# CHAPTER 3: RECOGNITION

## LOGISTIC REGRESSION AND RANKING

### I. Overview

Chapter 2 is about "**Recognition**". This is a very important capability in real-world practices of analytics. It is a capability to recognize the same "abstracted" analytic problem embedded in seemly different real-world problems. This is not to say that all the real-world problems can be reduced to one single abstracted problem. A real-world problem usually contains multiple perspectives and layers, presenting itself as a combination or composition of multiple abstracted problems. Being able to recognize these abstracted problems holds the key to solve these real-world problems effectively. After all, to solve a real-world problem, at a certain point or a certain level, you have to bring the problem or part of the problem or a certain aspect of the problem into the territory of a classic analytic problem, because only in those classic territories we know we can solve problems for sure.

## II. Logistic Regression Model

### II.1 Rationale and Formulation

Linear regression model is introduced as a tool to predict a continuous response (or called outcome) variable $y$ using a few input variables $\boldsymbol{x}$. In some applications, the response is a binary variable that denotes two classes. For example, in the AD dataset, we may wonder if we could use some variables to predict if the subject is a normal person or diseased.

We have learned about linear regression model to connect some input variables with the outcome variable. It is natural to wonder if and how the linear regression framework could still be useful here. Specifically, here, the input variables are still $\boldsymbol{x}$, but the outcome is not simply $y$. Rather, we may be more interested to predict probability $Pr(y = 1)$. Thus, we want to create probability by a function $p(\boldsymbol{x})$ such that $Pr(y = 1) = p(\boldsymbol{x})$. Following the mentality of linear regression, we somehow envision that the linear form, $\beta_0 + \sum_{i=1}^{p} \beta_i x_i$, should be used here as a constitutional component to define $p(\boldsymbol{x})$. It is hard to directly link $p(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$ though, since $p(x)$ as a probability has to be in the range of [0,1] but $\beta_0 + \sum_{i=1}^{p} \beta_i x_i$ has no limitation in the range. If we look closer into the idea of using a linear form to encode the predictive information in $\boldsymbol{x}$, we may realize that the linear form is very useful in ranking the possibilities rather than directly being eligible probabilities. In other words, a linear form is advantageous to make a comparison of two inputs, say, $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$, and evaluates which one leads to a higher probability of $Pr(y = 1)$. Thus, we don't have to let $p(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$, but only need $p(\boldsymbol{x}) \propto \beta_0 + \sum_{i=1}^{p} \beta_i x_i$.

To fix this problem, statisticians soon found that it is better to link these two entities as:

$$\log \frac{p(x)}{1-p(\boldsymbol{x})} = \beta_0 + \sum_{i=1}^{p} \beta_i x_i.$$

This is the so-called logistic regression model. The name stems from the transformation of $p(\boldsymbol{x})$ used here, i.e., the $\log \frac{p(\boldsymbol{x})}{1-p(\boldsymbol{x})}$, which is the logistic

transformation that has been widely used in many areas such as physics and signal processing.

The logistic regression model can be further transformed:

$$p(\boldsymbol{x}) = \frac{1}{1+e^{-\left(\beta_0+\sum_{i=1}^{p}\beta_i x_i\right)}}.$$

Based on this, we could predict $y = 1$ if $p(\boldsymbol{x}) \geq 0.5$, and $y = 0$ if $p(\boldsymbol{x}) < 0.5$.

As the thought process revealed above, logistic regression model is not the only eligible model for doing classification using linear form of the input variables. This is just one of the possibilities motivated by the linkage we can establish between $p(\boldsymbol{x})$ and $\beta_0 + \sum_{i=1}^{p}\beta_i x_i$ by $\log\frac{p(x)}{1-p(x)}$. Later we will learn more such models such as the Support Vector Machine (SVM) that still keep the linear form but follow different linkage functions. It is very important to know that this is a choice made by us, rather than a reality imposed on us nor a mathematical necessity that we have to accept.

### II.2 Theory/Method

Now we show how to estimate the regression parameters in a logistic regression model.

