

Text Classification with 'Logistic Regression'

(Maximum Entropy Model)

How Make a good classifier

Domain : The data must match the real scenario.

data ต้องอิงสถานการณ์จริง

Data quality and quantity : The annotation must be consistent, and the size must be large .

ข้อมูลต้องมีคุณภาพดีและใหญ่

Feature engineering : What does the model need to pay attention in order to make a good decision ?

ต้องคิดหาฟีเจอร์ที่เหมาะสมกับงาน

Model : Some models are better than others.

Logistic Regression

Logistic regression (or Maximum Entropy Model) is a statistical model that computes $P(Y|X)$ from linear combination of input features. It models a link between each label and each feature (infavor or against).

มันเป็นการใช้สถิติของค่า เพื่อคำนวณ $(PY|X)$ $x = \text{feature vector}$ $y = \text{label}$
เป็น การหาผลรวม ++ เพื่อให้เรารู้และหาความเชื่อมโยง ระหว่างฟีเจอร์และเรเบล

if Y is multiclass (e.g. positive, neutral, and neagative sentiment), logistic regression should be called 'multinomial logistic regression. In NLP , we call it gogistic regression or MaxEnt.

ถ้า Y เป็นมัลติคลาส เช่น ขิงบวกลบ เจยๆ เราจะเรียกว่า *multinomial logistic regression* ส่วนใหญ่ก็จะเป็นมัลติคลาส



Components of ML classifier ส่วนประกอบ ของ ML

1. **Representation** - How do we convert from text to feature vector ?
หาตัวแทน เพราะข้อความไม่สามารถนำไปคำนวณได้ ex *bagOfWord*
2. Inference/prediction - How do we compute $P(Y|X)$?
3. Training
 - a. Objective function
 - b. Optimization algorithm for training

Representation

A feature vector represents text. A vector is a list of numbers that can be compared with another vector to measure similarity.

- + Unigram count feature = bag-of-word features
- + Unigram binary feature
- + Unigram TF-IDF feature
- + Bigram count feature

text1 => feature vector

A	B	C	D	E
1	0	1	1	0

text ตัวต่อไปก็ต้องมีแพทเทิลเดียวกัน คือ คอลัมต้องทั่วถึง ทุกแถว

feature vector to represent the text 'predictable and boring'

Text	Label (Y)	predictable	and	boring	very	few	laughs	short	but	powerful	fun	good
predictable and boring	negative	1	1	1								
very few laughs	negative				1	1	1					
short but boring	negative			1				1	1			
very very powerful	positive				2					1		
fun and good good laughs	positive		1				1				1	2

Uni = 1 gram = word = นับทีละคำ such as bag-of-word feature

feature vector to represent the text 'predictable and boring'

Text	Label (Y)	predictable	and	boring	very	few	laughs	short	but	powerful	fun	good
predictable and boring	negative	1	1	1								
very few laughs	negative				1	1	1					
short but boring	negative			1				1	1			
very very powerful	positive				1					1		
fun and good good laughs	positive		1				1				1	1

อันนี้แล้วแต่งงาน ต้องพิจารณา คืออันนี้จะนับคำที่เจอบ่อยเป็น 1 อยู่ดี **Unigram binary**

feature vector to represent the text 'predictable and boring'

Text	Label (Y)	predictable	and	boring	very	few	laughs	short	but	powerful	fun	good
predictable and boring	negative	1	1/2	1/2								
very few laughs	negative				1/2	1	1/2					
short but boring	negative			1/2				1	1			
very very powerful	positive				2/2					1		
fun and good good laughs	positive		1/2				1/2				1	2

$\frac{TF}{IDF}$ — $\frac{\text{term frequency}}{\text{\#doc which contain that word}}$

TF-IDF = TF = term frequency / IDF #doc which contain that word

หมายความว่า คำที่เจอบ่อยๆ อาจจะไม่ช่วย หรือ สื่อความหมายถึง label เท่าไร

feature vector to represent the text 'predictable and boring'

Text	Label (Y)	predictable-and	and-boring	very-few	few-laugh	short-but	...
predictable and boring	negative	1	1				...
very few laughs	negative			1	1		...
short but boring	negative					1	...
very very powerful	positive						...
fun and good good laughs	positive						...

bigram
↓
word

Bigram feature คือหาคำที่อยู่ติดๆกัน 2 คำ
เช่น very-few few-laugh

มันเป็นการโมเดลของเรา จะได้ฟีเจอร์ที่ยาวมาก เยอะ

Text Representation (Features)

1. Unigram count feature assumes each label has a list of associated keywords. this is very good baseline feature.
2. Unigram binary feature is like unigram count feature but ignores the effects of duplicate words.
3. Unigram TF-IDF feature is like unigram count feature but downweights some of the words that appear in too many documents.
4. Bigram count feature assumes that we must consider at least two adjacent word to be able to predict the label. This feature is always too sparse i.e. too many zeros in the feature vector.
5. These features do work well if we have a good amount of data, but we will see more advanced representation called 'word embeddings.' Stay tuned.

