CONDITIONAL RANDOM FIELDS WITH LASSO AND ITS APPLICATION TO THE CLASSIFICATION OF BARLEY GENES BASED ON EXPRESSION LEVEL AFFECTED BY FUNGAL INFECTION

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Xiyuan Liu

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Statistics

April 2019

Fargo, North Dakota

NORTH DAKOTA STATE UNIVERSITY

Graduate School

Title

CONDITIONAL RANDOM FIELDS WITH LASSO AND ITS APPLICATION TO THE CLASSIFICATION OF BARLEY GENES BASED ON EXPRESSION LEVEL AFFECTED BY FUNGAL INFECTION

By

Xiyuan Liu

The supervisory committee certifies that this dissertation complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Prof. Gang Shen

Chair

Prof. Megan Orr

Prof. Bong-Jin Choi

Prof. Shaobin Zhong

Approved:

09 April 2019	Rhonda Magel
Date	Department Chair

ABSTRACT

The classification problem of gene expression level, more specifically, gene expression analysis, is a major research area in statistics. There are several classical methods to solve the classification problem. To apply Logistic Regression Model (LRM) and other classical methods, the observations in the dataset should fit the assumption of independence. That is, the observations in the dataset are independent to each other, and the predictor (independent variable) should be independent. These assumptions are usually violated in gene expression analysis. Although the Classical Hidden Markov Chain Model (HMM) can solve the independence of observation problem, the classical HMM requires the independent variables in the dataset are discrete and independent. Unfortunately, the gene expression level is a continuous variable. To solve the classification problem of Gene Expression Level data, the Conditional Random Field(CRF) is introduce. Finally, the Least Absolute Selection and Shrinkage Operator (LASSO) penalty, a dimensional reduction method, is introduced to improve the CRF model.

ACKNOWLEDGEMENTS

This thesis was supported by Dr. Gang Shen, who provided insight and expertise that greatly assisted the thesis.

I would also like to show my gratitude to Dr. Megan Orr, Dr. Bong-Jin Choi, and Dr. Shaobin Zhong for their time and feedback.

I thank Neville Tammi for comments and language support that greatly improved the manuscript.

This thesis would not have been done without my family behind me. I am indebted for the mental and physical support they have provided that has spurred me on through tough times during not only my thesis but also my entire graduate study. I am thankful for their love and prayers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

There are several classical models to solve classification problem of gene expression level, for example Logistic regression model, Mixture Model, and hypothesis test with Benjamini-Hochberg method. To apply these classical models, the observation in the dataset should fit the assumption of independence. That is, the observations in the dataset are independent to each other. However, when it comes to gene expression analysis, or similarly, natural language processing [Sha and Pereira, 2003], it is impossible to assume the independence for each of the observations. In addition, the classical models are only widely used when the dependent variable has only two possible outcomes. When the dependent variables have more than two possible outcomes, these classical models cannot be used directly. Hence, to solve these problems, the Hidden Markov Chain Model (HMM) was introduced.

Because HMM assumes that the dependent variables have the Markov Chain structure, it is suitable for a dataset that fails to assume independence. The HMM has three assumptions. First, it assumes that the dependent variable (or, in this study, the label) cannot be observed directly. But the outcomes for the dependent variable, which are independent variables, can be observed. Second, the HMM assumes that the labels for each of the observations have correlations. Third, the outcomes (independent variables) are independent of each other. Based on these three assumptions, the HMM can be divided into two major components.

The first component is the transition probability. The transition probability is a function that describes the relationship between different labels. The second component for the HMM is the emission probability. The emission probability describes the relationship between the independent variables and the dependent variables. For the classical HMM, when both dependent and independent variables are discrete, the emission probability is easy to estimate. Unfortunately, for the gene expression analysis of the gene expression level, the independent variable is continuous. Hence, the classical HMM is not very useful in this case. To design a model that could fit the gene expression dataset, Conditional Random Field (CRF) was introduced.

The CRF is a very generalized Machine Learning Model that is designed to solve the classification problem. The CRF does not contain the assumption of independence, and does not require the independent variable to be discrete. Furthermore, both HMM and LRM for the classification problems are special cases for the use of CRF [Sutton and McCallum, 2012]. Because of the flexibility, the CRF can adapt itself into different models to fit different datasets. However, the number of dimensions for the parameters in the CRF is very large, which will cause problems during the training step. To solve this problem, least absolute shrinkage and selection operator (LASSO) penalty is applied.

LASSO is very widely used for the dimensional reduction problem. It can improve any generalized linear model and Support Vector Machines (SVM) by reducing predictors while keeping the accuracy of the predictions [Hastie et al., 2015]. Applying LASSO to CRF will largely decrease the number of dimensions for the parameters of the CRF and hence, improve the CRF.

This study demonstrated the flexibility of the CRF and improved it with the LASSO penalty. A CRF model was designed for the barley gene expression data, which was collected in order to classify different reactions of host genes under pathogens to improve the immunity of barley. Since the gene expression level for each host gene may have strong correlations, the classical classification method was not appropriate. Furthermore, since the outcomes (i.e. the independent variables) in the gene expression data were not discrete variables, the classical HMM was also not suitable. Hence, CRF with a proper design for this dataset was introduced.

This thesis contains several sections to explain the CRF in detail. The literature review chapter explains the problems of the CRF and why LASSO can improve the CRF. The methodology chapter shows the technical details about how to design the CRF for this barley data and how to implement the LASSO into the CRF. The numeric experiment chapter demonstrates the reliability of the CRF and compare it with the classical generalized linear model. The data analysis chapter applies the CRF to the barley dataset and provides the conclusion for the results. Finally, we will introduce further research areas for the study.

2. LITERATURE REVIEW

The Conditional Random Field (CRF) was first introduced by Lafferty and McCallum in 2001 [Lafferty et al., 2001] in order to classify natural-language data. Furthermore, the paper pointed out that CRF can be developed for different data structures. Ten years later, Sutton and McCallum published a summary paper to provide a more complete explanation of the model [Sutton and McCallum, 2012] and its applications. As a flexible probabilistic model, CRF can adapt itself into several different scenarios, such as text process [Ammar et al., 2014], bioinformatics [Thiagarajan and Bremananth, 2015], image process [Qianwen et al., 2018], etc.

There are two components to CRF: feature functions and the normalization function [Sutton and McCallum, 2012]. Feature functions are function with two arguments, a dependent variable (usually is a countable finite variable) and independent variables. Although there are no strict roles on how to define the feature function, the function is usually constructed with one indicating function and one Real number function. This design is called the Label-observation features [Sutton and McCallum, 2012]. In the case of HMM, there are two types of feature functions. One is the joint probability between dependent variables and independent variables, and the other is the transition probability between dependent variables. On the other hand, in the case of classical Logistics Regression, the feature function is a linear function between dependent variables and independent variables. Because of the flexibility of the feature function design, the CRF is very handy in almost every circumstance.

The other important component of the CRF is the normalization function. The normalization function is a marginal function, which only depends on the independent variables. In order to design a conditional probability function for the classification problem given independent variables (i.e. $p(y|x)$), we need to know two functions: The joint probability function, which contains independent and dependent variables (i.e. $p(x, y)$), and the marginal function of the independent variable (i.e. $p(x)$). Using the feature functions, we can define $p(x, y)$. On the other hand, to obtain the marginal function, we need to integrate out the dependent variable in $p(x, y)$. When the dependent variable is a scalar that has only a few possible countable outcomes, it is easy to integrate out the dependent variable and compute the $p(x)$. However, when the dependent variable is a finite vector, in which each element has 2 outcomes, the set of possible outcomes for the dependent variable can be expend exponentially. Hence, one problem of the CRF is to find an efficient way to compute the marginal function. Sutton [Sutton and McCallum, 2012] proposed that using a backward-forward algorithm (more specifically, Viterbi Algorithm) can solve the problem efficiently. However, to compute the marginal function via Viterbi Algorithm, the parameters for $p(x, y)$ are required. Hence, another problem for the CRF is how to estimate parameters. Maximum Likelihood Estimation (MLE) and Least Square Estimation (LSE) are two of the most well known estimation methods for this problem. Yet, due to the nature of the CRF (i.e. high dimensional parameters), classical computational methods, such as the Newton-Raphson method, are not practical. Sutton [Sutton and McCallum, 2012] proposed Gradient Descent to solve the problem.

Unlike the Newton-Raphson Method, which needs to compute both the first and second derivatives for the object function (e.g. log-likelihood function), Gradient Descent only needs to compute the first derivative. However, the Gradient Descent still needs a sequence to substitute the second derivative. Kiefer and Wolfowitz [Kiefer and Wolfowitz, 1952] proved that the sequence for Gradient Descent belongs to l_2 norm but does not belong to l_1 norm. That means the critical aspect to approaching the problem is to select a sequence with a proper convergence speed.

Another problem of the CRF is the a large number of parameters it contains (i.e. high dimensional parameters). This is because of the flexibility of the feature function design. Containing a large number of parameters will cause the model to over fit the data. In addition, it is not very practical to estimate these many parameters through traditional computational methods. Besides, within these many parameters, it is possible that some of them may not have a significant influence of the model. Hence, it is very important to eliminate these parameters to avoid the risk of over fit and make the model more efficient. To solve this problem, LASSO is introduced.

LASSO, a dimensional reduction technique, can improve any generalized linear model or Support Vector Machines (SVM) by reducing the number of predictors while keeping the accuracy of the predictions [Hastie et al., 2015]. Additionally, as a classical High Dimensional Data Analysis tool, LASSO is widely used in may areas, such as Neural Networks [Cui and Wang, 2016], Bioinformatics [J Motyer et al., 2011], Biostatistics, etc.

The combination of both CRF and LASSO can indeed, largely improve the efficiency and accuracy of the statistical analysis for big data. For the next section, some basic settings for the barley dataset are introduced then the methodology is provided for the CRF with respect to the dataset.

3. PLANT SCIENCE DATA

A barley experiment was conducted in order to determine what genes in the plant genome will be affected by a fungus. If the study can identify genes in the plant genome that are associated with plant disease resistance, then the study will provide more information about the interaction between pathogen and host, and hence provide better control of the disease.

In this study, there are two treatments: Barley cv. Bowman infected by the original isolate (ND90Pr), which is highly virulent on Bowman, and the plants infected by the mutated isolate with low virulence on Bowman. The mutant was generated by deleting a NPS gene conforming high virulence of the isolate. For each treatment, 3 leaf samples were collected at 0, 6, 12, 18, 24, 36, 48, and 72 hours after inoculation. Hence, a total of 48 samples, that is, 3 replicates in 2 treatments at 8 time points were used to generate the dataset.

