

GLMS (General Linear Models)

Contents

1. Linear Regression
2. Logistic Regression
 - Reasoning
 - Method 1
 - Method 2
 - Loss Function
 - Walkthrough

Linear Regression

- With linear regression, each coefficient can be interpreted as the the change in the response for a one unit change in the predictor, holding all else constant.
- When there are interaction terms in the equation, such as

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$$

you can rewrite the equation to be

$$y = \beta_0 + \tilde{\beta}_1 X_1 + \beta_2 X_2 \quad \text{where} \quad \tilde{\beta}_1 = \beta_1 + \beta_3 X_2$$

where β_3 can be interpreted as the change in the **effectiveness of X_1 on y for a one unit change in X_2** .

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import Pipeline

mod = LinearRegression()
mod.fit(x_train, y_train)
y_hat = mod.predict(x_test)

rmse = np.sqrt(mean_squared_error(y_test, y_hat))

lin_reg_pipe = Pipeline([
    ('column', ColumnSelector(name='column')),
    ('regression', LinearRegression())
])

y_hat = lin_reg_pipe.predict(x_test)
rmse = np.sqrt(mean_squared_error(y_test, y_hat))
```

Logistic Regression

Logistic regression is used in a classification setting, namely when the response is dichotomous (binary). When this is the case, one seeks to model the probability of an observation being a “success” (or “of interest”, encoded as a 1) or a failure (encoded as a 0).

Logistic regression is similar to linear regression in that it still assumes the relationship between the attributes \mathbf{X} and the response y is linear, however with one small caveat; logistic regression assumes that the log odds of a success are linear in \mathbf{X} .

Reasoning

Since the response in logistic regression is dichotomous, and one would like to model the probability of an observation being a success, a good place to start would be:

$$P(Y = 1|X = x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

$$P(Y = 1|X = x) = \beta_0 + \sum_{j=1}^P \beta_j x_j$$

However, this opens up the possibility of the output $P(Y = 1|X = x)$ taking on a value greater than 1 or less than 0, which violates probabilities. To remedy this, one makes use of the *logistic function* (hence the name), which restricts all output to be within the bounds of 0 and 1, and therefore a valid probability (note that there are two ways to write the logistic function, but they are equivalent and will lead to the same result):

$$\frac{1}{1 + e^{-x}} = f(x) = \frac{e^x}{e^x + 1}$$

So, knowing that one would like the output to be a valid probability, we simply substitute $P(Y = 1|X = x)$ into the above equation in place of $f(x)$, where x is equal to the linear model $\beta_0 + \sum_{j=1}^P \beta_j x_j$:

$$\frac{1}{1 + e^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} = P(Y = 1|X = x) = \frac{e^{\beta_0 + \sum_{j=1}^P \beta_j x_j}}{e^{\beta_0 + \sum_{j=1}^P \beta_j x_j} + 1}$$

Quick Aside on Probability and Odds

In order to illustrate how everything comes together, it is important to note the relationship between the *probability* of an event occurring and the *odds* of an event occurring, given below:

$$Odds = \frac{P}{1 - P} \quad \text{and} \quad P = \frac{Odds}{Odds + 1}$$

Reasoning (continued)

Method 1

Now knowing the relationship between probabilities and odds, is clear how the right hand side of the two equations above are related, reproduced below:

$$\frac{Odds}{Odds + 1} = P(Y = 1|X = x) = \frac{e^{\beta_0 + \sum_{j=1}^P \beta_j x_j}}{e^{\beta_0 + \sum_{j=1}^P \beta_j x_j} + 1}$$

Therefore, we can express the odds of a success as:

$$Odds = e^{\beta_0 + \sum_{j=1}^P \beta_j x_j}$$

Using the natural logarithm on both sides of the equation, it is evident that **the log odds of a success are linear in the attributes X**.

$$\log(Odds) = \log(\epsilon^{\beta_0 + \sum_{j=1}^P \beta_j x_j})$$

$$\log(Odds) = \beta_0 + \sum_{j=1}^P \beta_j x_j$$

Method 2

The other way to reconcile logistic regression is to look at the other expression of logistic regression, reproduced below:

$$P(Y = 1|X = x) = \frac{1}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}}$$

Knowing the above, one can substitute the right side of the equation into the odds ratio and solve for $\beta_0 + \sum_{j=1}^P \beta_j x_j$:

$$\text{Given } Odds = \frac{P}{1-P} \quad \text{and} \quad P(Y = 1|X = x) = \frac{1}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}}$$

Then:

$$\begin{aligned} Odds &= \frac{\left(\frac{1}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} \right)}{1 - \left(\frac{1}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} \right)} \\ &= \frac{\left(\frac{1}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} \right)}{\left(\frac{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} \right) - \left(\frac{1}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} \right)} \\ &= \frac{\left(\frac{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} \right)}{\left(\frac{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} \right)^2 - \left(\frac{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} \right)} \\ &= \frac{1}{1 + \epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)} - 1} \\ &= \frac{1}{\epsilon^{-(\beta_0 + \sum_{j=1}^P \beta_j x_j)}} \\ &= \epsilon^{(\beta_0 + \sum_{j=1}^P \beta_j x_j)} \\ \text{Log}(Odds) &= \text{Log}(\epsilon^{(\beta_0 + \sum_{j=1}^P \beta_j x_j)}) \end{aligned}$$

And, once again, the conclusion is made that the log odds of a success are linear in the attributes \mathbf{X} .

$$\text{Log}(\text{Odds}) = \beta_0 + \sum_{j=1}^P \beta_j x_j$$

Loss Function

The loss function (or cost function) for logistic regression, called the log loss, is given below:

$$\text{Log Loss} = - \left(\frac{\sum_{i=1}^N (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))}{N} \right)$$

Walkthrough

The key to understanding the loss function is to look at each term in the numerator, reproduced below:

$$\text{Log loss} = y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

Since logistic regression is used in a *binary* setting, the response \mathbf{y} will be a vector of 1's and 0's. Therefore, y_i will be either a 1 or a 0. With that in mind, it is clear that the first part of both terms in the above equation, y_i and $(1 - y_i)$, serve the same functionality that an indicator function serves; if y_i is a 1, the second term in the equation will zero out, and conversely, if y_i is a 0, the first term in the equation will zero out:

$$\begin{aligned} \text{If } y_i = 1, \quad \text{Log loss} & \begin{cases} = (1) \log(\hat{y}_i) + (1 - 1) \log(1 - \hat{y}_i) \\ = (1) \log(\hat{y}_i) \\ = \log(\hat{y}_i) \end{cases} \\ \text{If } y_i = 0, \quad \text{Log loss} & \begin{cases} = (0) \log(\hat{y}_i) + (1 - 0) \log(1 - \hat{y}_i) \\ = (1) \log(1 - \hat{y}_i) \\ = \log(1 - \hat{y}_i) \end{cases} \end{aligned}$$

Now, the output of logistic regression is a **probability**, which is to say \hat{y}_i will be the probability that observation i is a 1. Therefore, the loss function will always be taking the log of a probability. Referencing the formulas above, if $y_i = 1$, $\text{Log loss} = \log(\hat{y}_i)$ and if $y_i = 0$, $\text{Log loss} = \log(1 - \hat{y}_i)$.

Since all models seek to **minimize the loss**, ideally the log loss is as close to 0 as possible. As it turns out, $\log(1) = 0$. So, when $y_i = 1$, $\text{Log loss} = \log(1)$ will minimize the loss, and if $y_i = 0$, $\text{Log loss} = \log(1 - 0)$ will minimize the loss. **It is clear that this setup minimizes the error when \hat{y}_i is as close to y_i as possible, averaged over all observations.**

Interpretation

- Each coefficient of logistic regression is interpreted as the change in the **log of the odds** of a success for a one unit change in the predictor, holding all else constant. To translate this to a probability of a success:

1. Plug each predictor value of a test observation into the linear equation, using the respective coefficients and the intercept; this returns the **log odds** of a success for that test observation.

$$\log(\text{Odds}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \dots$$

2. Exponentiate this value; this removes the log and **returns the odds of a success** for the test observation.

$$\text{Odds} = e^{\log(\text{Odds})}$$

3. Divide the odds by one plus the odds (odds / (1 + odds)) and you have the **probability that the test observation is a success**.

$$\text{Prob}(y_i = 1) = \frac{\text{Odds}}{1 + \text{Odds}}$$

- Note that your *X* matrix (NOT your *y* vector) must be scaled before being used by sklearn's *LogisticRegression()* class (it utilizes Lasso, which must have scaled data)

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score

mod = LogisticRegression
mod.fit(x_train, y_train)
# return predicted probability of being a success
y_hat_probs = mod.predict_proba(x_test)[:, 1]
# returns the predicted class (success or failure)
y_hat = mod.predict(x_test)

acc = accuracy_score(y_test, y_hat)
prec = precision_score(y_test, y_hat)
recall = recall_score(y_test, y_hat)

def log_odds_to_prob(log_odds):
    """
    Converts the log odds of a success into the probability of a success.

    Input:
        log_odds: (int) The log odds of a success / the output of a logistic regression model
    Output:
        Probability: (float) The probability of a success
    """
    odds = np.exp(log_odds)
    prob = odds / (1 + odds)
    return prob
```

