

# **R V COLLEGE OF ENGINEERING**

(Affiliated to Visvesvaraya Technological University, Belagavi)



**An Assignment Report on**

## **“Sentiment Analysis with Logistic Regression”**

*Submitted in partial fulfillment for Assignment component of*

**Machine Learning  
18MCS2C2**

*of*

**MASTER OF TECHNOLOGY  
in  
COMPUTER NETWORK ENGINEERING**

*Submitted by*

**AKASH HEGDE**

**1RV18SCN01**

**ANIL KUMAR M. S**

**1RV18SCN02**

**KARTHIK M. V**

**1RV18SCN06**

*Under the guidance of*

**Dr. HEMAVATHY R.**

**ASSOCIATE PROFESSOR**

Department of Computer Science and Engineering

**R V COLLEGE OF ENGINEERING**

**R.V.VIDYANIKETAN POST, MYSURU ROAD,**

**BENGALURU- 560059**

**2018-19**

## **ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned the efforts with success.

It is indeed a great pleasure for us to present this self-study report on “Sentiment Analysis with Logistic Regression” as a part of the course Machine Learning in the 2<sup>nd</sup> semester of Master of Technology in Computer Network Engineering.

We would like to thank Management of R V College of Engineering for providing such a healthy environment for the successful completion of self-study work.

It gives us immense pleasure to thank Dr. Ramakanth Kumar P, Professor and Head of Department for his constant support and encouragement.

Also, we would like to express our gratitude to our course guide Dr. Hemavathy R., Associate Professor, Department of Computer Science & Engineering and all other teaching and non-teaching staff of Computer Science Department who have directly or indirectly helped us in the completion of the self-study work.

Last, but not the least, we would hereby acknowledge and thank our parents who have been a source of inspiration and also instrumental in the successful completion of the self-study work.

Akash Hegde  
Anil Kumar M. S  
Karthik M. V  
Dept. of CNE

## List of Figures

Sl.No.	Figure Name	Page No.
1.1	General model of logistic regression	1
1.2	Graph of gradient descent	2
2.1	Structure of dataset	3
2.2	Bag-of-words representation	4
2.3	Array of the feature vectors	4
2.4	Reduction in score of the most frequent words	5
2.5	Most common words in the vocabulary	5
2.6	Word distribution across all tweets	6
2.7	Reduced vocabulary	6
2.8	Processed tweet	7
2.9	Output of the tokenizer	7
2.10	Training output for 2 folds	7
2.11	Accuracy for 2 folds	8
2.12	Model prediction for random tweet inputs	8

## Table of Contents

<b>Chapter 1</b>  <b>1. Introduction</b>  1.1 Logistic Regression 1.2 Sentiment Analysis 1.3 Gradient Descent	<b>1-2</b>
<b>Chapter 2</b>  <b>2. Experimental Procedure and Results</b>  2.1 Procedure 2.2 Results 2.3 Conclusion	<b>3-8</b>

## Chapter 1

### Introduction

#### 1.1 Logistic Regression

Logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail, win/lose, alive/dead or healthy/sick; these are represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name.

Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Figure 1.1 depicts a general model of logistic regression.

