VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

Article

Ridge and Lasso logistic regression

Eligijus Bujokas

VILNIUS (2018.12.21)

Logistic Ridge and Lasso regression

Abstract

The aim of this paper is to apply Ridge, Lasso and elastic net binary logistic regression to a data set regarding a binary response variable and text as a feature matrix. Additionally, some clasic text preprocesing techniques will be introduced.

The calculations and visuals are made using the statistical software **R**. The main framework for text preprocesing is the **text2vec** (Selivanov and Wang, 2018).

Key words : Logistic regression, Ridge, Lasso, Elastic Net

Contents

Introduction **4**

Logistic regression for textual data **5**

Introductory case 5

General case for binary dependant variable 6

Maximum likelihood for estimating the coefficients 7

Ridge logistic regression 8

Lasso logistic regression 8

Elastic net regression 8

Text preprocesing definitions techniques **9**

Definitions and notations 9

Document corpus 9

Tokenization 9

N - grams 10

Tf-idf tranformation 10

Term frequency 10

Inverse document frequency 10

Tf - idf calculation 11

Quora question example **12**

References **13**

Introduction

In the modern world there are many clasification tasks that revolve around clasifying textual data. There are algorythms that based on a user inputed string can talk with a person, after reading lyrics of a song it can label the genre of the song and so on. Typical features (independant variables, regresors) are unique words thus even a small text can have thousands of collumns in the design matrix.

A big design matrix leads to big computational times and often leads to multicolinearity. The reduction of the number of features in a dataset or reweighting the importance of features often leads to a speed up in the computational time and better results. Ridge and Lasso regression are often the tools of choice when dealing with the mentioned problems.

Logistic regression for textual data

Introductory case

Let us assume that our $\mathbb{Y}$ variable is binary. Each class of $\mathbb{Y}$ could indicate whether a string is 'positive' or 'negative' in a semantic sense, 'sincere' or 'not sincere' and so on. Each column in the $\mathbb{X}$ matrix are also binary. Each collumn represent a unique word and each row value represent whether a word was observed in a given string or not.

For example, let us say we have a 'positive' review -'This was delicious'- and a 'negative' one -'The food was awfull'. Then raw document will have two rows:

sentiment	review
1	This was delicious
0	This was awfull

The matrix that is used for computations is often called the document term matrix (dtm for short). The dtm of our raw document would look like this:

delicious	awfull	was	this
1	0	1	1
0	1	1	1

As we can see, the matrix indicates that the first review had words *delicious*, *was* and *this* while the second review had the words *awfull*, *was* and *this*.

We can define a general linear model for this case:

$$log\frac{P(Y=1)}{P(Y=0)} = \beta_0 + \beta_1 \mathbb{1}_{delicious} + \beta_2 \mathbb{1}_{was} + \beta_3 \mathbb{1}_{this} + \beta_4 \mathbb{1}_{awfull}$$

According to the data, the coefficient β_1 should be positive and β_4 should be negative implying positive and negative sentiments respectivelly. The words *was* and *this* do not help in distinguishing between the positive and the negative sentiments thus the coefficients near these features need to be 0 or very small.

General case for binary dependant variable

Let us assume that our data is the set:

$$D_{\mathbb{Y},\mathbb{X}} = \{\mathbb{Y}_i \in \{0,1\}, \mathbb{X} = [1, X_{i1}, X_{i2}, ..., X_{ik}], \forall i \in \{1, ..., n\}\}$$

In most practical cases, the intercept is added to the design matrix.

In matrix form:

$$\mathbb{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \mathbb{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & ... & x_{1k} \\ 1 & x_{21} & x_{22} & ... & x_{2k} \\ ... & ... & ... & ... & \\ 1 & x_{n1} & x_{n2} & ... & x_{nk} \end{bmatrix}$$

In practise, all x_{ij} are either 0 or 1, indicating existance in certain row of the document term matrix, or the count of occurance in row in the document term matrix. The number of columns in the $\mathbb{X}$ matrix is equal to the number of unique words in our document.

