On the Inference of the Logistic Regression Model

1. Model

\[ \ln \frac{p}{1-p} = f(\tilde{x}; \tilde{\beta}), \]  i.e. \[ p = \frac{\exp(f(\tilde{x}; \tilde{\beta}))}{1+\exp(f(\tilde{x}; \tilde{\beta}))}, \] where \( p = P(Y = 1|\tilde{x}; \tilde{\beta}) \), with “1” representing “true”, “0” representing “false”.

The linear form of \( f(\tilde{x}; \tilde{\beta}) \) is entertained, i.e.

\[ f(\tilde{x}; \tilde{\beta}) = \beta_0 1 + \beta_1 x_1 + \cdots + \beta_m x_m = \tilde{x}^T \tilde{\beta}. \] \( (x_0 = 1 \text{ here}) \)

2. Inference

2.1. Maximum Likelihood Method

Data: \( D = \{\tilde{y}, X\} \), where \( \tilde{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \) is vector of binary responses, \( X = \begin{bmatrix} \tilde{x}^{(1)}^T \\ \vdots \\ \tilde{x}^{(n)}^T \end{bmatrix} \) is the feature matrix with the first column being all 1’s.

Log-likelihood function:

\[ l(\tilde{\beta}) = \sum_{i=1}^{n} \ln \left( \left( p_i(\tilde{\beta}) \right)^{y_i} \left( 1 - p_i(\tilde{\beta}) \right)^{1-y_i} \right) \]

\[ = \sum_{i=1}^{n} \left( y_i \ln p_i(\tilde{\beta}) + (1 - y_i) \ln (1 - p_i(\tilde{\beta})) \right). \] \( (1) \)

First Derivative (Gradient):

\[ \frac{dt}{d\tilde{\beta}} = \sum_{i=1}^{n} \frac{\partial l}{\partial p_i} \cdot \frac{dp_i}{d\tilde{\beta}}. \]

\[ \frac{\partial l}{\partial p_i} = \frac{y_i}{p_i} - \frac{1-y_i}{1-p_i} = \frac{y_i(1-p_i) - p_i(1-y_i)}{p_i(1-p_i)} = \frac{y_i - p_i}{p_i(1-p_i)}. \]

\[ \frac{dp_i}{df_i} = \frac{dp_i}{d\tilde{\beta}} \cdot \frac{df_i}{d\tilde{\beta}}. \]

\[ \frac{dp_i}{df_i} = \frac{\exp(f_i) - \exp(f_i)}{(1+\exp(f_i))^2} = \frac{\exp(f_i) - \exp(f_i)}{(1+\exp(f_i))^2} = \frac{1}{p_i(1-p_i)}. \]

\[ \frac{df_i}{d\tilde{\beta}} = \tilde{x}^{(i)}. \]

\[ \frac{dp_i}{d\tilde{\beta}} = p_i(1-p_i)\tilde{x}^{(i)}. \]

Therefore,

\[ \frac{dt}{d\tilde{\beta}} = \sum_{i=1}^{n} \frac{y_i - p_i}{p_i(1-p_i)} \cdot p_i(1-p_i) \tilde{x}^{(i)}. \]
\[= \sum_{i=1}^{n} (y_i - p_i) \tilde{x}^{(i)}\]
\[= \left[ \begin{array}{c} \tilde{x}^{(1)} \\ \vdots \\ \tilde{x}^{(n)} \end{array} \right] \left[ \begin{array}{c} y_1 - p_1 \\ \vdots \\ y_n - p_n \end{array} \right] = X^T (\bar{y} - \bar{p}). \quad (2)\]