ไม่มีอันไหนดีกว่าอันไหน ต้องลองเอา

Inference/Prediction - How do we compute $P(Y|X)$?

การคำนวณ หาความน่าจะเป็น

Logistic Regression - inference

Naive Bayes computes $P(Y|X)$ by multiplying up $P(x|Y=y)$ *model parameter form training
(product of all features x for each label y)

Logistic regression compute $P(Y|X)$ by using sigmoid function (if 2 classes) or softmax function (if > 2 classes)

Consider this model

- Text: tweet
- Feature: bag-of-word ('against', 'love')+ text length
- Label: {positive (1), negative (0)} *binary*

*class
category*

*ใส่พจน์ /
สิ่งที่สัมพันธ์กับ label*

Multiply features with parameters and sum up

Compute the unnormalized score (z) by summing up (linear combination) the product between feature and parameter

text		'against'	'love'	text length
The protester is against the ...		1	0	100

feature vector

Weight	bias	'against'	'love'	text length
positive	0.05	-1	2	-0.0004

model parameter *model weight* *positive evidence*

score (z)		'against'	'love'	text length
positive	0.05	1 x -1	0 x 2	100 x -0.0004

linear combination

$= -0.99$

for i in range(cnt): $-\text{set } \sum_{i=1}^n x[i] \cdot w[i]$

Bias + dot product
between $w \cdot x$

$$z = w \cdot x + b$$

$$z = \left(\sum_{i=1}^n w_i x_i \right) + b$$

พจน์ $w_i x_i$ คือ $\vec{w} \cdot \vec{x}$ vector

Feature vector	x_1	x_2	x_3
$x =$	1	0	100

Weight	w_1	w_2	w_3
$w =$	-1	2	-0.0004

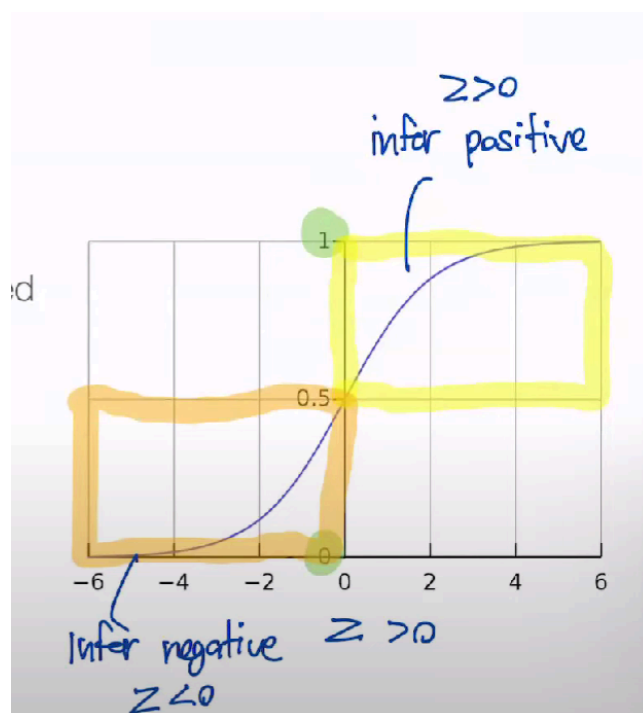
score (z)	intercept bias + $\text{dot product} = w \cdot x$
$z =$	$0.05 + (1 \times -1) + (0 \times 2) + (100 \times -0.0004)$ $b + w_1 x_1 + w_2 x_2 + w_3 x_3$

$z =$ สิ่งที่จะนำไปแปลงเป็น probability

Sigmoid !!!

We convert z into probability $P(Y=1|X)$ by passing Z into a sigmoid function (also called logistic function)

$e == 2.71828$



Compute $P(Y|X)$

Using complementation, if Y is not 1, then Y must be 0.

$$P(Y=0) = 1 - P(Y=1)$$

$$\begin{aligned} P(y=1) &= \sigma(\overbrace{\mathbf{w} \cdot \mathbf{x} + b}^z) \\ &= \frac{1}{1 + \exp(\underbrace{-(\mathbf{w} \cdot \mathbf{x} + b)}_z)} \\ P(y=0) &= 1 - \underbrace{\sigma(\mathbf{w} \cdot \mathbf{x} + b)}_{P(Y=1)} \end{aligned}$$

อธิบายของ Negative class ให้เอา 1 ไป ลบ

(Binary) Logistic Regression

- One instance of the text is represented by a feature vector $\mathbf{x} = [x_1, x_2, \dots, x_n]$
- The logistic regression model has a bias term b (also called intercept term) and weight vector $\mathbf{w} = [w_1, w_2, \dots, w_n]$. b and \mathbf{w} are the model parameters that need to be learned/trained from the training set.
- Bias b represents the score in favor of the positive label regardless of the feature vector.
- Each weight w_i represents the score in favor of the positive label associated with each feature x_i .
- z = unnormalized score computed by bias + the dot product of \mathbf{w} and \mathbf{x}
- The probability of the positive label $P(Y=1|X)$ is computed by passing z into a sigmoid function.

