Applied Data Science

Session 20: Logistic Regression

Dr. Soharab Hossain Shaikh

1

Logistic Regression Classifier

2

Logistic Regression

Name is somewhat misleading. It is really a **technique for classification, ~~not regression~~**.

"Regression" comes from fact that we fit a linear model to the feature space.

Involves a more probabilistic view of classification. In regression analysis, logistic regression or logit regression is estimating the parameters of a logistic model.

Models relationship between set of input variables X_i (**categorical or continuous**) and the response variable **Y** (**categorical**).

The dependent variable is binary rather than continuous and it can also be applied to ordered categories (ordinal data).

3

3

The term "Odds"

- Popular in horse races, sports, gambling etc.
- Instead of talking about the *probability* of winning or contacting a disease, people talk about the *odds* of winning or contacting a disease.

4

Logit Function in Logistic Regression

In logistic regression, the dependent variable is a logit, which is the natural log of the odds, that is,

$$odds = \frac{P}{1-P}$$

$$\log(\text{odds}) = \text{logit}(P) = \ln\left(\frac{P}{1-P}\right)$$

5

Math behind Logistic Regression

So a logit is a log of odds.
Odds is a function of P. P is the probability of a 1.
In Logistic Regression, we find : **logit(P) = *a + bX***

$$\ln\left(\frac{P}{1-P}\right) = a + bX$$

$$\frac{P}{1-P} = e^{a+bX}$$

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

6

Different ways of expressing Probabilities

- Consider a two-outcome probability space, where:
  - $p(O_1) = p$
  - $p(O_2) = 1 - p = q$
- Can express probability of O_1 as:

	notation	range equivalents		
standard probability	p	0	0.5	1
odds	p / q	0	1	+ ∞
log odds (logit)	log(p / q)	- ∞	0	+ ∞

7

7

Odds vs. Probability

- What is probability of A– P(A)?
- $Odds\ ratio = \frac{Probability\ of\ event\ occuring}{Probability\ of\ event\ not\ occuring}$
- Odds ratio = $\frac{P}{1-P}$
- Probability= $\frac{Odds\ ratio}{1+Odd\ \ ratio}$

8

Math Behind Logistic Regression

- Predict likelihood or probability
- Predicted value *P* should lie between 0 and 1
- Use Sigmoid function to achieve this

$$Probability, P = \frac{e^z}{1+e^z}$$ $z = \beta_0 + \beta_1 x$

$$Odds\ Ratio = \frac{P}{1-P}$$

Substituting for *P*, $Odds\ Ratio = \frac{P}{1-P} = e^z = e^{(\beta_0+\beta_1 x)}$

$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x$

Log(Odds) takes the form of linear regression intercept β_0 and slope β_1
β_0 and slope β_1 estimated using maximum likelihood estimation

9

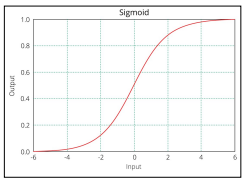
Equation of Logistic Regression

log - odds or **odds ratio** or **logit** function and is the link function for Logistic Regression

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

Regression intercept & coefficient

This link function follows a sigmoid function which limits its range of probabilities between 0 and 1.



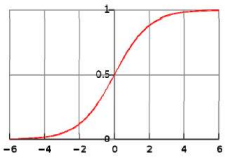
10

From Probabilities to log Odds (and back again)

$$z = \log\left(\frac{p}{1-p}\right)$$ logit function

$$\frac{p}{1-p} = e^z$$

$$p = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$$ logistic function



Standard Logistic Function

11

11

Logistic Regression

- A multidimensional feature space (features can be categorical or continuous).
- – Outcome is **discrete**, not continuous.

We'll focus on case of two classes.
– A linear decision boundary (hyperplane) will give good predictive accuracy.