In the dataset there are 6325 differential expressed genes. Each gene contains 8 time points records. When the gene expression level is missing in some of the 8 hours records, we assume that the gene has a very low gene expression level and set the level to 0. To analyze the data, the ratio between two treatments gene expression levels, namely, relative gene expression level, was used. The relative gene expression level is

$$g_j = \frac{g'_j + 10^{-6}}{g'_0 + 10^{-6}},$$

where $j = 0, 6, 12, 18, 24, 36, 48, 72$, g_j is the relative gene expression level on the time j, and g'_j is the raw gene expression level on the time j. To avoid the denominator of g_j equaling 0, 10^{-6} is added. The Methodology section will introduce both Hypothesis testing with the Benjamini-Hochberg method, which is a classical method for gene expression analysis, and CRF model for this dataset.

4. METHODOLOGY

4.1. Hypothesis test

The purpose of the hypothesis test is to classify the gene data into two groups. First group contains the plant genes that have no reaction with the fungus. Second group contains the plant genes that have relation with the fungus. The classical method that serves this purpose is hypothesis test. In order to apply the hypothesis test, the log-fold of the gene expression level is introduced. The log-fold is the nature log of the result calculated from the division of the current gene expression level by the previous one. There are some basic assumptions for the log-folds:

- the log-folds of an unchanged gene expression level are following $N_T(\tilde{0}, \Sigma)$
- the log-folds of a changed gene expression level are following $N_T(\tilde{\mu}, \Sigma)$ where $\max_t |\mu_t| > 0$, $t = 1, 2, \cdots, T$.

Both $\tilde{0}$ and $\tilde{\mu}$ are T dimensional vectors, where there are $T + 1$ records for each gene including the record at time 0. Furthermore,

$$\tilde{0} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}_T ,$$

and Σ is the covariance matrix for the distribution. Based on these assumptions, the hypothesis test will be conducted, that is:

- H_o: $\max_t |\mu_t| = 0$
- H_α: $\max_t |\mu_t| > 0$.

Notice that the hypothesis test only works when the distribution under the H_o hypothesis is known. However, in this case, the distribution under the null hypothesis (H_o) is an extreme value distribution whose CDF is impossible to be calculated directly. To solve this problem, a simulation for this extreme value distribution is introduced.

4.2. The simulation of the extreme value distribution

To establish a hypothesis test for the maximum log-fold, the extreme value distribution under the null hypothesis has to be known. However, it is difficult to derive the exact CDF for the extreme value distribution. This is because the current gene activities can be affected by the previous one, hence the data cannot be simply treated as independent. Therefore, the extreme value distribution under the null hypothesis can only be obtained by simulation. This simulation draws 100,000 random variables from $N(\tilde{0}, \Sigma)$, where:

$$\tilde{0} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \text{ and } \Sigma = \begin{pmatrix} 1 & \rho & 0 & 0 & 0 & 0 \\ \rho & 1 & \rho & 0 & 0 & 0 \\ 0 & \rho & 1 & \rho & 0 & 0 \\ 0 & 0 & \rho & 1 & \rho & 0 \\ 0 & 0 & 0 & \rho & 1 & \rho \\ 0 & 0 & 0 & 0 & \rho & 1 \end{pmatrix}.$$

An example of the histogram for the simulated data under the null hypothesis is shown below.

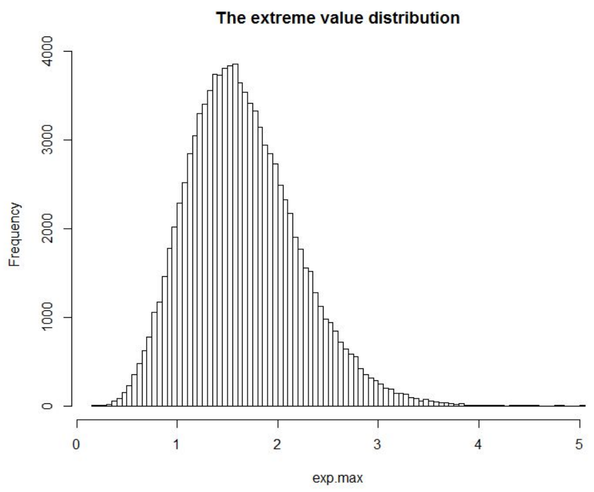

Figure 4.1. The extreme value distribution under the null hypothesis when $\rho = 0.05$.

After the extreme value distribution under the null hypothesis test is simulated, the critical value, which is observed from the 95% quantile, is used to determine when to reject the null hypothesis test. To prove that the simulation data under the null hypothesis is correct, a comparison between the theoretical quantile and the simulated quantile is necessary. To obtain the theoretical quantile, a theoretical CDF is necessary. Although it is impossible to derive the CDF for the extreme value distribution in general, it is possible to derive the CDF for the extreme distribution when all variables are independent.

Assume $\phi(\cdot)$ is the PDF of $N(0,1)$ and $x_t \overset{iid}{\sim} N(0,1)$, where $t = 1, 2, ..., 6$. The CDF of $\max_t |x_t|$ is

$$F(y) = Pr(\max_t |x_t| < y) = [\int_0^y 2\phi(z)dz]^6 \qquad \text{, where } y \geq 0.$$

The value of the theoretical quantile can be easily calculated using the theoretical CDF since $1 - \alpha = F(y)$,

$$y = \Phi^{-1}(\frac{1}{2}[1 + (1-\alpha)^{\frac{1}{6}}]),$$

where y is the theoretical quantile for the upper α^{th} percentile, and $\Phi^{-1}(\cdot)$ is the inverse function for the CDF of $N(0,1)$.

The table of the simulation quantiles and the theoretical quantiles for the upper 95%, 97.5%, and 99% are shown below.

Table 4.1. The quantiles for both simulated and theoretical

upper percentile	95%	97.5%	99%
simulated quantile	2.63	2.86	3.15
theoretical quantile	2.63	2.86	3.14

To visualize the relationship between the distribution of simulated data and the theoretical distribution, a histogram is presented. The histogram is based on the PDF of y that can be derived according to the theoretical CDF of y. The PDF of y is shown below.

$$f(y) = 12\phi(y)[\int_0^y 2\phi(z)dz]^5 \qquad \text{, where } y \geq 0,$$

and $\phi(\cdot)$ is the PDF of $N(0,1)$.

The histogram of the simulation dataset and the theoretical PDF is shown below.

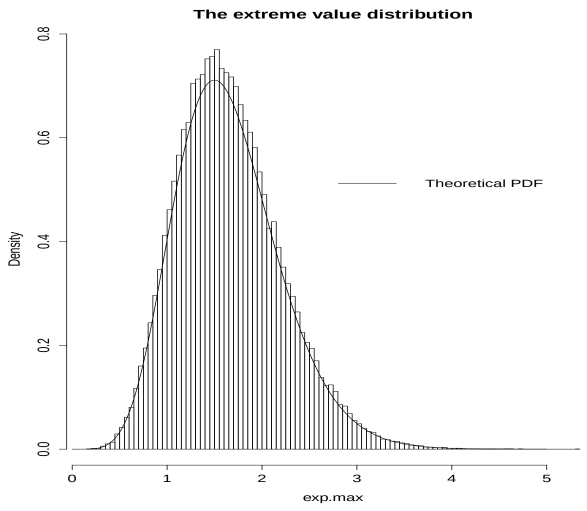

Figure 4.2. The theoretical PDF and the simulated data

Figure 4.2 show that the theoretical CDF and PDF for the extreme value distribution fit the simulation value. That is, both quantiles in the table are almost the same, and the theoretical PDF fits the histogram.

Although the problem for the hypothesis test is solved, the hypothesis test still cannot be conducted directly. It is because of the family-wise type one error in the hypothesis test cannot be ignored. The hypothesis test is testing all 6325 genes, which means 6325 tests are conducted. Because of the large number of the tests, the family-wise type one error is too large to be ignored. To solve this problem, the false discovery rate, which is used to measure the family-wise type one error in multiple hypothesis testing, needs to be introduced first.

4.3. False discovery rate

The false discovery rate (FDR) is a well known measurement for the family-wise type one error. FDR is the ratio between the number of false positives, which means reject the null hypothesis when the null is true, and the number of rejected null hypotheses. The FDR is defined as

$$Q_e = E(Q) = E\left[\frac{R-S}{R}\right],$$

where R is the number of rejected null hypotheses and S is the number of true positives, which means the null hypothesis is rejected when the alternative hypothesis is true. Hence, the FDR measures the model's accuracy, the model that contains a smaller FDR is preferred.

Given the definition of FDR, the problem of how to control the family-wise type one error is transferred into the problem of how to control (minimize) the FDR. This problem is solved by the Benjamini-Hochberg method.

4.4. Benjamini-hochberg

The purpose of the Benjamini-Hochberg method (B-H) is to control the FDR by controlling every type one error for the hypothesis tests. In this study, the B-H method is modified since the distribution of null hypothesis is unknown. Therefore, instead of comparing the p-value with α, which the significant level for the hypothesis test, $\max_t |x'_t|$ and the quantile for the simulated distribution are compared. The steps for this Benjamini-Hochberg method are shown below.

1. Find the $x_n = \max_t |x'_t|$, $t = 1, 2, \cdots, T$, where x'_t is the observed log-fold at time t. x_n is the maximum of the absolute log-fold for gene n, $n = 1, 2, \cdots, N$.

2. Rank x_n by ascending order, $x_{(1)} < x_{(2)} < ... < x_{(n)}$.

3. Reject all $H_{(i)}$, where $i = 1, 2, ..., k$ when

$$x_{(k)} \geq Crit = quantile(\frac{k}{N}\alpha),$$

   where α is the significance level that controls the family-wise error. Usually, $\alpha = 0.05$. The critical value is the $\frac{k}{n}\alpha$th quantile for the simulated distribution.

4. Stop rejecting until $x_{(k)} < Crit$, where $Crit$ is the $\frac{k}{N}\alpha$th quantile for the simulated distribution.

Up to this point, the hypothesis test for the plant gene expression level is well defined. However, as mentioned above, this hypothesis test contains two limitations. First, the hypothesis test can only solve the problem when there are only two possible outcomes for the classification problems. Second, the hypothesis test needs to have the iid normal distribution assumption for the independent variables. Hence, when the classification problem tries to separate the observations into three groups, and the independent variables have no assumptions, the hypothesis test is not suitable. In addition, the hypothesis test only uses $\max_t |\mu_t|$ to determine the groups for each gene; Therefore, it wastes the remaining 6 records. To solve all these problems, the Conditional Random Field is introduced.

4.5. Conditional random field

Conditional Random Field (CRF) is a Machine Learning model that is designed specifically for the classification problem. As mentioned in the plant science data chapter, each gene expression level has 8 records. Without reducing the 8 dimensional instance into 1 a dimensional instance, the CRF model is applied. The CRF model contains two major functions, the feature function and the normalization function. To construct this model, the feature function $f_l(x'_{ij}, y)$, needs to be defined first. For this study, the feature function is defined as:

$$f_l(x'_{ij}, y) = I_{\{y=m\}} x'_{ij},$$

where $l = 1, 2, 3$, y is the label of the gene, x'_{ij} is the ratio between the original relative gene expression level on time i, $(i = 1, 2, \cdots, 8)$, and the mutant relative gene expression level on time j, $(j = 1, 2, \cdots, 8)$, that is,

$$x'_{ij} = \frac{g_{i.original}}{g_{j.mutant}},$$

where $i \geq j$. There are 36 different combinations for the ratio. Returning to the feature function, $I_{\{y=m\}}$ is the indicating function such that,

$$I_{\{y=m\}} = \begin{cases} 1 & \text{if } y = m \\ 0 & \text{if } y \neq m \end{cases},$$

and m is the label that indicates the relation between the plant gene and the specific fungus gene where:

- $m = 0$: No relation between the plant gene and the specific fungus gene,
- $m = 1$: Positive relation between the plant gene and the specific fungus gene, and
- $m = -1$: Negative relation between the plant gene and the specific fungus gene.