Multiclass Lg1aistic Regression

Multiclass logistic regression is just like binary logistic regression, but it supports the scenario where class are > 2 .

- ส่วนใหญ่คลาสมากกว่า 2 อยู่แล้ว

The idea is the same but we have weight vector for each class . If there are 3 class , we have three weight vector . We concatenate these vectors and call them weight matrix .

- vector ต่อกันหลายๆกันเรียก matrix

Multiply features with parameters and sum up

Compute the unnormalized score z_j for each class j

$$z_j = \sum_{i=1}^n w_{ji} x_i + b_j$$

\uparrow label j $w_j \cdot x + b_j$

text		'against'	'love'	text length
The protester is against the ...	$x =$	1	0	100

Weight matrix (W)	bias	'against'	'love'	text length
positive	0.15	-2	-1	0.0004
negative	-0.2	2	-0.2	0.005
neutral	1	-1	0.4	-0.00001

score (z)	bias	'against'	'love'	text length
positive	0.15	1×-2	0×-1	100×0.0004
negative	-0.2	1×2	0×-0.2	100×0.005
neutral	1	1×-1	0×0.4	100×-0.00001

Softmax Function

- Convert (normalize) \mathbf{z} into a probability vector by passing it to softmax function.

$$\text{softmax}(\mathbf{z}) = \left[\frac{\exp(z_1)}{\sum_{i=1}^K \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^K \exp(z_i)}, \dots, \frac{\exp(z_K)}{\sum_{i=1}^K \exp(z_i)} \right]$$

	bias	'against'	'love'	text length	score (z)
positive	0.15	1×-2	0×-1	100×0.0004	$= -1.81$
negative	-0.2	1×2	0×-0.2	100×0.005	$= 2.3$
neutral	1	1×-1	0×0.4	100×-0.00001	$= -0.001$

	z	exp(z)	P(Y)
positive	-1.81	0.1636541368	0.0147
negative	2.3	9.974182455	0.8956
neutral	-0.001	0.9990004998	0.0897

score vector

$e^{-1.81}$

$e^{-1.81}$

$e^{-1.81}$

$e^{-1.81}$

$e^{-1.81}$

$e^{-1.81}$

$P(Y)$ = normalize คือ แถวแรก ของ positive = $e^{\text{pow } -1.81} / e^{\text{pow } -1.81} + e^{\text{pow } 2.3} + e^{\text{pow } -0.001}$

Matrix Multiplication

Matrix Multiplication

- Compute dot product for each row

$$\begin{array}{c}
 \begin{array}{ccc}
 \text{'against'} & \text{'love'} & \text{text length} \\
 \hline
 -2 & -1 & 0.0004 \\
 2 & -0.2 & 0.005 \\
 -1 & 0.4 & -0.00001
 \end{array} \\
 \mathbf{z} = \mathbf{W}
 \end{array}
 \cdot
 \begin{array}{c}
 \begin{array}{c}
 1 \\
 0 \\
 100
 \end{array} \\
 \mathbf{x}
 \end{array}
 +
 \begin{array}{c}
 \begin{array}{c}
 0.15 \\
 -0.2 \\
 1
 \end{array} \\
 \mathbf{b}
 \end{array}$$

$$\begin{array}{c}
 \begin{array}{c}
 -1.96 \\
 2.5 \\
 -1.001
 \end{array} \\
 \mathbf{z} =
 \end{array}
 +
 \begin{array}{c}
 \begin{array}{c}
 0.15 \\
 -0.2 \\
 1
 \end{array} \\
 \mathbf{b}
 \end{array}$$

Matrix Multiplication

- Compute dot product for each row
- Add the result with the bias vector

$$\begin{array}{c}
 \begin{array}{ccc}
 \text{'against'} & \text{'love'} & \text{text length} \\
 \hline
 -2 & -1 & 0.0004 \\
 2 & -0.2 & 0.005 \\
 -1 & 0.4 & -0.00001
 \end{array} \\
 \mathbf{z} = \mathbf{W}
 \end{array}
 \cdot
 \begin{array}{c}
 \begin{array}{c}
 1 \\
 0 \\
 100
 \end{array} \\
 \mathbf{x}
 \end{array}
 +
 \begin{array}{c}
 \begin{array}{c}
 0.15 \\
 -0.2 \\
 1
 \end{array} \\
 \mathbf{b}
 \end{array}$$

$$\begin{array}{c}
 \begin{array}{c}
 -1.96 \\
 2.5 \\
 -1.001
 \end{array} \\
 \mathbf{z} =
 \end{array}
 +
 \begin{array}{c}
 \begin{array}{c}
 0.15 \\
 -0.2 \\
 1
 \end{array} \\
 \mathbf{b}
 \end{array}$$

$$\begin{array}{c}
 \begin{array}{c}
 -1.81 \\
 2.3 \\
 -0.001
 \end{array} \\
 \mathbf{z} =
 \end{array}$$