Logistic regression

Using python.

Iván Andrés Trujilllo Abella

ivantrujillo1229@gmail.com

Logistic regression

$$
\begin{aligned}
P(y_i = 1) =& P_i \\
P(y_i = 0) =& 1 - P_i
\end{aligned} \tag{1}
$$

...

$$P_i = \frac{1}{1 + e^{-\Theta^T x}} \tag{2}$$

MLE

In a sample of n observations.

$$
\begin{aligned}
f(y_1, y_2, ..., y_n) &= \prod_{i=1}^{n} f(y_i) \\
&= \prod_{i=1}^{n} P(y_i) \\
&= \prod_{i=1}^{n} P_i^{y_i}(1 - P_i)^{1-y_i}
\end{aligned} \tag{3}
$$

$$f(y_1, y_2, ..., y_n) = \prod_{i=1}^{n} P_i^{y_i} (1 - P_i)^{1-y_i} \tag{4}$$

With logarithm is more easy, in a convenient we write $P_i = \hat{y}_i$.

$$\ln f(y_1, y_2, ..., y_n) = \sum_{i=1}^{n} y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i) \tag{5}$$

Related with the cross binary entropy.

Cost function

Now we need minimize the function, then could be apply

$$-\frac{1}{n}\sum_{i=1}^{n} y_i \ln(\hat{y}_i) + (1 - y_i)\ln(1 - \hat{y}_i) \tag{6}$$

In dataset y_i and x_i are fixed quantities, we need find the values of Θ that minimize the expression.

Cost Function

$$\frac{\partial J(\Theta)}{\partial \Theta_j} = \sum_{i=1}^{n} \frac{\partial J(\Theta)}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial \Theta_j} \tag{7}$$

Dertivatives

In one sample the cost for $i - th$ pattern.

$$-\frac{1}{n}\left(y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i)\right) \tag{8}$$

Proof

$$\frac{\partial J(\Theta)}{\partial \hat{y}_i} = -\frac{1}{n}\left(\frac{y_i}{\hat{y}_i} - \frac{(1-y_i)}{(1-\hat{y}_i)}\right) \tag{9}$$

Sigmoid derivative

...

$$\sigma(x)' = \sigma(x)(1 - \sigma(x)) \tag{10}$$

$$\tag{11}$$

Using quotien and chain rule we have

$$\sigma(x)' = \frac{e^{-x}}{(1 + e^{-x})^2} \tag{12}$$

$$\frac{1 + e^{-x} - 1}{(1 + e^{-x})^2} \tag{13}$$

$$\frac{1}{1 + e^{-x}} - \frac{1}{(1 + e^{-x})^2} \tag{14}$$

$$\frac{1}{1 + e^{-x}}(1 - \frac{1}{1 + e^{-x}}) \tag{15}$$

$$\sigma(x)(1 - \sigma(x)) \tag{16}$$

$$\frac{\partial \hat{y}_i}{\partial \Theta_j} = \sigma(x)(1 - \sigma(x))x_j \tag{17}$$

$$\frac{\partial J(\Theta)}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial \Theta_j} = -\frac{1}{n}\left(\frac{y_i}{\hat{y}_i} - \frac{(1 - y_i)}{(1 - \hat{y}_i)}\right)\hat{y}_i(1 - \hat{y}_i)x_{ij} \tag{18}$$

$$-\frac{1}{n}\left(y_i(1-\hat{y}_i)-\hat{y}_i(1-y_i)\right)x_j \tag{19}$$

$$-\frac{1}{n}(y_i-\hat{y}_i)x_j \tag{20}$$

$$\frac{1}{n}(\hat{y}_i-y_i)x_j \tag{21}$$

$$
\begin{aligned}
\frac{\partial J(\Theta)}{\partial \Theta_j} &= \sum_{i=1}^{n} \frac{\partial J(\Theta)}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial \Theta_j} \\
&= \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) x_{ij}
\end{aligned} \tag{22}
$$

Gradient matricial way

...

$$\nabla J(\vec{\Theta}) = \frac{1}{N}\boldsymbol{X}^T(\vec{\hat{y}} - \vec{y}) \\ = \frac{1}{N}\boldsymbol{X}^T(\sigma(\boldsymbol{X}\vec{\Theta}) - \vec{y}) \tag{23}$$

...

Implementation logistic from scratch(Click here)

python implementation

Statsmodels

```
formula = 'died ~ studytime + C(drug) ' # logit died studytime
    i.drug
model = smf.logit(formula= formula, data=df)
results = model.fit()
print(results.summary())
coefs = pd.DataFrame({
    'coef': results.params.values,
    'odds ratio': np.exp(results.params.values),
    'name': results.params.index
})
coefs
```

Contingency table

		Diagnose	
		Disease	No-Disease
Risk Factor	Smoke	a	b
	Not Smoke	c	d

From this table we can calculate the odds, increase the probability of a diagnose smoke, or not?

$$odds_{smoke} = \frac{a}{b} = \frac{P(D \mid smoke)}{P(ND \mid smoke)} \tag{24}$$

e

Odds ratio

We can calculate the odds ratio as:

$$OR = \frac{odds_{smoke}}{odds_{NoSmoke}} \tag{25}$$

according to the information of the table, we can calculate the odds ratio as $OR = \frac{\frac{a}{b}}{\frac{c}{d}}$.
How we can interpret this OR?.

From odds to probability

$$\frac{P(A)}{P(A^c)} + 1 = odds + 1 \tag{26}$$

$$\frac{P(A) + P(A^c)}{P(A^c)} = odds + 1 \tag{27}$$

$$\frac{1}{P(A^c)} = odds + 1 \tag{28}$$

$$\frac{odds}{odds + 1} = P(A) \tag{29}$$

Odds Ratio

```
testing = pd.DataFrame({
'smoke':[1,1,0,1,1,0,1,1,0,1,0,0,1,1,1,0,0,1,1,1],
'diagnose':[1,0,1,1,0,0,1,1,0,0,0,1,1,0,0,0,0,0,1,0]})
pd.crosstab(testing['smoke'], testing['diagnose'])
```

```
# Odds ratio
oddsnum = 6/7
oddsdem = 2/5
print(oddsnum, oddsdem, oddsnum/oddsdem)
```

Logistic regression

Using statsmodels

```
import statsmodels.formula.api as smf
model_logit = smf.logit(formula="diagnose ~ smoke", data=testing)
res = model_logit.fit()
# How we can convert to dummy variable??
res.summary()
np.exp(res.params) # This give us the odds ratio getted
    previously.
```

ORs lessert than 1

$$(1 - OR) * 100 \tag{30}$$