We will define no relation, positive relation, and negative relation in the data analysis chapter.

To simplify the notation for the 36 combinations of x'_{ij}s, we denoted x'_{ij} into x_i. Hence, the feature function $f_l(x'_{ij}, y)$ above is changed to

$$f_l(x_i, y) = I_{\{y=m\}} x_i, \text{ where } i = 1, 2, \cdots, 36.$$

After defining the feature function, the normalization function is well defined as

$$Z(x) = \sum_y \exp\{\sum_{i,l} \theta_{i,l} f_l(x_i, y)\},$$

where $\theta_{i,l}$ are the coefficients for each $x_i, i = 1, 2, \cdots 36$. The footnote $l = 1, 2, 3$ corresponding to the $m = 0, 1, -1$, which is the possible outcome of y. $Z(x)$ is the normalization function, adding $Z(x)$ can promise $\sum_y p(y|x) = 1$, where $\sum_y$ means the sum of all the possible outcomes for y. Since both the feature and normalization functions are introduced, the CRF model is completed, that is,

$$p(y|x) = \frac{1}{Z(x)} \exp\{\sum_{i,l} \theta_{i,l} f_l(x_i, y)\} = \frac{1}{Z(x)} \exp\{\sum_{i,l} \theta_{i,l} I_{\{y=m\}} x_i\}.$$

There are two steps to determine the label for each gene via $p(y|x)$. First, the training step. In this step, the parameters of of $p(y|x)$, which are $\theta_{i,l}$s will be estimated by the training data. After estimating the $\theta_{i,l}$s, the model can be applied to the second step, testing step. In the testing step, for each gene, the label that is determined by the CRF model is compared to the label that is well known. However, there are several difficulties for each steps. The next section will discuss the first step (i.e. training step) for the CRF.

4.6. CRF training

The major problem in the first step is how to estimate the $\theta_{i,l}$s. In this study, maximum likelihood is applied. That is, instead of directly estimating $\theta_{i,l}$s in $p(y|x)$, we maximize the log-likelihood, $l(\theta)$. In this study, the log-likelihood is:

$$l(\theta) = \sum_{k=1}^{n} \log p(y^{(k)}|x^{(k)}) = \sum_{k=1}^{n} \left(\sum_{i,l} \theta_{i,l} f_l(x_i^{(k)}, y^{(k)}) - \log Z(x^{(k)}) \right),$$

where $x_i^{(k)}$ and $y^{(k)}$ are the ratio and the label for gene k, respectively. Furthermore,

$$\theta = (\theta_{1,1}, \theta_{2,1}, \cdots, \theta_{36,1}, \theta_{1,2}, \cdots \theta_{36,2}, \theta_{1,3} \cdots, \theta_{36,3})$$

is a 108 dimensional vector. As mentioned in the introduction chapter, since the number of parameters for CRF is large, estimating the parameters via traditional computational methods is not practical. Hence, gradient descent is applied in order to maximize the log-likelihood function.

The gradient descent needs to compute the partial derivatives for $l(\theta)$ with respect to each $\theta_{i,l}$. The main difficulty of computing the log-likelihood and the derivative for the log-likelihood is computing $Z(x^{(k)})$. This is because, for the general linear-chain CRF, $y^{(k)}$ is not required to be independent. As a result, instead of being a one dimension number (i.e. a scalar), $y^{(k)}$ is a n dimension vector, where $n \geq 2$. Therefore, it is very difficult to permute all the possible outcomes for $y^{(k)}$, which is a required process to compute $Z(x^{(k)})$. However, in this study, we assume $y^{(k)}$s are independent, hence, the difficulty of computing $Z(x^{(k)})$ is eliminated. Therefore, we only need to focus on the gradient descent to maximize $l(\theta)$.

4.6.1. Gradient descent

As mentioned above, one of the most common methods to maximize $l(\theta)$ is gradient descent. Gradient descent is a first-order iterative algorithm to find the minimum of the objective function.

Assuming the objective function $f(x) : \mathbb{R} \to \mathbb{R}$ is differentiable and the global minimum can be achieved. Then, by applying the update rule:

$$x^{(t+1)} = x^{(t)} - s^{(t)} f'(x^{(t)}),$$

where $f'(x^{(t)})$ is the first derivative of $f(x)$ at point $x^{(t)}$, t is the current iteration, and $s^{(t)}$ is the step length at time t, the minimum of $f(x)$ can be achieved after Tth iterations, that is:

$$f(x^{(T)}) = \min f(x), \text{ and } f'(x^{(T)}) = 0.$$

In this study, the objective function is $l(\theta)$, to maximize $l(\theta)$ is equivalently to minimize $-l(\theta)$, hence, the objective function for gradient descent in this study is :

$$-l(\theta) = \sum_{k=1}^{n} \left(\log Z(x^{(k)}) - \sum_{i,l} \theta_{i,l} f_l(x_i^{(k)}, y^{(k)}) \right).$$

Furthermore, the first derivative of $-l(\theta)$ with respect to $\theta_{i,l}$ is:

$$\frac{-\partial l(\theta)}{\partial \theta_{i,l}} = \sum_{k=1}^{n} \left(x_i^{(k)} p(y = m | x^{(k)}, \theta_{i,l}) - I_{\{y=m\}} x_i^{(k)} \right),$$

where $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, \cdots, x_{36}^{(k)})$, $i = 1, 2, \cdots, 36$, $l = 1, 2, 3$ and

$$p(y = m | x^{(k)}, \theta_{i,l}) = \frac{1}{Z(x)} \exp\{\sum_{i,l} \theta_{i,l} I_{\{y=m\}} x_i\},$$

with $Z(x) = \sum_y \exp\{\sum_{i,l} \theta_{i,l} f_l(x_i, y)\}$ and m as the label corresponds to l. Therefore, the update rule for $\theta_{i,l}$ is:

$$\theta_{i,l}^{(t+1)} = \theta_{i,l}^{(t)} - s^{(t)} \sum_{k=1}^{n} \left(x_i^{(k)} p(y = m | x^{(k)}, \theta_{i,l}^{(t)}) - I_{\{y=m\}} x_i^{(k)} \right).$$

Notice that when m only has two outcomes, that is, $m = 0, 1$, $l = 1, 2$, and $i = 1, 2, \cdots, I$, the first derivatives of $l(\theta)$ with respect to $\theta_{i,1}$ and $\theta_{i,2}$ are:

$$\begin{aligned}
\frac{-\partial l(\theta)}{\partial \theta_{i,1}} &= \sum_{k=1}^{n} \left(x_i^{(k)} p(y = 0 | x^{(k)}, \theta_{i,1}^{(t)}) - I_{\{y=0\}} x_i^{(k)} \right) \\
\frac{-\partial l(\theta)}{\partial \theta_{i,2}} &= \sum_{k=1}^{n} \left(x_i^{(k)} p(y = 1 | x^{(k)}, \theta_{i,2}^{(t)}) - I_{\{y=1\}} x_i^{(k)} \right) \\
&= \sum_{k=1}^{n} \left(x_i^{(k)} [1 - p(y = 0 | x^{(k)}, \theta_{i,1}^{(t)})] - I_{\{y=1\}} x_i^{(k)} \right) \\
&= \sum_{k=1}^{n} \left((x_i^{(k)} - I_{\{y=1\}} x_i^{(k)}) - x_i^{(k)} p(y = 0 | x^{(k)}, \theta_{i,1}^{(t)}) \right) \\
&= \sum_{k=1}^{n} \left(I_{\{y=0\}} x_i^{(k)} - x_i^{(k)} p(y = 0 | x^{(k)}, \theta_{i,1}^{(t)}) \right) \\
&= -\left(\frac{\partial - l(\theta)}{\partial \theta_{i,1}} \right).
\end{aligned}$$

Hence, the update rule for $l(\theta)$ when $m = 0, 1$ is

$$\theta_{i,1}^{(t+1)} = \theta_{i,1}^{(t)} - s^{(t)} \frac{-\partial l(\theta)}{\partial \theta_{i,1}}$$
$$\theta_{i,2}^{(t+1)} = \theta_{i,2}^{(t)} + s^{(t)} \frac{-\partial l(\theta)}{\partial \theta_{i,1}}$$

Therefore, when the initial value for $\theta_{i,l} = 0$, where $\forall i \in \{1, 2, \cdots I\}$ and $\forall l \in \{1, 2\}$, we have

$$\theta_{i,1}^{(T)} = -\theta_{i,2}^{(T)}, \forall i = 1, 2, \cdots, I$$

after T several iterations.

After computing the first derivative, the step-size function (i.e. $s^{(t)}$) need to be defined. The classical sequence of $s^{(t)}$ requires two criteria [Kiefer and Wolfowitz, 1952]

$$\textstyle\sum_t s^{(t)} = \infty \text{ and } \sum_t (s^{(t)})^2 < \infty$$

These two criteria imply that $s^{(t)}$, as an infinite dimensional vector, must belong to l_2 norm but cannot belong to l_1 norm. Therefore, $s^{(t)}$ must converge as $t \to \infty$ but cannot converge very fast. For a more practical reason, the sequence of $s^{(t)}$ is defined as:

$$s^{(t)} = \frac{1}{p(t_0 + t)},$$

where p and t_0 are two user-defined constant numbers [Bottou, 2010]. Hence, Considering both theoretical and the practical reasons, $s^{(t)}$ in this study is defined as

$$s^{(t)} = \sqrt{\frac{1}{10^8(0.005 + t)}}.$$

Therefore, $s^{(t)}$ will decrease when the number of iterations increases, which ensures that the vector belongs to l_2 norm. Furthermore, the square root will ensures $s^{(t)}$ can converge in a not very fast way.

