The Machine Learning of Spam

YingCong Kuang
Computer Science
Bishop's University
Sherbrooke, Canada
victorkuang336@gmail.com

Xinyu Wang
Computer Science
Bishop's University
Sherbrooke, Canada
ruobai5211@gmail.com

Zhaorong Hong
Computer Science
Bishop's University
Sherbrooke, Canada
hongzhaorong365@gmail.com

Lu Yang
Computer Science
Bishop's University
Sherbrooke, Canada
lyang17@ubishops.ca

***Abstract*—This is the paper of a computer programming project for spam machine learning. In order to illustrate the effort of achieving the project, this paper will mainly describe the idea of process E-mail content and the method of Naive Bayes which applies for machine learning to detect spam. Related details such as background information, code and reference also will be provided in following content.**

Keywords—spam, Naive Bayes, machine learning

I. INTRODUCTION (*HEADING 1*)

Recently, most of IT brands are dedicated to develop the spam filter. It is important to improve customer experience and prevent information overload because there is a pretty high interaction frequency in this information age. This is how the idea comes with the project of spam machine learning which uses the theorem of Naive Bayes and related Gaussian and Multinomial method. In this programming project, it aims to use the knowledge of pattern recognition to analyze the characteristics of sensitive words in spam and establish a machine learning model for detecting spam.

II. DEVELOP ENVIRONMENT

A. Eclipse

Eclipse is an integrated development environment used in computer programming, and is the most widely used IDE for Java and Python. It contains a base workspace and an extensible plug-in system for customizing the environment. The version of Eclipse Oxygen.1a Release (4.7.1a)

B. PyDev(python)

PyDev is a third-party plug-in for Eclipse. It is an Integrated Development Environment used for programming in Python supporting code refactoring, graphical debugging, code analysis among other features. It is convenient for developer to apply the algorithms of Naive Bayes and Gaussian method from the library of PyDev.

III. APPROACH

This project uses the data set downloaded from an open source website. It provides more than 4000 E-mails for training or testing and contains a list of label for each E-mail (label "1" stand for normal E-mail, label "0" stand for spam). Since data set is ready, the next step is data processing. The project consists of two main parts. The first part is dealing the content of the mail. This part provides the dictionary of "feature" that can be used in the Naive Bayes algorithm. The second part is machine learning. Using the algorithm from python's library (Naive Bayes classifier) to process training and testing path. There are the principle of the main function

A. Make dictionary for valid word

The contents of each E-mail will be separated into minor section and the body part, because only the main part of the content is the valid information that study in machine learning. Then it needs to extract the key word further. For example, there are many words like "1", "a", "@" and so on, those are the unnecessary items in processing information. In this program, the valid word from each E-mail will be extracted and saved in a "dictionary" variable which has a capacity with 1500 (a default value set by code). In terms of the key word, for instance, it could be "sex", "loan", "wager" and so on.

B. Extract features

The program also builds a matrix that links the E-mail with "dictionary". It is a kind of relationship between all the E-mails and all key words in the "dictionary". So, the matrix here records the occurrences of all the key words in each E-mail. It means that in the generated matrix, rows denote the files of training set and columns denote the words of dictionary. The dataset has a file call "SPAMTrain.label" which contains the labels of the E-mails, with 1 stands for a HAM (non-spam) and 0 stands for a SPAM. Once comparing the matrix and the labels, it is able for the machine to learn what is the features of SPAM and HAM. Therefore, the program has established the fundamental for the machine learning of spam.

C. Training and testing method

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable [1]. Bayes' theorem states the following relationship, given class variable Y and dependent feature vector X1 through Xn. That is in equation (1). In a real calculation based on the content of E-mail, Y could stand for "Spam" or "Ham", Xn could stand for "loan" or some key word like that.

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)} \quad (1)$$

Based on the naive conditional independence assumption:

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y)$$

And this relationship can be simplified into following change for all i .

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Since the probabilities of X1…..Xn are constant given the input, the following classification (2) with likelihood can be used.

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y) \quad (2)$$

D. *Gaussian Naive Bayes and Multinomial Naive Bayes*

GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian which its parameters should be estimated using maximum likelihood:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (3)$$

MultinomialNB is more usually applied in the situation that the data is multinomially distributed. This paper is not going to describe its algorithm because we use it in this project just in order to compare with GaussianNB by their result. Related reference of this method can be found in [1].

IV. IMPLEMENTATION

A significant part to achieve the method above is how the program implement. The project applies python to process data based on the principle of Naive Bayes algorithm which can import from "sklearn.naive bayes" library in python develop environment.

A. *ExtractContent*

This step is a pre-processing of the E-mail. In order to roughly separate the main body from the redundant text to reduce the amount of content which use for processing. This function saved as ExtractConten.py in program file. And this step already output the E-mails we need for training and testing. It is available to check the difference between before and after extraction by compare the E-mail in file "RESULT" and "TRAINING".

B. *Features*

In order to further filtering symbols and letters from the body, those texts will be deleted if they are alphabet or only have 1 length. And it set up a dictionary with the capacity of saving 1500 key word after filter. Related function "Make Dictionary" saved in program file "features".

```
removeList = dictionary.keys()
for item in list(removeList):
    if item.isalpha() == False:
        del dictionary[item]
    elif len(item) == 1:
        del dictionary[item]

dictionary = dictionary.most_common(1500)
return dictionary
```

Fig. 1. Code for filtering

Once the dictionary is ready, next step is extracting word count vector (our feature here) of 1500 dimensions for each email of training set. Each word count vector contains the frequency of 1500 words in the training file. Of course, maybe most of them will be zero. Let us take an example. Suppose it has 500 words in our dictionary. Each word count vector contains the frequency of 500 dictionary words in the training file. Suppose text in training file was "if you want it, click here" then the occurrence of these word will be encoded as [0,0,0,0,0,…….0,0,2,0,0,0,……,0,0,1,0,0,…0,0,1,0,0,……2,0,0,0,0,0]. Here, all the word counts are placed at 296th, 359th, 415th, 495th index of 500 length word count vector and the rest are zero.