The likelihood function is:

$$L(\boldsymbol{\beta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n)^{y_n}\big(1 - p(\boldsymbol{x}_n)\big)^{1-y_n}.$$

We use the log-likelihood to turn products into sums:

$$l(\boldsymbol{\beta}) = \sum_{n=1}^{N}\big\{y_n \log p(\boldsymbol{x}_n) + (1 - y_n)\log\big(1 - p(\boldsymbol{x}_n)\big)\big\}.$$

This could be further transformed into

$$l(\boldsymbol{\beta}) = \sum_{n=1}^{N} -\log\left(1 + e^{\beta_0+\sum_{i=1}^{p}\beta_i x_{ni}}\right) - \sum_{n=1}^{N} y_n\big(\beta_0 + \sum_{i=1}^{p}\beta_i x_{ni}\big),$$

since

$$\sum_{n=1}^{N}\big\{y_n \log p(\boldsymbol{x}_n) + (1 - y_n)\log\big(1 - p(\boldsymbol{x}_n)\big)\big\},$$

$$= \sum_{n=1}^{N}\log\big(1 - p(\boldsymbol{x}_n)\big) - \sum_{n=1}^{N} y_n \log\frac{p(\boldsymbol{x}_n)}{1-p(\boldsymbol{x}_n)},$$

$$= \sum_{n=1}^{N} -\log\left(1 + e^{\beta_0+\sum_{i=1}^{p}\beta_i x_{ni}}\right) - \sum_{n=1}^{N} y_n\big(\beta_0 + \sum_{i=1}^{p}\beta_i x_{ni}\big).$$

The Newton-Raphson algorithm is commonly used to optimize the log-likelihood function of the logistic regression model to identify the optimal regression parameters. The Newton-Raphson algorithm is an iterative algorithm that seeks updates of the current solution using the following formula:

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} - \left(\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}\right)^{-1} \frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}.$$

Here, $\boldsymbol{\beta}$ is the column vector form of the regression parameters.

We can show that

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{n=1}^{N} \boldsymbol{x}_n \big(y_n - p(\boldsymbol{x}_n)\big),$$

$$\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = -\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T p(\boldsymbol{x}_n)\big(1 - p(\boldsymbol{x}_n)\big).$$

A certain structure can then be revealed if we rewrite it in matrix form:

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbf{X}^T(\boldsymbol{y} - \boldsymbol{p}),$$

$$\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = -\mathbf{X}^T \mathbf{W} \mathbf{X}.$$

where $\mathbf{X}$ is the $N \times (p+1)$ input matrix, $\boldsymbol{y}$ is the $N \times 1$ column vector of $y_i$, $\boldsymbol{p}$ is the $N \times 1$ column vector of $p(\boldsymbol{x}_n)$, and $\mathbf{W}$ is a $N \times N$ diagonal matrix of weights with the n$^{th}$ diagonal element as $p(\boldsymbol{x}_n)\big(1 - p(\boldsymbol{x}_n)\big)$.

Then, plugging this into the updating formula of the Newton-Raphson algorithm, we can derive that

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} + (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} (\boldsymbol{y} - \boldsymbol{p}),$$

$$= (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \Big(\mathbf{X} \boldsymbol{\beta}^{old} + \mathbf{W}^{-1} (\boldsymbol{y} - \boldsymbol{p})\Big),$$

$$= (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \boldsymbol{z},$$

where $\mathbf{z} = \mathbf{X} \boldsymbol{\beta}^{old} + \mathbf{W}^{-1} (\boldsymbol{y} - \boldsymbol{p})$.

This resembles the generalized least squares (GLS) estimator of a regression model, where each data point $(\boldsymbol{x}_n, y_n)$ is associated with a weight $w_n$ to reduce the influence of potential outliers in fitting the regression model. This insight revealed by the Newton-Raphson algorithm suggests a new perspective to look at the logistic regression model. The

updating formula suggests that we are actually solving a weighted regression model as:

$$\boldsymbol{\beta}^{new} \leftarrow \arg\min_{\boldsymbol{\beta}} (\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta})^T \mathbf{W} (\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta}).$$

For this reason, this algorithm is also called the Iteratively Reweighted Least Square or IRLS algorithm. $\boldsymbol{z}$ is referred as the adjusted response.

Putting all these together, a complete flow of the IRLS is shown in below:

1. Initialize $\boldsymbol{\beta}$.
2. Compute $\boldsymbol{p}$ by its definition: $p(\boldsymbol{x}_n) = \frac{1}{1+e^{-\left(\beta_0 + \sum_{i=1}^{p} \beta_i x_{ni}\right)}}$ for $n = 1,2,\dots,N$.
3. Compute the diagonal matrix $\mathbf{W}$, while the n[th] diagonal element as $p(\boldsymbol{x}_n)\big(1 - p(\boldsymbol{x}_n)\big)$ for $n = 1,2,\dots,N$.
4. Set $\boldsymbol{z}$ as $= \mathbf{X}\boldsymbol{\beta} + \mathbf{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})$.
5. Set $\boldsymbol{\beta}$ as $(\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\boldsymbol{z}$.
6. If the stopping criteria is met, stop; otherwise go back to step 2.