After both the step-size function and the first derivative of $l(\theta)$ are known, the gradient descent for maximizing the $l(\theta)$ is well defined. However, there is another difficulty to estimate $\theta_{i,l}$

for CRF. As shown above, the CRF model is defined as:

$$p(y|x) = \frac{1}{Z(x)} \exp\{\sum_{i,l} \theta_{i,l} f_l(x_i, y)\},$$

where

$$Z(x) = \sum_{y} \exp\{\sum_{i} \theta_{i,l} f_l(x_i, y)\}.$$

Apply the equation for the feature function $f_l(x_i, y) = I_{\{y=m\}} x_i$, we have:

$$p(y|x) = \frac{\exp\{\sum_{i,l} \theta_{i,l} I_{\{y=m\}} x_i\}}{\exp\{\sum_i \theta_{i,1} x_i\} + \exp\{\sum_i \theta_{i,2} x_i\} + \exp\{\sum_i \theta_{i,3} x_i\}}.$$

Let $\theta_{a,1} = \theta_{a,2} = \theta_{a,3} = C$ for some $a \in \{1, 2, \cdots, 36\}$ and $C \in \mathbb{R}$, we have:

$$\begin{aligned} p(y|x) &= \frac{\exp\{\sum_{i,l} \theta_{i,l} I_{\{y=m\}} x_i\}}{\exp\{\sum_i \theta_{i,1} x_i\} + \exp\{\sum_i \theta_{i,2} x_i\} + \exp\{\sum_i \theta_{i,3} x_i\}} \\ &= \frac{\exp\{\sum_{i\in/a,l} (\theta_{i,l} I_{\{y=m\}} x_i) + (\theta_{a,1} I_{\{y=-1\}} + \theta_{a,2} I_{\{y=0\}} + \theta_{a,3} I_{\{y=1\}}) x_a\}}{\exp\{\sum_{i\in/a} \theta_{i,1} x_i + \theta_{a,1} x_a\} + \exp\{\sum_{i\in/a} \theta_{i,2} x_i + \theta_{a,2} x_a\} + \exp\{\sum_{i\in/a} \theta_{i,3} x_i + \theta_{a,3} x_a\}} \\ &= \frac{\exp\{\sum_{i\in/a,l} (\theta_{i,l} I_{\{y=m\}} x_i) + (C I_{\{y=-1\}} + C I_{\{y=0\}} + C I_{\{y=1\}}) x_a\}}{\exp\{\sum_{i\in/a} \theta_{i,1} x_i + C x_a\} + \exp\{\sum_{i\in/a} \theta_{i,2} x_i + C x_a\} + \exp\{\sum_{i\in/a} \theta_{i,3} x_i + C x_a\}} \\ &= \frac{\exp\{\sum_{i\in/a,l} (\theta_{i,l} I_{\{y=m\}} x_i) + C x_a\}}{\exp\{\sum_{i\in/a} \theta_{i,1} x_i + C x_a\} + \exp\{\sum_{i\in/a} \theta_{i,2} x_i + C x_a\} + \exp\{\sum_{i\in/a} \theta_{i,3} x_i + C x_a\}} \\ &= \frac{\exp\{\sum_{i\in/a,l} (\theta_{i,l} I_{\{y=m\}} x_i)\}}{\exp\{\sum_{i\in/a} \theta_{i,1} x_i\} + \exp\{\sum_{i\in/a} \theta_{i,2} x_i\} + \exp\{\sum_{i\in/a} \theta_{i,3} x_i\}}, \end{aligned}$$

where $/a = \{1, 2, \cdots, a-1, a+1, \cdots, 36\}$. Hence, when $\theta_{a,1} = \theta_{a,2} = \theta_{a,3} = C$, the observation x_a is not useful in the model. Therefore, in order to reduce the dimension, we need to force $C = 0$. Therefore, when $\theta_{a,1} = \theta_{a,2} = \theta_{a,3}$ appears, we have

$$\theta_{a,1} = \theta_{a,2} = \theta_{a,3} = 0.$$

To further reduce the number of dimensions for θ, we set a baseline for the model. That is, let $\theta_{i,1} = 0$, $i = 1, 2, \cdots, 36$. $\theta_{i,1}$ is the parameters when $y = 0$. Hence, $p(y|x)$ will be changed into

$$p(y|x) = \frac{\exp\{\sum_{i,l} \theta_{i,l} I_{\{y=m\}} x_i\}}{1 + \exp\{\sum_i \theta_{i,1} x_i\} + \exp\{\sum_i \theta_{i,2} x_i\}}$$

where $l = 1, 2$ with respect to $m = -1, 1$. That is, $\theta_{i,1}$ represents the parameter with respect to x_i and $y = 1$, and $\theta_{i,2}$ represents the parameter with respect to x_i and $y = -1$. These actions above are called dimensionality reduction. To achieve the goal, Least Absolute Selection and Shrinkage Operator (LASSO) is introduced.

4.7. LASSO

As mentioned above, LASSO can reduce the dimensions of θ, that is, force

$$\theta_{a,1} = \theta_{a,2} = \theta_{a,3} = 0$$

when $\theta_{a,1} = \theta_{a,2} = \theta_{a,3} = C$, for some $a \in \{1, 2, \cdots, 36\}$ and $C \in \mathbb{R}$. The LASSO method used l_1-norm and Lagrange multiplier to achieve the goal. That is, instead of only minimizing $-l(\theta)$, we minimize

$$L(\theta) = -l(\theta) + \lambda \left\|\theta\right\|_1,$$

where θ is the 108-dimensional vector (as $3 \times 36 = 108$) with $\theta_{i,l}$ as its elements, $\lambda \in [0, \infty)$ is the Lagrange multiplier, and $\left\|\cdot\right\|_1$ is the norm-1 that is defined as:

$$\left\|\theta\right\|_1 = \sum_{i,l} |\theta_{i,l}|.$$

Notice that when λ is large enough, the norm-1 penalty will force $\theta = \mathbf{0}$, where $\mathbf{0} = (0, 0, 0 \cdots, 0)$ is a 108-dim vector. Hence, by applying an appropriate λ, the CRF model can provide more efficient and less confusing information about the correlation between the label y and x_i, where $i = 1, 2, \cdots, 36$. Because of the duality of the Lagrange form, minimizing $L(\theta)$ is equivalent to solving the problem of

$$\begin{gathered}\min_\theta -l(\theta) \\ \text{subject to } \left\|\theta\right\|_1 \leq t, \text{ for some } t \in [0, \infty).\end{gathered}$$

Since $-l(\theta)$ and the norm function are both convex functions, it implies that a unique solution for the problem exists.

However, there are mainly two difficulties when LASSO is applied. The first problem is, assuming the λ is given, the Lagrange function $(L(\theta))$ is not differentiable when $\theta_i = 0$, for some

$i \in \{1, 2, \cdots, 108\}$. As a result, the gradient descent method, which is mentioned above, can not be applied directly. To solve the problem, proximal gradient descent is introduced.

Proximal gradient descent is a general case for the projected gradient method. The idea is: First, use traditional gradient descent to find the solution for $-l(\theta)$, which is a differentiable function, then, use the project function to project the result onto the restrict function. In this case, the restrict function is the norm-1 function. The proximal gradient descent can effectively avoid the problem of non-differentiable function $L(\theta)$. The project function for this study is defined as:

$$Prox_{\tau^{(t)}}(z_{i,l}^{(t)}) = sign(z_{i,l}^{(t)})(|z_{i,l}^{(t)}| - \tau^{(t)})_+,$$

where

$$z_{i,l}^{(t)} = \theta_{i,l}^{(t)} - s^{(t)} \sum_{k=1}^{n} \left(x_i^{(k)} p(l|x^{(k)}, \theta_{i,l}^{(t)}) - I_{\{y=m\}} x_i^{(k)} \right),$$

and $\tau^{(t)} = \lambda s^{(t)}$. At this point, the proximal gradient descent is well defined, and the update rule can be rewritten as:

1. $z_{i,l}^{(t)} = \theta_{i,l}^{(t)} - \sqrt{\frac{1}{10^8(0.005+t)}} \sum_{k=1}^{n} \left(x_i^{(k)} p(l|x^{(k)}, \theta_{i,l}^{(t)}) - I_{\{y=m\}} x_i^{(k)} \right)$
2. $\theta_{i,l}^{(t+1)} = sign(z_{i,l}^{(t)})(|z_{i,l}^{(t)}| - \lambda\sqrt{\frac{1}{10^8(0.005+t)}})_+$,

where λ is the Lagrange multiplier,

$$sign(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}, \text{ and } (x)_+ = \max\{0, x\}.$$

The next difficulty is the choice of λ. To solve the problem, Pathwise Coordinate Descent (PCD) is applied. That is, first, select a λ large enough, say λ_0, so that the result of proximal gradient descent, $\theta^{(0)} = \mathbf{0}$. Then, reduce the λ a little bit, say λ_1, re-apply the proximal gradient descent using the result for the previous proximal gradient descent, (i.e. $\theta^{(0)}$) as the initial value to apply PCD again. After several iterations, find the best λ.

By applying both Proximal Gradient Descent and Pathwise Coordinate Descent, the LASSO method is complete. Hence, the Conditional Random Field can now, be applied. Up to this point, the CRF with LASSO regulation can be computed. To test the accuracy of the CRF, a numeric experiment is necessary. The numeric experiment will discuss the results under two different types of data.

5. NUMERIC EXPERIMENT

5.1. Binary response data

To demonstrate the efficiency of classical method and the CRF, the simulation under two possible outcomes situation was conducted. The gene simulation included 5 plants. Each plant contained 500 genes. There were 3 different scenarios in the Experiment. First, 10 out of 500 genes genes were changed by the fungus. Second, 250 out of 500 genes were changed. Last, 490 out of 500 genes were changed by the fungus.

The simulation assumed that these genes' activities were observed 7 times during the experiment period and the gene activity data was read and qualified into gene expression level by CUFFLINK and CUFFDIFF [Trapnell et al., 2012]. The dataset selected out the maximum log-fold from all the log-folds of the gene's expression level. The log-fold was the nature log of the result that calculated by dividing the previous gene expression level from the current one. In addition, the simulation assumes that the log-fold was following a normal distribution ($N_6(\tilde{0}, \Sigma)$) when the gene cannot pass the threshold. Otherwise, the log-fold was following a normal distribution ($N_6(\tilde{\mu}, \Sigma)$), where $\max_t |\mu_t| > 0$. The Σ of both distributions was the covariance matrix that

$$\Sigma = \begin{pmatrix} 1 & \rho & 0 & 0 & 0 & 0 \\ \rho & 1 & \rho & 0 & 0 & 0 \\ 0 & \rho & 1 & \rho & 0 & 0 \\ 0 & 0 & \rho & 1 & \rho & 0 \\ 0 & 0 & 0 & \rho & 1 & \rho \\ 0 & 0 & 0 & 0 & \rho & 1 \end{pmatrix},$$

where ρ is set to be 0.5.

The following plots are the log-fold of those 10 genes that are changed by the fungus.

