	Linf (σ_{Linf})
Lasso	1.00 (0.00)	0.95 (0.03)	0.91 (0.09)	0.83 (0.08)	0.86 (0.06)
ENet	0.95 (0.03)	1.00 (0.00)	0.95 (0.07)	0.93 (0.03)	0.93 (0.02)
Grace	0.91 (0.09)	0.95 (0.07)	1.00 (0.00)	0.89 (0.06)	0.92 (0.03)
GBLasso	0.83 (0.08)	0.93 (0.03)	0.89 (0.06)	1.00 (0.00)	0.98 (0.01)
Linf	0.86 (0.06)	0.93 (0.02)	0.92 (0.03)	0.98 (0.01)	1.00 (0.00)

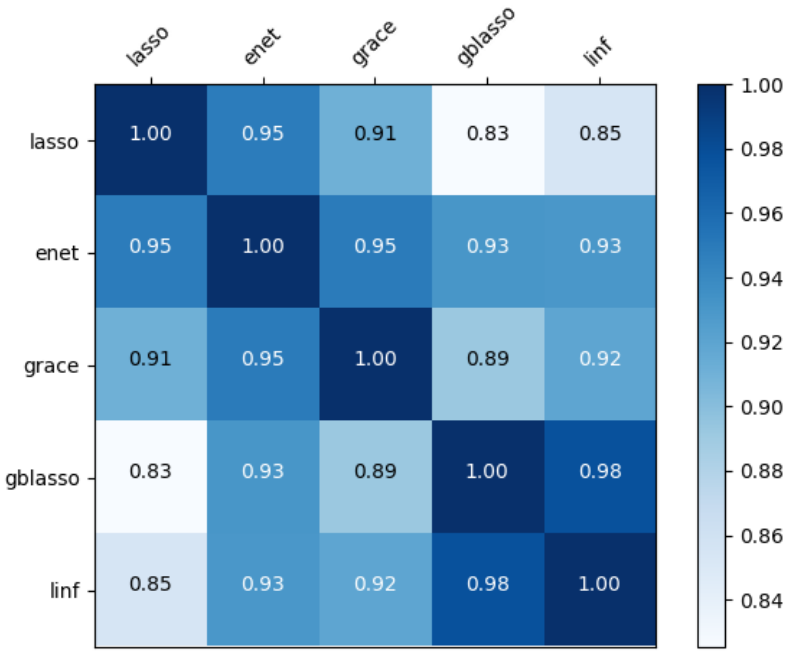

Figure 8.2: Orchestrated tuning mean regression method similarities

Figure 8.3 shows the distribution of iterations elapsed before the orchestrated tuning procedure converged for all synthetic datasets.

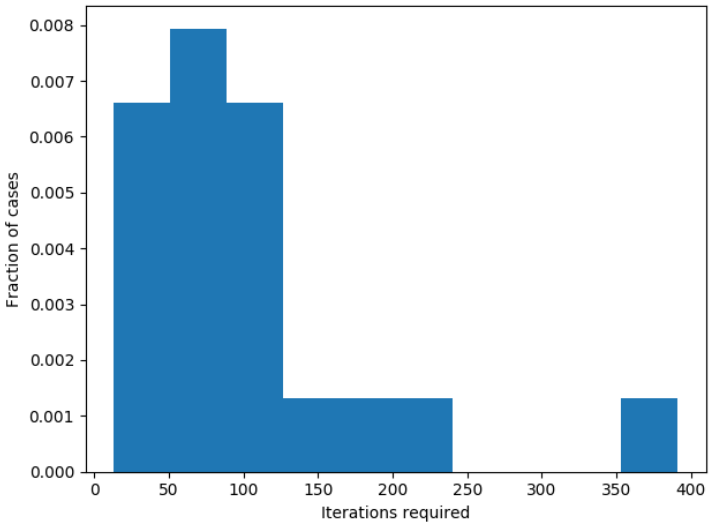

Figure 8.3: Iterations elapsed before convergence for all synthetic datasets

8.4 Similarity comparison by tuning method

The regression method similarities discussed in sections 8.2 and 8.3 have been merged in Table 8.3. Only the similarities regarding methods in the orchestrated tuning ensemble are retained from the CV-MSE results. This is done to enable a more visible comparison between similarities obtained through both hyperparameter tuning approaches.

The mean similarity between each pair of regression methods is increased in all of the cases. For some pairs, an increase of over 0.1 cosine similarity is visible. The overall mean similarity between different methods, which in the case of CV-MSE tuning is 0.836, reaches a value of 0.915 when orchestrated tuning is used.

Table 8.3: Comparison of regression method similarities by tuning method

Method	Tuning	Lasso	ENet	Grace	GBLasso	Linf
Lasso	CV-MSE	1	0.94	0.81	0.7	0.82
	Orchestrated	1	0.95	0.91	0.83	0.86
ENet	CV-MSE	0.94	1	0.85	0.78	0.85
	Orchestrated	0.95	1	0.95	0.93	0.93
Grace	CV-MSE	0.81	0.85	1	0.81	0.91
	Orchestrated	0.91	0.95	1	0.89	0.92
GBLasso	CV-MSE	0.7	0.78	0.81	1	0.89
	Orchestrated	0.83	0.93	0.89	1	0.98
Linf	CV-MSE	0.82	0.85	0.91	0.89	1
	Orchestrated	0.86	0.93	0.92	0.98	1

The comparison between method similarities obtained in the context of CV-MSE and orchestrated hyperparameter tuning clearly shows a similarity increase when our proposed tuning approach is used. This indicates that the desired effect of improved cross-method consensus is achieved.

Bibliography

[1] Robin Holliday. Epigenetics: a historical overview. *Epigenetics*, 1(2):76–80, 2006.

[2] Rudolf Jaenisch and Adrian Bird. Epigenetic regulation of gene expression: how the genome integrates intrinsic and environmental signals. *Nature genetics*, 33:245–254, 2003.

[3] Gerda Egger, Gangning Liang, Ana Aparicio, and Peter A Jones. Epigenetics in human disease and prospects for epigenetic therapy. *Nature*, 429(6990):457–463, 2004.

[4] Manel Esteller. Epigenetics in cancer. *New England Journal of Medicine*, 358(11):1148–1159, 2008.

[5] Cancer Genome Atlas Network et al. Comprehensive molecular portraits of human breast tumors. *Nature*, 490(7418):61, 2012.

[6] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[7] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[8] Caiyan Li and Hongzhe Li. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24(9):1175–1182, 2008.

[9] Caiyan Li and Hongzhe Li. Variable selection and regression analysis for graph-structured covariates with an application to genomics. *The annals of applied statistics*, 4(3):1498, 2010.

[10] Wei Pan, Benhuai Xie, and Xiaotong Shen. Incorporating predictor network in penalized regression with application to microarray data. *Biometrics*, 66(2):474–484, 2010.

[11] Chong Luo, Wei Pan, and Xiaotong Shen. A two-step penalized regression method with networked predictors. *Statistics in biosciences*, 4(1):27–46, 2012.

[12] Sunkyung Kim, Wei Pan, and Xiaotong Shen. Network-based penalized regression with application to genomic data. *Biometrics*, 69(3):582–593, 2013.

[13] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[14] Xiaotong Shen, Wei Pan, and Yunzhang Zhu. Likelihood-based selection and sharp parameter estimation. *Journal of the American Statistical Association*, 107(497):223–232, 2012.

[15] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1, March 2014.

[16] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[17] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 6. Springer, 2013.

[18] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[19] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.