**Second Derivative (Hessian):**

\[\frac{d}{d\beta} \left( \frac{d}{d\beta} (X^T (\bar{y} - \bar{p})) \right)\]
\[= - \frac{d}{d\beta} (X^T \bar{p})\]
\[= - \frac{d}{d\beta} (\sum_{i=1}^{n} p_i \tilde{x}^{(i)})\]
\[= - \sum_{i=1}^{n} \frac{d}{d\beta} (p_i \tilde{x}^{(i)})\]
\[= - \sum_{i=1}^{n} \frac{d}{d\beta} \left( \begin{array}{c} x_0^{(i)} p_i \\ \vdots \\ x_m^{(i)} p_i \end{array} \right)\]
\[= - \sum_{i=1}^{n} \left[ \begin{array}{c} x_0^{(i)} \\ \vdots \\ x_m^{(i)} \end{array} \right] \frac{d}{d\beta} \left( \begin{array}{c} p_i \\ \vdots \\ p_i \end{array} \right)\]
\[= - \sum_{i=1}^{n} \left[ \begin{array}{c} x_0^{(i)} \\ \vdots \\ x_m^{(i)} \end{array} \right] \left( \begin{array}{c} \frac{\partial p_i}{\partial \beta_0} \\ \vdots \\ \frac{\partial p_i}{\partial \beta_m} \end{array} \right)\]
\[= - \sum_{i=1}^{n} \left[ \begin{array}{c} x_0^{(i)} \\ \vdots \\ x_m^{(i)} \end{array} \right] \frac{d}{d\beta} \left( \begin{array}{c} p_i \\ \vdots \\ p_i \end{array} \right)^T\]
\[= - \sum_{i=1}^{n} \tilde{x}^{(i)} (p_i (1 - p_i) \tilde{x}^{(i)})^T\]
\[= - \sum_{i=1}^{n} p_i (1 - p_i) \tilde{x}^{(i)} \tilde{x}^{(i)^T}.\]
\[= - \left[ \begin{array}{c} \tilde{x}^{(1)} \\ \vdots \\ \tilde{x}^{(n)} \end{array} \right] \left[ \begin{array}{cccc} p_1 (1 - p_1) & \cdots & 0 & \vdots \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & p_n (1 - p_n) & \vdots \end{array} \right] \left[ \begin{array}{c} \tilde{x}^{(1)} \\ \vdots \\ \tilde{x}^{(n)} \end{array} \right]^T\]
\[= -X^T \Lambda X, \quad (3) \quad \Rightarrow \text{concave!}\]

where \( \Lambda = diag \{ p_1 (1 - p_1), \ldots, p_n (1 - p_n) \} \).
Newton’s Method:

\[
\mathbf{X}^T \left( \mathbf{y} - \mathbf{p}^{(k-1)} \right) + \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} \left( \mathbf{p}^{(k)} - \mathbf{p}^{(k-1)} \right) = 0 \quad \text{(by letting the first derivative of the quadratic approximation at } \mathbf{p}^{(k-1)} \text{ be zero)}
\]

\[
\mathbf{X}^T \left( \mathbf{y} - \mathbf{p}^{(k-1)} \right) = \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} \left( \mathbf{p}^{(k)} - \mathbf{p}^{(k-1)} \right)
\]

\[
\overrightarrow{\mathbf{p}}^{(k)} = \mathbf{p}^{(k-1)} - \mathbf{p}^{(k-1)} = \left( \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} \right)^{-1} \mathbf{X}^T \left( \mathbf{y} - \mathbf{p}^{(k-1)} \right)
\]

\[
\overrightarrow{\mathbf{p}}^{(k)} = \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X}^{-1} \mathbf{X}^T \left( \mathbf{y} - \mathbf{p}^{(k-1)} \right) + \overrightarrow{\mathbf{p}}^{(k-1)}.
\]  \hspace{1cm} (4)

Note that both \(\mathbf{p}^{(k-1)}\) and \(\mathbf{A}^{(k-1)}\) are calculated using \(\overrightarrow{\mathbf{p}}^{(k-1)}\).

Discussion 1: Let’s push Eqn. (4) further.

\[
\overrightarrow{\mathbf{p}}^{(k)} = \left( \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} \right)^{-1} \mathbf{X}^T \left( \mathbf{y} - \mathbf{p}^{(k-1)} \right) + \left( \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} \overrightarrow{\mathbf{p}}^{(k-1)}
\]

\[
= \left( \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} \right)^{-1} \mathbf{X}^T \left( \mathbf{y} - \mathbf{p}^{(k-1)} \right) + \mathbf{A}^{(k-1)} \mathbf{X} \overrightarrow{\mathbf{p}}^{(k-1)}
\]

\[
= \left( \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{A}^{(k-1)} \left( \mathbf{A}^{(k-1)} \right)^{-1} \left( \mathbf{y} - \mathbf{p}^{(k-1)} \right) + \mathbf{X} \overrightarrow{\mathbf{p}}^{(k-1)}
\]

\[
= \left( \mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{A}^{(k-1)} \left( \mathbf{z}^{(k-1)} \right).
\]  \hspace{1cm} (5)

The Eqn (5) says that

\[
\overrightarrow{\mathbf{p}}^{(k)} = \arg\min_{\overrightarrow{\mathbf{p}}} \left( \mathbf{z}^{(k-1)} - \mathbf{X} \overrightarrow{\mathbf{p}} \right)^T \mathbf{A}^{(k-1)} \left( \mathbf{z}^{(k-1)} - \mathbf{X} \overrightarrow{\mathbf{p}} \right),
\]

that is, iteratively, \(\overrightarrow{\mathbf{p}}^{(k)}\) is the solution to a weighted least square minimization problem.

