Chapter 5

REGRESSION LEARNING

CONTENTS

5.1 Linear Regression

Linear Regression models the linear relationship between feature (explanatory) variables in $\mathbb{R}^D$ to response y let say in $\mathbb{R}$ for simplicity:

$$y = b + \boldsymbol{\omega}^\top \boldsymbol{x} + \varepsilon\,,$$

where ε is the idiosyncratic noise, usually assumed to follow a Gaussian distribution with a mean of 0 and a variance of σ^2, *i.e.* $\varepsilon \sim \mathcal{N}(0, \sigma^2)$; sometimes it is modelled as a martingale difference if one deal with time series data. Denote $f_{b,\boldsymbol{\omega}}(\boldsymbol{x}) = b + \boldsymbol{\omega}^\top \boldsymbol{x}$.

With a collection of N independent samples $\mathcal{S} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$, a well-fitted linear regression model means that the errors $\{\varepsilon_n\}_{n=1}^N$ between the real labels $\{y_n\}_{n=1}^N$ and the predicted label $\{f_{\boldsymbol{\omega},b}(\boldsymbol{x}_n)\}_{n=1}^N$ are minimized. However, these errors have both positive and negative values, simply summing up all the errors will mostly cancel out each other. Therefore, a chosen loss / cost (resp. satisfaction) function[1] $\mathcal{E}$, that turns all errors to have the same sign, is minimized with suitable choices of b and $\boldsymbol{\omega}$ instead. In practice, the two commonly adopted approaches in a regression problem are:

1. **Mean Squared Error** (MSE):
$$\mathcal{E} = \frac{1}{N}\sum_{n=1}^N \varepsilon_n^2 := \frac{1}{N}\sum_{n=1}^N \{y_n - f_{\boldsymbol{\omega},b}(\boldsymbol{x}_n)\}^2 . \tag{5.1.1}$$
Under MSE, we first augment b into $\boldsymbol{\omega}$, such that we define $\tilde{\boldsymbol{\omega}} := (b, \boldsymbol{\omega}^\top)^\top \in \mathbb{R}^{D+1}$, and then also add an additional column vector of 1's in $\mathbb{R}^N$ in front of the data matrix $\boldsymbol{X}$, that means to form $\tilde{\boldsymbol{X}} = (\mathbf{1}|\boldsymbol{X}) \in \mathbb{R}^{N\times(1+D)}$, then we aim to minimize the following MSE with respect to $\tilde{\boldsymbol{\omega}}$:
$$\frac{1}{N}\sum_{n=1}^N \varepsilon_n^2 = \frac{1}{N}(\boldsymbol{y} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\omega}})^\top(\boldsymbol{y} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\omega}}) .$$
Differentiating this MSE with respect to $\tilde{\boldsymbol{\omega}}$ gives:
$$\nabla_{\tilde{\boldsymbol{\omega}}}\left(\frac{1}{N}\sum_{n=1}^N \varepsilon_n^2\right) = \frac{1}{N}\nabla_{\tilde{\boldsymbol{\omega}}}(\boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{y}^\top\tilde{\boldsymbol{X}}\tilde{\boldsymbol{\omega}} + \tilde{\boldsymbol{\omega}}^\top\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{X}}\tilde{\boldsymbol{\omega}}) = \frac{1}{N}\left(-2(\boldsymbol{y}^\top\tilde{\boldsymbol{X}})^\top + 2(\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{X}})\tilde{\boldsymbol{\omega}}\right) .$$
Equating the equation above to zero yields the following well-known *Least Squares Estimate*:
$$\hat{\boldsymbol{\omega}} = (\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^\top\boldsymbol{y} ,$$
where we require the matrix $\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{X}}$ being invertible, *i.e.* $\boldsymbol{X}$ does not suffer from the problem of multicollinearity; or in other words, we apply a suitable feature engineering in Chapter **??** to identify out independent, or at least uncorrelated, factors.

2. **Mean Absolute Error** (MAE):
$$\mathcal{E} = \frac{1}{N}\sum_{n=1}^N |\varepsilon_n| := \frac{1}{N}\sum_{n=1}^N |y_n - f_{\boldsymbol{\omega},b}(\boldsymbol{x}_n)| . \tag{5.1.2}$$

Although MSE provides most tractible closed-form solution, MSE overweights the outliers, because outliers often incur large errors so that ε_n^2 is far more exaggerated in MSE than $|\varepsilon_n|$ in MAE. On the other hand, MAE as a function in $\boldsymbol{x}_n$ does not have a continuous derivative, that makes harder to find an explicit solution, fortunately, we can still use gradient descent method for finding the optimal solution, more details will be given in Chapter 6.

[1] A loss or cost function is a function that we aim to minimize, while a satisfaction function is a function that we aim to maximize; both these types of functions are called objective function of the optimization problem. In this book, we shall use the term loss function or cost function interchangeably.

5.2 Regularization

Regularization is a collective term that encompasses methods that force the learning algorithm to return a less complex model. In practice, that often leads to slightly higher bias yet it significantly reduces the variance implied by the model, which is the bias-variance tradeoff discussed in the Subsection 4.3.2. The two most widely used types of regularization are called $\mathcal{L}^1$ and $\mathcal{L}^2$ regularizations. The main idea is through an introduction of a penalty term to the complex model, so as to obtain a simplier model without sacrificing much of its prediction effectiveness.

For simplicity, we illustrate this concept via the linear regression with MSE in (5.1.1). The $\mathcal{L}^1$ and $\mathcal{L}^2$ regularized loss functions are given respectively by:

$$\text{(LASSO) } \mathcal{L}^1: \quad \min_{\boldsymbol{\omega},b}\left\{\frac{1}{N}\sum_{n=1}^{N}\{y_n - f_{\boldsymbol{\omega},b}(\boldsymbol{x}_n)\}^2 + \frac{C}{N}\sum_{d=1}^{D}|\omega_d|\right\}, \tag{5.2.1}$$

$$\text{(Ridge) } \mathcal{L}^2: \quad \min_{\boldsymbol{\omega},b}\left\{\frac{1}{N}\sum_{n=1}^{N}\{y_n - f_{\boldsymbol{\omega},b}(\boldsymbol{x}_n)\}^2 + \frac{C}{N}\sum_{d=1}^{D}\omega_d^2\right\}, \tag{5.2.2}$$

where LASSO stands for *Least Absolute Shrinkage and Selection Operator*. Note that the intercept constant term b is not included in the penalty term to avoid the problem of multicollinearity. In particular, Ridge regularization is a special case of the Tikhonov regularization with the Tikhonov matrix $\boldsymbol{\Gamma} = \sqrt{C/N}\boldsymbol{I}_D$:

$$\text{(Tikhonov) } \mathcal{L}^2_{\text{Tikh}}: \quad \min_{\boldsymbol{\omega},b}\left\{\frac{1}{N}\sum_{n=1}^{N}\{y_n - f_{\boldsymbol{\omega},b}(\boldsymbol{x}_n)\}^2 + \|\boldsymbol{\Gamma}\boldsymbol{\omega}\|_2^2\right\}.$$

Moreover, similar to the C in SVM (dealing with noise in Section **??**), this $C > 0$ is also a hyperparameter that controls the overall effect of regularization:

1. when C is small, the model is not much different from a standard non-regularized linear regression model;
2. when C takes a large value, the learning algorithm tends to set most ω_d to a very small value or zero to minimize the objective, the model trained would become relatively simple which may lead to a potential threat of underfitting.

It is an art to find a suitable value C such that it will not increase the bias too much yet it reduces the variance of the model to a reasonable level. To illustrate this idea, withouout loss of generality, consider the ridge Regression ($\mathcal{L}^2$) without the intercept term; in practice, we include b as $\omega^{(0)}$ also by augmenting a column vector with all entries 1 next to $\boldsymbol{X}$:

$$\mathcal{E} = \frac{1}{N}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\omega})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\omega}) + \frac{C}{N}\boldsymbol{\omega}^\top\boldsymbol{\omega}.$$

Then, the gradient of the loss function with respect to $\boldsymbol{\omega}$ is:

$$\nabla_{\boldsymbol{\omega}}\mathcal{E} = -\frac{1}{N}\left(2\boldsymbol{X}^\top\boldsymbol{y} + 2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\omega}\right) + \frac{2C}{N}\boldsymbol{\omega} = \boldsymbol{0} \qquad \Leftrightarrow \qquad \hat{\boldsymbol{\omega}} = (\boldsymbol{X}^\top\boldsymbol{X} + C\boldsymbol{I}_D)^{-1}\boldsymbol{X}^\top\boldsymbol{y}.$$