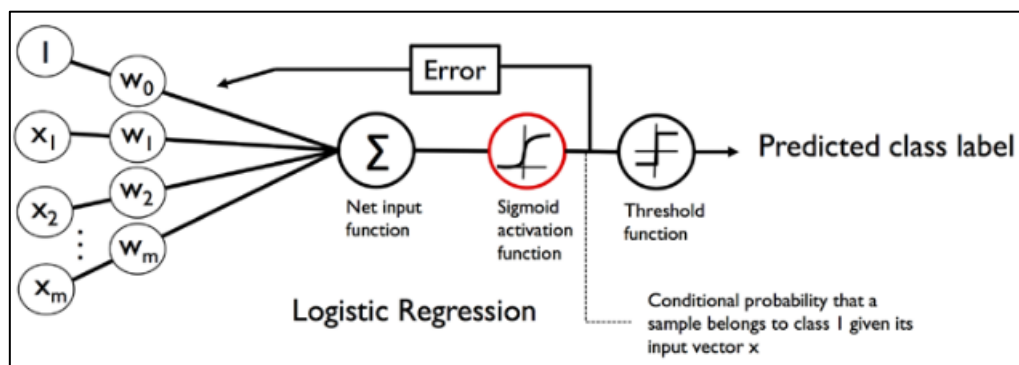


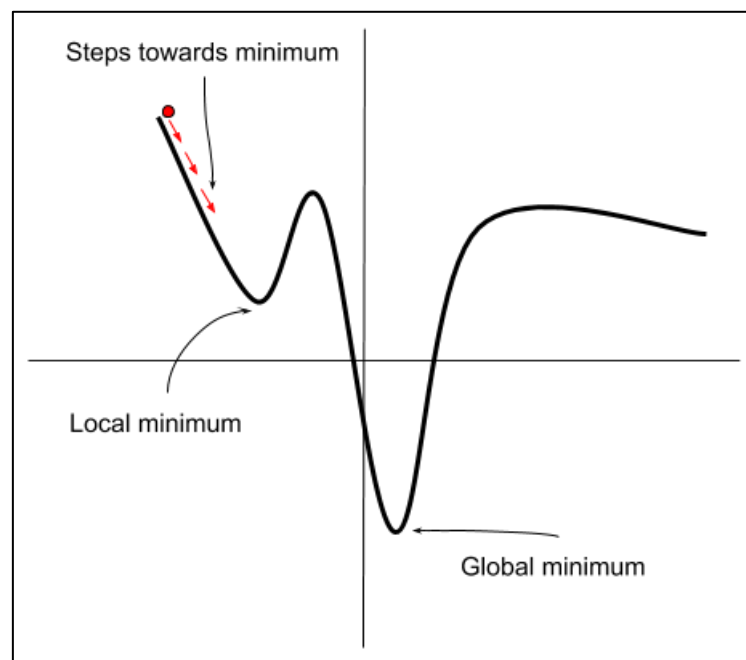
Figure 1.1 General model of logistic regression

## 1.2 Sentiment Analysis

Sentiment analysis (also known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

## 1.3 Gradient Descent

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. Gradient descent is used to update the parameters of the model. Parameters refer to coefficients in Logistic Regression. Figure 1.2 depicts the graph of gradient descent.



**Figure 1.2 Graph of gradient descent**

## Chapter 2

### Experimental Procedure and Results

The detailed procedure of the work done and the obtained results is described in this section. The conclusion drawn is also highlighted.

#### 2.1 Procedure

Dataset taken is a Twitter dataset consisting of 100000 tweets of varying lengths and sentiment. The structure of the dataset is shown in Figure 2.1.

ItemID	Sentiment	SentimentText
0	1	0 is so sad for my APL frie...
1	2	0 I missed the New Moon trail...
2	3	1 omg its already 7:30 :O
3	4	0 .. Omgaga. Im sooo im gunna CRy. I'...
4	5	0 i think mi bf is cheating on me!!! ...
5	6	0 or i just worry too much?
6	7	1 Juuuuuuuuuuuuuuuuuusssst Chillin!!
7	8	0 Sunny Again Work Tomorrow :-  ...
8	9	1 handed in my uniform today . i miss you ...
9	10	1 hmmm.... i wonder how she my number @-)

**Figure 2.1 Structure of dataset**

The structure of a tweet varies a lot across different tweets. Tweets have different lengths, letters, numbers and special characters. A lot of words are not correctly spelled; for example, the word "Juuuuuuuuuuuuuuuuusssst" or the word "frie" instead of "friend." This makes it hard to measure how positive or negative are the words within the corpus of tweets.

If the words were all correct dictionary words, a lexicon could be used to punctuate words. However because of the nature of social media language, it cannot be done. Thus, a way of scoring the words such that words that appear in positive tweets have greater score than those that appear in negative tweets is found out.

The tweets are represented as equal-sized vectors of numbers as follows:

- A list (vocabulary) with all the unique words in the whole corpus of tweets is created.
- A feature vector is constructed from each tweet that contains the counts of how often each word occurs in the particular tweet.

