C3_W1_Lab_3_logistic_regression_model_interpretation

December 29, 2025

1 Logistic Regression Model Interpretation

Welcome to this exercise! You'll review how to interpret the coefficients in a logistic regression model. - The logistic regression is considered a **Generalized Linear Model**. - In general, you would employ one of these models to interpret the relationship between variables. - The logistic regression can be interpreted in terms of the **odds** and **OR** (Odds Ratio), which you'll learn about here.

1.1 Import Libraries

```
In [1]: # Import libraries that you will use in this notebook
        import numpy as np
        import pandas as pd
        from sklearn.linear_model import LogisticRegression
```

1.2 Load the Data

```
In [2]: # Read in the data
        data = pd.read_csv("data/dummy_data.csv", index_col=0)

        # View a few rows of the data
        data.head()
```

Out[2]:

	sex	age	obstruct	outcome	TRTMT
1	0	57	0	1	True
2	1	68	0	0	False
3	0	72	0	0	True
4	0	66	1	1	True
5	1	69	0	1	False

Here is a description of all the fields:

- sex (binary): 1 if Male, 0 if Female
- age (int): age of patient at the beginning of the study
- obstruct (binary): obstruction of colon by tumor
- outcome (binary): 1 if patient died within 5 years
- TRTMT (binary): if patient was treated

You'll want to pay close attention to the `TRTMT` and `outcome` columns. - `TRTMT`: Whether a treatment was given or not. - `outcome`: To measure the effective of treatment, you'll have the 5-year survival rate. This is stored in the `outcome` variable, which is a binary variable with two possible values. 1 indicates that the patient died, and 0 indicates that the patient did not die during the 5-year period.

1.3 Logistic Regression

The formula for computing a logistic regression has the following form:

$$\sigma(\theta^T x^{(i)}) = \frac{1}{1 + e^{\left(-\theta^T x^{(i)}\right)}},$$

$x^{(i)}$ refers to example 'i' (a particular patient, or generally, a single row in a data table).

$\theta^T x^{(i)} = \sum_j \theta_j x_j^{(i)}$ is the linear combination of the features $x_1^{(i)}$, $x_2^{(i)}$, $x_2^{(i)}$ etc., weighted by the coefficients θ_1, θ_2, θ_3 etc.

So for this example, $\theta^T x^{(i)} = \theta_{TRTMT} x_{TRTMT}^{(i)} + \theta_{AGE} x_{AGE}^{(i)} + \theta_{SEX} x_{SEX}^{(i)}$

Also, σ is the sigmoid function, defined as $\sigma(a) = \frac{1}{1+e^{(-a)}}$ for some variable a. The output of the sigmoid function ranges from 0 to 1, so it's useful. in representing probabilities (whose values also range from 0 to 1).

If $x^{(i)}$ is the input vector and $OUTCOME$ is the target variable, then $\sigma(\theta^T x^{(i)})$ models the probability of death within 5 years.

For example, if the data has three features, $TRTMT$, AGE, and SEX, then the patient's probability of death is estimated by:

$$Prob(OUTCOME = 1) = \sigma(\theta^T x^{(i)}) = \frac{1}{1 + e^{\left(-\theta_{TRTMT} x_{TRTMT}^{(i)} - \theta_{AGE} x_{AGE}^{(i)} - \theta_{SEX} x_{SEX}^{(i)}\right)}}$$

1.4 Fit the Model

Let's separate the data into the target variable and the features and fit a logistic regression to it. Notice that in this case you are **not separating the data into train and test sets** because you're interested in the **interpretation of the model**, not its predictive capabilities.

```
In [4]: # Get the labels
        y = data.outcome

        # Get the features (exclude the label) #elimina columna outcome "y"
        X = data.drop('outcome', axis=1)

        # Fit the logistic regression on the features and labels
        classifier = LogisticRegression(solver='lbfgs').fit(X, y)
```

1.5 Odds

Looking at the underlying equation, you can't interpret the model in the same way as with a regular linear regression. - With a linear regression such as $y = 2x$ if the x increases by 1 unit, then y increases by 2 units. - How do you interpret the coefficient of a logistic regression model now that there is a sigmoid function?

Let's introduce the concept of **odds**, and you'll see how this helps with the interpretation of the logistic regression.