Heuristically, especially in one-dimensional case, if C increases, $\boldsymbol{\omega}$ tend to decrease, and so C can be regarded as a shrinkage factor; in general, since $\boldsymbol{X}^\top\boldsymbol{X}$ is symmetric, we have the spectual decomposition of $\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{H}\boldsymbol{\Lambda}\boldsymbol{H}^\top$, for some orthogonal matrix $\boldsymbol{H}$ that $\boldsymbol{H}\boldsymbol{H}^\top = \boldsymbol{H}^\top\boldsymbol{H} = \boldsymbol{I}_D$ and a diagonal matrix $\boldsymbol{\Lambda}$ of non-negative eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$; see Item 16 in Section 3.3. On the other hand, we can adopt the singular

decomposition on $\boldsymbol{X}^\top = \boldsymbol{H}\boldsymbol{\Sigma}\boldsymbol{V}$, where $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times N}$ is a rectangular block matrix of the form:

$$\boldsymbol{\Sigma} = \left(\begin{array}{c|c} \boldsymbol{\Lambda}^{1/2} & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{0} \end{array}\right),$$

and $\boldsymbol{V} \in \mathbb{R}^{N\times N}$ is the orthogonal matrix of $\boldsymbol{X}\boldsymbol{X}^\top$. Therefore, the least-square solution to $\boldsymbol{\omega}$ can be re-written as:

$$\begin{aligned}\hat{\boldsymbol{\omega}} &= (\boldsymbol{H}\boldsymbol{\Lambda}\boldsymbol{H}^\top + C\boldsymbol{I}_D)^{-1}\boldsymbol{H}\boldsymbol{\Sigma}\boldsymbol{V}\boldsymbol{y} = \Big(\boldsymbol{H}(\boldsymbol{\Lambda}+C\boldsymbol{I}_D)\boldsymbol{H}^\top\Big)^{-1}\boldsymbol{H}\boldsymbol{\Sigma}\boldsymbol{V}\boldsymbol{y} \\ &= (\boldsymbol{H}^\top)^{-1}(\boldsymbol{\Lambda}+C\boldsymbol{I}_D)^{-1}\boldsymbol{H}^{-1}\boldsymbol{H}\boldsymbol{\Sigma}\boldsymbol{V}\boldsymbol{y} = \boldsymbol{H}\boldsymbol{\Lambda}^{-1/2}\Big(\boldsymbol{\Lambda}^{1/2}(\boldsymbol{\Lambda}+C\boldsymbol{I}_D)^{-1}\boldsymbol{\Sigma}\Big)\boldsymbol{V}\boldsymbol{y} \\ &= \boldsymbol{H}\boldsymbol{\Lambda}^{-1/2}\left(\begin{array}{ccc|c} \frac{\lambda_1}{\lambda_1+C} & & & \boldsymbol{0} \\ & \ddots & & \vdots \\ & & \frac{\lambda_D}{\lambda_D+C} & \boldsymbol{0} \\ \hline \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \end{array}\right)\boldsymbol{V}\boldsymbol{y}\,.\end{aligned}$$

Therefore, with singular-values $\sqrt{\lambda_1} \geq \sqrt{\lambda_2} \geq \cdots \geq \sqrt{\lambda_D} \geq 0$, we have a shrinkage factor of $\lambda_d/(\lambda_d + C)$, for $d = 1, \cdots, D$, that:

1. if C is relatively smaller than λ_d, *i.e.* $C \ll \lambda_d$, then the regularization effect on the weight ω_d is small, since all shrinkage factors are close to 1;
2. if C is relatively larger than λ_d, *i.e.* $C \gg \lambda_d$, then the weight ω_d shrinks to zero; further discussion will be introduced in Subsection 5.2.1.

Remark 5.2.1. In general, instead of the usual Lagrangian multiplier C/N, we can pick C/N^α for some $\alpha > 0$ such that

$$\nabla_{\boldsymbol{\omega}}\mathcal{E} = -\frac{1}{N}\Big(2\boldsymbol{X}^\top\boldsymbol{y} + 2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\omega}\Big) + \frac{2C}{N^\alpha}\boldsymbol{\omega} = \boldsymbol{0} \qquad \Leftrightarrow \qquad \hat{\boldsymbol{\omega}} = \left(\boldsymbol{X}^\top\boldsymbol{X} + \frac{C}{N^{\alpha-1}}\boldsymbol{I}_D\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}\,.$$

Therefore,

1. if α is small, *i.e.* $\alpha < 1$, the penalty term increases exponentially in α. The effect of $\frac{C}{N^{\alpha-1}}\boldsymbol{I}_D$ to the parameter $\boldsymbol{\omega}$ dominates that of the training dataset $\boldsymbol{X}$, which ultimately makes the solution irrelevant to the original problem;
2. if α is huge, *i.e.* $\alpha > 1$, the penalty term decays exponentially in α, which ultimately reduces the regularizing effect.

Therefore, $\alpha = 1$, *i.e.* C/N, seems to be the most suitable choice.

On the other hand, consider the LASSO Regression ($\mathcal{L}^1$), again for simplicity, in the absence of the intercept term,

$$\mathcal{E} = \frac{1}{N}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\omega})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\omega}) + \frac{C}{N}\boldsymbol{1}_D^\top|\boldsymbol{\omega}|, \qquad \text{where} \quad \boldsymbol{1}_D = (1,\cdots,1)^\top, \quad \text{and} \quad |\boldsymbol{\omega}| = (|\omega_1|,\cdots,|\omega_D|)^\top\,.$$

In general, it is unlikely to solve for an explicit solution, we can still adopt the iterative optimization methods, such as the *Cyclic Coordinate Descent* algorithm (see Subsection 5.2.3), or Quadratic Programming (see Subsection 5.2.4 and Section 3.5).

5.2.1 Sparsity for coefficients under $\mathcal{L}^1$ Regularization

1. $\mathcal{L}^1$ **Regularization**: Produce a sparse model, which is a trained model with a huge number of parameters originally yet most of them are actually vanished. Hence, $\mathcal{L}^1$ is good at **feature selection**, by identifying which features are essential for prediction or not, and the trained model possesses a higher explainable nature;
2. $\mathcal{L}^2$ **Regularization**: Maximizes the performance of the model, and the underlying differentiability ensures the convenient use of various gradient descent method for parameter estimation.

In practice, for example in image pattern detection, $\mathcal{L}^1$ regularization helps to capture the edges and shapes, because $\mathcal{L}^1$ regularization shrinks the inessential parameters to zero; for example in deep learning, regularization helps to significantly reduce the variance of the deep learning model at the cost of a moderate increase in bias, which favourably produces a less overfitted model; see more discussions of **bias-variance tradeoff** in Subsection 4.3.2 and regularization in deep learning in Bengio et. al (2017).

To futher illustrate the sparsity of $\mathcal{L}^1$ regularization, we consider a collection of contour lines depicted in Figure 5.2.1 of a simple linear regression model with only two parameters, *i.e.* $\boldsymbol{\omega} = (\omega_1, \omega_2)^\top$.

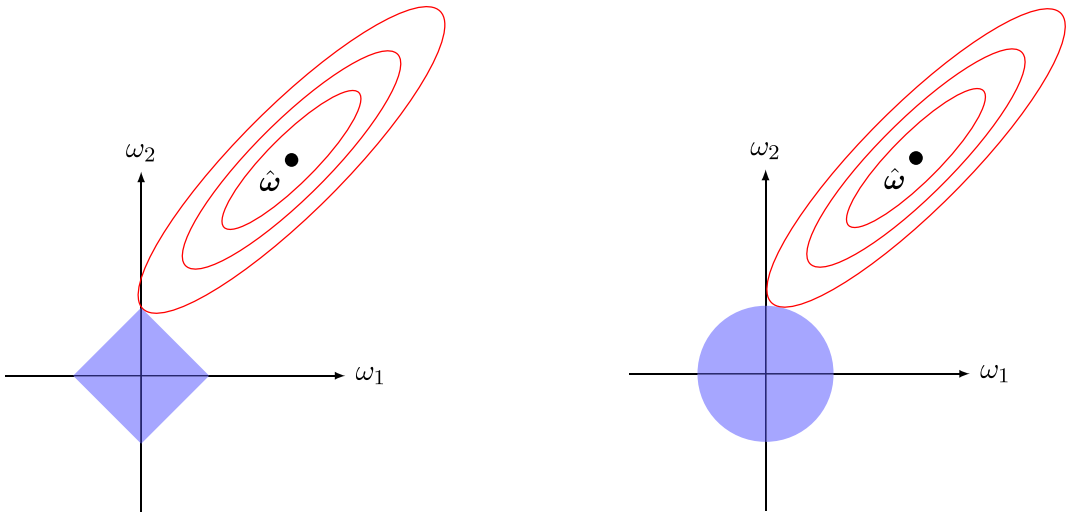

Figure 5.2.1: Left: $\mathcal{L}^1$ regularization; and Right: $\mathcal{L}^2$ regularization[2].

Here the red ellipses represent the contours of the MSE loss function, each of which contains points having the same MSE, while the black dot $\hat{\boldsymbol{\omega}}$ represents the non-constrainted Least Squares Estimate for $\boldsymbol{\omega}$, *i.e.* it has the smallest possible MSE value. On the other hand, the blue area represents the constrainted region for $\mathcal{L}^1$ (Left) regularization and $\mathcal{L}^2$ (Right) regularization, where

1. in $\mathcal{L}^1$ regularization, the contraint is $|\omega_1| + |\omega_2| \leq t$, so the $\mathcal{L}^1$ regularization problem becomes

$$\min_{\boldsymbol{\omega}, b} \frac{1}{N} \sum_{n=1}^{N} \{y_n - f_{\boldsymbol{\omega},b}(\boldsymbol{x}_n)\}^2 , \qquad \text{subject to} \quad \sum_{d=1}^{2} |\omega_d| < t \, . \tag{5.2.3}$$

   This constrainted problem is also known as the Ivanov regularization;

[2]Some datasets with MSE as loss function may produce a contour which shrinks the parameter ω_d to zero, even in $\mathcal{L}^2$ regularization; nevertheless, Figure 5.2.1 shows the generic situation.

2. in $\mathcal{L}^2$ regularization, the contraint is $\omega_1^2 + \omega_2^2 \le t$, so the $\mathcal{L}^2$ regularization problem becomes

$$\min_{\boldsymbol{\omega},b} \frac{1}{N}\sum_{n=1}^{N}\{y_n - f_{\boldsymbol{\omega},b}(\boldsymbol{x}_n)\}^2\,, \qquad \text{subject to} \quad \sum_{d=1}^{2}\omega_d^2 < t\,. \tag{5.2.4}$$

Under mild conditions with a suitable choice of the threshold t, the constrainted $\mathcal{L}^1$ (resp. $\mathcal{L}^2$) problem in (5.2.3) (resp. (5.2.4)) can be reformulated into an unconstrainted Lagrangian problem in maximizing (5.2.1) (resp. (5.2.2)) with respect to $C \ge 0$; also see Quadratic Programming in Section 3.5.