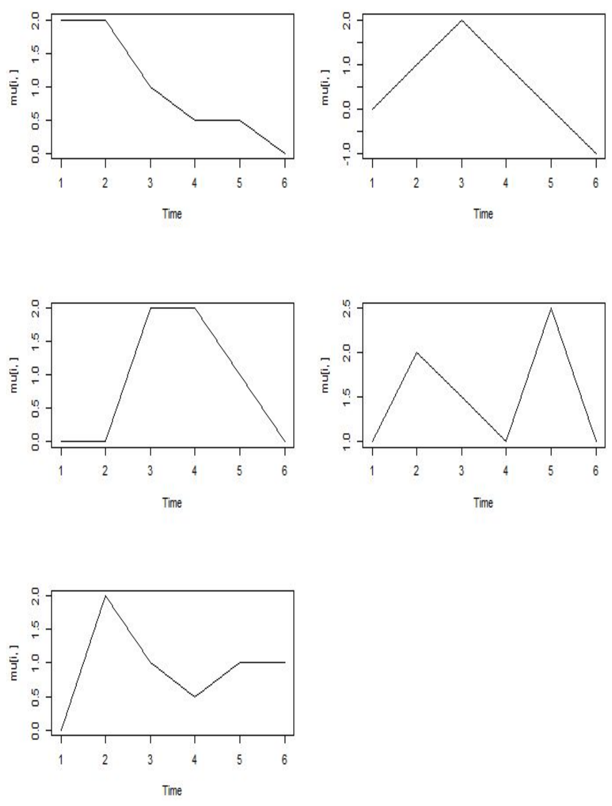

Figure 5.1. The changed genes' expression level

The plots have 5 different situations, and each situation has different replications depends on the scenarios, which are mentioned above. Moreover, to simulate the sensitive of the model, the magnitude, which is set from 1 to 3 by 1, are also added in the maximum log-fold dataset.

The dataset that simulated based on the setting above is used to test the efficiency of the CRF.

5.1.1. The result

After introducing the data simulation situation, both Benjamini-Hochberg method (B-H) and CRF are applied. This binary numeric experiment focus on comparing two models' accuracy with respect to different scenario and different magnitude of the data.

In this numeric experiment, the B-H method used $\alpha = 0.05$ and the CRF used $p(y = 1|x) > 0.65$ as criteria to determine and label the changed genes. The table below shows the result generated from the binary numeric experiment using both B-H method and CRF method.

Table 5.1. Binary numeric experiment

Scenario (changed/total)	Magenitude	B-H			CRF			
		Reject	Success	FDR	Reject	Success	FDR	λ
10/500	1	21	4	0.810	3	1	0.667	0.987
	2	25	10	0.600	5	5	0.000	7.562
	3	20	10	0.500	5	5	0.000	6.576
250/500	1	104	96	0.077	297	240	0.192	12.821
	2	256	244	0.047	293	250	0.147	12.493
	3	258	250	0.031	274	250	0.088	12.821
490/500	1	224	224	0.000	476	472	0.008	12.821
	2	482	482	0.000	494	490	0.008	12.821
	3	490	490	0.000	490	490	0.000	10.849

There are 4 conclusions can be draw from table 5.1. First, in the scenario of 10 out of 500 genes are changed by fungus, the FDRs of CRF are lower than the FDRs of B-H method under all 3 magnitudes. However, B-H detected all 10 genes whereas the CRF detected only 5 when the magnitude set to 2 and 3.

Second, The B-H methods performs better than CRF when the scenario was set to 250 out of 500 genes are changed. However, CRF detected all 250 changed genes when the magnitude equals 2.

Third, When the scenario was set to 490 out of 500 genes are changed, both methods had a good performs. Moreover, CRF detected all 490 genes when magnitude equals 2 and 3.

Last, when magnitude increased, the FDR for both B-H method and CRF decreased.

In summary, when the data is under a normal scenario, that is, greater than 50% of the genes can be detected as changed genes, the B-H method has a stable and relatively efficient performance. However, when it comes to a more extreme case, the CRF outperforms the classical method. Furthermore, although the CRF did not have a lower FDR comparing with B-H method for the normal scenarios, it still has a relatively accurate predictions.

The purpose of this binary numeric experiment is to compare the efficiency of the traditional classification method for the gene expression analysis (i.e. B-H method) and the CRF. However, since the purpose of the study is to classify the gene in the plant genome into three groups, the ternary numeric experiment is necessary.

5.2. Ternary response data

As mentioned above, the final goal is to separate the gene into three groups (positive reaction, no reaction, and negative reaction) using CRF. In order to estimate the accuracy of the CRF, the simulation study is very necessary. The dataset of the simulation is conducted in a scenario of a 1000 observations with 100 features. The observations is labeled into three groups: 0, 1 and -1. The first 5 features of the observation determines the label. Therefore, the rest of 95-dim features are useless.

5.2.1. Label 1

For the observation of label 1, we have

$$x_1 \sim U(0.075, 1.025), x_2 \sim N(4x_1, 0.025^2), x_3 \sim U(x_2 + 1.075, x_2 + 2.025),$$
$$x_4 \sim N(7, 0.05^2), x_5 = x_4 + U(0.075, 1.025)$$

For the other dimensions of the observation, we have

$$(x_6, x_7, x_8) \sim N_3((6, 7, 8), \Sigma_0) \text{ and } (x_{15}, x_{16}, \cdots, x_{100}) \sim N_{86}(\mathbf{8}, \Sigma_1),$$

$\mathbf{8}$ are vectors, in which each element is 8. The covariance matrix

$$\Sigma_0 = \begin{pmatrix} 0.05 & 0.025 & 0.01 \\ 0.025 & 0.05 & 0.01 \\ 0.01 & 0.01 & 0.05 \end{pmatrix} \text{ and } \Sigma_1 = diag(1)_{86\times 86}.$$

Letting the rest of 95 dimensions independent from the first 5 dimensions will let the parameters of these dimensions equal to 0. Notes that, the mean of the first 5 dimensions is increasing, this is because the label 1 is simulating the positive affect for the gene expression.

5.2.2. Label 0

For the observation of label 0, we have

$$x_1 \sim N(1, 0.05^2), x_2 \sim N(2, 0.025^2), x_3 \sim U(x_2 - 0.025, x_2 + 2.025),$$
$$x_4 \sim U(x_3 - 1.025, x_3 - 0.075), x_5 \sim N(x_4 + 1, 0.05^2)$$

For the other dimensions of the observation, we have

$$x_6 \sim U(7,8), (x_7, x_8, x_9) \sim N_3((7,8,9), \Sigma_2),\ (x_{15}, x_{100}) \sim N_{86}(\mathbf{8}, \Sigma_1),$$

where **8** are vectors, in which each element is 8. The covariance matrix

$$\Sigma_2 = \begin{pmatrix} 0.05 & 0.01 & 0.01 \\ 0.01 & 0.05 & 0.01 \\ 0.01 & 0.01 & 0.05 \end{pmatrix}$$

Again, letting the rest 95 dimensions independent from first 5 dimensions will let the parameters of these dimensions equal to 0. Notes that, the mean of the first 5 dimensions is neither increasing nor decreasing, this is because the label 0 is simulating the no reaction for the gene expression.

5.2.3. Label -1

For the observation of label -1, we have

$$x_1 \sim N(8, 0.05^2), x_2 \sim N(x_1 - 2, 0.025^2), x_3 \sim U(3.075, 4.025),$$
$$x_4 \sim U(x_3 - 1.025, x_3 - 0.075), x_5 \sim N(x_4 - 2, 0.025^2)$$

For the other dimensions of the observation, we have

$$x_6 \sim U(7,8), (x_7, x_8, x_9) \sim N_3((7,8,9), \Sigma_2),\ (x_{15}, x_{100}) \sim N_{86}(\mathbf{8}, \Sigma_1),$$

where the covariance matrix

$$\Sigma_2 = \begin{pmatrix} 0.05 & 0.01 & 0.01 \\ 0.01 & 0.05 & 0.01 \\ 0.01 & 0.01 & 0.05 \end{pmatrix}$$

Notes that, the mean of the first 5 dimensions is decreasing, this is because the label 0 is simulating the negative effect for the gene expression.

5.2.4. Additional setting

For x_9 in label 0, $x_{10}, \cdots, x_{14}$, we designed as following:

- $x_9 \sim N(9, 0.05^2)$ in label 0.
- $x_{10} \sim N(x_9, 0.05^2)$.
- $x_{12} \sim U(x_{11} - 0.025, x_{11} + 0.025)$.
- $x_{14} \sim U(\frac{x_{13}+x_{15}-0.05}{2}, \frac{x_{13}+x_{15}+0.05}{2})$.

- $x_{11}, x_{13}, x_{14} \sim N(8, 0.05^2)$

Moreover, Similar to the scenario settings for the binary numeric experiment, the ternary numeric experiments contains 3 different scenarios.

1. 20 out of 1000 genes are changed by the fungus
   - 10 out of 1000 observations have 98.5% chance to be label 1.
   - 10 out of 1000 observations have 97.6% chance to be label -1.
   - 980 out of 1000 observations have 94.8% chance to be label 0.
2. 500 out of 1000 genes are changed by the fungus
   - 250 out of 1000 observations have 98.5% chance to be label 1.
   - 250 out of 1000 observations have 97.6% chance to be label -1.
   - 500 out of 1000 observations have 94.8% chance to be label 0.
3. 980 out of 1000 genes are changed by the fungus
   - 490 out of 1000 observations have 98.5% chance to be label 1.
   - 490 out of 1000 observations have 97.6% chance to be label -1.
   - 20 out of 1000 observations have 94.8% chance to be label 0.

The following plot is the mean of 100-dim features for label 0, label 1, and label -1.

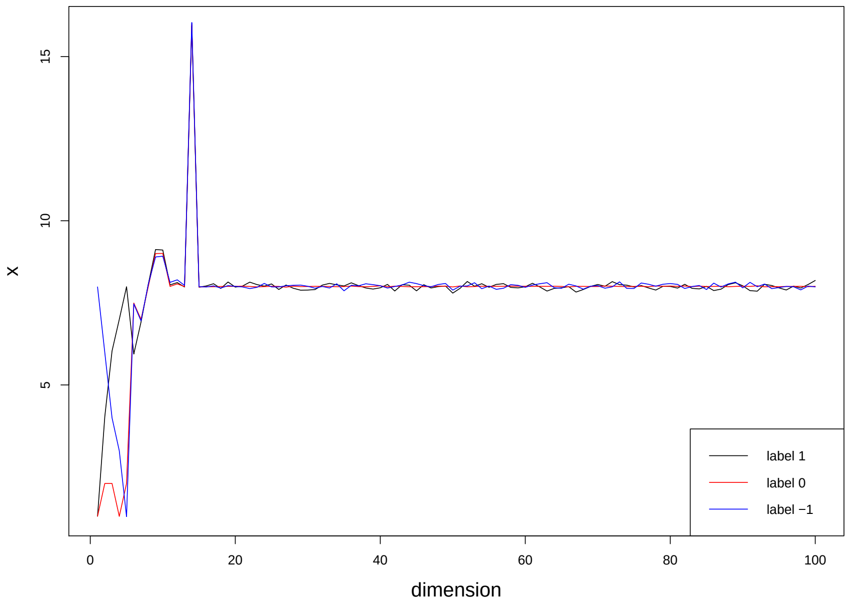

Figure 5.2. Ternary response data

In figure 5.2, the black line is the mean of 100-dim features labeled as 1. This line is simulating the positively changed genes. The red line is the mean of 100-dim features labeled as 0, which is simulating the unchanged genes. The blue line is the mean of 100-dim features labeled as -1, which is simulating the negatively changed genes. As the plot shows, after 5th dimension, the features tends to be alike, therefore, the feature after 5th dimension are useless. The CRF model results is generated under this simulation setting. In order to compare the result with some classical model, the Generalized Linear Model (GLM) under the assumption of multinomial is used.