### II.3 R Lab

Focusing on the AD dataset, now let's predict the diagnosis of the subjects either as normal or diseased. The variable, `DX_bl`, encodes the diagnosis information, i.e., "0" denotes normal while "1" denotes diseased.

```
#### Dataset of Alzheimer's Disease
#### Objective: prediction of diagnosis
# filename
AD <- read.csv('AD_bl.csv', header = TRUE)
str(AD)
```

First, let's examine a simple logistic regression model using the function `glm()` with only one predictor, `FDG`.

```
# Fit a logistic regression model with FDG
logit.AD <- glm(DX_bl ~ FDG, data = AD, family = "binomial")
summary(logit.AD)
```

```
##
## Call:
## glm(formula = DX_bl ~ FDG, family = "binomial", data = AD)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4686  -0.8166  -0.2758   0.7679   2.7812
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  18.3300     1.7676   10.37   <2e-16 ***
## FDG          -2.9370     0.2798  -10.50   <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: 711.27  on 516  degrees of freedom
## Residual deviance: 499.00  on 515  degrees of freedom
## AIC: 503
##
## Number of Fisher Scoring iterations: 5
```

It can be seen from the summary of the built model that the predictor `FDG` is significant, as the `p-value` is `<2e-16` that is far less than 0.05. Although it is not as straightforward as in linear regression model that we could derive the metric `R-squared` to examine how much proportion of the variability of the outcome variable could be explained away by the predictor, here, we could observe that, out of the total deviance of `711.27`, `711.27`–`499.00` = `212.27` could be explained by the predictor `FDG`.

We could then query what is the 95% CI of the regression parameter:

```
## CIs of the regression parameters using profiled log-likelihood
confint(logit.AD)

##                 2.5 %    97.5 %
## (Intercept) 15.033585 21.974091
## FDG         -3.513878 -2.415248
```

It is worthy of mentioning that, the `wald.test()` that builds on the Chi-squared test, is also another method that is often used in testing the significance of the regression parameters in logistic regression model. The

results by the `wald.test()` is usually in compliance with the approximated t-test results as shown in the `summary()` function as we have seen above.

```
## wald test for the regression coefficients
library(aod)

## Warning: package 'aod' was built under R version 3.3.3

wald.test(b = coef(logit.AD), Sigma = vcov(logit.AD), Terms = 2)
```

For instance, we can see that, by setting "`Terms = 2`", the Chi-squared test used in the `wald.test()` shows significance of the predictor `FDG` in predicting `DX_bl`.

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 110.2, df = 1, P(> X2) = 0.0
```

With a built model, it is of interest to see how it can be used to predict on a given dataset. Here, for presentational purpose, we randomly pick up 200 samples from the AD dataset to form the `AD.pred`, our imagined dataset to be predicted by the model.

```
# To predict on a given dataset
AD.pred <- AD[sample(1:dim(AD)[1], 200),]
# predict() uses all the temp values in dataset, including append
ed values
pred <- predict(logit.AD, AD.pred, type = "link", se.fit = TRUE)
AD.pred$fit    <- pred$fit
AD.pred$se.fit <- pred$se.fit
```

We can readily convert these information into the 95% CIs of the predictions (the way these 95% CIs are derived are again, only in approximated sense).

```
# CI for fitted values
AD.pred <- within(AD.pred, {
  # added "fitted" to make predictions at appended temp values
  fitted    = exp(fit) / (1 + exp(fit))
  fit.lower = exp(fit - 1.96 * se.fit) / (1 + exp(fit - 1.96 * se.
fit))
  fit.upper = exp(fit + 1.96 * se.fit) / (1 + exp(fit + 1.96 * se.
fit))
})
```

We can draw the following figure to visualize the prediction.