The optimal solution is then the contact point where the red ellipse contour curve first touches the blue shaded region, which has the smallest MSE value than those of other points lying in the interior. Note that

1. the blue area for $\mathcal{L}^1$ regularization is a diamond, the optimal solution is more likely to be one of the four corners of the diamond, unless the tangent of the red ellipse contour line evaluated at the optimal solution is equal to one of the four sides of the diamond. For the example depicted in Figure 5.2.1, we have $\omega_2 \gg 0$ while $\omega_1 \approx 0$ that shrinks ω_1 to zero.
2. the corresponding blue area for $\mathcal{L}^2$ regularization is a circle, the optimal solution is the contact point between the circle and the critical elliptical contour line, which seldom has ω_1 or ω_2 to be approximately zero.

5.2.2 Different Types of Regularizer

In general, a loss function with a $\mathcal{L}^q$ regularizer takes the following form:

$$\mathcal{L}^q: \quad \min_{\boldsymbol{\omega},b}\left(\frac{1}{N}\sum_{n=1}^{N}\{y_n - f_{\boldsymbol{\omega},b}(\boldsymbol{x}_n)\}^2 + \lambda\sum_{d=1}^{D}|\omega_d|^q\right).$$

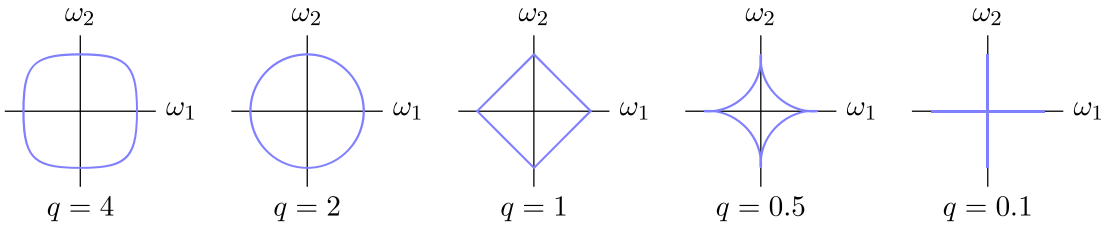

Figure 5.2.2: Constrainted contours with different values of q.

In practice, we can augment several $\mathcal{L}^q$ regularizers to our loss function. For instance, Zou and Hastie (2005) proposed the *elastic net* which combines $\mathcal{L}^1$ and $\mathcal{L}^2$ regularizations with the following penalty:

$$\frac{C}{N}\sum_{d=1}^{D}\Big\{(1-\alpha)|\omega_d| + \alpha\omega_d^2\Big\}, \qquad \text{for some} \quad \alpha \in (0,1)\,,$$

where this hyperparameter α controls the ratio of the effect of $\mathcal{L}^2$ to that of $\mathcal{L}^1$ to be adopted for the loss function.

5.2.3 Cyclic Coordinate Descent Algorithm

The *Cyclic Coordinate Descent Algorithm* for LASSO regularization was first introduced in the work of Fu (1998), but it was not commonly noticed until an emphasis in the work of Friedman et al. (2007):

We first fix the parameters $\tilde{\omega}_d$, for $d = 1, \cdots, D$ but not ℓ (same notation applies for the intercept term b by $\tilde{b}$) and ω_ℓ is the parameter to be updated. Consider the MSE loss function in (5.2.1) so that ω_ℓ is set free, but other $\tilde{\omega}_d$'s are fixed:

$$\begin{aligned}\mathcal{E}(\omega_\ell; \tilde{\boldsymbol{\omega}}) &:= \frac{1}{N}\sum_{n=1}^{N}\left(y_n - \tilde{b} - \sum_{d\neq\ell} x_n^{(d)}\tilde{\omega}_d - x_n^{(\ell)}\omega_\ell\right)^2 + \lambda\sum_{d\neq\ell}|\tilde{\omega}_d| + \lambda|\omega_\ell| \\ &= \frac{1}{N}\sum_{n=1}^{N}\left(r_n^{(\ell)} - x_n^{(\ell)}\omega_\ell\right)^2 + \lambda|\omega_\ell| + \lambda\sum_{d\neq\ell}|\tilde{\omega}_d| \\ &= \frac{1}{N}\sum_{n=1}^{N}\left(x_n^{(\ell)}\right)^2\omega_\ell^2 - 2\cdot\frac{1}{N}\sum_{n=1}^{N} r_n^{(\ell)}x_n^{(\ell)}\omega_\ell + \lambda|\omega_\ell| + \gamma =: \alpha\omega_\ell^2 - 2\beta\omega_\ell + \lambda|\omega_\ell| + \gamma\,,\end{aligned}$$

where γ, and $r_n^{(\ell)}$ does not depend on ω_ℓ,

$$r_n^{(\ell)} := y_n - \tilde{b} - \sum_{d\neq\ell} x_n^{(d)}\tilde{\omega}_d\,, \qquad \beta := \frac{1}{N}\sum_{n=1}^{N} r_n^{(\ell)}x_n^{(\ell)}\,; \tag{5.2.5}$$

$$\alpha := \frac{1}{N}\sum_{n=1}^{N}\left(x_n^{(\ell)}\right)^2 > 0\,, \qquad \gamma := \frac{1}{N}\sum_{n=1}^{N}\left(r_n^{(\ell)}\right)^2 + \lambda\sum_{d\neq\ell}|\tilde{\omega}_d|\,. \tag{5.2.6}$$

We aim to find $\hat{\omega}_\ell$ that minimizes $\mathcal{E}(\omega_\ell; \tilde{\boldsymbol{\omega}})$:

$$\arg\min_{\omega_\ell}\left(\alpha\omega_\ell^2 - 2\beta\omega_\ell + \lambda|\omega_\ell| + \gamma\right) = \arg\min_{\omega_\ell}\left(\omega_\ell^2 - \frac{2\beta\omega_\ell - \lambda|\omega_\ell|}{\alpha}\right) \qquad (\because \alpha > 0)\,.$$

Due to the non-differentiable modules function, we then consider the following cases:

1. If $\omega_\ell > 0$, then
$$\hat{\omega}_\ell = \arg\min_{\omega_\ell}\left(\omega_\ell^2 - \frac{2\beta\omega_\ell - \lambda\omega_\ell}{\alpha}\right) = \arg\min_{\omega_\ell}\left(\omega_\ell - \frac{2\beta-\lambda}{2\alpha}\right)^2 = \frac{2\beta-\lambda}{2\alpha}\,.$$
which is required to be greater than zero given that $2\beta - \lambda > 0$ or $\beta > \lambda/2$ as $\alpha > 0$.

2. If $\omega_\ell < 0$, then
$$\hat{\omega}_\ell = \arg\min_{\omega_\ell}\left(\omega_\ell^2 - \frac{2\beta\omega_\ell + \lambda\omega_\ell}{\alpha}\right) = \arg\min_{\omega_\ell}\left(\omega_\ell - \frac{2\beta+\lambda}{2\alpha}\right)^2 = \frac{2\beta+\lambda}{2\alpha}\,,$$
which is less than zero if $2\beta + \lambda < 0$ or $\beta < -\lambda/2$ as $\alpha > 0$.

3. In light of cases 1 and 2, we still miss out the remaining possibility of $\beta \in [-\lambda/2, \lambda/2]$, we then consider the loss function $\mathcal{E}(\omega_\ell; \tilde{\boldsymbol{\omega}})$ at 3 different possible ranges of values of ω_ℓ:

   (a) $(\omega_\ell = 0)$:
   $$\mathcal{E}(\omega_\ell; \tilde{\boldsymbol{\omega}}) = \alpha(0)^2 - 2\beta\times 0 + \lambda|0| + \gamma = \gamma > 0\,.$$

   (b) $(\omega_\ell > 0)$:
   $$\mathcal{E}(\omega_\ell; \tilde{\boldsymbol{\omega}}) = \underbrace{\alpha\left(\omega_\ell - \frac{2\beta-\lambda}{2\alpha}\right)^2 + \gamma'}_{>0} + \gamma > \gamma\,.$$

   (c) $(\omega_\ell < 0)$:
   $$\mathcal{E}(\omega_\ell; \tilde{\boldsymbol{\omega}}) = \underbrace{\alpha\left(\omega_\ell - \frac{2\beta+\lambda}{2\alpha}\right)^2 + \gamma''}_{>0} + \gamma > \gamma\,.$$

   In summary, we conclude that $\omega_\ell = 0$ is the minimizer under this case.

Therefore, we have the following possible update for ω_ℓ:

$$\omega_\ell = \begin{cases} \dfrac{2\beta - \lambda}{2\alpha}, & \text{if } \beta > \dfrac{\lambda}{2} > 0\,; \\ 0, & \text{if } -\dfrac{\lambda}{2} \le \beta \le \dfrac{\lambda}{2}\,; \\ \dfrac{2\beta + \lambda}{2\alpha}, & \text{if } \beta < -\dfrac{\lambda}{2} < 0\,. \end{cases} ,$$

where β is defined in (5.2.5) and α is defined in (5.2.6).

On the other hand, consider the update for b such that all $\tilde{\omega}_d$, for $d = 1, \cdots, D$, are given; consider the MSE loss function in (5.2.1) so that b is set free but not all $\tilde{\omega}_d$:

$$\mathcal{E}(b; \tilde{\boldsymbol{\omega}}) = \frac{1}{N}\sum_{n=1}^{N}\Big(y_n - \tilde{\boldsymbol{\omega}}^\top \boldsymbol{x}_n - b\Big)^2 + \lambda \mathbf{1}_D^\top |\tilde{\boldsymbol{\omega}}|\,,$$

where $\mathbf{1}_D = (1, \cdots, 1)^\top \in \mathbb{R}^D$. Then the partial derivative of $\mathcal{E}(b; \tilde{\boldsymbol{\omega}})$ with respect to b gives:

$$\frac{\partial \mathcal{E}}{\partial b}(b; \tilde{\boldsymbol{\omega}}) = -2 \cdot \frac{1}{N}\sum_{n=1}^{N}(y_n - \boldsymbol{x}_n\tilde{\boldsymbol{\omega}} - b) = 0 \qquad \Leftrightarrow \qquad b = \frac{1}{N}\sum_{n=1}^{N} r_n^{(0)}\,,$$

where $r_n^{(0)} = y_n - \tilde{\boldsymbol{\omega}}^\top \boldsymbol{x}_n$.

