Implement a spamfilter based on naive bayes theorem

Yuxuan Zhou

08.02.2018

Contents

01.
Theory

- Bayes rule
- ‚Bag of words ‘ and Independence assumptions
- MAP, ML

02.
Implementation

- Naive Bayes Classifier
- Spamfilter

Classification Methods: Supervised Machine Learning

- Input:
  - A fixed set of classes C = { ‘spam’ ,’ ham’ }
  - A test set of n prelabeled documents $(\boldsymbol{d_i}, \boldsymbol{c_i})$
  - A training set of m prelabeled documents $(\boldsymbol{d_i}, \boldsymbol{c_i})$

- Output:
  - A learned classifier $\boldsymbol{\gamma}: \boldsymbol{d} \rightarrow \boldsymbol{c}$

- Classifier art:
  - Naïve Bayes
  - Support-Vector machines
  - Logistic regression

Naive Bayes Classifier

- Bayes rule for a document d and a class c

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

- Assumptions
- Bag of words Model:

  assume position of words doesn’t matter

- Conditional Independence:

  Assume the feature probabilities are independent given the class c

$$P(x_1,\ldots,x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

Maximum a posteriori

- Most likely class according to the known result

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(c \mid d)$$

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c) P(c)}{P(d)}$$

- In der Implementation is P(d) a fixed value

$$= \underset{c \in C}{\operatorname{argmax}}\, P(d \mid c) P(c)$$

- Document d is represented as features x1...xn

$$= \underset{c \in C}{\operatorname{argmax}}\, P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

- Applying the assumptions above

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}}\, P(c_j) \prod_{x \in X} P(x \mid c)$$

Maximum Likelihood estimates

- Just count the frequencies in the labeled data(in the case of binomial distribution)

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

- Count of word wi among all words in documents of class cj

- Problem arises a new word which is not in the training documents appears

Laplace smoothing

- Extract the Vocabulary from training corpus

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1\right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Underflow Prevention

- By transforming into the log space, multiplying of probabilities becomes sum logs of probabilities
- Calculation error because of the limited floating point number could be avoided

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

Underflow Prevention

- Direct multiply the probabilities using Bayes theorem

```
zhouyuxuan@zhouyuxuan-Lenovo-Y50-70:~/spamfilter$ python spamfilter_test.py trai
n -m /home/zhouyuxuan/spamfilter/spamdata/model -c /home/zhouyuxuan/spamfilter/s
pamdata/train
{'--corpuspath': '/home/zhouyuxuan/spamfilter/spamdata/train',
 '--modelpath': '/home/zhouyuxuan/spamfilter/spamdata/model',
 '<path>': None,
 'classify': False,
 'train': True}
Training size is: 9000
Accuracy is 0.7006666666666667
```

```
zhouyuxuan@zhouyuxuan-Lenovo-Y50-70:~/spamfilter$ python spamfilter_test.py clas
sify -m /home/zhouyuxuan/spamfilter/spamdata/model /home/zhouyuxuan/spamfilter/s
pamdata/test
{'--corpuspath': None,
 '--modelpath': '/home/zhouyuxuan/spamfilter/spamdata/model',
 '<path>': '/home/zhouyuxuan/spamfilter/spamdata/test',
 'classify': True,
 'train': False}
Test size is: 3000
Accuracy is 0.736
```

Underflow Prevention

- Direct multiply the probabilities using Bayes theorem

```
zhouyuxuan@zhouyuxuan-Lenovo-Y50-70:~/spamfilter$ python spamfilter.py classify
-m /home/zhouyuxuan/spamfilter/spamdata/model /home/zhouyuxuan/spamfilter/spamda
ta/test
{'--corpuspath': None,
 '--modelpath': '/home/zhouyuxuan/spamfilter/spamdata/model',
 '<path>': '/home/zhouyuxuan/spamfilter/spamdata/test',
 'classify': True,
 'train': False}
Test size is: 3000
Accuracy is 0.855
```

```
zhouyuxuan@zhouyuxuan-Lenovo-Y50-70:~/spamfilter$ python spamfilter.py train -m
/home/zhouyuxuan/spamfilter/spamdata/model -c /home/zhouyuxuan/spamfilter/spamda
ta/train
{'--corpuspath': '/home/zhouyuxuan/spamfilter/spamdata/train',
 '--modelpath': '/home/zhouyuxuan/spamfilter/spamdata/model',
 '<path>': None,
 'classify': False,
 'train': True}
Training size is: 9000
Accuracy is 0.7954444444444444
```

Programm Design

- Naive Bayes Classifier:
- Defined as a class object

```
class classifier:
    def __init__(self,p_ham,p_spam,p_w_ham,p_w_spam,V):
        self.p_ham= p_ham
        self.p_spam= p_spam
        self.p_w_ham = p_w_ham
        self.p_w_spam = p_w_spam
        self.V = V
        self.classified = []
```

- Spamfilter:
- Handle the data, and define the Interface for user

Program Design

- Save the Class Object ,Classifier ‘

```
import _pickle as cPickle

 # save the classifier
 if not os.path.exists(opts['--modelpath']):
     os.makedirs(opts['--modelpath'])
 with open(opts['--modelpath']+ '/NBclassifier.pkl', 'wb+') as fid:
     cPickle.dump(clas, fid)
```

- Load it by classifying

```
# load it again
with open(opts['--modelpath']+ '/NBclassifier.pkl', 'rb') as fid:
    clas = cPickle.load(fid)
```

Naive Bayes Classifier

- Mehtods:
- Train

```
@classmethod
def train(cls,train_features,alpha=0.7):
```

......

```
return cls(p_ham,p_spam,p_w_ham,p_w_spam,V)
```

Then it can be called and assigned at the same time

```
#train a classifier
clas = classifier.train(train_features)
```

- Evaluate

```
def evaluate(self,test_features):
```

,Compare the labels ‘

```
accuracy = cor / (cor + wro)
print(f'Accuracy is {accuracy}')
```

Spamfilter

- Command line interaction(docopt)

```
def main(opts):
    if opts['train']:
```

- Load Data(Save in suitable form)

```
for File in FileList:
    with open(folder + File.encoding="utf-8", errors="surrogateescape") as f:
```

- Preprocessing Data(Token, Lemmatization, Shuffle)

```
from nltk import word_tokenize, WordNetLemmatizer
```

- Extracting the Features(Stopwords, Frequency)

```
from nltk.corpus import stopwords
```

```
def get_features(text):
    return {word: count for word, count in Counter(preprocess(text)).items() \
    if not word in stoplist}
```