5.2.5. Result

For the simulation, The λ in CRF is designed in the range from 500 to 0.01 since when $\lambda = 500$, $\theta_{i,l} \approx 0, \forall i = 1, 2, \cdots, 100, \forall l = 1, 2.$

To illustrate the efficiency of the CRF model, the classical generalized linear model (GLM) under the assumption of multinomial is used. There are mainly 2 aspects to analyze, the accuracy prediction of the label, and l_1 norm between the estimator and the true parameter.

5.2.6. Accuracy prediction of the label

The accuracy report for the model, that is, the proportion of accurately predict the observation when the model labels the observation as 1 or -1. There are two comparisons for this aspect: All CRF using different λ values, and the accuracy between the best CRF and the GLM model.

First, All CRF using different λ values. The following table shows the result under the scenario 20 out of 1000 genes are changed by fungus. The λ for the CRF model is in between 4.274 and 2.960.

Table 5.2. CRF model for the scenario 20 out of 1000

λ	4.274	3.946	3.617	3.288	2.960
n_1'	10	10	10	10	10
n_1	9	9	9	9	9
n_2'	7	0	10	10	10
n_2	7	0	10	10	10
r_1	0.900	0.900	0.900	0.900	0.900
r_2	1.000	NA	1.000	1.000	1.000
FDR	0.059	0.100	0.050	0.050	0.050

This table reports the result when λ is in a range from 2.960 to 4.274. The n_1' and the n_2' represent the number of data that are labeled as 1 and -1 by the CRF respectively and the n_1 and the n_2 are the number of correct reports within those reports. r_1 and r_2 are the true ratio for label 1 and label -1 respectively, which are

$$r_1 = \frac{n_1'}{n_1} \text{ and } r_2 = \frac{n_2'}{n_2}.$$

Lastly, the FDR is computed by the formula

$$FDR = 1 - \frac{n_1+n_2}{n_1'+n_2'}$$

The table shows that when λ is in between 2.960 and 3.617, the CRF gives the best result. That is, within 10 observations reported as label 1, 9 of them are correct. Also, 10 out of 10 observations reported as -1 are correct. For the rest λ, there are two problems, either no report for label 1 and label -1 or shows higher FDR.

As mentioned in the methodology LASSO section, when λ is increasing, the estimator will closer to 0, hence, will have more 0 dimensions. For this simulation, since the CRF with

$2.960 \leq \lambda \leq 3.617$ have the same result, the model with higher λ is more efficient (use less dimensions to predict the accurate result). Hence, the CRF with $\lambda = 3.617$ is selected.

The next comparison is between CRF and GLM. The following table shows the result:

Table 5.3. CRF and GLM for the scenario 20 out of 1000

Model	n_1'	n_1	n_2'	n_2	FDR
CRF $\lambda = 3.617$	10	9	10	10	0.05
GLM	47	0	19	0	1

Table 5.3 implies that the CRF is more accurate since the FDR for CRF is 0.05 whereas the FDR for GLM is 1.

The next table shows the result of CRF for the scenario that 500 out of 1000 genes are changed by the fungus.

Table 5.4. CRF model for the scenario 500 out of 1000

λ	89.752	76.932	64.111	51.291	38.471
n_1'	198	240	250	250	250
n_1	184	225	234	234	234
n_2'	0	10	250	250	250
n_2	0	8	219	219	219
r_1	0.929	0.938	0.936	0.936	0.936
r_2	NA	0.800	0.876	0.876	0.876
FDR	0.071	0.068	0.094	0.094	0.094

Similar to the first scenario, $\lambda = 64.111$ was selected because CRFs with $38.471 \leq \lambda \leq 64.111$ has the smallest FDR with the largest λ. The next table shows the result for the comparison between CRF and GLM.

Table 5.5. CRF and GLM for the scenario 500 out of 1000

Model	n_1'	n_1	n_2'	n_2	FDR
CRF $\lambda = 64.111$	250	234	250	219	0.094
GLM	246	197	278	214	0.216

Table 5.5 implies that the CRF has a more accurate prediction since it contains the smallest FDR (FDR = 0.094).

The next table shows the result of CRF for the scenario that 980 out of 1000 genes are changed by the fungus.

Table 5.6. CRF model for the scenario 980 out of 1000

λ	141.033	128.213	115.392	102.572	89.752
n_1'	490	490	490	490	490
n_1	460	460	460	460	460
n_2'	308	478	490	490	490
n_2	258	420	431	431	431
r_1	0.939	0.939	0.939	0.939	0.939
r_2	0.838	0.879	0.880	0.880	0.880
FDR	0.100	0.091	0.091	0.091	0.091

Table 5.6 implies that when λ is in the range of $89.752 \leq \lambda \leq 128.213$, the FDR for CRF corresponds to the λ are the same. Moreover, when $\lambda = 115.392$, r_2 reached to the highest value (i.e. 0.880). Hence, the CRF with $\lambda = 115.392$ was selected in the study.

The next table is the comparison result between CRF with $\lambda = 115.392$ and GLM

Table 5.7. CRF and GLM for the scenario 980 out of 1000

Model	n_1'	n_1	n_2'	n_2	FDR
CRF $\lambda = 115.392$	490	460	490	431	0.091
GLM	408	382	389	358	0.072

Unlike previous two scenarios, the GLM in the scenario 980 out of 1000 generated a more accurate result. That is, the FDR in GLM is smaller than the FDR in CRF, $(0.072 < 0.091)$.

Two conclusions can be drawn when combining all three scenarios together. First, the FDR in GLM largely decreased when the ratio of changed gene increased. $(1 \geq 0.216 \geq 0.072)$. Furthermore, the FDR for GLM in the last scenario is less than the one for CRF. Hence, the GLM is suitable for the data in which there are significant proportion of changed genes. Second, the CRF outperformed GLM when the ratio of changed/total genes is small. In addition, although the FDR for the GLM is smaller than the one for CRF in the last scenario, the difference between these two FDR is not that significant (0.019). Hence, the CRF is suitable for overall scenarios. The next important study for the model is to demonstrate the effectiveness of LASSO penalty.

5.2.7. l_1 norm between the estimator and the true parameter

To determine the effectiveness of the LASSO, the l_1 norm distance between the true parameter and the estimated parameter is used. As discussed in the previous section, the comparison is between the CRF and the GLM model. The following table shows the result.

Table 5.8. l_1 norm between GLM and CRF

Scenario (changed/total)	GLM	CRF
20/1000	170502.800	140.357
500/1000	456889.900	141.258
980/1000	3686.477	140.578

The large difference of l_1 norm between GLM and CRF is expected. For example, under the first scenario, the l_1 distance between the true parameter and the CRF is 140.357, whereas the distance between the true parameter and the estimator of GLM model is 170502.800. The large different between these two distance is because the true parameter only have 5 dimensions of the data are not equal to 0. The estimator of CRF parameter contains 65 dimensions that are estimated as 0. On the other hand, GLM assumed 31 dimensions are not significantly equals 0, which leads to a higher l_1 distance.

5.3. Numeric experiment summary

By analyzing these two data scenarios, which are binary and ternary response data, there are 2 conclusions. First, the CRF was outperformed both the Hypothesis test with B-H method and the GLM under the extreme data scenario. Although those classical method were more accurate when the changed-total ratio is large, the CRF still had a relatively good result. Second, the CRF with LASSO effectively reduced the dimensions of the parameters so that the model can use less variable to make a relatively accurate prediction.

Since the numeric experiment provided enough evidence of the usefulness and the effectiveness of CRF with LASSO, the CRF with LASSO can now be applied to the plant science dataset.

6. CRF WITH PLANT SCIENCE DATA

6.1. Prepare the data

After demonstrating the effectiveness of the CRF, the model can be applied to the plant science dataset, whose details are introduced in the plant science data section. Before applying the model, there are two problems that need to be solved. First, the range of the data is too large (from 4.5×10^{-11} to 1.4×10^{10}). Because of the range problem, the CRF can not be applied directly. Hence, it is very important to shrink the range of the data to a reasonable size. In this study, the range was shrank in to the range from 8.2×10^{-17} to 25000.

In addition to the range problem, the second problem is, this data have no training dataset, that means, we need to manually label some of the observations to create a training dataset. The problem of the manually labeling is that it is very time consuming and can only label these observations by visually check the observation's plot. Because of this, the training data was not totally certified by the plant science, and the result can only be used as a suggestion for the future analysis. We selected 200 observations and labeled it as following:

- Label 0: The gene expression have no significant different pattern in both environments, which are original isolate and mutant isolate.
- Label 1: The gene expression does not change in the mutant isolate environment and have changes in the original one.
- Label -1: The gene expression does not change in the original isolate environment and have changes in the mutant one.

Finally, 171 out of 200 genes were labeled as 0, 19 out of 200 genes were labeled as 1, and 10 out of 200 genes were labeled as -1.

The plot for one example of the raw training data that is labeled as 0 is shown below.

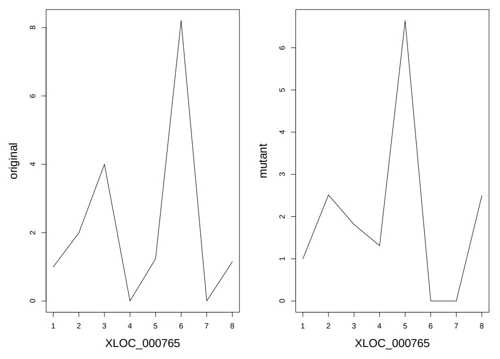

Figure 6.1. Gene labeled as 0

Figure 6.1 shows that if the gene shows no significant different pattern between different environments, then the gene is labeled as 0.

The plot for one example of the raw training data that is labeled as 1 is shown below.

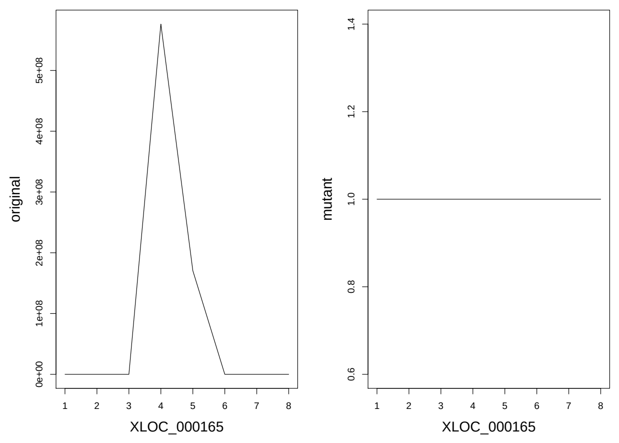

Figure 6.2. Gene labeled as 1

Figure 6.2 shows that if the gene shows activities in original isolate environment and shows no activities in mutant isolate environment, then the gene is labeled as 1.

The plot for one example of the raw training data that is labeled as -1 is shown below.