Discussion 2: Let’s check what effect there will be when \(\mathbf{X}\) has linearly dependent columns (linealy correlated variables).

The main concern is the Hessain matrix \(\mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X}\).

Note that \(\mathbf{X}^T \mathbf{A}^{(k-1)} \mathbf{X} = \mathbf{X}^T \mathbf{X} \Sigma^{(k-1)} \left( \Sigma^{(k-1)} \right)^T \mathbf{X} = \Phi^{(k-1)^T} \Phi^{(k-1)},\) where \(\Sigma^{(k-1)} = \left( \mathbf{A}^{(k-1)} \right)^{1/2}, \Phi^{(k-1)} = \left( \Sigma^{(k-1)} \right)^T \mathbf{X}\)

Consider the matrix’s rank, we have
\[ \text{rank}(X^T \Lambda^{(k-1)} X) = \text{rank}(\Phi^{(k-1)^T} \Phi^{(k-1)}) \]
\[ = \text{rank}(\Phi^{(k-1)} \Phi^{(k-1)^T}) \]
\[ = \text{rank} \left( (\Sigma^{(k-1)})^T X X^T \Sigma^{(k-1)} \right) \]
\[ = \text{rank}(XX^T) \]
\[ = \text{rank}(X^T X), \]

Which implies that if \(X^T X\) is rank deficient due to the co-linearity of the variables, then \(X^T \Lambda^{(k-1)} X\) is also rank deficient, which is not invertible. So, co-linearity of the variables not only affects the inference in linear regression, it also affects the logistic regression.

R Example:

```r
> ## Read and process the data:
> inputfile_path = 'G:/AnalyticsCompetitions/others/data/';
> inputfile_name01 = 'binary.csv';
> inputfile01 = paste(inputfile_path, inputfile_name01, sep='');
> trainData = read.csv(inputfile01,colClasses = "character");
> featureColumns = names(trainData)[2:4];
> trainData[, featureColumns] <- apply(trainData[, featureColumns], 2, as.numeric);
> trainData$admit = as.numeric(trainData$admit);
> print(summary(trainData));

admit         gre         gpa         rank
Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000
1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
Median :0.0000   Median :580.0   Median :3.395   Median :2.000
Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485
3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000

> # Build dummy variables for the rank levels 2,3,4
> trainData[, 'rank2'] <- 0;
> trainData[, 'rank3'] <- 0;
> trainData[, 'rank4'] <- 0;
> trainData$rank2[trainData$rank == 2] <- 1;
> trainData$rank3[trainData$rank == 3] <- 1;
> trainData$rank4[trainData$rank == 4] <- 1;
> featureColumns = names(trainData)[c(2:3,5:7)];
> print(trainData[1:20,]);
> print(summary(trainData));

```

```
## Build simple logistic regression model using just GRE

```r
eqn = paste('admit', '~', 'gre', sep='');
mylogistic <- glm(eqn, data = trainData, family='binomial');
print(summary(mylogistic));
```

```
Call:
  glm(formula = eqn, family = "binomial", data = trainData)

Deviance Residuals:
  Min       1Q   Median       3Q      Max
-1.1623  -0.9052  -0.7547   1.3486   1.9879

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)  
(Intercept)             -2.901344   0.606038  -4.787 1.69e-06 ***  
gre                      0.003582   0.000986   3.633  0.00028 ***

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 499.98  on 399  degrees of freedom  
Residual deviance: 486.06  on 398  degrees of freedom  
AIC: 490.06
```

## Build simple logistic regression model using just GPA

```r
eqn = paste('admit', '~', 'gpa', sep='');
mylogistic <- glm(eqn, data = trainData, family='binomial');
print(summary(mylogistic));
```

```
Call:
  glm(formula = eqn, family = "binomial", data = trainData)

Deviance Residuals:
  Min       1Q   Median       3Q      Max
-1.1131  -0.8874  -0.7566   1.3305   1.9824