```
# visualize the prediction
library(ggplot2)
newData <- AD.pred[order(AD.pred$FDG),]
p <- ggplot(newData, aes(x = FDG, y = DX_bl))
# predicted curve and point-wise 95% CI
p <- p + geom_ribbon(aes(x = FDG, ymin = fit.lower, ymax = fit.up
per), alpha = 0.2)
p <- p + geom_line(aes(x = FDG, y = fitted), colour="red")
# fitted values
p <- p + geom_point(aes(y = fitted), size=2, colour="red")
# observed values
p <- p + geom_point(size = 2)
p <- p + ylab("Probability")
p <- p + labs(title = "Observed and predicted probability of dise
ase")
print(p)
```

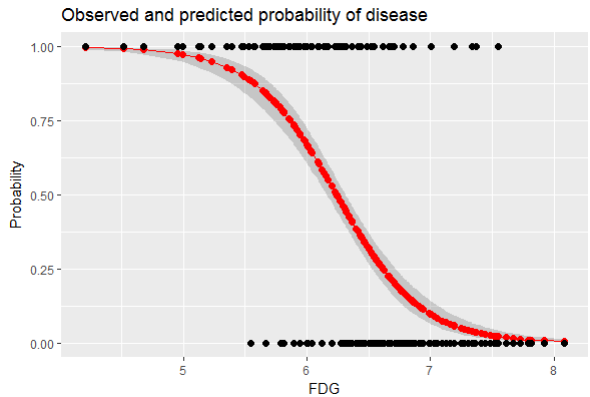


**Figure 3.1**: Predicted probabilities (with their 95% CIs) versus observed outcomes

The figure is shown in Figure 3.1. It can be seen that, the model prediction is significant and captures the relationship between `FDG` with `DX_bl` with a smooth logit curve, and the prediction confidences are fairly small (evidenced by the tight 95% CIs).

On the other hand, from Figure 3.1, we can also see that, while the single-predictor model is significant and does well on the two extreme ends of the probability range, in the middle part its prediction power is limited,

calling for more predictors to enhance its prediction power. Thus, in the next step, we decide to add more variables to the logistic regression model. Before we build the model, it is of interest to visualize the relationships between the predictors with the outcome variable. For example, in the following we show how the continuous variables could be presented in Boxplot form to see if the distribution of the continuous variable is different across the two classes of samples.

```
# install.packages("reshape2")
require(reshape2)

AD.long <- melt(AD[,c(1,2,4,5,6,7,19)], id.vars = c("ID", "DX_bl
"))
# Plot the data using ggplot
require(ggplot2)
p <- ggplot(AD.long, aes(x = factor(DX_bl), y = value))
# boxplot, size=.75 to stand out behind CI
p <- p + geom_boxplot(size = 0.75, alpha = 0.5)
# points for observed data
p <- p + geom_point(position = position_jitter(w = 0.05, h = 0),
alpha = 0.1)
# diamond at mean for each group
p <- p + stat_summary(fun.y = mean, geom = "point", shape = 18, s
ize = 6,
                      alpha = 0.75, colour = "red")
# confidence limits based on normal distribution
p <- p + stat_summary(fun.data = "mean_cl_normal", geom = "errorb
ar",
                      width = .2, alpha = 0.8)
p <- p + facet_wrap( ~ variable, scales = "free_y", ncol = 3)
p <- p + labs(title = "Boxplots of variables by diagnosis (0 - no
rmal; 1 - patient)")
print(p)
```

This code will generate Figure 3.2 which shows that some variables, such as `FDG` and `HippoNV`, could separate the two classes significantly. Some variables such as `AV45` and `AGE` have less prediction power, but still look promising. It is important to recognize that these figures only show marginal relationship among variables. Thus, while it is helpful, it is also important to keep in mind its limitations such as the inability to show synergistic effects among the variables.

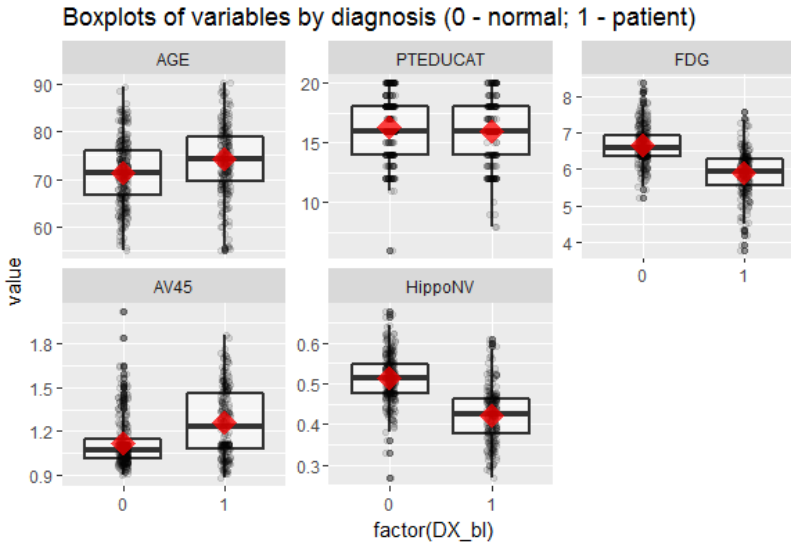


**Figure 3.2**: Boxplots of the continuous predictors in the two classes

Just like in the linear regression model, we could use the function `step()` to automatically select the best model given a set of variables.