The unique words in each tweet represent only a small subset of all the words in the bag-of-words vocabulary, thus the feature vectors will mostly consist of zeros. Figure 2.2 shows how the bag of words is created. The vocabulary is stored in a Python dictionary that maps the unique words to integer indices. Figure 2.3 shows the array representation of the same.

```
{'this': 13,  
'is': 7,  
'amazing': 2,  
'ml': 9,  
'the': 12,  
'best': 3,  
'yes': 15,  
'it': 8,  
'am': 1,  
'not': 10,  
'sure': 11,  
'about': 0,  
'how': 6,  
'going': 5,  
'to': 14,  
'end': 4}
```

**Figure 2.2 Bag-of-words representation**

```
array([[0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0],  
       [0, 0, 0, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1, 0, 0, 1],  
       [1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0]], dtype=int64)
```

**Figure 2.3 Array of the feature vectors**

Each index position in the feature vectors corresponds to the integer values that are stored as dictionary items in the CountVectorizer vocabulary. For example, the first feature at index position 0 resembles the count of the word 'about' , which only occurs in the last document, and the word 'is' , at index position 7, occurs in all three tweets (two times in the second tweet). These values in the feature vectors are

also called the raw term frequencies:  $tf(t, d)$  — the number of times a term  $t$  occurs in a document  $d$ .

Scikit-learn uses term frequency-inverse document frequency in order to reduce the score of the most frequent word across all the tweets. Figure 2.4 shows how this occurs, in the form of an array representation. Words that appear in all documents like “is” (with 0.47), get a lower score than others that don't appear in all documents, like “amazing” (with 0.72).

```
array([[0. , 0. , 0.72, 0. , 0. , 0. , 0. , 0.43, 0. , 0. , 0. ,
        0. , 0. , 0.55, 0. , 0. ],
       [0. , 0. , 0. , 0.4 , 0. , 0. , 0. , 0.47, 0.4 , 0.4 , 0. ,
        0. , 0.4 , 0. , 0. , 0.4 ],
       [0.33, 0.33, 0. , 0. , 0.33, 0.33, 0.33, 0.2 , 0. , 0. , 0.33,
        0.33, 0. , 0.25, 0.33, 0. ]])
```

**Figure 2.4 Reduction in score of most frequent words**

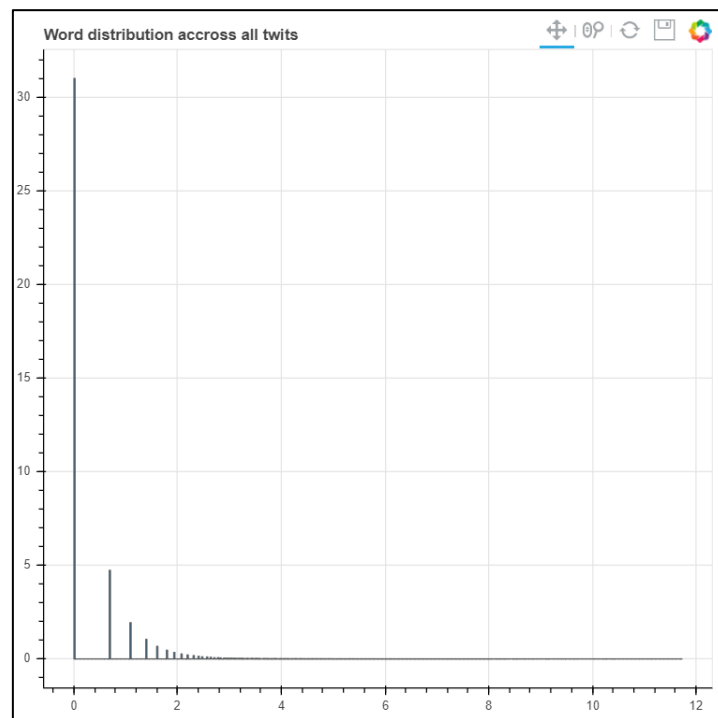
The most common words are meaningless in terms of sentiment: *I*, *to*, *the*, *and*. They do not give any information on positiveness or negativeness. They are basically noise that can most probably be eliminated. Figure 2.5 represents these common words.

```
[('', 123916),
 ('I', 32879),
 ('to', 28810),
 ('the', 28087),
 ('a', 21321),
 ('you', 21180),
 ('i', 15995),
 ('and', 14565),
 ('it', 12818),
 ('my', 12385),
 ('for', 12149),
 ('in', 11199),
 ('is', 11185),
 ('of', 10326),
 ('that', 9181),
 ('on', 9020),
 ('have', 8991),
 ('me', 8255),
 ('so', 7612),
 ('but', 7220)]
```

**Figure 2.5 Most common words in the vocabulary**



The word distribution across all tweets is shown in Figure 2.6. It is plotted using BokehJS.