The general model is then:

$$log\frac{P(Y=1)}{P(Y=0)} = \sum_{i=0}^{k} \beta_i X_i \, (1)$$

Maximum likelihood for estimating the coefficients

Let us define:

$$\theta := \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}$$

From (1), the probability of ‘success’ for each observation i can be rewritten as [1 , pp. 3-5]:

$$\pi_i := P(\mathbb{Y}_i = 1) = \frac{e^{\theta^T \mathbb{X}_i}}{1 + e^{\theta^T \mathbb{X}_i}} = \frac{1}{1 + e^{-\theta^T \mathbb{X}_i}}$$

$$P(\mathbb{Y}_i = 0) = 1 - \pi_i$$

In every binary case, the $\mathbb{Y}$ can be encoded as a vector consisting of 0 and 1. Thus, we want θ such that the product:

$$l(\theta) = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i} \ (2)$$

is the biggest .

Logarythm is a monotone function thus the maximum of $l(\theta)$ is the same as $log(l(\theta))$ [2 , pp. 636 - 637].

$$L(\theta) := log(l(\theta)) = \sum_{i=1}^{n} [y_i log(\pi_i) + (1 - y_i) log(1 - \pi_i)] = \sum_{i=1}^{n} \left[y_i log(\frac{\pi_i}{1 - \pi_i}) + log(1 - \pi_i) \right]$$

$$L(\theta) = \sum_{i} \left[y_i \theta^T \mathbb{X}_i + log(1 + e^{\theta^T \mathbb{X}_i}) \right] \ (3)$$

The $\widehat{\theta}$ that maximizes the (3) equation will give a weight to every unique word in our text. This, in practise, is ussualy not ideal, because even a small text document can have thousands of unique words.

Ridge logistic regression

Ridgre logistic regression introduces an additional term to the (3) equation - the L2 penalty.

$$L^R(\theta, \lambda) = \sum_{i}^{n} \left[y_i \theta^T \mathbb{X}_i + log(1 + e^{\theta^T \mathbb{X}_i}) \right] - \lambda \sum_{j=1}^{k} \beta_j^2$$

Often in practise, the λ parameter is fixed to a certain value. As the parameter λ increases, the ridge coefficient estimates will tend to approach zero. However, the penalty introduced in the log-likelihood function will shrink all of the coefficients towards zero, but it will not set any of them exactly to zero. Hence, ridge regression has the disadvantage over model selection, of including all the predictors in the final model.

On the other hand, Ridge regression estimates gives us more uniformally distributed weights to all words. Depending on the problem, one may view this as an advantage.

Lasso logistic regression

Lasso logistic regression is very similar to that of Ridge, but the penalty term is L1:

$$L^L(\theta, \lambda) = \sum_{i}^{n} \left[y_i \theta^T \mathbb{X}_i + log(1 + e^{\theta^T \mathbb{X}_i}) \right] - \lambda \sum_{j=1}^{k} |\beta_j|$$

The L1 penalty used in the lasso is used for both variable selection and shrinkage, since it has the effect, when the λ is sufficiently large, of forcing some of the coefficient estimates to be exactly equal to zero [2 , pp.637]. In Lasso regression, the final model may involve only a subset of the predictors, which in turn improves model interpretability.

Depending on the research subject and the problem, having less predictors is beneficial.

Elastic net regression

One is not confined to just using either Ridge or Lasso regression. The elastic net procedure tries to implement both of these methods:

$$L^{EN}(\theta, \lambda) = \sum_{i}^{n} \left[y_i \theta^T \mathbb{X}_i + log(1 + e^{\theta^T \mathbb{X}_i}) \right] - (\lambda \sum_{j=1}^{k} |\beta_j| + (1 - \lambda) \sum_{j=1}^{k} \beta_j^2)$$

$\lambda \in (0, 1)$.

This approach is particularly useful when the number of predictors is much larger than the number of observations [3].

Text preprocesing definitions techniques

Definitions and notations

The whole text corpus (all the data) is denoted $\mathbb{D}$. Typically, the corpus is the vector containing the text data. In the introduction chapter, the text corpus would be [‘This was delicious’, ‘This was awfull’].

Each entry in the corpus is denoted as d. In our example case we have two documents: ‘This was delicious’ and ‘This was awfull’.

Each term (token) in a given corpus is denoted as t.

Document corpus

A document or text corpus are large and structured set of texts. Usually in the case natural language procesing (NLP for short) each row of the $\mathbb{X}$ is treated as a separate document (in the introduction chapter, the number of documents would be 2). Each document are made of several tokens.