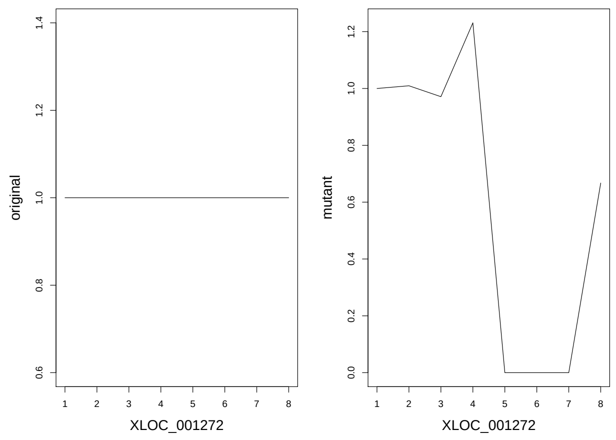

Figure 6.3. Gene labeled as -1

Figure 6.3 shows that if the gene shows activities in mutant isolate environment but shows no activities in original isolate environment, then the gene is labeled as -1.

The training step of CRF used these 200 genes and their gene expression ratio, which is introduced in the plant science data section, as the training data to train the CRF model, and the training data ratio plot is shown below.

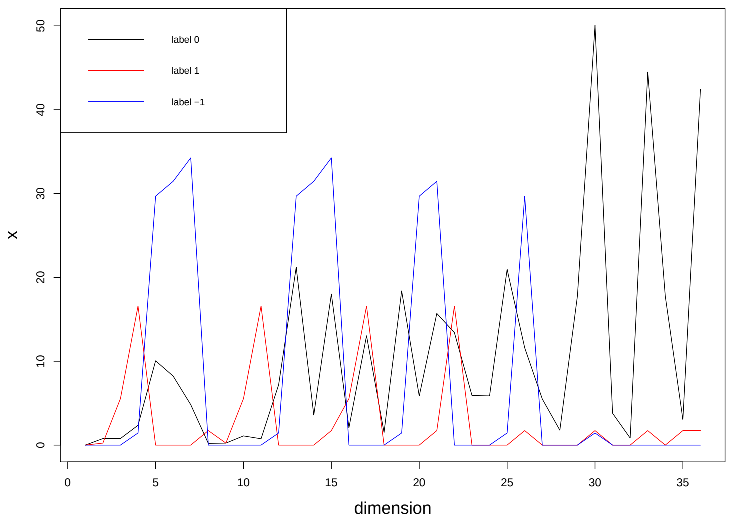

Figure 6.4. The training data ratio plot

Figure 6.4 shows 3 ratio patterns with respect to three different labels. Although these patterns showed less significant difference comparing with the simulation data, the CRF model with $\lambda = 0.001$ still can classify the data with a reasonable low FDR. The following table is the FDR ratio for the CRF model with different λ.

Table 6.1. The FDR for different λ

λ	51.283	38.462	25.642	12.821	0.001
n'_1	1	1	1	1	1
n_1	1	1	1	1	1
n'_2	0	0	0	1	3
n'_2	0	0	0	0	2
r_1	1	1	1	1	1
r_2	NA	NA	NA	0	0.67
FDR	0	0	0	0.5	0.25

Table 6.1 shows that the only useful CRF is when $\lambda = 0.001$ as the rest of λ cannot accurately label -1. In addition, the FDR for the CRF when $\lambda = 0.001$ is 0.25, which is acceptable. When $\lambda = 0.001$, the CRF can correctly detect one gene labeled as positive, 2 out of 3 genes labeled as negative. After determined the CRF model, the model was applied to label the rest of the data.

6.2. Data result summary

The CRF model with $\lambda = 0.001$ was applied to the rest 6125 genes. Within these data, the model labeled 41 genes as positively affected. 263 genes as negatively affected. The plant science has more interest in the positively affected gene since these genes have a relatively high activities under the environment of original isolation, which is highly virulent on Bowman.

Although the training data were not totally certified by plant science, the result for the data analysis still can provide some research directions for plant science. 28 out of 41 are certified by visually check. 3 out of 41 should belongs to negative group, and 10 out of 41 cannot be certified by visually check. Up to this point, this study suggest that these 28 genes should be further studied, and 10 genes that cannot be certified by visually check also can be studied. The result can be found in the appendix.

REFERENCES

[Ammar et al., 2014] Ammar, W., Dyer, C., and Smith, N. A. (2014). Conditional random field autoencoders for unsupervised structured prediction. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3311–3319. Curran Associates, Inc.

[Ash and Doleans-Dade, 1999] Ash, R. B. and Doleans-Dade, C. A. (1999). *Probability and Measure Theory*. Harcourt/Academic Press, second edition.

[Bottou, 2010] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *in COMPSTAT*.

[Casella and Berger, 2001] Casella, G. and Berger, R. (2001). *Statistical Inference*. Duxbury Resource Center.

[Ciuperca, 2016] Ciuperca, G. (2016). Adaptive lasso model selection in a multiphase quantile regression. *Statistics*, 50(5):1100–1131.

[Cui and Wang, 2016] Cui, C. and Wang, D. (2016). High dimensional data regression using lasso model and neural networks with random weights. *Inf. Sci.*, 372:505–517.

[Hastie et al., 2015] Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC.

[Izenman, 2008] Izenman, A. J. (2008). *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer Publishing Company, Incorporated, 1 edition.

[J Motyer et al., 2011] J Motyer, A., McKendry, C., Galbraith, S., and Wilson, S. (2011). Lasso model selection with post-processing for a genome-wide association study data set. *BMC proceedings*, 5 Suppl 9:S24.

[Kiefer and Wolfowitz, 1952] Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23.

[Lafferty et al., 2001] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Montgomery, 2006] Montgomery, D. C. (2006). *Design and Analysis of Experiments.* John Wiley & Sons, Inc., USA.

[Qianwen et al., 2018] Qianwen, L., Qingchuan, T., Yalin, Z., and Manxiao, L. (2018). Sketch simplification based on conditional random field and least squares generative adversarial networks. *Neurocomputing*, 316.

[Sha and Pereira, 2003] Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 134–141, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Shalev-Shwartz and Ben-David, 2014] Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, New York, NY, USA.

[Shao, 2003] Shao, J. (2003). *Mathematical Statistics.* Springer Texts in Statistics. Springer.

[Sutton and McCallum, 2012] Sutton, C. and McCallum, A. (2012). An introduction to conditional random fields. *Found. Trends Mach. Learn.*, 4(4):267–373.

[Thiagarajan and Bremananth, 2015] Thiagarajan, B. and Bremananth, R. (2015). Brain image segmentation using conditional random field based on modified artificial bee colony optimization algorithm.

[Trapnell et al., 2012] Trapnell, C., Roberts, A., Goff, L., Pertea, G., Kim, D., R Kelley, D., Pimentel, H., Salzberg, S., Rinn, J., and Pachter, L. (2012). Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature protocols*, 7:562–78.

[Zaki and Jr, 2014] Zaki, M. J. and Jr, W. M. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms.* Cambridge University Press, New York, NY, USA.

APPENDIX A. THE GENE STATUS

Gene id	Status	Gene name
XLOC_000165	Sure	gene:MLOC_34615
XLOC_000206	Sure	gene:MLOC_70053
XLOC_000446	Sure	gene:MLOC_53979
XLOC_001271	Sure	gene:MLOC_38003
XLOC_001548	Sure	gene:MLOC_81846
XLOC_002973	Sure	gene:MLOC_2643
XLOC_003663	Sure	gene:MLOC_45012
XLOC_005291	Sure	gene:MLOC_77287
XLOC_007644	Sure	gene:MLOC_7939
XLOC_008226	Sure	gene:MLOC_56670
XLOC_008882	Sure	gene:MLOC_14530
XLOC_010024	Sure	gene:MLOC_3643
XLOC_010931	Sure	gene:MLOC_56924
XLOC_012053	Sure	gene:MLOC_59105
XLOC_012661	Sure	gene:MLOC_63737
XLOC_013304	Sure	gene:MLOC_44080
XLOC_014150	Sure	gene:MLOC_62978
XLOC_014369	Sure	gene:MLOC_4747
XLOC_015012	Sure	gene:MLOC_44410
XLOC_015369	Sure	gene:MLOC_35028
XLOC_015734	Sure	gene:MLOC_19822
XLOC_016272	Sure	gene:MLOC_68290
XLOC_017008	Sure	gene:MLOC_37653
XLOC_017070	Sure	gene:MLOC_64575
XLOC_017078	Sure	gene:MLOC_60337
XLOC_017510	Sure	gene:MLOC_28088
XLOC_017817	Sure	gene:MLOC_9498
XLOC_020205	Sure	gene:MLOC_7742

Gene id	Status	Gene name
XLOC_004384	Error	gene:MLOC_37835
XLOC_007363	Error	BBBI
XLOC_020189	Error	gene:MLOC_65964
XLOC_001594	Not sure	gene:MLOC_7036
XLOC_005095	Not sure	gene:MLOC_60152
XLOC_006087	Not sure	gene:MLOC_61500
XLOC_006876	Not sure	gene:MLOC_55637
XLOC_006995	Not sure	gene:MLOC_70921
XLOC_009096	Not sure	gene:MLOC_37446
XLOC_016348	Not sure	gene:MLOC_78460
XLOC_021823	Not sure	gene:MLOC_61260
XLOC_024611	Not sure	psbK
XLOC_025581	Not sure	gene:MLOC_47215