So, the following Algorithm 5.3 proposed by Fu (1998) can be used to update for LASSO regularization:

Algorithm 5.3 Cyclic Coordinate Descent Algorithm

1: Initialize the parameters $\boldsymbol{\theta}^{(0)} = (b^{(0)}, \boldsymbol{\omega}^{(0)})$ with the least squares estimate for the unconstrainted problem;
2: Fix the parameters $\omega_1^{(t)}, \omega_2^{(t)}, \cdots, \omega_D^{(t)}$ and find $b^{(t+1)} = \arg\min_b \mathcal{E}^{(t)}(b) := \arg\min_b \mathcal{E}(b; \boldsymbol{\omega}^{(t)})$;
3: **for** $d = 1, \cdots, D$ **do**
4: Fix the parameters $b^{(t+1)}, \omega_1^{(t+1)}, \cdots, \omega_{d-1}^{(t+1)}, \omega_{d+1}^{(t)}, \cdots, \omega_D^{(t)}$ and find $\omega_d^{(t+1)} = \arg\min_{\omega_d} \mathcal{E}^{(t)}(\omega_d)$, where $\mathcal{E}^{(t)}(\omega_d) := \mathcal{E}(\omega_d; b^{(t+1)}, \omega_1^{(t+1)}, \cdots, \omega_{d-1}^{(t+1)}, \omega_{d+1}^{(t)}, \cdots, \omega_D^{(t)})$;
5: **end for**
6: Repeat steps 2 to 5 for $t = 0, 1, 2, \cdots$ until $\|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\|_2 \le \delta$, for some pre-assigned threshold $\delta > 0$.

5.2.4 Solving LASSO via Quadratic Programming

Recall the LASSO problem in (5.2.3):

$$\min_{\boldsymbol{\omega}} \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\omega})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\omega})\,, \qquad \text{subject to} \quad \sum_{d=1}^{D}|\omega_d| < t\,,$$

where $\boldsymbol{\omega} = (b, \omega_1, \cdots, \omega_D)^\top \in \mathbb{R}^{1+D}$. After some simplifications, the problem becomes:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\boldsymbol{\omega}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{\omega} - \boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{\omega}\,, \\ &\text{subject to} && \mathbf{1}_D^\top |\boldsymbol{\omega}| \le t\,, \end{aligned} \tag{5.2.7}$$

where $\mathbf{1}_D = (1, \cdots, 1)^\top \in \mathbb{R}^D$. The problem is then a Quadratic Programming problem under nonconvex domain, we therefore mimic the strategy in Example 3.5.2 by introducing $\boldsymbol{\omega}^+$ and $\boldsymbol{\omega}^-$:

$$\omega_d^+ = \frac{|\omega_d| + \omega_d}{2} \ge 0 \qquad \text{and} \qquad \omega_d^- = \frac{|\omega_d| - \omega_d}{2} \ge 0\,,$$

such that $\omega_d^+ \cdot \omega_d^- = 0$. Therefore, in matrix form, we obtain

$$\boldsymbol{\omega} := \boldsymbol{\omega}^+ - \boldsymbol{\omega}^- \quad \text{and} \quad |\boldsymbol{\omega}| := \boldsymbol{\omega}^+ + \boldsymbol{\omega}^-, \quad \boldsymbol{\omega}^+ \geq \mathbf{0}, \quad \boldsymbol{\omega}^- \geq \mathbf{0}\,, \quad \boldsymbol{\omega}^+ \odot \boldsymbol{\omega}^- = \mathbf{0}\,, \quad \text{for} \quad d = 1, \cdots, D\,.$$

The problem in (5.2.7) becomes

$$\begin{aligned} &\text{minimize} && \frac{1}{2}(\boldsymbol{\omega}^+ - \boldsymbol{\omega}^-)^\top \boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{\omega}^+ - \boldsymbol{\omega}^-) - \boldsymbol{y}^\top \boldsymbol{X}(\boldsymbol{\omega}^+ - \boldsymbol{\omega}^-)\,, \\ &\text{subject to} && \text{i)} \quad \mathbf{1}_D^\top(\boldsymbol{\omega}^+ + \boldsymbol{\omega}^-) \leq t\,; \\ &&& \text{ii)} \quad \boldsymbol{\omega}^+ \geq \mathbf{0}\,; \\ &&& \text{iii)} \quad \boldsymbol{\omega}^- \geq \mathbf{0}\,; \\ &&& \text{iv)} \quad \boldsymbol{\omega}^+ \odot \boldsymbol{\omega}^- = \mathbf{0}\,. \end{aligned} \tag{5.2.8}$$

Or equivalently,

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\begin{pmatrix} \boldsymbol{\omega}^+ \\ \boldsymbol{\omega}^- \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{X}^\top\boldsymbol{X} & -\boldsymbol{X}^\top\boldsymbol{X} \\ -\boldsymbol{X}^\top\boldsymbol{X} & \boldsymbol{X}^\top\boldsymbol{X} \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega}^+ \\ \boldsymbol{\omega}^- \end{pmatrix} - \begin{pmatrix} \boldsymbol{y}^\top\boldsymbol{X}, & -\boldsymbol{y}^\top\boldsymbol{X} \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega}^+ \\ \boldsymbol{\omega}^- \end{pmatrix}, \\ &\text{subject to} && \text{i)} \quad \begin{pmatrix} \mathbf{1}_D^\top, & \mathbf{1}_D^\top \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega}^+ \\ \boldsymbol{\omega}^- \end{pmatrix} \leq t\,; \\ &&& \text{ii)} \quad \boldsymbol{I}_{D+D} \begin{pmatrix} \boldsymbol{\omega}^+ \\ \boldsymbol{\omega}^- \end{pmatrix} \geq \mathbf{0}\,; \\ &&& \text{iii)} \quad \boldsymbol{\omega}^+ \odot \boldsymbol{\omega}^- = \mathbf{0}\,. \end{aligned} \tag{5.2.9}$$

Generally $\boldsymbol{P} = \boldsymbol{X}^\top\boldsymbol{X}$ is often positive-definite in the absence of multicollinearity, however, the problem in (5.2.9) is still not in the standard form of a Quadratic Programming problem. As mentioned in Example 3.5.2, if $\boldsymbol{\omega}$ is of small dimension, then we shall construct 2^D number of Quadratic Programming problems by setting either $\omega_d^+ = 0$ or $\omega_d^- = 0$, for all $d = 1, \cdots, D$.

5.5 Logistic Regression

Logistic regression is a supervised classification (mostly) learner. However, logistic regression is fundamentally different from multiple linear regression as the latter specifies a prediction in value, while the former gives the chance of taking a certain label. In this section, we shall mainly focus on binary classification and its extension to multi-label classification is straightforward.

5.5.1 Introduction with Binary Response

The ordinary linear regression has its limitations and can be easily abused; indeed, recall that ordinary linear regression has the following assumptions:

1. The regression function of y is a linear function in the independent variables $\boldsymbol{x}$'s, *i.e.*
$$\mathbb{E}(y|\boldsymbol{x}) = \beta_0 + \beta_1 x^{(1)} + \cdots + \beta_D x^{(D)}\,;$$
2. The idiosyncratic noises ε_i's are at least uncorrelated, if not totally iid, with a common distribution with the mean zero and finite variance $\sigma^2 < \infty$.

Usually, residuals plots are used to check for these assumptions. A Q-Q normal plot of the residuals is used to further check the normality assumption on the idiosyncratic noises, normally closer the plot to the main diagonal, more trust on their normality can be put. The plot of residuals versus fitted values of the observation is used to check the linearity and constant error variance assumptions, for instance, a uniform horizontal band of the plot means that the linear model is well-fitted with a homogeneous variance for the noise term. The plot of residuals versus lag (residuals) is used to check uncorrelation or the independence of noises, again a uniform horizontal band without any pattern stands for the absense of correlation. Let us look at the following financial example which demonstrates how the linear regression can be abused.

The data file "*fin-ratio.csv*" contains financial ratios of 680 securities listed in the main board of Hong Kong Stock Exchange in 2002, in which there are six financial variables, namely, **Earning Yield** (EY), **Cash Flow to Price** (CFTP), **logarithm of Market Value** (ln_MV), **Dividend Yield** (DY), **Book to Market Equity** (BTME), **Debt to Equity Ratio** (DTE). These financial variables are publicly available information, for instance, found in Yahoo Finance (`https://finance.yahoo.com/`). Among these companies, there are 32 Blue Chips which are the Hang Seng Index Constituent Stocks. The last column HSI is a binary variable indicating whether the stock is a Blue Chip or not. We now want to establish any relationship between HSI and these mentioned six financial variables. One can run an ordinary least square regression of HSI against these six financial variables by using **R**'s built-in function `lm()` (Linear Model), see the codes and output in the following Programme 5.5.1:

```
> d <- read.csv("fin-ratio.csv")
> names(d)
[1] "EY"    "CFTP"  "ln_MV" "DY"    "BTME"  "DTE"   "HSI"
> summary(lm(HSI~EY+CFTP+ln_MV+DY+BTME+DTE,data=d))
# linear model

```

```
Call:
lm(formula = d$HSI ~ d$EY + d$CFTP + d$ln_MV + d$DY + d$BTME + d$DTE)

Residuals:
     Min       1Q   Median       3Q      Max
-0.32104 -0.08546 -0.01672  0.05592  0.73866
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.4591209  0.0268310 -17.112  < 2e-16 ***
d$EY        -0.0017172  0.0016181  -1.061  0.28896
d$CFTP      -0.0103792  0.0037321  -2.781  0.00557 **
d$ln_MV      0.0810286  0.0040887  19.818  < 2e-16 ***
d$DY        -0.0027336  0.0017826  -1.534  0.12561
d$BTME       0.0004798  0.0007938   0.604  0.54575
d$DTE        0.0010610  0.0018035   0.588  0.55655
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 0.1689 on 673 degrees of freedom
Multiple R-Squared: 0.3708,     Adjusted R-squared: 0.3652
F-statistic: 66.09 on 6 and 673 DF,  p-value: < 2.2e-16
```

