Report

Naive Bayes text classification

Task overview

Text classification is a supervised machine learning method used to classify sentences or text documents into one or more defined categories. It’s a widely used natural language processing task playing an important role in spam filtering, sentiment analysis, categorisation of news articles and many other business related issues.

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Algorithm concept

A naive Bayesian classifier is a simple probabilistic classifier based on the application of the Bays theorem with strict (naive) assumptions about independence.

Bayesian classifiers can be trained very effectively. Many practical applications for estimating parameters for naive Bayes models use plausible methods; probability of occurrence of the Bayesian technique.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

Algorithm steps

1. Pre-processing data. This step consists of:
   - Read texts from csv file.
   - Convert all texts to lowercase.
   - Delete stopwords
2. Split dataset on train | verification datasets.
3. Fit model on train dataset.
4. Check model adequacy on verification dataset, calculate accuracy.

Naive Bayes with classic probability calculation

In this algorithm for each word in classification document (test line) calculate conditional probability this word for each class. Conditional probabilities are multiplied or sum their logarithms (in the case of a lot of words).

For example, input data are:

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Number of classes is 2. Calculate class priors:

$$P(c) = \frac{3}{4}$$
$$P(j) = \frac{1}{4}$$

Calculate conditional probability for test doc for each class. Test doc consist of 3 unique words.

$$P(\text{Chinese}|c) = (5+1) / (8+6) = 6/14 = 3/7$$
$$P(\text{Tokyo}|c) = (0+1) / (8+6) = 1/14$$
$$P(\text{Japan}|c) = (0+1) / (8+6) = 1/14$$
$$P(\text{Chinese}|j) = (1+1) / (3+6) = 2/9$$
$$P(\text{Tokyo}|j) = (1+1) / (3+6) = 2/9$$
$$P(\text{Japan}|j) = (1+1) / (3+6) = 2/9$$

Calculation probabilities affiliation test document to each class:

$$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \approx 0.0003$$

$$P(j|d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \approx 0.0001$$

Choosing a class by choosing the maximum probability. in this case is 0.0003.

For testing Naïve Bayes model was taken dataset:

```
demo_data = [["Chinese Beijing Chinese","0"],
             ["Chinese Chinese Shanghai","0"],
             ["Chinese Macao","0"],
             ["Tokyo Japan Chinese","1"]]
demo_pred =  "Chinese Chinese Chinese Tokyo Japan"
```

Model must return:

```
# Must return[ ('Chinese Chinese Chinese Tokyo Japan', '0')]
# pobability {'1': 0.00013548070246744226, '0': 0.00030121377997263036}
# or log     {'1': -7.906681345001262, '0': -7.10769031284391}
```

Model return

```
*** NaiveBayes data statistic ***
Corpus length =  4
classes count =  {'0': 3, '1': 1}
classes ratio =  {'0': 0.75, '1': 0.25}
unique words in corpus =  6
*** ------------------------ ***
pobability    0 {'0': 0.00030121377997263036, '1': 0.00013548070246744226}
log           0 {'0': -8.10769031284391, '1': -8.906681345001262}
```

Model validation on a large dataset. Dataset consist of 1118 e-mails. 33% emails are spam class.

Pre-processing data – read data from csv file, split data on train-verification datasets in proportion 80% / 20%. Check class ratio in train and verification set.

```
1118 = 894 + 224
------------
*** NaiveBayes data statistic ***
Corpus length =  1118
classes count =  {'0': 380, '1': 738}
classes ratio =  {'0': 0.33989266547406083, '1': 0.6601073345259392}
unique words in corpus =  33697
*** ------------------------ ***
*** NaiveBayes data statistic ***
Corpus length =  894
classes count =  {'0': 297, '1': 597}
classes ratio =  {'0': 0.33221476510067116, '1': 0.6677852348993288}
unique words in corpus =  26275
*** ------------------------ ***
*** NaiveBayes data statistic ***
Corpus length =  224
classes count =  {'1': 141, '0': 83}
classes ratio =  {'1': 0.6294642857142857, '0': 0.3705357142857143}
unique words in corpus =  15295
*** ------------------------ ***
Wall time: 1.67 s
```

As optimization was selected deleting stopwords in corpus, using nltk library.