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)  
(Intercept)             -4.35768     1.0353   -4.209 2.57e-05 ***
gpa                      1.05109     0.2989    3.517  0.000437 ***

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 499.98  on 399  degrees of freedom  
Residual deviance: 486.06  on 398  degrees of freedom  
AIC: 490.06
```
## Build simple logistic regression model using all the feature variables

```r
print("Use dummy variables for the rank levels 2,3,4");

## Use factor function to specify the variable rank as categorical variable

```
> # Confidence intervals
> confint(mylogistic);
# Waiting for profiling to be done...
2.5 %       97.5 %
(Intercept) -6.2716202334 -1.792547080
gre          0.0001375921  0.004435874
gpa          0.1602959439  1.464142727
rank2       -1.3008888002 -0.056745722
rank3       -2.0276713127 -0.670372346
rank4       -2.4000265384 -0.753542605

> # Look at the correlation matrix of all the feature variables
> print(cor(trainData[,featureColumns]));
gre         gpa       rank2       rank3       rank4
gre    1.00000000  0.38426588  0.05620179 -0.07320022 -0.06823548
gpa    0.38426588  1.00000000 -0.05786732  0.07448976 -0.08442825
rank2  0.05620179 -0.05786732  1.00000000 -0.51283704 -0.34930442
rank3 -0.07320022  0.07448976 -0.51283704  1.00000000 -0.29539686
rank4 -0.06823548 -0.08442825 -0.34930442 -0.29539686  1.00000000

##

> # Input: x - feature matrix
> n - number of rows (data points)
> m - number of columns (features)
> y - target column
> Output: b - regressional parameters (m+1 parameters including the intercept)

newtonLogistic <- function(x,n,m,y)
{
b = matrix(rep(0,m), m, 1);
b_old = b;

print('Newton Iterations:');
print(as.numeric(b));
for (k in 1:10)
{
f = as.numeric(x %*% b_old); # n-by-1 matrix to a R-vector
f = exp(f);
p = f / (1+f);

lam = diag(p*(1-p)); # Diagonal matrix
p = matrix(p, n, 1); # Column vector
b = solve( t(x) %*% lam %*% x) %*% t(x) %*% (y-p) + b_old;
print(as.numeric(b));
if (sum(abs(b-b_old)) < 1.0e-4)
  break;
  b_old = b;
}
return (b);
}

##

> # Perform Newton's iterations for infereing the model parameters
> n = dim(trainData)[1];
> x = as.matrix(trainData[,featureColumns]);
> x = cbind(matrix(rep(1,n), n, 1), x); # Feature matrix with first column as 1
> y = matrix(trainData$admit, n, 1); # Target (binary) vector
> m = length(featureColumns) + 1; # Number of columns in x
> b = newtonLogistic(x,n,m,y);
[1] "Newton Iterations:"
 [1] 0 0 0 0 0 0
 [1] -3.035640842  0.001718288  0.622140099 -0.649461398 -1.162281919 -1.292105462
 [1] -3.913973614  0.002218669  0.789983846 -0.674212541 -1.327645335 -1.527241669
 [1] -3.989459770  0.002264096  0.803944664 -0.675335206 -1.340127398 -1.551235371
### Parameters inferred by Newton method:

<table>
<thead>
<tr>
<th>gre</th>
<th>0.002264426</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpa</td>
<td>0.804037549</td>
</tr>
<tr>
<td>rank2</td>
<td>-0.675442928</td>
</tr>
<tr>
<td>rank3</td>
<td>-1.340203916</td>
</tr>
<tr>
<td>rank4</td>
<td>-1.551463677</td>
</tr>
</tbody>
</table>

Note: the interpretation of the inferred model includes:

1. For every one unit change in `gre`, the log odds of admission (versus non-admission) increases by 0.002.
2. For a one unit increase in `gpa`, the log odds of being admitted to graduate school increases by 0.804.
3. The indicator variables for `rank` have a slightly different interpretation. For example, having attended an undergraduate institution with `rank` of 2, versus an institution with a `rank` of 1, changes the log odds of admission by -0.675.

For 3), consider

\[
r = \log \left( \frac{P(\text{admit} = 1)}{1 - P(\text{admit} = 1)} \right) = b_0 + b_1 \cdot \text{gre} + b_2 \cdot \text{gpa} + b_3 \cdot \text{rank2} + b_4 \cdot \text{rank3} + b_5 \cdot \text{rank4}
\]

\[
b_0 = E(r | \text{gre} = 0, \text{gpa} = 0, \text{rank2} = 0, \text{rank3} = 0, \text{rank4} = 0)
\]

\[
b_0 + b_3 = E(r | \text{gre} = 0, \text{gpa} = 0, \text{rank2} = 1, \text{rank3} = 0, \text{rank4} = 0)
\]

\[
b_3 = E(r | \text{gre} = 0, \text{gpa} = 0, \text{rank2} = 1, \text{rank3} = 0, \text{rank4} = 0) - E(r | \text{gre} = 0, \text{gpa} = 0, \text{rank2} = 0, \text{rank3} = 0, \text{rank4} = 0)
\]

(rank2 = 0, rank3 = 0, rank4 = 0) corresponds to the rank value 1

(rank2 = 1, rank3 = 0, rank4 = 0) corresponds to the rank value 2

### Add a column that is highly linearly correlated with `gre`