```
# Automatic selection of the model
logit.AD.full <- glm(DX_bl ~ ., data = AD[,c(1:16)], family = "bi
nomial")
logit.AD.final <- step(logit.AD.full, direction="both", trace = 0)
summary(logit.AD.final)

##
## Call:
## glm(formula = DX_bl ~ AGE + PTEDUCAT + FDG + AV45 + HippoNV +
##     rs3818361 + rs610932 + rs3851179, family = "binomial", dat
a = AD[,
##     c(1:16)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6385  -0.5022  -0.1146   0.3651   3.0694
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  31.17834    3.76132   8.289  < 2e-16 ***
## AGE          -0.03128    0.02134  -1.466   0.1427
```

```
## PTEDUCAT     -0.12833    0.05087  -2.523   0.0116 *
## FDG          -2.73262    0.33499  -8.157 3.43e-16 ***
## AV45          1.58053    0.72007   2.195   0.0282 *
## HippoNV     -24.42793    2.74972  -8.884  < 2e-16 ***
## rs3818361    -0.43672    0.28703  -1.522   0.1281
## rs610932      0.47909    0.28390   1.688   0.0915 .
## rs3851179    -0.48461    0.27440  -1.766   0.0774 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 711.27  on 516  degrees of freedom
## Residual deviance: 344.47  on 508  degrees of freedom
## AIC: 362.47
##
## Number of Fisher Scoring iterations: 6
```

The final model selected by the backward-forward selection procedure implemented in the `step()` function is shown in below, which includes 8 predictors and one intercept. You may have noticed that some variables included in this model are actually not significant by the approximated t-test.

The model as a whole could be evaluated by the Chi-square test against the null hypothesis that there is a lack of fit. And it can be seen that the model shows no lack of fit as the `p-value` is 1.

```
# Test residual deviance for lack-of-fit (if > 0.10, little-to-no
 lack-of-fit)
dev.p.val <- 1 - pchisq(logit.AD.final$deviance, logit.AD.final$d
f.residual)
dev.p.val

## [1] 1
```

Again, we can derive the 95% CIs of the regression coefficients:

```
# coefficients and 95% CI
cbind(coef = coef(logit.AD.final), confint(logit.AD.final))
```

And we can observe that:

```
##                      coef        2.5 %       97.5 %
## (Intercept)  31.17834039  24.15644635  38.94011124
## AGE          -0.03127754  -0.07367180   0.01022977
## PTEDUCAT     -0.12833007  -0.23028570  -0.03019323
## FDG          -2.73262447  -3.42455191  -2.10810353
```

```
## AV45          1.58052749   0.17699902   3.00488253
## HippoNV     -24.42793042 -30.12412328 -19.31065470
## rs3818361    -0.43672386  -1.00682550   0.12136335
## rs610932      0.47909019  -0.07263237   1.04320233
## rs3851179    -0.48460576  -1.02757559   0.05088704
```

Regarding the regression coefficients of logistic regression model, it is also of interest to convert them into odds ratios for another interpretation. This could be done by directly calling upon the definition of the odds ratio, as done in the codes shown in below:

```
## odds ratios and 95% CI
exp(cbind(OR = coef(logit.AD.final), confint(logit.AD.final)))
```

Then, we can derive the odds ratios and their 95% CIs.

```
##                         OR        2.5 %       97.5 %
## (Intercept) 3.472012e+13 3.097500e+10 8.155967e+16
## AGE         9.692065e-01 9.289765e-01 1.010282e+00
## PTEDUCAT    8.795630e-01 7.943066e-01 9.702580e-01
## FDG         6.504835e-02 3.256387e-02 1.214681e-01
## AV45        4.857517e+00 1.193630e+00 2.018384e+01
## HippoNV     2.460847e-11 8.265316e-14 4.106664e-09
## rs3818361   6.461498e-01 3.653770e-01 1.129035e+00
## rs610932    1.614605e+00 9.299426e-01 2.838292e+00
## rs3851179   6.159400e-01 3.578735e-01 1.052204e+00
```

Besides these significant tests and presentations of the model parameters, we can also look at the predictions of the model on the samples. We can use the function, `fitted()`, to derive the probabilities of diseased given by the model for the samples. Then, we could visualize the predictions using the boxplots:

```
# visualize the correlation
tempData = cbind(Yhat,AD$DX_bl)
require(ggplot2)
qplot(factor(AD$DX_bl), Yhat, data = AD,
      geom=c("boxplot"), fill = factor(AD$DX_bl),title="Predictio
n versus Observed")
```

The result is shown in Figure 3.3, which indicates that the model can separate the two classes significantly.

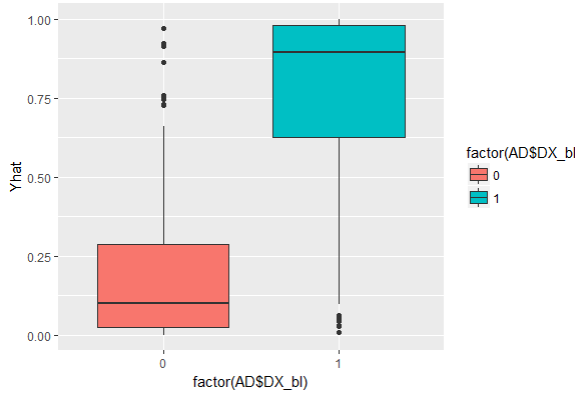


**Figure 3.3**: Boxplots of the predicted probabilities of diseased in the two classes

### II.4 Remarks

Another perspective to look into the origin of logistic regression model is motivated by an empirical observation. Here, let's use the AD dataset, and pick up the variables, `HippoNV` and `DX_bl`, to see empirically what is the shape the relationship between the two takes. Specifically, here, we categorize the continuous variable `HippoNV` into distinct levels, and observe what is the prevalence of AD incidences within each level. The following R code serves this data processing purpose.

```
# Create the frequency table in accordance of categorization of H
ippoNV
temp = quantile(AD$HippoNV,seq(from = 0.05, to = 0.95, by = 0.05))
AD$HippoNV.category <- cut(AD$HippoNV, breaks=c(-Inf, temp, Inf))
tempData <- data.frame(xtabs(~DX_bl + HippoNV.category, data = A
D))
tempData <- tempData[seq(from = 2, to = 2*length(unique(AD$HippoN
V.category)), by = 2),]
summary(xtabs(~DX_bl + HippoNV.category, data = AD))

tempData$Total <- colSums(as.matrix(xtabs(~DX_bl + HippoNV.catego
ry, data = AD)))
tempData$p.hat <- 1 - tempData$Freq/tempData$Total
tempData$HippoNV.category = as.numeric(tempData$HippoNV.category)
str(tempData)
```

We can use the `str(tempData)` to visualize the data we have converted, where 20 levels of `HippoNV` has been created; "`Total`" denotes the total number of subjects within each level, and "`p.hat`" denotes the proportion of the diseased subjects within each level (the prevalence).

```
str(tempData)

## 'data.frame':    20 obs. of  5 variables:
##  $ DX_bl           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 ...
##  $ HippoNV.category: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Freq            : int  24 25 25 21 22 15 17 17 19 11 ...
##  $ Total           : num  26 26 26 26 26 25 26 26 26 34 ...
##  $ p.hat           : num  0.0769 0.0385 0.0385 0.1923 0.1538 ...
```

We are now ready to further visualize the relationship between the two variables by drawing a scatterplot, as shown in Figure 3.4. We also use the "`loess`" method, which is a nonparametric smoothing method, to fit the relationship, which clearly shows a logit type functional shape of the relationship. This provides an empirical justification of the use of the logic transformation $\log\frac{p(x)}{1-p(x)}$, to $p(x)$ and $\beta_0 + \sum_{i=1}^{p}\beta_i x_i$, which gives birth to the logistic regression model.

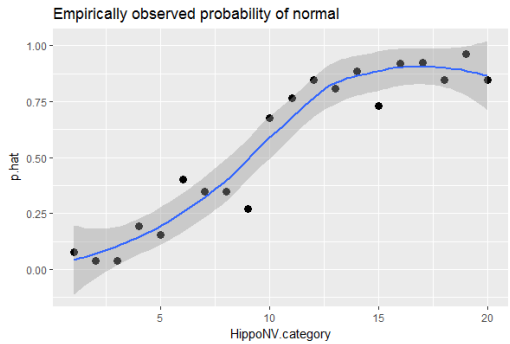


**Figure 3.4**: The empirical relationship between `HippoNV` and `DX_bl` takes a shape as the logit function