**Figure 2.6 Word distribution across all tweets**

The NLTK (Natural Language Tool Kit) package for these stop words is then downloaded. These stop words are then eliminated and the reduced vocabulary is calculated, as shown in Figure 2.7.

```
[(' ', 123916),
 ('I', 32879),
 ('I'm', 6416),
 ('like', 5086),
 ('-', 4922),
 ('get', 4864),
 ('u', 4194),
 ('good', 3953),
 ('love', 3494),
 ('know', 3472),
 ('go', 2990),
 ('see', 2868),
 ('one', 2787),
 ('got', 2774),
 ('think', 2613),
 ('&', 2556),
 ('lol', 2419),
 ('going', 2396),
 ('really', 2287),
 ('im', 2200)]
```

**Figure 2.7 Reduced vocabulary**

The special characters and other unnecessary signs and tags are then removed. Emoticons are not removed as they are integral to sentiment expressed. Figure 2.8 shows how an input tweet with many special characters and signs is modified to obtain processed data.

```
this tweet man is nice :)
```

**Figure 2.8 Processed tweet**

There is another trick that is used to reduce the vocabulary and consolidate words. Words like *love*, *loving* express the same positivity. It is then replaced with only one: *lov*. This process of reducing a word to its root is called stemming. A tokenizer is also needed to break down the tweets in to individual words. Two tokenizers are implemented – a regular one and one that does stemming. Figure 2.9 shows the output of the tokenizer.

```
['Hi', 'there,', 'I', 'am', 'loving', 'this,', 'like', 'with', 'a', 'lot', 'of', 'love']  
['Hi', 'there,', 'I', 'am', 'love', 'this,', 'like', 'with', 'a', 'lot', 'of', 'love']
```

**Figure 2.9 Output of the tokenizer**

Training the model requires the best hyperparameters like the learning rate or regularization strength to be chosen. The algorithm should be trained such that it performs better with stemming words, removing tags and special characters, among other operations. To take these decisions methodically, GridSearch is used. GridSearch is a method of training an algorithm with different variations of parameters to select the best combination.

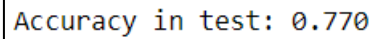
Training is done on parallel processors for number of folds of cross-validation. Figure 2.10 shows the training output for 2 folds.

```
Fitting 2 folds for each of 96 candidates, totalling 192 fits  
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.  
[Parallel(n_jobs=-1)]: Done 34 tasks      | elapsed: 3.9min  
[Parallel(n_jobs=-1)]: Done 192 out of 192 | elapsed: 25.0min finished
```

**Figure 2.10 Training output for 2 folds**

## 2.2 Results

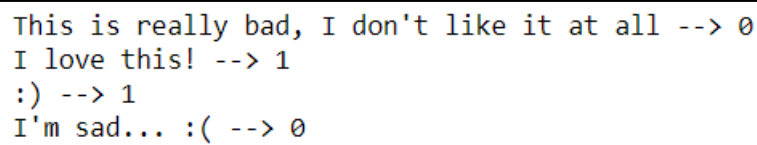
Accuracy is then calculated using in-built scoring metrics. Figure 2.11 shows accuracy obtained for the 2 fold cross-validation.



Accuracy in test: 0.770

**Figure 2.11 Accuracy for 2 folds**

Finally, for test data of some random tweets, the logistic outputs are checked in terms of 0 or 1. Figure 2.12 shows these outputs.



```
This is really bad, I don't like it at all --> 0
I love this! --> 1
:) --> 1
I'm sad... :( --> 0
```

**Figure 2.12 Model prediction for random tweet inputs**

## 2.3 Conclusion

Logistic regression is primarily used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Here, a dataset consisting of random tweets is analyzed with respect to sentiment, using logistic regression.

The machine learning model is built and trained considering the cross-validation procedure of GridSearch. Accuracy of nearly 77% is achieved with the model being trained for 2 folds. This accuracy can be further improved by changing the hyperparameters accordingly.