HW4 Report

인대영(2022311154)
2022-10-03

1. $Y_i \overset{iid}{\sim} \text{Bernoulli}(\theta_i)$

pdf: $f(y_i;\theta_i) = \theta_i^{y_i}(1-\theta_i)^{1-y_i} = \left(\frac{\theta_i}{1-\theta_i}\right)^{y_i}(1-\theta_i)$

If there are n observations,

$$\prod_{i=1}^{n} f(y_i,\theta_i) = L(\theta) = \prod_{i=1}^{n}\left(\frac{\theta_i}{1-\theta_i}\right)^{y_i}(1-\theta_i) = \exp\left\{\sum y_i \log\left(\frac{\theta_i}{1-\theta_i}\right) + \sum \log(1-\theta_i)\right\}$$

$$\log\left(\frac{\theta_i}{1-\theta_i}\right) = \log\left(e^{\beta_0+\beta_1 X_i}\right) = \beta_0+\beta_1 X_i \quad \cdots\cdots ①$$

$$\theta_i = \frac{e^{\beta_0+\beta_1 X_i}}{1+e^{\beta_0+\beta_1 X_i}}, \quad 1-\theta_i = 1-\frac{e^{\beta_0+\beta_1 X_i}}{1+e^{\beta_0+\beta_1 X_i}} = \frac{1}{1+e^{\beta_0+\beta_1 X_i}}$$

$$\therefore \log(1-\theta_i) = \log\left(\frac{1}{1+e^{\beta_0+\beta_1 X_i}}\right) \quad \cdots\cdots ②$$

By ① and ②,

$$L(\theta) = L(\beta_0,\beta_1) = \exp\left\{\sum y_i(\beta_0+\beta_1 X_i) + \sum \log\left(\frac{1}{1+\exp(\beta_0+\beta_1 X_i)}\right)\right\}$$

$$\log L(\theta) = \sum y_i(\beta_0+\beta_1 X_i) + \sum \log\left(\frac{1}{1+\exp(\beta_0+\beta_1 X_i)}\right)$$

$$\frac{\partial \log L(\theta)}{\partial \beta_0} = \sum y_i + \sum\left(\frac{1}{1+\exp(\beta_0+\beta_1 X_i)}\right)' \cdot (1+\exp(\beta_0+\beta_1 X_i))$$

$$= \sum y_i + \sum \frac{-\exp(\beta_0+\beta_1 X_i)(1+\exp(\beta_0+\beta_1 X_i))}{(1+\exp(\beta_0+\beta_1 X_i))^2}$$

$$= \sum y_i - \sum \frac{\exp(\beta_0+\beta_1 X_i)}{1+\exp(\beta_0+\beta_1 X_i)} = 0$$

$$\therefore \sum \frac{1}{1+\exp(-(\beta_0+\beta_1 X_i))} = \sum y_i$$

$$\frac{\partial \log L(\theta)}{\partial \beta_1} = \sum y_i X_i + \sum\left(\frac{1}{1+\exp(\beta_0+\beta_1 X_i)}\right)' \cdot (1+\exp(\beta_0+\beta_1 X_i))$$

$$= \sum y_i X_i + \sum \frac{-X_i\exp(\beta_0+\beta_1 X_i)(1+\exp(\beta_0+\beta_1 X_i))}{(1+\exp(\beta_0+\beta_1 X_i))^2}$$

$$= \sum X_i y_i - \sum \frac{X_i\exp(\beta_0+\beta_1 X_i)}{1+\exp(\beta_0+\beta_1 X_i)} = 0$$

$$\therefore \sum \frac{X_i}{1+\exp(-(\beta_0+\beta_1 X_i))} = \sum X_i y_i$$

2. $h_\theta(z) = \dfrac{e^z}{1+e^z}$ where $z = \theta' x$

$$1 - h_\theta(z) = \frac{1}{1+e^z}$$

$$\frac{dh_\theta(z)}{dz} = \frac{e^z(1+e^z) - e^z \cdot e^z}{(1+e^z)^2}$$

$$= \frac{e^z}{(1+e^z)^2}$$

$$= h_\theta(z)\,(1 - h_\theta(z))$$

3. $J(\theta) = -\dfrac{1}{m}\displaystyle\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\left\{\log(h_\theta(x^{(i)})) - \log(1-h_\theta(x^{(i)}))\right\} + \log(1-h_\theta(x^{(i)}))\right]$$