Programme 5.5.1: Codes and output of the ordinary least square regression of HSI against the six financial variables

From the output, the p-value of `d$DTE` is the largest and it is far greater than the acceptance threshold (say 0.1), this means the coefficient of `DTE` is not significantly different from zero. Therefore, we should exclude it in our regression. We then continue to fit the regression using only the remaining five independent financial variables:

```
> summary(lm(HSI~EY+CFTP+ln_MV+DY+BTME,data=d))
```

,

which we further found that the p-value of `d$BTME` was the largest far greater than 0.1, and this suggests that `d$BTME` should be excluded. We keep on excluding the variable with the largest p-value one by one until all the p-values are small and less than 0.1; this procedure is one of the common model selection methods in regression known as the **backward elimination**. Finally, we arrive at the following model, see Programme 5.5.2:

```
> summary(lm(HSI~CFTP+ln_MV,data=d))
Call:
lm(formula = d$HSI ~ d$CFTP + d$ln_MV)
Residuals:
 Min       1Q   Median       3Q      Max
-0.32409 -0.08559 -0.01729  0.05688  0.73488

```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.454781   0.026284 -17.303  < 2e-16 ***
d$CFTP      -0.012026   0.003475  -3.461 0.000573 ***
d$ln_MV      0.079630   0.004032  19.751  < 2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 0.1689 on 677 degrees of freedom
Multiple R-Squared: 0.3666,     Adjusted R-squared: 0.3648
F-statistic: 195.9 on 2 and 677 DF,  p-value: < 2.2e-16
```

Programme 5.5.2: The final regression model after the backward elimination

Although we arrive at the least square regression model:

$$\text{HSI} = -0.454781 - 0.01026(\text{CFTP}) + 0.07963(\text{ln\_MV})\,,$$

it does not mean that this model is useful in telling us whether the company is HSI or not. Firstly, the fitted values of HSI can be any real number rather than 0 and 1. Secondly, we need to look at the residuals of this regression model to check for the validity of the assumptions;

```
> reg <- lm(HSI~CFTP+ln_MV,data=d)      # save regression results
> names(reg)                            # display items in reg
[1] "coefficients"  "residuals"     "effects"       "rank"
[5] "fitted.values" "assign"        "qr"            "df.residual"
[9] "xlevels"       "call"          "terms"         "model"
> par(mfrow=c(3,2))                         # set a 3x2 graphical frame
> hist(reg$fitted.values)                   # fitted values histogram
> hist(reg$residuals)                       # residuals histogram
> plot(reg$fitted.values,reg$residuals) # residuals vs fitted values
> qqnorm(reg$residuals)                     # qq-normal plot of residuals
> qqline(reg$residuals)                     # add reference line
> res <- as.ts(reg$residuals)               # change res to time series
> plot(res,lag(res))                        # residuals vs lag(residuals)
> plot(reg$residuals)                       # residuals vs index number
```

These generate the plots below:

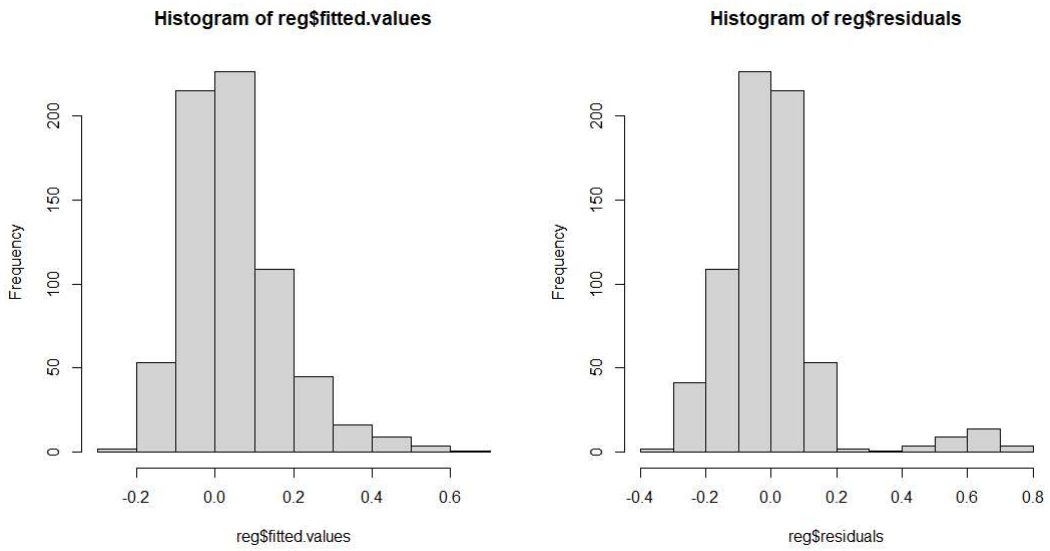

Figure 5.5.1: Histograms of the fitted values (Left) and the residuals (Right).

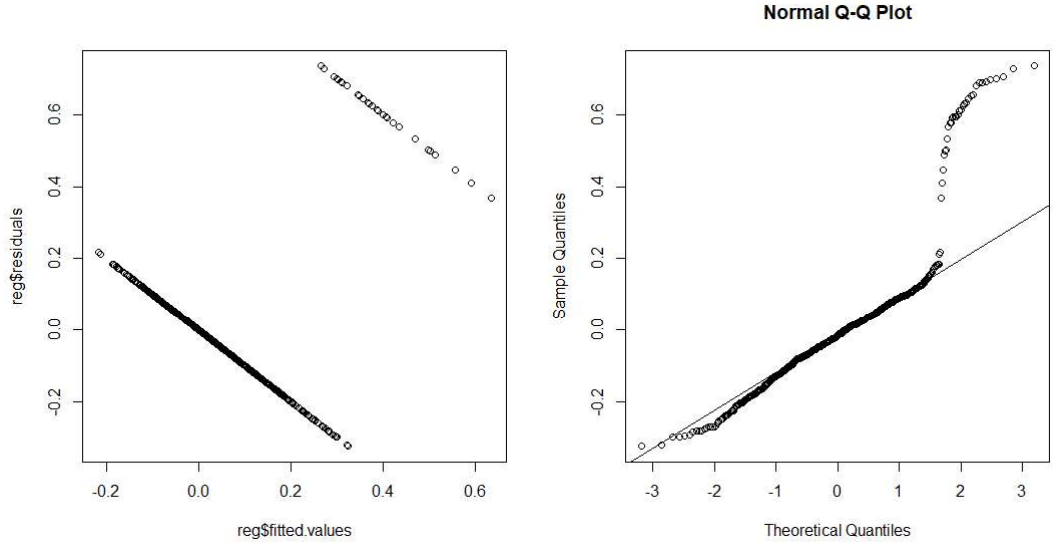

Figure 5.5.2: Plots of residuals vs fitted values (Left) and the normal Q-Q plot (Right).

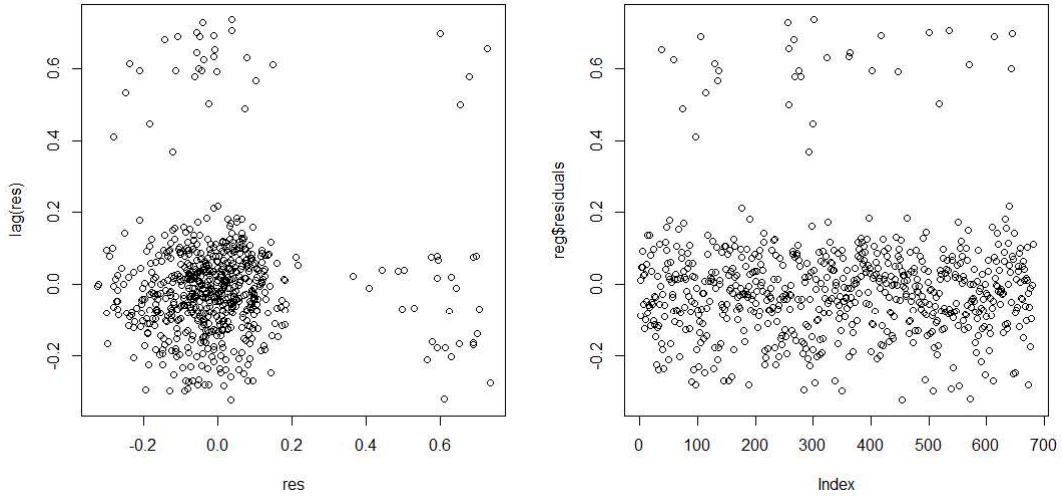

Figure 5.5.3: Residual plots against lag (Left) and against index number (Right).

If the normal assumption is correct, the normal Q-Q plot should be close to a straight line, but instead it is fairly fitted with an indication of heavy tail feature, the histograms of residuals on the right of Figure 5.5.1 should be normally distribted, and the serial residual plots in Figure 5.5.3 should be randomly distributed as a cluster without any patern. Also see Figure 5.5.2, while HSI takes only binary values of 0 or 1, the fitted values of HSI are distributed over a widespread interval. Also, the residuals, while dominantly distribute around 0, has a secondary peak with a large positive value around 0.6, see the right plot of Figure 5.5.3, and this secondary peak of large positive residual values is likely associated with those data having HSI = 1. In addition for the output of Programme 5.5.2, the linear regression assumptions are seriously violated since the adjusted R^2 is 0.3648, which is not surprising as the response y (=HSI) is binary.

So far, we have seen the shortcomings of using a simple linear regression model in dealing with binary response variables; meanwhile, the *logistic regression* can be invoked to model the binary response variables. For logistic regression, we first define a parameter

$$\pi_n := \mathbb{P}(y_n = 1|\boldsymbol{x}_n)\,, \tag{5.5.1}$$

which is interpreted as the probability of $y_n = 1$ (probability of "success") given $\boldsymbol{x}_n$. A crucial assumption for the logistic regression model is:

$$\ln\left(\frac{\pi_n}{1-\pi_n}\right) = \beta_0 + \beta_1 x_n^{(1)} + \cdots + \beta_D x_n^{(D)} = \boldsymbol{x}_n^\top \boldsymbol{\beta}\,, \tag{5.5.2}$$

where $\boldsymbol{x}_n = (1, x_n^{(1)}, \cdots, x_n^{(D)}) \in \mathbb{R}^{D+1}$ and $\boldsymbol{\beta} = (\beta_0, \cdots, \beta_D)$. The left hand side of (5.5.2) is the **log-odd ratio** of probability of success, and here we assume that the log-odd ratio of success probability is an affine function in $x_n^{(\cdot)}$'s. The entire (5.5.2) is called the Bernoulli **link function**, the link function is used in the generalized linear model (GLM) setting to relate the linear predictor $\boldsymbol{x}_n^\top\boldsymbol{\beta}$ to the mean of the distribution function, now a Bernoulli one, $\mu = \pi_n$. Equivalently, (5.5.2) can be rewritten as

$$\pi_n = \frac{\exp(\boldsymbol{x}_n^\top\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_n^\top\boldsymbol{\beta})}\,, \tag{5.5.3}$$

which is called the **expit** transformation of $\boldsymbol{x}_n^\top\boldsymbol{\beta}$, which is also known as the **sigmoid** function in ANN. Obviously, π_n always lies between 0 and 1 and this is consistent with the interpretation as a probability of success. Also, (5.5.3) satisfies

$$\nabla_{\boldsymbol{\beta}}\pi_n = \pi_n(1-\pi_n)\boldsymbol{x}_n \in \mathbb{R}^{D+1}\,. \tag{5.5.4}$$

The likelihood function for the data $(\boldsymbol{x}_n, y_n)$, for $n = 1, \cdots, N$, is

$$L(\boldsymbol{\beta}) = \prod_{n=1}^{N}\left(\pi_n^{y_n}(1-\pi_n)^{(1-y_n)}\right), \tag{5.5.5}$$

or the log-likelihood function is

$$\ln L(\boldsymbol{\beta}) = \sum_{n=1}^{N}\Big(y_n \ln\pi_n + (1-y_n)\ln(1-\pi_n)\Big)\,. \tag{5.5.6}$$

Once we obtain the maximum likelihood estimate for $\boldsymbol{\beta}$ by maximizing (5.5.6), we can then compute the predicted probability of success conditional on $\boldsymbol{x}_n$ using (5.5.3); indeed, the gradient of $\ln L(\boldsymbol{\beta})$ in $\boldsymbol{\beta}$ is given

by

$$\begin{aligned}\nabla_{\boldsymbol{\beta}} \ln L(\boldsymbol{\beta}) &= \sum_{n=1}^{N}\left(y_n\frac{\nabla_{\boldsymbol{\beta}}\pi_n}{\pi_n} - (1-y_n)\frac{\nabla_{\boldsymbol{\beta}}\pi_n}{1-\pi_n}\right)\\ &= \sum_{n=1}^{N}\left(y_n\frac{\pi_n(1-\pi_n)}{\pi_n}\boldsymbol{x}_n - (1-y_n)\frac{\pi_n(1-\pi_n)}{1-\pi_n}\boldsymbol{x}_n\right)\\ &= \sum_{n=1}^{N}\left(y_n(1-\pi_n)\boldsymbol{x}_n - (1-y_n)\pi_n\boldsymbol{x}_n\right) = \sum_{n=1}^{N}(y_n-\pi_n)\boldsymbol{x}_n\,.\end{aligned}$$

Generally, it is difficult to look for the MLE by solving for $\nabla_{\boldsymbol{\beta}} \ln L(\boldsymbol{\beta}) = 0$ directly due to the nonlinearity of the expit function in $\boldsymbol{\beta}$. Therefore, Gauss-Newton algorithm is adopted in solving this MLE, which will be introduced in the next subsection.

We first demonstrate the effectiveness of this logistic regression through the HSI example as shown below. To this end, **R** has a built-in `glm()` function (standing for generalized linear model) to perform logistic regression. The output format is quite similar to `lm()`. The argument `family=binomial` indicates that a binomial (Bernoulli) **link function** is used in the logistic regression programmed by `glm()`. Referring to Programme 5.5.3, the MLEs of the coefficients are extremely large, this may be caused by many outliers existing in the dataset. We shall further investigate the MLEs of the coefficients after removing the outliers in Subsection 5.5.3. Anyhow, let us keep these logistic regression output for the moment and see how the model performs.