APPENDIX B. CODE

```
############## The true parameters and the data ##############
# theta.true = matrix(c(    0,     0,   0,  0,  0, rep(0, 95),
#                       -23.5, -16.6,  32, 10, -6, rep(0, 95),
#                        -7.7,  -8.7, 8.4, -5,  7, rep(0, 95)),
#                        nrow = 3, byrow = T)
#
# data = read.table(file = "data/simulation data test.txt", header = T)
# theta.new = matrix(rep(0, 300), nrow = 3, byrow = T)
# theta.ini = theta.new

###### Function sections ######
#-------------- Create p(y|x) --------------
#---------- Input ----------
# theta: The parameters of theta_{ij}, a 2X100 matrix where
# theta =
#         theta_01 theta_02,..., theta_(0 100)
#         theta_11 theta_12,..., theta_(1 100)
#
# data: The data frame that includes:
#       label (i): The label corresponds to the gene
#       X1-X100: The observed variable for the gene:
#       x_{i1}, x_{i2},..., x_{i 100}
#---------- Output ----------
# output: The data frame includes:
# theta.x: The data frame include:
#          p.0: p(y = 0|x)
#          p.1: p(y = 1|x)
#          other treatments the same as data.
#------------------------------------------------------------

p.y.x = function(theta, data){
x = as.matrix(data[, -1])
temp = theta%*%t(x)
temp[temp>=709] = 709
temp[2, ] = temp[1,]+temp[2,]
temp[3, ] = temp[1,]+temp[3,]
result = exp(temp)

p.y.x = apply(result, 2, sum)
p.0 = result[1, ]/p.y.x
p.1 = result[2, ]/p.y.x
p.2 = result[3, ]/p.y.x
result = t(rbind(p.0, p.1, p.2))

theta.x = data.frame(result, data)
names(theta.x)[1:3] = c("p.0", "p.1", "p.2")

theta.x$p.0[theta.x$p.0 == -Inf] = 0
theta.x$p.1[theta.x$p.1 == -Inf] = 0
theta.x$p.2[theta.x$p.2 == -Inf] = 0

theta.x$p.0[theta.x$p.0 == Inf] = 1
theta.x$p.1[theta.x$p.1 == Inf] = 1
theta.x$p.2[theta.x$p.2 == Inf] = 1
return(theta.x)
}
# new.data = p.y.x(theta.ini, data)
# new.data = p.y.x(theta.new, data)

# sapply(new.data[1:10, 1:3], mean)

# sapply(new.data[11:20, 1:3], mean)

# sapply(new.data[21:30, 1:3], mean)

# rm(new.data)


#-------------- The first derivative of log(p(y|x)) --------------

# Require: the outcome from p.y.x

#---------- Input ----------

# new.data: The data frame that generated by p.y.x function with:

#           p.0: p(y = 0|x)

#           p.1: p(y = 1|x)

#           label (i): The label corresponds to the gene

#           X1-X100: The observed variable for the gene:

#           x_{i1}, x_{i2},..., x_{i 100}

#---------- Output ----------

# output: The matrix of d(log(p(y|x))/theta_{ij} where

#         i = 0, 1

#         j = 1, 2,...,100

#------------------------------------------------------------


d.log = function(new.data){

new.data.temp = data.frame(new.data,

I.1 = 0,

I.2 = as.numeric(new.data$label == 1),

I.3 = as.numeric(new.data$label == -1))

x = as.matrix(new.data[-(1:4)])

p = t(as.matrix(new.data[1:3]))

p[1, ] = 0

ncol = dim(new.data.temp)[2]
```

```
I = t(as.matrix(new.data.temp[(ncol-2):ncol]))
d.log = p%*%x-I%*%x
return(d.log)
}
# d.l = d.log(new.data)
# rm(d.l)

#------------- The L(theta) = sum_n^N(log(p(y|x))) -------------
#---------- Input ----------
# theta: The parameters of theta_{ij}, a 2X2 matrix where
# theta =
#         theta_01 theta_02,..., theta_(0 100)
#         theta_11 theta_12,..., theta_(1 100)
#
# data: The data frame that includes:
#       label (i): The label corresponds to the gene
#       X1-X100: The observed variable for the gene:
#       x_{i1}, x_{i2},..., x_{i 100}
#---------- Output ----------
# output:
# L(theta) = sum_n^N(log(p(y|x)))
#------------------------------------------------------------

log.l = function(theta, data){
new.data = data.frame(data,
I.1 = 1,
I.2 = as.numeric(data$label == 1),
I.3 = as.numeric(data$label == -1))
x = t(as.matrix(data[-1]))
ncol = dim(new.data)[2]
```

```
I = t(as.matrix(new.data[(ncol-2):ncol]))
theta.x = theta%*%x
Ix = I*theta.x

result.1 = exp(Ix[1, ])
result.2 = exp(Ix[2, ])
result.3 = exp(Ix[3, ])

z = result.1+result.1*result.2+result.1*result.3

log.l = sum(Ix)-sum(log(z))
if(log.l == -Inf){log.l = -2^31-1}
return(log.l)
}

# l.l = log.l(theta.true, data)
# l.l = log.l(theta.ini, data)
# rm(l.l)

#-------------- The Proximal gradient method --------------
# The proximal gradient method specially for the CRF
# with L1 norm restrict. The initial guess is 0.
# This function needs:
#                      p.y.x(theta.new, data)
#                      d.log(new.data)
#                      log.l(theta.new, data.label)
#
#---------- Input ----------
# theta.new: The parameters of theta_{ij}, a 2X2 matrix where
# theta =
#         theta_01 theta_02,..., theta_(0 100)
#         theta_11 theta_12,..., theta_(1 100)
#
# data: The data frame that includes:
#       label (i): The label corresponds to the gene
#       X1-X100: The observed variable for the gene:
#       x_{i1}, x_{i2},..., x_{i 100}
#
# lambda: The Lagrangian multiplier parameter [0, 1] for lasso restriction
# Stepwise: The length for one steps with the formula
#                 (stepwise = 10^8 by default): eta = 1/stepwise*(m0+m)
# m0: The initial length for one steps with the formula
#        (m0 = 0.005 by default): eta = 1/stepwise*(m0+m)
#---------- Output ----------
# output:
# A list of:
#            Time: How many iterations for the algorithm
#            theta.est: The parameters estimated
#------------------------------------------------------------

Proximal = function(theta.new, data, lambda, stepwise = 10^8, m0 = 0.005){
m = 0; criteria = 2^31-1; ll.list = c(); lagrange = c()

while (criteria >= 10^-5){
theta = theta.new
new.data = p.y.x(theta, data)
d.l = d.log(new.data)
ll.old = log.l(theta, data)
theta.old = theta
```

```
ll.list = c(ll.list, ll.old)
lagrange = c(lagrange, -ll.old+lambda*sum(abs(theta.old)))

# eta = sqrt(B^2/(rho^2*m))
eta = sqrt(1/(stepwise*(m0+m)))
z = theta.old-eta*d.l
z.minus.tau = abs(z)-eta*lambda
z.minus.tau[z.minus.tau<0] = 0

theta.new = sign(z)*z.minus.tau
ll.new = log.l(theta.new, data)

# if(max(is.na(theta.new))|sum(abs(theta.new-theta.old))<=10^-5){
if(max(is.na(theta.new))){
cat("m = ", m, " and theta is NULL or theta is unchanged", "\n")
break
}else{
# theta.sum = theta.sum+theta
if(ll.new == -2^31-1 & ll.old == -2^31-1){
criteria = 2^31-1
}else{
criteria = abs(ll.old-ll.new)
}
cat("m = ", m, ", and diff = ", criteria, ",
and -L(theta) = ", -ll.new, ",and lambda = ", lambda, "\n")
m = m+1
}
}
Time = m-1
return(list(Time = Time, theta.est = theta.new,
          ll.list = ll.list, lagrange = lagrange))
}


# Proximal(theta.true, data, 500)
# Proximal(theta.ini, data, 500)



#-------------- The function computes the label --------------
# Require: p.y.x(theta, data)
#---------- Input ----------
# theta.new: The parameters of theta_{ij}, a 2X2 matrix where
# theta =
#         theta_01 theta_02,..., theta_(0 100)
#         theta_11 theta_12,..., theta_(1 100)
#
# data: The data frame that includes:
#       label (i): The label corresponds to the gene
#       X1-X100: The observed variable for the gene:
#       x_{i1}, x_{i2},..., x_{i 100}
#---------- Output ----------
# output:
# A data set of the labels for observed data
#------------------------------------------------------------


test = function(theta, data, p.value){
test = p.y.x(theta, data)
test = data.frame(test[, 1:4])
test$label.est = NA
test$label.est[test$p.1>=p.value] = 1
test$label.est[test$p.2>=p.value] = -1
```

```
return(test)

}


# test.est = test(theta.true, data)

# sum(test.est$label.est == data$label)


#------ The function computes the log-likelihood

#------ and the first derivation of log-l ------

# Require: p.y.x(theta, data)

#---------- Input ----------

# theta =

#         theta_01 theta_02,..., theta_(0 100)

#         theta_11 theta_12,..., theta_(1 100)

#

# data: The data frame that includes:

#       label (i): The label corresponds to the gene

#       X1-X100: The observed variable for the gene:

#       x_{i1}, x_{i2},..., x_{i 100}

#---------- Output ----------

# output:

# ll: Log-likelihood

# d.l: First derivation of log-likelihood

#------------------------------------------------------------


Log.l.d.log = function(theta, data){

new.data = p.y.x(theta, data)

ll = log.l(theta, data)

d.l = d.log(new.data)

return(list(ll = ll, d.l = d.l))

}
```

```
#----- The function computes the maximum of lagrange -----
#---------- Input ----------
# data: The data frame that includes:
#       label (i): The label corresponds to the gene
#       X1-X100: The observed variable for the gene:
#       x_{i1}, x_{i2},..., x_{i 100}
#---------- Output ----------
# output:
# ll: the maximum of lagrange
#------------------------------------------------------------

lagrange.max.function = function(data){
N = length(data[, 1])
dim.x = dim(data)[2]

p.0 = sum(data[, 1]==0)/N
p.1 = sum(data[, 1]==1)/N

data.x0 = as.matrix(data[data[, 1]==0, -1])
data.x1 = as.matrix(data[data[, 1]==1, -1])

n0 = dim(data.x0)[1]
n1 = dim(data.x1)[1]

I.0 = matrix(1, ncol = 1, nrow = n0)
I.1 = matrix(1, ncol = 1, nrow = n1)

lambda.0 = t(data.x0)%*%(p.0*I.0)
lambda.1 = t(data.x1)%*%(p.1*I.1)
lambda.max = max(c(lambda.0, lambda.1))
return(lambda.max)
}


# setwd("/Users/xiyuanliu/Desktop/temp_cluster/CRF_real")
library(MASS)
rm(list = ls())
load("The_CRF_functions.RData")
####### Import the data #######
data.org = read.csv(file = "training_data.csv", header = T)
data = data.org[, -1]

##### Main Section #####
theta.new = matrix(rep(0, 108), nrow = 3, byrow = T)
theta.ini = theta.new
# Result = Proximal(theta.ini, data, 500)
# test(Result$theta.est, data, 0.85)
# new.data = p.y.x(theta.ini, data)

lambda.list = seq(500, 0.001, length.out = 40)
l = length(lambda.list)
theta.est.list = list()
start.time = c()
end.time = c()
Time.list = c()

i = 1
while(i<=l){
```

```
lambda = lambda.list[i]

start.time[i] = Sys.time()

Result = Proximal(theta.new, data, lambda)

end.time[i] = Sys.time()

theta.new = Result$theta.est

theta.est.list[[i]] = theta.new

Time.list[i] = Result$Time

i = i+1

}

time.data1 = data.frame(start = start.time,

end = end.time, elapsed = end.time-start.time)



# lambda.first = lambda.list[39]

# lambda.list = seq(lambda.first, 0.001, length.out = 40)

# l = length(lambda.list)

# theta.new = theta.est.list[[38]]

#

# theta.est.list = list()

# start.time = c()

# end.time = c()

# Time.list = c()

#

# i = 1

# while(i<=l){

#   lambda = lambda.list[i]

#   start.time[i] = Sys.time()

#   Result = Proximal(theta.new, data, lambda)

#   end.time[i] = Sys.time()

#   theta.new = Result$theta.est
#   theta.est.list[[i]] = theta.new
#   Time.list[i] = Result$Time
#   i = i+1
# }
#
# time.data2 = data.frame(start = start.time, end = end.time,
elapsed = end.time-start.time)

save.image("Real_data.RData")
```