```
# Draw the scatterplot of HippoNV.category versus the probability
 of normal
library(ggplot2)

p <- ggplot(tempData, aes(x = HippoNV.category, y = p.hat))
p <- p + geom_point(size=3)
p <- p + geom_smooth(method = "loess")
p <- p + labs(title ="Empirically observed probability of normal",
 xlab = "HippoNV")
print(p)
```

## III. A Product Ranking Problem by Pairwise Comparison

### III.1 Rationale and Formulation

In recent years, we have witnessed a growing interest in estimating the ranks of a list of items. This same problem could be found in a variety of applications, such as the online advertisement of products on Amazon or movie recommendation by Netflix. These problems could be analytically summarized as: given a list of items denoted by $\boldsymbol{M} = \{M_1, M_2, \dots, M_p\}$, what is the rank (denoted by $\boldsymbol{\phi} = \{\phi_1, \phi_2, \dots, \phi_p\}$) we should attribute to them? Here, $\boldsymbol{\phi}$ is a vector of real values, i.e., the larger the $\phi_i$, the higher the rank of $M_i$.

To obtain this ranking of the items, comparison data (either by domain expert or users) is often collected, e.g., a pair of items in $\boldsymbol{M}$, let's say, $M_i$ and $M_j$, will be pushed to the expert/user who conduct the comparison to see if $M_i$ is better than $M_j$, and then, a score, denoted as $y_k$, will be returned, i.e., a positive $y_k$ indicates that the expert/user knowledge more tends to support that $M_i$ is better than $M_j$, while a negative $y_k$ indicates the opposite. Note that the larger the $y_k$, stronger the knowledge.

Following this line, we denote the initial data set as $D_0$, which consists of the set of pairwise comparisons that are queried (denoted as a set $\boldsymbol{S}_0$) and the corresponding expert response data (denoted as a vector $\boldsymbol{y}_0$). The next question is, how to estimate the underlying ranking $\boldsymbol{\phi}$? And further, how to further collect pairwise comparison data to enhance the estimation

of $\boldsymbol{\phi}$, i.e., in other words, what should be the new comparisons in $\boldsymbol{S_1}$ so we can collect the corresponding new data $\boldsymbol{y_1}$?

### III. 2 Theory/Method

Obviously, these are statistical questions. From the first glance, it looks unfamiliar. But in this paper[1], it is revealed that the underlying statistical model is a linear regression model! This surprising recognition indicates that we can readily use the rich array of methods and conclusions in linear regression framework to solve many problems in ranking!

To see that, first, it is important to make explicit what probabilistic relationship is implied in the pairwise comparison mechanism, that can be used to model the relationship between the parameter to be estimated ($\boldsymbol{\phi}$) and the data ($D_0$). Specifically, we could establish a probabilistic relationship between $\boldsymbol{\phi}$ and the observed $D_0$, i.e., for the k$^{th}$ comparison that involves items $M_i$ and $M_j$, we could assume that $y_k$ is distributed as:

$$y_k \sim N\left(\phi_i - \phi_j, \sigma^2/w_k\right).$$

This essentially assumes that if the item $M_i$ is more (or less) important than the model $M_j$, we will expect to see positive (or negative) values of $y_k$. This is consistent with the nature of the expert/user comparison data in many applications. Note that, $\sigma^2$ encodes the overall accuracy level of the expert/user knowledge, as more knowledgeable expert/user will tend to have smaller $\sigma^2$. Also, $w_k$ encodes uncertainty in this particular comparison, acting as the local accuracy level of the expert/user knowledge. In practice, expert/user could also provide their confidence level, i.e., $w_k$, along with $y_k$. Alternatively, when this information is lacking, we could simply assume $w_k = 1$ for all the comparison data. Following this line, we could further illustrate how we could represent the comparison data in a more compact matrix form. This is shown in Figure 3.5.

---

[1] *Osting, B., Brune, C. and Osher, S. Enhanced statistical rankings via targeted data collection. Proceedings of the 30th International Conference on Machine Learning (ICML) 2013.*

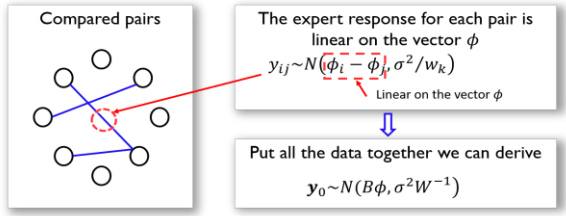