```
> summary(glm(HSI~EY+CFTP+ln_MV+DY+BTME+DTE,data=d,family=binomial))
Call:
glm(formula = d$HSI ~ d$EY + d$CFTP + d$ln_MV + d$DY + d$BTME + d$DTE,
family = binomial)
Deviance Residuals:
       Min            1Q          Median            3Q          Max
-8.490e+00   -2.107e-08     -2.107e-08   -2.107e-08  8.490e+00
Coefficients:
               Estimate Std. Error    z value Pr(>|z|)
(Intercept) -4.121e+15  1.066e+07 -386410689   <2e-16 ***
d$EY         1.516e+13  6.431e+05   23570628   <2e-16 ***
d$CFTP      -6.364e+13  1.483e+06  -42902735   <2e-16 ***
d$ln_MV      4.945e+14  1.625e+06  304287297   <2e-16 ***
d$DY        -1.144e+14  7.085e+05 -161536188   <2e-16 ***
d$BTME      -7.907e+12  3.155e+05  -25063060   <2e-16 ***
d$DTE        8.744e+12  7.168e+05   12198713   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

(Dispersion parameter for binomial family taken to be 1)

```

```
Null deviance:  258.08  on 679  degrees of freedom
Residual deviance: 1153.40  on 673  degrees of freedom
AIC: 1167.4
```

Programme 5.5.3: Fitted coefficients of the first primitive logistic regression for HSI using the 2002 data

Here since all the p-values are small, we do not need to eliminate any variables.

```
> lreg <- glm(HSI~EY+CFTP+ln_MV+DY+BTME+DTE,data=d,family=binomial)
> names(lreg)           # display items in lreg
[1] "coefficients"      "residuals"         "fitted.values"
[4] "effects"           "R"                 "rank"
[7] "qr"                "family"           "linear.predictors"
[10] "deviance"          "aic"               "null.deviance"
[13] "iter"              "weights"           "prior.weights"
[16] "df.residual"       "df.null"           "y"
[19] "converged"         "boundary"          "model"
[22] "call"              "formula"           "terms"
[25] "data"              "offset"            "control"
[28] "method"            "contrasts"         "xlevels"

> pr <- (lreg$fitted.values>0.5)            # pr=True if fitted >0.
> table(d$HSI, pr)                       # tabulation of pr & HSI
 pr
  FALSE TRUE
0   634   14
1     2   30
```

Programme 5.5.4: Output and prediction of the first primitive logistic regression for HSI using the 2002 data

Referring to Programme 5.5.4, the `glm()` output is now saved in `lreg`, which contains many items as shown in the output of `names(lreg)`. An important one is the `fitted.values`, which is the estimated success probability in accordance with π_n in (5.5.3) for each of these 680 stocks in 2002. All these numbers lie between 0 and 1 and represent the respective probability of HSI $= 1$ for each stock. Since our MLEs of the coefficients are extremely large, the fitted values are either very close to 0 or 1. Furthermore, we can assign a value of `pr=True` if the fitted value is greater than 0.5 let say, or `pr=False` otherwise. This can be interpreted as the stock being predicted as an HSI constituent stock or not. Finally, we can produce a confusion matrix of pr versus HSI to see how well this logistic model predicts. From the output, there are $634 + 30 = 664$ correct classifications, and $14 + 2 = 16$ misclassifications. There are 2 stocks with HSI $= 1$ being misclassified as `pr=False` while there are 14 stocks with HSI $= 0$ being misclassified as `pr=True`. The correct classification rate, *i.e.* the accuracy, is then given by $664/680 = 97.65\%$.

Next, we obtain the prediction `y_pred` by fitting the HSI dataset to the `lreg` model. Then, unlike in Example 4.4.2, we plot the ROC with the `ROCR` library.