```r
gre_sd = sd(trainData$gre);
trainData[, 'gre1'] <- trainData$gre + rnorm(n, mean = gre_mean, sd = 0.2 * gre_sd);
```

### Build simple logistic regression model using all the feature variables (with the newly added `gre1`)

```r
eqn = paste('admit', '~', paste(featureColumns, collapse = '+'), sep = ' ');
mylogistic <- glm(eqn, data = trainData, family = 'binomial');
print(summary(mylogistic));
```
## Deviance Residuals:

<table>
<thead>
<tr>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.6568</td>
<td>-0.8713</td>
<td>-0.6454</td>
<td>1.0826</td>
<td>2.1229</td>
</tr>
</tbody>
</table>

## Coefficients:

| Estimate | Std. Error | z value | Pr(>|z|) | Significance |
|----------|------------|---------|---------|-------------|
| (Intercept) | -0.096485 | 0.004992 | 1.795 | 0.072717 | significance affected |
| gre | 0.008958 | 0.333718 | 2.410 | 0.015950 | * |
| gpa | 0.804279 | 0.316921 | 2.574 | 0.010001 | ** |
| rank2 | -0.657404 | 0.345867 | -3.825 | 0.000131 | *** |
| rank3 | -1.322875 | 0.345867 | -3.825 | 0.000131 | *** |
| rank4 | -1.530626 | 0.345867 | -3.825 | 0.000131 | *** |

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 499.98 on 399 degrees of freedom
Residual deviance: 456.60 on 393 degrees of freedom
AIC: 470.6

Number of Fisher Scoring iterations: 4

---

2.2. Maximum Accuracy Method (Minimum Error Method)

Given the ground truths $y_1, ..., y_n$ and the predictions $p_1, ..., p_n$, we have that the accuracy (or hit rate) is

$$
\hat{h}(\hat{\beta}) = \frac{1}{n} \sum_{i=1}^{n} [y_i \cdot I(p_i(\hat{\beta}) > 0.5) + (1 - y_i) \cdot I(1 - p_i(\hat{\beta}) > 0.5)].
$$

(6)
where the summation $h(\tilde{\beta})$ is total “number” of hits.

Let’s look at the first derivative

$$\frac{dh}{d\beta} = \sum_{i=1}^{n} \frac{\partial h}{\partial p_i} \cdot \frac{dp_i}{d\beta}$$

$$\frac{dh}{dp_i} = \sum_{i=1}^{n} (2y_i - 1) \cdot \frac{dp_i}{d\beta} = p_i(1 - p_i) \tilde{x}^{(i)}$$

$$\frac{dh}{d\beta} = \sum_{i=1}^{n} (2y_i - 1) \cdot p_i(1 - p_i) \tilde{x}^{(i)} ,$$

which looks not as nice as the Eqn. (2)

Equivalently we can consider the error rate, which is

$$\frac{1}{n} m(\tilde{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \left[ 1 - y_i p_i(\tilde{\beta}) - (1 - y_i) \left( 1 - p_i(\tilde{\beta}) \right) \right],$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left[ y_i \left( 1 - p_i(\tilde{\beta}) \right) + (1 - y_i)p_i(\tilde{\beta}) \right]$$

where the summation $m(\tilde{\beta})$ is total “number” of misses.

The first derivative is

$$\frac{dm}{d\beta} = \sum_{i=1}^{n} (2y_i - 1) \cdot p_i(1 - p_i) \tilde{x}^{(i)}.$$

2.2. Minimum Cost Method

Although the Eqn. (7) and the Eqn. (9) look not as nice as the Eqn. (2), we can easily generalize them to fit more scenarios. One scenario is that $\frac{1}{n} \sum_{i=1}^{n} y_i$ is very small but the cost $C_i$ of misclassifying any point $i$ with $y_i = 1$ is very high.

Look at the following cost metrics

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Model output</th>
