Contents

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 following chapter discusses the details of each linear regression approach. After introducing the synthetic dataset generation process, we define our parameter tuning method and compare the results with those from the standard approach. Next we present the real dataset used for exploration of the relationship between gene methylation and expression. After introducing of our result merging approach we present and discuss the 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:

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

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:

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

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 inversion 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.

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

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.

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

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.

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

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.

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

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.

$$
\begin{array}{l|ccccc}
 & Predictor\ 1 & Predictor\ 2 & & ... & Predictor\ p \\
\hline
Method\ 1 & M_1(\beta_1) & M_1(\beta_2) & & ... & M_1(\beta_p) \\
Method\ 2 & M_2(\beta_1) & M_2(\beta_2) & & ... & M_2(\beta_p) \\
... & ... & ... & & ... & ... \\
Method\ k & M_k(\beta_1) & M_k(\beta_2) & & ... & M_k(\beta_p)
\end{array}
\tag{5.1}
$$

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.

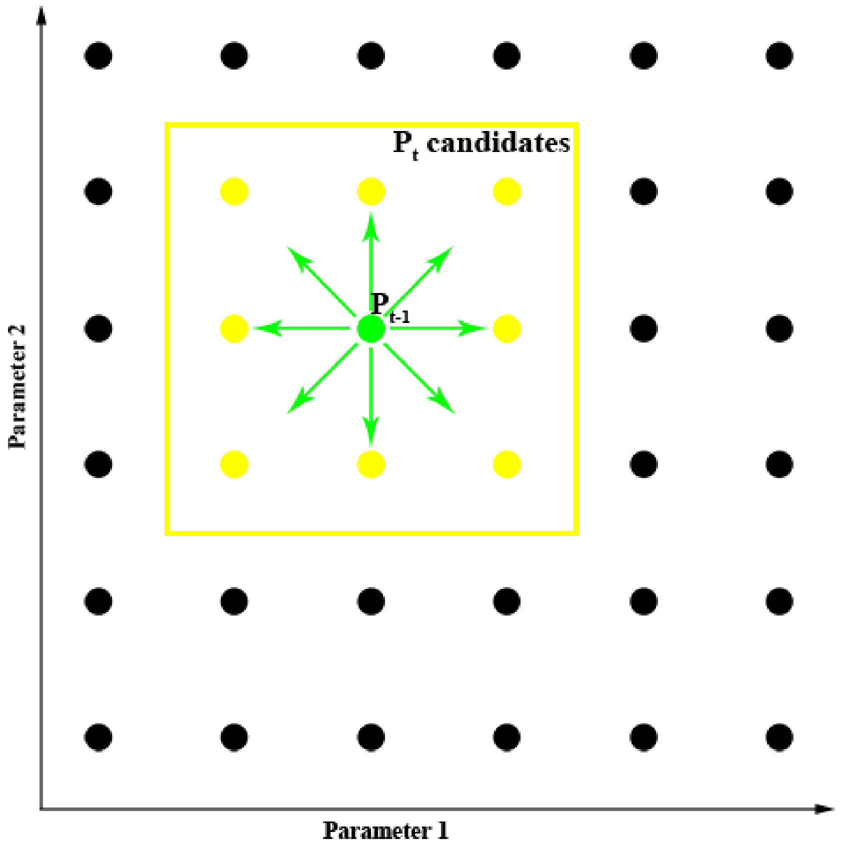

Figure 6.1: Local search grid neighborhood, two tuning hyperparameters; P_{t-1} is the parameter combination selected in iteration $t-1$, parameter values in the P_t neighborhood are considered in iteration t

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.

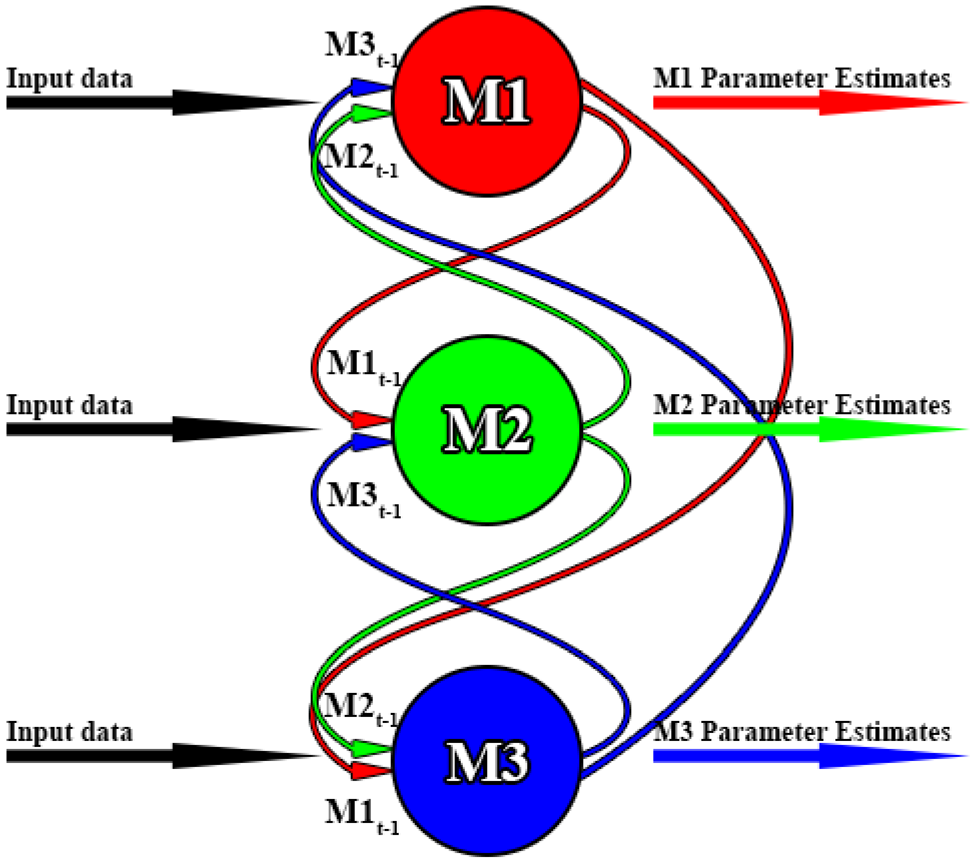

Figure 6.2: Orchestrated tuning abstract structure, three regression methods; $M1_{t-1}$ denotes the parameter estimates of method 1 for iteration $t-1$; Each tuning iteration t of a method uses the common input data and the parameter estimates of all other methods for the previous iteration

Choice of starting points for the search

The choice of hyperparameter starting points in the search grid for the ensemble of regression methods should greatly affect the convergence state of the tuning process. This initial state of the system can be chosen in one of several ways.

1. A naive approach could initialize the tuning process with hyperparameter values located in the centers of the search grids of all methods.

2. An alternative approach would be to start the tuning procedure from a corner of the search grids. Depending on whether we wish to promote

simpler or more complete models, the appropriate griid corner should be chosen.

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 ran CV-MSE tuning. Such a fused tuning procedure could potentially combine the benefits of both tuning approaches - selection of models with good generalized prediction capabilities, but also coordinated for better agreement across regression methods.

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

Discussion

Such a parameter tuning process aims to find combinations of values for the various methods' hyperparameters that lead to the production of similar coefficient vectors across methods. As previously shown, the focus is on the estimated predictor coefficients and 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, it uses the fact that we have multiple methods and they perform their hyperparameter tuning cooperatively so as to promote consensus across methods after the training with arbitrary data.

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.