```
> library(ROCR)
> y_pred <- predict(lreg, d)
> pred <- prediction(predictions=y_pred, labels=d$HSI)
> perf <- performance(pred, measure="tpr", x.measure="fpr")
> plot(perf, col="blue", lwd=3)
> segments(0, 0, 1, 1, col="red", lwd=3)
```

Programme 5.5.5: Plotting ROC of HSI dataset using the `ROCR` library in **R**

Here both functions `prediction()` and `performance()` return an object defined by the `ROCR` library. In particular, for the `prediction()` function, the parameter `predictions` only accepts continuous attributes, and the parameter `labels` is the true label; meanwhile for the `performance()` function, the arguments `measure="tpr"` indicates that TPR is used in the y-axis and `x.measure="fpr"` indicates that FPR is used in the x-axis. Finally, we plot the ROC curve with a blue line and add a diagonal red line to represent the ROC curve for a random classifier.

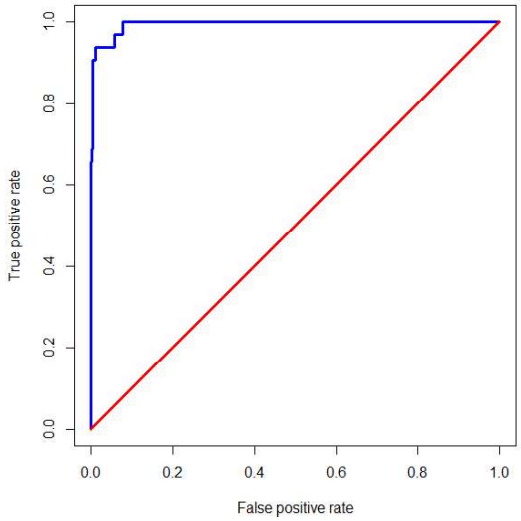

Figure 5.5.4: ROC curve for logistic regression with HSI dataset.

Apart from logistic regression, another commonly used method to tackle binary responses is via perceptron (See Section 7.1), which is essentially logistic regression but with a step function in place of the expit transform, and we here provide a generic comparison between these two methods under some representative cases. Particularly, referring to Figure 5.5.5, the datasets illustrate the classifications of linearly separable (two on the top) and inseparable (another pair at the bottom) ones, using perceptron (represented by the broken line) and logistic regression (represented by the solid line) respectively[6]. The advantage of logistic regression against perceptron is clear in classifying linearly separable data; indeed, though both methods correctly classify the data, the separating line generated by logistic regression tends to separate the two groups of data more evenly, and the one offered by perceptron is slightly off-center, especially shown in the

[6]In both algorithms using IRLS (See Subsection 5.5.2), 30 iterations are performed with a prior termination condition if, for instance in logistic regression, (5.5.9) is satisfied for $\varepsilon = 0.01\%$.

top left plot in Figre 5.5.5. This is due to the fact that the stochastic gradient descent used in perceptron ceases to update the slope parameter β_1 once all the data are correctly classified, while IRLS in addition picks the very one which maximizes the likelihood.

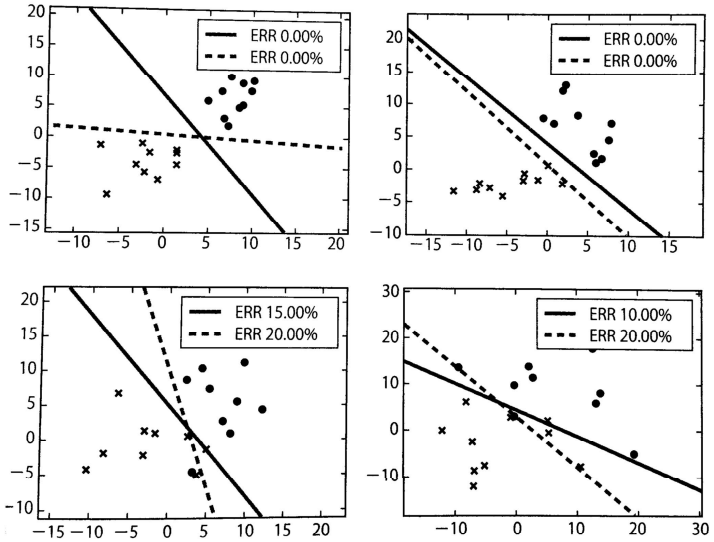

Figure 5.5.5: Comparsions for perceptron (broken line) and IRLS for logistic regression (solid line).

5.5.2 Gauss-Newton Algorithm for Logistic Regression: Iteratively Reweighted Least Square (IRLS)

In this section, we shall introduce the Iteratively Reweighted Least Square method based on the Gauss-Newton Algorithm, see Subsection 6.1.2. Denote the log-likelihood $\ln L(\boldsymbol{\beta})$ (as defined in (5.5.6)) by $l(\boldsymbol{\beta})$ for the sake of notational simplicity. With an initial $\boldsymbol{\beta}^{(0)}$, we construct recursively the sequence of approximants with (6.1.18), for $t = 0, 1, \cdots$:

$$\boldsymbol{\beta}^{(t+1)} := \boldsymbol{\beta}^{(t)} - [(\nabla_{\boldsymbol{\beta}} \nabla_{\boldsymbol{\beta}}^\top) l(\boldsymbol{\beta}^{(t)})]^{-1} \nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta}^{(t)}) = \boldsymbol{\beta}^{(t)} - \boldsymbol{H}^{-1}(\boldsymbol{\beta}^{(t)}) \nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta}^{(t)}) \,, \tag{5.5.7}$$

where $\boldsymbol{H}(\boldsymbol{\beta})$ is the Hessian matrix of $l(\boldsymbol{\beta})$, which we are about to compute out explicitly. Recall from (5.5.3) and (5.5.4), the logit function and its derivative are respectively given by:

$$\pi_n = \operatorname{expit}(\boldsymbol{x}_n^\top \boldsymbol{\beta}) = \frac{\exp(\boldsymbol{x}_n^\top \boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}_n^\top \boldsymbol{\beta})} \qquad \text{and} \qquad \nabla_{\boldsymbol{\beta}} \pi_n = \pi_n (1 - \pi_n) \boldsymbol{x}_n \in \mathbb{R}^{D+1} \,.$$

Hence for $i, j = 0, 1, \cdots, D$ and $n = 1, \cdots, N$, we have

$$\left. \frac{\partial \pi_n}{\partial \beta_i} \right|_{\boldsymbol{x}_n} = \pi_n (1 - \pi_n) x_{ni} \,,$$

$$\left. \frac{\partial l}{\partial \beta_i}(\boldsymbol{\beta}) \right|_{\boldsymbol{x}_n} = \sum_{n=1}^{N} \left(\frac{y_n}{\pi_n} - \frac{1 - y_n}{1 - \pi_n} \right) \frac{\partial \pi_n}{\partial \beta_i} = \sum_{n=1}^{N} (y_n - \pi_n) x_{ni} \,,$$

$$\left. \frac{\partial^2 l}{\partial \beta_i \partial \beta_j}(\boldsymbol{\beta}) \right|_{\boldsymbol{x}_n} = - \sum_{n=1}^{N} \pi_n (1 - \pi_n) x_{ni} x_{nj} \,,$$

where we also recall that $x_{n0} = 1$ for $n = 1, \cdots, N$. We may then write

$$\nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta}) = \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{\pi}) \qquad \text{and} \qquad \boldsymbol{H}(\boldsymbol{\beta}) = -\boldsymbol{X}^\top \boldsymbol{R} \boldsymbol{X} \,,$$

where $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N)^\top \in \mathbb{R}^{N\times(D+1)}$, $\boldsymbol{R} = \text{diag}[\pi_1(1-\pi_1), \cdots, \pi_N(1-\pi_N)] \in \mathbb{R}^{N\times N}$, $\boldsymbol{y} = (y_1, \cdots, y_N)^\top \in \mathbb{R}^N$, and $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_N)^\top \in \mathbb{R}^N$. Therefore, the iteration (5.5.7) can be rewritten neatly as:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + (\boldsymbol{X}^\top \boldsymbol{R}^{(t)} \boldsymbol{X})^{-1} \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{\pi}^{(t)}), \tag{5.5.8}$$

where $\boldsymbol{R}^{(t)}$ and $\boldsymbol{\pi}^{(t)}$ are respectively $\boldsymbol{R}$ and $\boldsymbol{\pi}$ evaluated at $\boldsymbol{\beta}^{(t)}$. We remark that since

$$\pi_n = \text{expit}(\boldsymbol{x}_n^\top \boldsymbol{\beta}) = \frac{\exp(\boldsymbol{x}_n^\top \boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}_n^\top \boldsymbol{\beta})} \in (0, 1), \qquad n = 1, \cdots, N.$$

The matrix $\boldsymbol{X}^\top \boldsymbol{R} \boldsymbol{X}$ is positive definite and hence the Hessian $\boldsymbol{H}$ is invertible. The process of finding $\boldsymbol{\beta}$ via (5.5.8) is called *Iteratively Reweighted Least Squares* (**IRLS**). As the change in the values of $\boldsymbol{\beta}^{(t)}$ tends to decrease as the log-likelihood $l(\boldsymbol{\beta})$ (as well as the likelihood $L(\boldsymbol{\beta})$) approaches its global maximum, we may terminate the iteration when the percentage change of $\boldsymbol{\beta}^{(t)}$ is smaller than a prescribed threshold $\varepsilon > 0$, *i.e.* stopping at iteration $t+1$ when

$$\frac{\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}\|_2}{\|\boldsymbol{\beta}^{(t)}\|_2} < \varepsilon. \tag{5.5.9}$$

In the scenarios when computing the inverse of Hessian $\boldsymbol{X}^\top \boldsymbol{R} \boldsymbol{X}$ for each iteration is expensive, supposing that $\boldsymbol{\beta}^{(0)}$ is already quite close to the maximum point sought, we can replace it by an approximation, for instance, $\boldsymbol{H}(\boldsymbol{\beta}^{(t)}) = \boldsymbol{X}^\top \boldsymbol{R}^{(t)} \boldsymbol{X}$ by $\boldsymbol{H}(\boldsymbol{\beta}^{(0)}) = \boldsymbol{X}^\top \boldsymbol{R}^{(0)} \boldsymbol{X}$; such a shortcut approach is called *quasi-Newton method* (see Nocedal and Wright (2006)). Besides, the same method can be also very useful as there could be some limitations, as iterations proceed, some of the $\pi_n^{(t)}$'s approach rapidly to 0 or 1. Their values may then be incorrectly recorded by the computer program as 0 or 1 due to usual rounding procedure. Therefore, the Hessian $\boldsymbol{H}$ becomes singular and the iteration (5.5.8) would then be ill-conditioned; also see Figure 5.5.6.

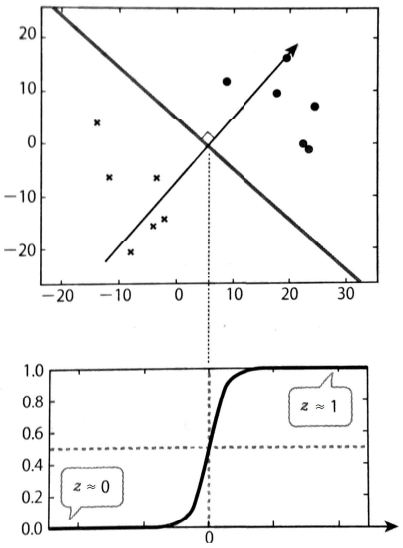

Figure 5.5.6: Causing singularity of $\boldsymbol{H}$ for logistic regression as $\pi_n^{(t)}$ approaching to 0 or 1, where the horizontal axis represents the values taken by $\boldsymbol{x}^\top \boldsymbol{\beta}$, and $z = \text{expit}(\boldsymbol{x}^\top \boldsymbol{\beta})$.

5.5.3 Outlier Detection

As pointed out for the output in Programme 5.5.3, the large magnitudes of coefficients could be caused by outliers, we next aim to use the Mahalanobis distance, as defined in (5.5.10), to detect outliers and then remove them, before we fit into a logistic regression learner. To this end, we first introduce a function in **R**,

`mdist()`, to compute the Mahalanobis distance; see Programme 5.5.6. The dataset "fin-ratio.csv" probably contains many outliers, through which we illustrate the effect of outliers. First, we read in the 2002 data and separate the data into two parts, `d0` for HSI = 0 and `d1` for HSI = 1, as shown in Programme 5.5.7. We detect and can only throw away the outliers in `d0`, since `d1` contains only 32 cases, with which we cannot tolerate to lose any more of them.