If an outcome is binary (either an event happens or the event doesn't happen): - Let p represent the probability of the event (such as death). - Let $1 - p$ represent the probability that the event doesn't happen (no death). - The odds are the probability of the event divided by 1 minus the probability of the event:

$$\text{odds} = \frac{p}{1-p}$$

Going back to the logistic regression, recall that the sigmoid function σ ranges between 0 and 1, and so it's a useful function for representing a probability. - So, let p, the probability of event, be estimated by $\sigma(\theta^T x^{(i)})$.

The **odds** defined in terms of the probability of an event p are:

$$\text{odds} = \frac{p}{1-p}$$

Substitute $p = \sigma(\theta^T x^{(i)})$ to get:

$$\text{odds} = \frac{\sigma(\theta^T x)}{1 - \sigma(\theta^T x)}$$

Substitute for the definition of sigmoid: $\sigma(\theta^T x^{(i)}) = \frac{1}{1+e^{(-\theta^T x)}}$

$$\text{odds} = \frac{\frac{1}{1+e^{(-\theta^T x)}}}{1 - \frac{1}{1+e^{(-\theta^T x)}}}$$

Multiply top and bottom by $1 + e^{(-\theta^T x)}$ and simplify to get:

$$\text{odds} = \frac{1}{\left(1 + e^{(-\theta^T x)}\right) - (1)}$$

Do some more cleanup to get:

$$\text{odds} = e^{\left(\theta^T x^{(i)}\right)}$$

So what is this saying? - The odds (probability of death divided by probability of not death) can be estimated using the features and their coefficients if you take the dot product of the coefficients and features, then exponentiate that dot product (take e to the power of the dot product).

Since working with the exponential of something isn't necessarily easier to think about, you can take one additional transformation to get rid of the exponential, coming up next.

1.6 Logit

Note that the inverse function of exponentiation is the natural log - $\log(e^a) = a$ - $e^{(\log(a))} = a$

So if you want to "remove" the exponential e, you can apply the natural log function, which we'll write as log. You may have seen natural log written as ln as well, but we'll use log because Python functions usually name natural log functions as `log`.

Note that the log of odds is defined as the **logit** function:

$$\text{logit}(a) = \log\frac{a}{1-a}$$

Apply the log() to the odds:

$$\text{logit} = \log(\text{odds}) = \log\left(\frac{p}{1-p}\right) = \log\left(e^{\left(\theta^T x^{(i)}\right)}\right) = \theta^T x^{(i)}$$

So, what's nice about this? - The right side of this equation is now a weighted sum of the features in $x^{(i)}$, weighted by coefficients in θ^T.

1.7 Interpreting the Coefficient's Effect on the Logit

This is an improvement in the interpretability of your model. - Now you can interpret a single coefficient θ_j in a similar way that you interpret the coefficient in regular linear regression.

For a small example, let's say the coefficient for age is 0.2, patient A has age=40, and the logit for patient A is 3.

$$\text{logit} = \theta_{age} \times x_{age} + \cdots$$

Patient A (now)

$$3 = \theta_{age} \times 40 + \cdots$$

If you increase patient A's age by 1 year, then this increases the logit by 0.2 (which is the coefficient for age).
Patient (A one year older):

$$3 + 0.2 = 0.2 \times (40 + 1) + \cdots$$

The range of possible values for the logit A nice feature of the logit (log odds) is the range of possible values it can have. The logit function can be any real number between $-\infty$ and $+\infty$.

One way to see this is to look at the ranges of values for the sigmoid, the odds, and then logit. - The sigmoid $\sigma(a)$ ranges from 0 to 1 for a variable a. Recall that we're letting $p = \sigma(a)$ - The odds $\frac{p}{1-p}$ can be as small as 0 (when $p = 0$) and as large as $+\infty$ (when $p \to 1$). So the odds range from 0 to $+\infty$. - log(odds) can range from $-\infty$ (when the odds are 0), to $+\infty$ (when the odds approach $+\infty$).
- So the range of the log odds is $-\infty$ to $+\infty$

To check the coefficients of the model, you can use the model's `coef_ attribute`.

```
In [7]: # Get the coefficients (the thetas, or weights for each feature)
        thetas = classifier.coef_
        print(thetas)
```

```
[[-0.21704833  0.0460642   0.37798496 -0.418984  ]]
```

This will return a numpy array containing the coefficient for each feature variable. Let's print it in a nicer way:

```
In [8]: # Print the name of the feature and the coefficient for each feature
        for i in range(len(X.columns)):
            print("Feature {:<9s}: coefficient = {:<10f}".format(X.columns[i], thetas[0, i]))
```