The whole data set in text related tasks is ussually defined as corpus and can be used instead of the terms ‘whole data set’ or ‘all data’.

Tokenization

Text tokenization is the process of segmenting running text into words and sentences.

Electronic text is a linear sequence of symbols (characters or words or phrases). Naturally, before any real text processing is to be done, text needs to be segmented into linguistic units such as words, punctuation, numbers, alpha-numerics, etc. This process is called tokenization [4].

In English, words are often separated from each other by blanks (white space), but not all white space is equal. Both ‘Los Angeles’ and ‘rock ’n’ roll’ are individual thoughts despite the fact that they contain multiple words and spaces. We may also need to separate single words like ‘I’m’ into separate words ‘I’ and ‘am’.

Tokenization is a kind of pre-processing in a sense; an identification of basic units to be processed. It is conventional to concentrate on pure analysis or generation while taking basic units for granted. Yet without these basic units clearly segregated it is impossible to carry out any analysis or generation.

For example, the string ‘I’m a from the city of Vilnius’ can be tokenized into several tokens: ‘i’, ‘am’, ‘from’, ‘the’, ‘city’, ‘of’, ‘vilnius’.

N - grams

An n-gram is a contiguous sequence of n items from a given sequence of text. Given a sentence, we can construct a list of n-grams from the sentence by finding pairs of words that occur next to each other. For example, given the sentence 'my name is Eligijus' you can construct bigrams (n-grams of length 2) by finding consecutive pairs of words: 'my name', 'name is' and 'is Eligijus'. Each of these bigrams would be considered a token and would be considered as a separate feature in the $\mathbb{X}$ matrix.

Tf-idf tranformation

Tf-idf, short for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The tf-idf is the product of two statistics, term frequency and inverse document frequency.

Term frequency

$tf(\cdot)$ is a function of two arguments - the term t and document d. If we denote the term frequency in a document as $f_{t,d}$ then

$$tf(t,d) = f_{t,d}$$

In simpler terms, the function tf() takes every unique token in our corpus and counts how many times it appeared in a document.

Inverse document frequency

The inverse document frequency ($idf(\cdot)$) is a measure of how much information the word provides, in other words, if it is common or rare across all documents.

$$idf(t,D) = log\left(\frac{N}{1 + |\{d \in D : t \in d\}|}\right)$$

N - total number of documents in our corpus. The corpus in the introduction chapter has two documents.

$|\{d \in D : t \in d\}|$ number of documents where the term t appears. We add 1 because if no document has a certain term, then we would be dividing by zero.

Tf - idf calculation

The tf-idf transformation for each term is a product of term frequency and inverse document frequency.

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

Recall our example case. Lets us calculate the tfidf statistic for the word ‘awfull’.

$tf('awfull', d_1) = 0$

$tf('awfull', d_2) = 1$

$idf('awfull', D) = log(\frac{2}{2}) = 0$

$tfidf('awfull', d_1, D) = 0$

$tfidf('awfull', d_2, D) = 0$

The transformation would yield that in both documents the term ‘awfull’ is equal to 0, altough this term is very important in distinguishing between the sentements. This is because we only used a corpus with two documents. In practise, the td - idf transformation is used when a corpus has a lot of documents.

Quora question example

References

[1] S. A. Czepiel. *Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation*. Article. 2017. <URL: https://czep.net/stat/mlelr.pdf>.

[2] A. F. d. S. Jose Manuel Pereira Mario Bastoa. *The logistic lasso and ridge regression in predicting corporate failure*. Article. 2016. <URL: https://www.sciencedirect.com/science/article/pii/S2212567116303100>.

[3] D. Selivanov and Q. Wang. *text2vec: Modern Text Mining Framework for R*. R package version 0.5.1. 2018. <URL: https://CRAN.R-project.org/package=text2vec>.

[4] C. Trim. *Tokenization*. Article. 2013. <URL: https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en>.

[5] H. Zou and T. Hastie. *Regularization and variable selection via the elastic net*. Article. 2005. <URL: https://web.stanford.edu/~hastie/Papers/B67.2%20(2005)%20301-320%20Zou%20&%20Hastie.pdf}.>