12

12

Using Logistic Regression Model

- Model consists of a vector β in *d*-dimensional feature space
- For a point **x** in feature space, project it onto β to convert it into a real number *z* in the range - ∞ to + ∞

$$z = \alpha + \boldsymbol{\beta} \cdot \mathbf{x} = \alpha + \beta_1 x_1 + \cdots + \beta_d x_d$$

- Map *z* to the range 0 to 1 using the logistic function

$$p = 1/(1+e^{-z})$$

- Overall, logistic regression maps a point **x** in *d*-dimensional feature space to a value in the range 0 to 1

Logit Transformation

The logistic regression model is given by

$$P(Y \mid X) = \frac{e^{\beta_o + \beta_1 X}}{1 + e^{\beta_o + \beta_1 X}}$$

which is equivalent to

$$\ln\left(\frac{P(Y \mid X)}{1 - P(Y \mid X)}\right) = \beta_o + \beta_1 X$$

This is called the Logit Transformation

13

13

Logit is Directly Related to Odds

The logistic model can be written

$$\ln\left(\frac{P(Y \mid X)}{1 - P(Y \mid X)}\right) = \ln\left(\frac{P}{1-P}\right) = \beta_o + \beta_1 X$$

This implies that the odds for success can be expressed as

$$\frac{P}{1-P} = e^{\beta_o + \beta_1 X}$$

This relationship is the key to interpreting the coefficients in a logistic regression model !!

9

14

Using Logistic Regression Model

Can interpret prediction from a logistic regression model as:

- – A probability of class membership
- – A class assignment, by **applying threshold to probability**
- Threshold represents decision boundary in feature space

15

15

Training a Logistic Regression Model

Optimization of model parameters:

Need to optimize $\boldsymbol{\beta}$ so the model gives the best possible reproduction of training set labels

– Usually done by numerical approximation or maximum likelihood

– On really large datasets, may use stochastic gradient descent

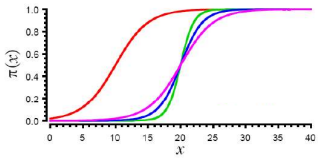
16

16

Parameters of the Model and Shape of the Function

- Parameters control shape and location of sigmoid curve
  - α controls location of midpoint
  - β controls slope of rise

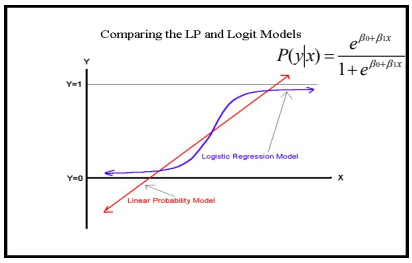
$\pi(x) = \exp(\alpha + \beta x) / (1 + \exp(\alpha + \beta x))$



17

17

Comparing Linear and Logistic Regression Models



18

18

The Logistic Regression Model

The "logit" model solves these problems:
ln[p/(1-p)] = $\beta_0 + \beta_1 X$

p is the probability that the event Y occurs, *p(Y=1)*
[range=0 to 1]

p/(1-p) is the "odds ratio" [range=0 to ∞]

ln[p/(1-p)]: log odds ratio, or logit [range=-∞ to +∞]

19

Making Prediction with Sigmoid

- Using our knowledge of sigmoid functions and decision boundaries, we can now write a prediction function.
- A prediction function in logistic regression returns the probability of our observation being positive, True, or "Yes".
- We call this class 1 and its notation is P(class=1).
- As the probability gets closer to 1, our model is more confident that the observation is in class 1.

20

Making Prediction with Sigmoid

Let's use the same multiple linear regression equation from our linear regression tutorial.

$$z = W_0 + W_1 Studied + W_2 Slept$$

This time however we will transform the output using the sigmoid function to return a probability value between 0 and 1.

$$P(class = 1) = \frac{1}{1 + e^{-z}}$$

If the model returns .4 it believes there is only a 40% chance of passing. If our decision boundary was .5, we would categorize this observation as "Fail.""

21

Cost Function

- Unfortunately we can't (or at least shouldn't) use the same cost function MSE as we did for linear regression. Why?
- Because our prediction function is non-linear (due to sigmoid transform).
- Squaring this prediction as we do in MSE results in a non-convex function with many local minimums. If our cost function has many local minimums, gradient descent may not find the optimal global minimum.
- Instead of Mean Squared Error, we use a cost function called Cross-Entropy, also known as Log Loss. Cross-entropy loss can be divided into two separate cost functions: one for and one for.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \quad \text{if } y = 1$$
$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \quad \text{if } y = 0$$

- The benefits of taking the logarithm reveal themselves when you look at the cost function graphs for y=1 and y=0. These smooth monotonic functions (always increasing or always decreasing) make it easy to calculate the gradient and minimize cost.

22

Cost Function

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

- Multiplying by y and (1-y)in the above equation is a sneaky trick that let's us use the same equation to solve for both y=1 and y=0 cases.
- If y=0, the first side cancels out.
- If y=1, the second side cancels out.
- In both cases we only perform the operation we need to perform.