<th>Ground truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 - p_i$</td>
<td>$\hat{y}_i = 0$</td>
<td>$y_i = 0$</td>
</tr>
<tr>
<td>$p_i$</td>
<td>$\hat{y}_i = 1$</td>
<td>$y_i = 0$</td>
</tr>
</tbody>
</table>

The minimum error method can then be generalized into the minimum cost method.
\[ c(\bar{\beta}) = \sum_{i=1}^{n} \left[ c_i y_i \left( 1 - p_i(\bar{\beta}) \right) + c_i (1 - y_i) p_i(\bar{\beta}) \right]. \]  

(10)

The first derivative is
\[ \frac{dc}{d\bar{\beta}} = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) p_i(1 - p_i) \bar{x}^{(i)}. \]  

(11)

If \( n \) is very large, an usual method is to resort to the stochastic gradient descent method using the Eqn. (11), that is
\[ \frac{\partial \bar{\beta}^{(k)}}{\partial \bar{\beta}^{(k-1)}} = \bar{\beta}^{(k-1)} - \eta \frac{d m(\bar{\beta}^{(k-1)})}{d\bar{\beta}}. \]

An interesting question is: is \( c(\bar{\beta}) \) convex? This is very tricky question. Let’s look at the second derivative
\[ \frac{d^2}{d\bar{\beta}^2} \frac{dc}{d\bar{\beta}} = \sum_{i=1}^{n} \left( \frac{dc}{d\bar{\beta}} \right)^2 \left( c_i(1 - y_i) - c_i y_i \right) p_i(1 - p_i) \bar{x}^{(i)} \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \frac{d^2}{d\bar{\beta}^2} \left( p_i(1 - p_i) \bar{x}^{(i)} \right) \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \frac{d^2}{d\bar{\beta}^2} \left( \begin{bmatrix} x^{(i)}_0 & \frac{\partial p_i(1-p_i)}{\partial \beta_0} & \ldots & \frac{\partial p_i(1-p_i)}{\partial \beta_m} \\ x^{(i)}_m & \frac{\partial p_i(1-p_i)}{\partial \beta_0} & \ldots & \frac{\partial p_i(1-p_i)}{\partial \beta_m} \end{bmatrix} \right) \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \left[ \begin{bmatrix} x^{(i)}_0 \\ x^{(i)}_m \end{bmatrix} & \frac{\partial p_i(1-p_i)}{\partial \beta_0} & \ldots & \frac{\partial p_i(1-p_i)}{\partial \beta_m} \right] \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \bar{x}^{(i)} \left( \frac{\partial p_i(1-p_i)}{\partial \beta} \right)^T \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \bar{x}^{(i)} \left( 1 - 2 p_i \frac{\partial p_i}{\partial \beta} \right)^T \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \bar{x}^{(i)} \left( 1 - 2 p_i \right) p_i(1 - p_i) \bar{x}^{(i)} \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \bar{x}^{(i)} \left( 1 - 2 p_i \right) p_i(1 - p_i) \bar{x}^{(i)} \bar{x}^{(i)T} \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \bar{x}^{(i)} \left( 1 - 2 p_i \right) p_i(1 - p_i) \bar{x}^{(i)} \bar{x}^{(i)T} \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \bar{x}^{(i)} \left( 1 - 2 p_i \right) p_i(1 - p_i) \bar{x}^{(i)} \bar{x}^{(i)T} \]

\[ = \sum_{i=1}^{n} (c_i(1 - y_i) - c_i y_i) \bar{x}^{(i)} \left( 1 - 2 p_i \right) p_i(1 - p_i) \bar{x}^{(i)} \bar{x}^{(i)T} \]

\[ \text{(12)} \]

Therefore, we can see that when \( c_i = 0 \) and \( p_i > 0.5 \) when \( y_i = 1 \), then the hessian matrix is positive semidefinite. That is true positive rate > 0.5 is an important condition.