$\dfrac{\partial h_\theta(x^{(i)})}{\partial(\theta_j)} = \sigma(z)(1-\sigma(z)) \times \dfrac{\partial z}{\partial \theta_j}$ where $\sigma(x) = \dfrac{1}{1+e^{-x}}$

$\dfrac{\partial z}{\partial \theta_j} = \dfrac{\partial(\theta' x)}{\partial \theta_j} = x_j$ $\left(\because \text{ for } k \neq j,\ \dfrac{\partial \theta_k}{\theta_j} = 0\right)$

$$\therefore \frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\left\{1 - h_\theta(x^{(i)}) + h_\theta(x^{(i)})\right\} - h_\theta(x^{(i)})\right]x_j^{(i)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

4. We first prompt the user to enter the data set. We then implement the gradient descent algorithm. First, we set the initial coefficients by random. Then, we set the learning rate as 0.5 and the number of epochs as 1000. Using the sigmoid function, we update the coefficients using the equation below:

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

Logistic Regression Analysis via Gradient Descent Algorithm

In [6]:
```
#initial coefficients
np.random.seed(0)
b = pd.Series(np.random.uniform(low=-1, high=1, size=3))
b
```

Out[6]:
```
0    0.097627
1    0.430379
2    0.205527
dtype: float64
```

In [7]:
```
alpha = 0.5 #learning rate
epochs = 1000 #number of iterations
n = len(x) #size
```

The implemented algorithm along with the resulting coefficients from 'sample1.csv' data are shown below. We also implement logistic regression using 'statsmodels' package.

```
In [10]: def sigmoid(x):
             return 1/(1+np.exp(-x))

In [11]: for i in range(epochs):
             b[0]=b[0]-alpha*((sigmoid(z.dot(b.T))-y)*z[0]).mean()
             b[1]=b[1]-alpha*((sigmoid(z.dot(b.T))-y)*z[1]).mean()
             b[2]=b[2]-alpha*((sigmoid(z.dot(b.T))-y)*z[2]).mean()

In [12]: #result
         b=np.round(b,4)
         b

Out[12]: 0    0.1682
         1    2.8210
         2    2.8184
         dtype: float64
```

Statsmodels

```
In [13]: import statsmodels.formula.api as sm

In [14]: log_reg = sm.logit('y ~ x1+x2', data = data).fit()

         Optimization terminated successfully.
                  Current function value: 0.283800
                  Iterations 8
```

Finally, we write the output file to show the comparison between both methods. We can observe below that the logistic regression coefficients estimated by each method are quite similar.

Write File

```
In [17]: f = open("HW4_output.txt",'w')
         text = "Coefficients by Gradient Descent Method\n-------------\nConstant: "+str(b[0])+"\n"
         for i in range(len(x.columns)):
             text += "Beta"+str(i+1)+": "+str(b[i+1])+"\n"
         text += "\nCoefficients by Statmodels\n-------------\nConstant: "+str(log_reg.params[0])+"\n"
         for i in range(len(x.columns)):
             text += "Beta"+str(i+1)+": "+str(log_reg.params[i+1])+"\n"
         f.write(text)
         f.close()
```

HW4_output.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
Coefficients by Gradient Descent Method
-------------
Constant: 0.1682
Beta1: 2.821
Beta2: 2.8184

Coefficients by Statmodels
-------------
Constant: 0.1683
Beta1: 2.8218
Beta2: 2.8193
```