Regularized Regression

- Both Ridge Regression and Lasso Regression both model the relationship between a set **X** of *p* predictors and a quantitative response **y** as a linear model (same as linear regression).
- However, both Ridge and Lasso have an additional term appended onto the loss function of a linear model, the RSS:
 - Ridge: λ multiplied by the summation of the *squares* of all coefficients. (*l2* penalty)

* So, the loss function becomes

$$RSS + \lambda \sum_{j=1}^p \beta_j^2$$

– Lasso: λ multiplied by the summation of the *absolute values* of all the coefficients. (*l1* penalty)

* So, the loss function becomes

$$RSS + \lambda \sum_{j=1}^p |\beta_j|$$

The tuning parameter λ is best selected using cross validation. The practical difference between Lasso and Ridge regression is that Lasso will set some of the coefficients equal to *exactly zero* while ridge regression shrinks the coefficients *towards zero*.

Lasso Regression

```
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error

lassie = Lasso()
lassie.fit(x_train, y_train)
y_hat = lassie.predict(x_test)
print(np.sqrt(mean_squared_error(y_test, y_hat)))
```

Ridge Regression

```
import numpy as np
from sklearn.linear_model import Ridge()
from sklearn.metrics import accuracy_score

ridge = Ridge()
ridge.fit(x_train, y_train)
y_hat = ridge.predict(x_test)
print(np.sqrt(accuracy_score(y_test, y_hat)))
```