23

Cost Function

- To minimize our cost, we use Gradient Descent just like before in Linear Regression.
- There are other more sophisticated optimization algorithms out there such as conjugate gradient like BFGS, but you don't have to worry about these.
- One of the neat properties of the sigmoid function is its derivative is easy to calculate.

$$s'(z) = s(z)(1 - s(z))$$

- Which leads to an equally beautiful and convenient cost function derivative:

$$C' = x(s(z) - y)$$

- C' is the derivative of cost with respect to weights
- y is the actual class label (0 or 1)
- $s(z)$ is your model's prediction
- x is your feature or feature vector.

Notice how this gradient is the same as the MSE gradient of Linear Regression, the only difference is the **hypothesis function**.

24

Multi-class Classification

- Instead of y=0,1 we will expand our definition y=0,1,2,3...n-1 (n-class classification problem).
- Basically we rerun binary classification multiple times, once for each class.

Procedure:

1. Divide the problem into **n binary classification problems**.
2. For each class we do the following:
3. Predict the probability the observations are in that single class.
4. Prediction = max {probability of all the classes}

For each sub-problem, we select one class (YES) and dump all the others into a second class (NO).

Then we take the class with the highest predicted value.

25

Softmax Classification

- The softmax function (*softargmax* or *normalized exponential function*) is a function that takes as input a vector of ***K*** real numbers; and normalizes it into a probability distribution consisting of ***K*** probabilities proportional to the exponentials of the input numbers.
- That is, prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after softmax, each component will be in the interval [0, 1], and the components will add up to 1, so that they can be interpreted as probabilities.
- The standard softmax function is defined by formula

$$\sigma(z_i) = \frac{e^{z_{(i)}}}{\sum_{j=1}^{K} e^{z_{(j)}}} \quad for\ i = 1, ., ., ., K\ and\ z = z_1, ., ., ., z_K$$

- In other words, we apply the standard exponential function to each element z_i of the input vector z and normalize these values by dividing the by the sum of all these exponentials; this normalization ensures that the sum of the components of the output vector σ(z) is 1.

26

Logistic Regression

Advantages:

– Makes no assumptions about distributions of classes in feature space

– Easily extended to multiple classes (multinomial regression)

– Can interpret model coefficients as indicators of feature importance

Disadvantages:

- Linear Decision Boundary

27

27

How to *build* a Logistic Regression Model in *Scikit-Learn*

28

Logistic Regression in Scikit-Learn

scikit-learn

`sklearn.linear_model`.LogisticRegression

class `sklearn.linear_model.`**LogisticRegression**(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

- Build an estimator model like **LogisticRegression()**
- Use **fit()** function to ***Train*** the model with **Training Dataset**
- Use **predict()** function to ***Test/Evaluate*** the model with **Test Dataset**
- Use **predict()** function to make ***prediction/inference*** on **New Unseen Data**

29

`sklearn.linear_model`.LogisticRegression

class `sklearn.linear_model.`**LogisticRegression**(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default**. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation, or no regularization. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty. The Elastic-Net regularization is only supported by the 'saga' solver.

30

Example Code

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

31

32

Logistic Regression Case-study

33

Case study on Logistic Regression

Context:

Credit risk is nothing but the default in payment of any loan by the borrower.

In Banking sector this is an important factor to be considered before approving the loan of an applicant.

Dream Housing Finance company deals in all home loans.

They have presence across all urban, semi urban and rural areas.

Customers first apply for home loan after that company validates a customer's eligibility for loan.

Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form.

34

Case study on Logistic Regression

Data Attributes:

1. Loan_ID	Unique Loan ID
2. Gender	Male/ Female
3. Married	Applicant married (Y/N)
4. Dependents	Number of dependents
5. Education	Applicant Education (Graduate/ Undergraduate)
6. Self_Employed	Self employed (Y/N)
7. ApplicantIncome	Applicant income
8. CoapplicantIncome	Co-applicant's income
9. LoanAmount	Loan amount in thousands
10. Loan_Amount_Term	Term of loan in months
11. Credit_History	Credit history meets guidelines
12. Property_Area	Urban/ Semi Urban/ Rural
13. Loan_Status	Loan approved (Y/N)

35

Let's go to the Coding Demo…

36

To be continued in the next session…..

37