```
> mdist <- function(x) {
+   t <- as.matrix(x)            # transform x to a matrix
+   m <- apply(t,2,mean)             # compute column mean, 2 stands for column
+   s <- var(t)                  # compute sample cov. matrix
+   mahalanobis(t,m,s)           # built-in mahalanobis func.
+ }
```

Programme 5.5.6: **R**-code for the function `mdist()` in computing the Mahalanobis distance for the whole sample.

```
> d <- read.csv("fin-ratio.csv")          # read in dataset
> d0 <- d[d$HSI==0,]                      # select HSI=0
> d1 <- d[d$HSI==1,]                      # select HSI=1
> dim(d0)
[1] 648   7 # 7 variables = 6 (financial) + 1 (HSI)
> dim(d1)
[1] 32  7
```

Programme 5.5.7: **R**-code for separating the data in "fin-ratio.csv" based on the value of the HSI variable.

```
> x <- d0[,1:6]                         # save d0 to x
> md <- mdist(x)                        # compute mdist
> plot(md)                              # plot md
```

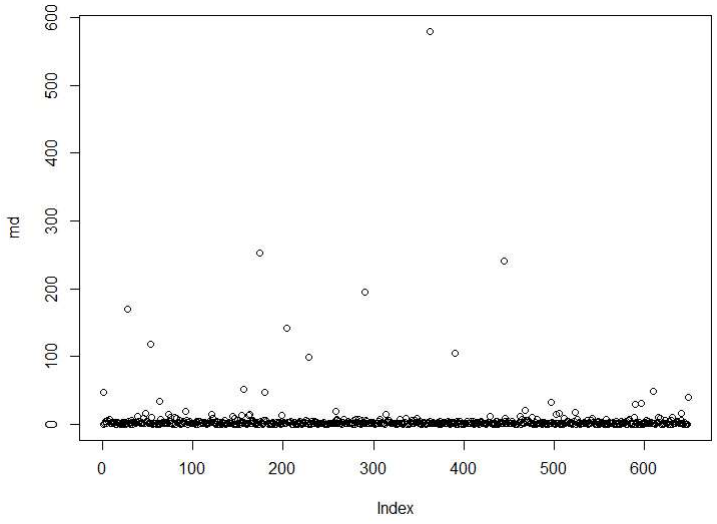

Figure 5.5.7: **R**-code and the resulting plot for the Mahalanobis distance of each stock in `d0`

As shown in Figure 5.5.7, points with a large Mahalanobis distance are the potential outliers which we want to discard. Yet, what is the cut-off value? To this end, we recall that for an iid sample of $\mathbf{X}_1, \cdots, \mathbf{X}_N \in \mathbb{R}^D$ following $\mathcal{N}_D(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, let $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$ denote their respective sample observations, then the Mahalanobis distance D_n for the datum $\boldsymbol{x}_n$ is given by:

$$D_n^2 = (\boldsymbol{x}_n - \bar{\boldsymbol{x}}_N)^\top \boldsymbol{S}^{-1} (\boldsymbol{x}_n - \bar{\boldsymbol{x}}_N) \sim \chi^2_D, \qquad n = 1, 2, \cdots, N, \tag{5.5.10}$$

where $\bar{\boldsymbol{x}}_n$ is the sample mean and $\boldsymbol{S}$ is the observed sample covariance matrix; especially if N is large enough, these D_n^2's are also approximately independent of each other. Based on this, a more scientific way is to inspect the percentiles from a chi-square distribution with degrees of freedom D (= 6 in our "fin-ratio.csv", the six financial variables). We then remove the stocks in `d0` with top 1% values of D_n, and combine the resulting class with `d1` to obtain the cleansed dataset `d3`, see Programme 5.5.8.

```
> (c <- qchisq(0.99,df=6))          # p=6, type-I error = 0.01
[1] 16.81189

> d2 <- d0[md<c,]                   # select case in d0 with md<c
> dim(d2)                           # throw away 648-626=22 cases
[1] 626   7
> d3 <- rbind(d1,d2)                    # combine d1 with d2
> dim(d3)
[1] 658   7
> # save the cleansed dataset
> write.csv(d3,file="fin-ratio1.csv",row.names=F)
```

Programme 5.5.8: **R**-code for cleansing the "fin-ratio.csv" dataset by dropping the top 1% outliers.

Next, we try to fit a logistic regression to this cleansed dataset and remove the financial variables with large p-value one by one by backward eliminations, and we finally arrive at the following model with the confusion matrix (cross tabulation table) as shown in Programme 5.5.9. The accuracy (correct classification rate) is now increased to $653/658 = 99.24\%$, and throwing away the outliers actually gives a simpler model and some better classification results.

```
> summary(glm(HSI~CFTP+ln_MV+BTME,data=d3,family=binomial))

Call:
glm(formula = HSI ~ CFTP + ln_MV + BTME,
family = binomial, data = d3)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-2.377e+00  -1.943e-04  -8.005e-06  -3.054e-07   1.738e+00

Coefficients:
```

```
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -69.9309    21.3821  -3.271  0.00107 **
CFTP         -3.0376     1.2178  -2.494  0.01262 *
ln_MV         7.2561     2.2284   3.256  0.00113 **
BTME          1.3222     0.6418   2.060  0.03940 *
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

> lreg <- glm(HSI~CFTP+ln_MV+BTME,data=d3,binomial)
> pr <- (lreg$fit>0.5)           # prediction
> table(pr,d3$HSI)               # classification table
> table(d3$HSI, pr)
 pr
  FALSE TRUE
0   624    2
1     3   29
```

Programme 5.5.9: **R** output and confusion matrix of the first primitive logistic regression on a cleansed dataset in 2002.

5.5.4 MM Algorithm and IRLS Algorithm

The first order derivative of (5.5.6) with respect to $\boldsymbol{\beta}$ is

$$\nabla_{\boldsymbol{\beta}} \ln L(\boldsymbol{\beta}) = \sum_{n=1}^{N} (y_n - \pi_n)\boldsymbol{x}_n = \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{\pi})$$

and that of the second order derivative is

$$(\nabla_{\boldsymbol{\beta}} \nabla_{\boldsymbol{\beta}}^\top) \ln L(\boldsymbol{\beta}) = -\sum_{n=1}^{N} \pi_n (1 - \pi_n) \boldsymbol{x}_n \boldsymbol{x}_n^\top .$$

With simple calculus (see Böhning and Lindsay (1988)), we know that with $0 < \pi_n < 1$, we have $\pi_n(1-\pi_n) \leq 1/4$, we may define the negative semi-definite matrix:

$$\boldsymbol{A} = -\frac{1}{4} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top = -\frac{1}{4} \boldsymbol{X}^\top \boldsymbol{X} \preceq 0 ,$$

such that $(\nabla_{\boldsymbol{\beta}} \nabla_{\boldsymbol{\beta}}^\top) \ln L(\boldsymbol{\beta}) - \boldsymbol{A} \succeq 0$, *i.e.* a positive semi-definite matrix; see Property 13 in Section 3.3.

Using the lower bound and the concave version of Proposition **??** and noting that $\ln L$ is concave, we have

$$\ln L(\boldsymbol{\beta}) \geq \ln L(\boldsymbol{\beta}^{(t)}) + (\nabla_{\boldsymbol{\beta}} \ln L(\boldsymbol{\beta}^{(t)}))^\top (\boldsymbol{\beta} - \boldsymbol{\beta}^{(t)}) + \frac{1}{2} (\boldsymbol{\beta} - \boldsymbol{\beta}^{(t)})^\top \boldsymbol{A} (\boldsymbol{\beta} - \boldsymbol{\beta}^{(t)}) := g(\boldsymbol{\beta} | \boldsymbol{\beta}^{(t)}) ,$$

such that $g(\boldsymbol{\beta}|\boldsymbol{\beta}^{(t)})$ minorizes $L(\boldsymbol{\beta})$ at $\boldsymbol{\beta}^{(t)}$. Consider the first order derivative of $g(\boldsymbol{\beta}|\boldsymbol{\beta}^{(t)})$ with respect to $\boldsymbol{\beta}$ and set it to zero yields:

$$\begin{aligned} & \nabla_{\boldsymbol{\beta}} g(\boldsymbol{\beta}|\boldsymbol{\beta}^{(t)}) = \nabla_{\boldsymbol{\beta}} \ln L(\boldsymbol{\beta}^{(t)}) + \boldsymbol{A}\boldsymbol{\beta} - \boldsymbol{A}\boldsymbol{\beta}^{(t)} = \boldsymbol{0} \\ \Rightarrow \quad & \boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \boldsymbol{A}^{-1} \nabla_{\boldsymbol{\beta}} \ln L(\boldsymbol{\beta}^{(t)}) = \boldsymbol{\beta}^{(t)} + 4(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{\pi}^{(t)}) , \end{aligned}$$

where

$$\pi_n^{(t)} = \frac{\exp(\boldsymbol{x}_n^\top \boldsymbol{\beta}^{(t)})}{1 + \exp(\boldsymbol{x}_n^\top \boldsymbol{\beta}^{(t)})} .$$