**Figure 3.5**: The data structure and its analytic formulation underlying the pairwise comparison. Each node is an item in $M$ while each arc represents a comparison of two items

Note that it is straightforward to derive the structure of the matrix $\mathbf{B}$ as shown in Figure 3.5:

$$\mathbf{B}_{kj} = \begin{cases} 1 & if\ j = head(k) \\ -1 & if\ j = tail(k) \\ 0 & otherwise \end{cases}$$

Here, $j = tail(k)$ if the k[th] comparison is asked in the form as "if $M_i$ is better than $M_j$" (i.e., denoted as $M_i \to M_j$); otherwise, $j = head(k)$ for question asked in the form as $M_j \to M_i$.

Then, we can derive that

$$\boldsymbol{y} \sim N(\mathbf{B}\boldsymbol{\phi}, \sigma^2\mathbf{W}^{-1}).$$

where $\mathbf{W}$ is the diagonal matrix of elements $w_k$ for $k = 1,2,\dots,K$. Thus, for the initial expert comparison data $D_0$, we could derive that

$$\boldsymbol{y}_0 \sim N(\mathbf{B}_0\boldsymbol{\phi}, \sigma^2\mathbf{W}_0^{-1}).$$

where $\mathbf{B}_0$ is defined on the set $S_0$. Using the framework developed in Chapter 2, we could derive the GLS estimator of $\boldsymbol{\phi}$ as

$$\widehat{\boldsymbol{\phi}} = (\mathbf{B}_0^T\mathbf{W}_0\mathbf{B}_0)^{-1}\mathbf{B}_0^T\mathbf{W}_0\boldsymbol{y}_0.$$

The recognition of the linear regression formulation underling the ranking problem brings more insights and operational possibilities to solve the problem better. For example, as many design of experiments techniques have been developed for optimal data collection, while most are based on the linear regression framework, these techniques could find relevance in

this ranking problem, e.g., what new comparison data we should collect to optimize statistical accuracy and efficiency given limited budget? As shown in the paper, an E-optimal design method could be introduced here to optimally decide on which new comparison should be conducted. As this process involves Bayesian statistics, optimal design, and optimization, interested readers are encouraged to read the paper.

## IV. Exercises

### *Data analysis*

1. Create a new binary variable based on `AGE`, by labeling the subjects whose age is above the mean of `AGE` to be class "1" and labeling the subjects whose age is below the mean of `AGE` to be class "0". Then, repeat the analysis shown in the R lab of this chapter for the logistic regression model and the analysis shown in the R lab of Chapter 2 for decision tree model. Identify the final models you would select, evaluate the models, and compare the regression model with the tree model.
2. Find two datasets from the UCI data repository or R datasets. Conduct a detailed analysis for both datasets using both logistic regression model and the tree model, e.g., for regression model, you may want to conduct model selection, model comparison, testing of the significance of the regression parameters, evaluation of the R-squared and significance of the model. Also comment on the application of your model on the context of the dataset you have selected.
3. Pick up any dataset you have used, and randomly split the data into two halves. Use one half to build the tree model and the regression model. Test the models' prediction performances on the second half. Report what you have found, adjust your way of model building, and suggest a strategy to find the model you consider as the best.

***Derivation***

4. Use your pen and paper, write up the optimization process to estimate the regression parameters using the dataset in Table 2.4. If your optimization process requires more than 3 iterations to converge, delinerate 3 iterations is sufficient.

***Programming***

5. Write your own R script to implement the IRLS algorithm of a logistic regression model. Compare the output from your script with the output from `glm()`.
6. Write your own R script to generate prediction of new data points using an estimated logistic regression model.
7. Write your own R script to implement the ranking problem formulated as a linear regression model. Test it on the dataset shown below to estimating the $\boldsymbol{\phi}$.

   $M_6 \to M_4 = 2.33$, $M_5 \to M_6 = 1.80$, $M_4 \to M_3 = -7.45$, $M_2 \to M_8 = 13.18$, $M_2 \to M_6 = 4.37$, $M_1 \to M_5 = 0.32$, $M_7 \to M_2 = -0.43$.

   (Here, for validation only, the comparison data is generated from $\boldsymbol{\phi} = \{3.9,10.2,6.7,1.7,5.2,3.4,7.8,2.3\}$.