The below python code will generate a feature vector matrix whose rows denote lots of files of training set and columns denote 1500 words of dictionary. The value at index 'ij' will be the number of occurrences of jth word of dictionary in ith file. This function

```
def extract_features(training_dir,dictionary):

    trainingFiles = [os.path.join(training_dir,f) for f in os.listdir(training_dir)]
    features_matrix = np.zeros((len(trainingFiles),1500))
    docID = 0;
    for file in trainingFiles:
      with open(file) as fileopen:
        for i,line in enumerate(fileopen):
          if i > 1:
            words = line.split()
            for word in words:
              compte = 0
              for i,d in enumerate(dictionary):
                if d[0] == word:
                  compte = i
                  features_matrix[docID,compte] = words.count(word)
        docID = docID + 1
    return features_matrix
```

Fig. 2. Code for features matrix (vector)

C. *Traning and testing*

The program chooses 139 E-mails from dataset and set up a training set "Train" (saved in program file). It assumes 88 E-mails is non-spam. Then, using the feature extracting function above, the program generates the training vector as the parameter for the input of 2 Naive Bayes classifier: MultinomialNB and GaussianNB. The function "fit" here is applying for fitting Gaussian or Multinomial Naive Bayes according to the features already processed so that the program can generate the model for testing.

```
##Create the dictionary
training_dir = 'Train'
dictionary = make_Dictionary(training_dir)

##vectors
##0 stands for a SPAM
traning = np.zeros(139)
traning[88:139] = 1
train_matrix = extract_features(training_dir,dictionary)

# NB classifier
model1 = MultinomialNB()
model2 = GaussianNB()
model1.fit(train_matrix,traning)
model2.fit(train_matrix,traning)
```

Fig. 3. Code for training process

```
fit(self, X, y, sample_weight=None):
"""Fit Gaussian Naive Bayes according to X, y

Parameters
----------
X : array-like, shape (n_samples, n_features)
    Training vectors, where n_samples is the number of samples
    and n_features is the number of features.

y : array-like, shape (n_samples,)
    Target values.

sample_weight : array-like, shape (n_samples,), optional (default=None)
    Weights applied to individual samples (1. for unweighted).

    .. versionadded:: 0.17
       Gaussian Naive Bayes supports fitting with *sample_weight*.

Returns
-------
self : object
    Returns self.
"""
```

Fig. 4. Definition of "fit" function in NB library

In terms of the testing, the program chooses 202 E-mails from dataset and set it up as testing set "TESTING_RESULT" (saved in program file). It assumes 180 E-mails is non-spam. Since the training model already generated, it can be applied for the classification to get probability of maximum correlation with features by using "predict", which use the method of likelihood and argmax.

```
test_dir = 'TESTING_RESULT'
test_matrix = extract_features(test_dir,dictionary)
test = np.zeros(202)
test[180:202] = 1
result1 = model1.predict(test_matrix)
result2 = model2.predict(test_matrix)
```

Fig. 5. Code for testing process

```
def _joint_log_likelihood(self, X):
    """Compute the unnormalized posterior log probability of X

    I.e. ``log P(c) + log P(x|c)`` for all rows x of X, as an array-like of
    shape [n_classes, n_samples].

    Input is passed to _joint_log_likelihood as-is by predict,
    predict_proba and predict_log_proba.
    """

def predict(self, X):
    """
    Perform classification on an array of test vectors X.

    Parameters
    ----------
    X : array-like, shape = [n_samples, n_features]

    Returns
    -------
    C : array, shape = [n_samples]
        Predicted target values for X
    """
    jll = self._joint_log_likelihood(X)
    return self.classes_[np.argmax(jll, axis=1)]
```

Fig. 6. Definition of likelihood and predict function in NB library

V. RESULT

Testing result will be displayed on the counsel of eclipse after running file "training.py". This file contains the main function of training and testing. Based on the 202 test E-mails, the program has shown the confusion matrix of the test-set for both the models. The diagonal elements represent the correctly identified (a.k.a. true identification) E-mails where as non-diagonal elements represents wrong classification (false identification) of E-mails. The result below shows that 173+0 is the number of HAM (non-SPAM) in Multinomial model, while the 143+3 is the result in Gaussian model. Also, the program provides the score by the function "accuracy score" from python library. As Fig.8 shown, Multinomial method has better accuracy than the Gaussian because the data distribution type for this example is discrete rather than continuous.

```
def accuracy_score(y_true, y_pred, normalize=True, sample_weight=None):
    """Accuracy classification score.

    In multilabel classification, this function computes subset accuracy:
    the set of labels predicted for a sample must *exactly* match the
    corresponding set of labels in y_true.
```

Fig. 7. Definition of function "accuracy score"

```
MultinomialNB | HAM | SPAM
[[173   7]
 [ 22   0]]
0.856435643564
GaussianNB | HAM | SPAM
[[143  37]
 [ 19   3]]
0.722772277228
```

Fig. 8. Result and accuracy of testing

REFERENCES

[1] H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.