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:

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

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:

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

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 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 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	-	Grace (2.4.1)
		-	aGrace (2.4.2)
		-	Linf (2.4.4)
		-	aLinf (2.4.4)
		-	TTLP (2.4.5)
		-	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.

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

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.