Results

Should pay attention, for large documents we can see low model accuracy because Python uses double-precision floats, which can hold values from about 10 to the -308 to 10 to the 308 power and calculation probabilitiy is out of range and takes an absolute zero value. Thereby better to use sum probabilities logarithms.

Model results for multiply probabilities, logarithms probabilities and logarithms probabilities for corpus without stopwords:

```
pobability
Matches:   Counter({True: 117, False: 107})
Accuracy:  {False: 0.47767857142857145, True: 0.5223214285714286}
----------
log
Matches:   Counter({True: 214, False: 10})
Accuracy:  {True: 0.9553571428571429, False: 0.044642857142857144}
----------
log without stopwords
Matches:   Counter({True: 211, False: 13})
Accuracy:  {True: 0.9419642857142857, False: 0.05803571428571429}
----------
Wall time: 3.6 s
```

Naive Bayes with tf-idf probability calculation

Term frequency–inverse document frequency (**TF/IDF**), is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. Tf–idf is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use tf–idf.

In this modification of Naive Bayes algorithm for each word instead conditional probability calculate if/idf value.

Fitting model consist of calculating IDF = (Total number of documents / Number of documents in which the term a is found) for each word in each class in corpus.

In prediction calculate TF = (Count of times the term a has appeared in the text / count of all words in the text) for classification document words.

Calculation probabilities affiliation test document to each class and choosing a class by choosing the maximum probability.

Testing Naïve Bayes with TF/IDF model on dataset

```
demo_data = [["Chinese Beijing Chinese","0"],
             ["Chinese Chinese Shanghai","0"],
             ["Chinese Macao","0"],
             ["Tokyo Japan Chinese","1"]]
demo_pred =  "Chinese Chinese Chinese Tokyo Japan"
```

Model return

```
pobability    0 {'0': 0.22515081802568107, '1': 0.19709985929564233}
log           0 {'0': -1.4909847989936278, '1': -1.6240447786966632}
```

Model validation on a large dataset. Dataset consist of 1118 e-mails. 33% emails are spam class.

Pre-processing data – read data from csv file, split data on train-verification datasets in proportion 80% / 20%. Check class ratio in train and verification set.

As optimization was selected deleting stopwords in corpus, using nltk library.

Results

```
*** NaiveBayes data statistic ***
Corpus length =  894
classes count =  {'0': 297, '1': 597}
classes ratio =  {'0': 0.33221476510067116, '1': 0.6677852348993288}
unique words in corpus =  26275
*** ------------------------ ***
pobability
Matches:   Counter({True: 133, False: 91})
Accuracy:  {True: 0.59375, False: 0.40625}
----------
log
Matches:   Counter({True: 181, False: 43})
Accuracy:  {True: 0.8080357142857143, False: 0.19196428571428573}
----------
log without stopwords
Matches:   Counter({True: 206, False: 18})
Accuracy:  {True: 0.9196428571428571, False: 0.08035714285714286}
----------
Wall time: 3min 47s
```

Compare accuracy

Model accuracy for different calculation

Naive Bayes	Calculation variant	Accuracy, %
Classic	Multiply probability	52.23
	Sum logarithms probabilities	**95.54**
	Sum logarithms probabilities and delete stopwords	94.20
TF-IDF	Multiply TF/IDF	59.38
	Sum logarithms TF-IDF	80.80
	Sum logarithms TF-IDF and delete stopwords	91.96

References

- http://nlpx.net/archives/57
- https://stevenloria.com/tf-idf/
- https://ru.wikipedia.org/wiki/TF-IDF

2019-05-09
Ievgen Shumelchyk
Ievgen.shumelchyk@gmail.com