DOI 10.37882/2223–2966.2022.03–2.15

ПРИМЕНЕНИЕ ВЕРОЯТНОСТНОГО АЛГОРИТМА К ФИЛЬТРАЦИИ СПАМА

APPLYING A PROBABILISTIC ALGORITHM TO SPAM FILTERING

O. Okhlupina
D. Murashko

Summary. Among the common methods of combating spam, a special place is occupied by a probabilistic machine learning algorithm, which is based on the well-known Bayes theorem. The so-called "naive" Bayesian classifier establishes the class of the document by determining the a posteriori maximum. With the development of machine learning methods, the Bayesian algorithm has not lost its relevance and continues to be very popular for solving a large number of tasks, including spam detection. The main advantages of this classifier are simplicity, fast learning, fairly high accuracy, reliability. The paper considers the solution of the problem of determining spam messages using a probabilistic machine learning algorithm. The mathematical justification and implementation of the Bayesian algorithm on a concrete example using program code in the Python programming language is given.

Keywords: spam, filtering, probabilistic algorithm, a posteriori probability, machine learning, classifier, training.

Охлупина Ольга Валентиновна
К.ф.-м.н., доцент, Брянский государственный инженерно-технологический университет
helga131081@yandex.ru
Мурашко Дмитрий Сергеевич
Брянский государственный инженерно-технологический университет
murashko100500@gmail.com

Аннотация. Среди распространённых методов борьбы со спамом особое место занимает вероятностный алгоритм машинного обучения, в основе которого лежит широко известная теорема Байеса. Так называемый «наивный» байесовский классификатор устанавливает класс документа посредством определения апостериорного максимума. С развитием методов машинного обучения байесовский алгоритм не потерял своей актуальности и продолжает оставаться весьма востребованным для решения большого количества задач, среди которых находится и обнаружение спама. Основными преимуществами данного классификатора являются простота, быстрая обучаемость, достаточно высокая точность, надёжность. В работе рассмотрено решение задачи определения спам-сообщений с использованием вероятностного алгоритма машинного обучения. Приводится математическое обоснование и реализация байесовского алгоритма на конкретном примере с помощью программного кода на языке программирования Python.

Ключевые слова: спам, фильтрация, вероятностный алгоритм, апостериорная вероятность, машинное обучение, классификатор, обучение.

Введение

Использование подходов и методов, реализуемых в системах искусственного интеллекта, даёт возможность существенно повысить эффективность решений большинства практических задач по сравнению с традиционными подходами. Человеческий опыт обнаружения спама играет важную роль в борьбе с подобными рассылками в силу нешаблонности и высокой результативности. Однако оптимизация процесса фильтрации и совершенствование её механизмов приобретает всё большую актуальность. Существует ряд методов определения спама, имеющих свои преимущества и недостатки в плане соответствия необходимым критериям простоты, обучаемости и надёжности, минимизации ложных выводов. В пункте 1 работы приводится математическая база вероятностного байесовского алгоритма. В пункте 2 решается задача обнаружения и фильтрации спам-сообщений с демонстрацией программного кода на языке программирования Python.

1. Байесовский алгоритм и фильтрация

В основе вероятного байесовского алгоритма лежит использование хорошо известной теоремы Байеса [1], тесно связанной с условными вероятностями. Данный алгоритм относится к алгоритмам машинного обучения.

Введём следующие обозначения. Пусть X — множество объектов, Y — множество классов (конечное). Вероятностное пространство $X \times Y$ имеет плотность $p(x,y) = P(y)\,p(x|y)$, где $P(y)$ — априорные вероятности появления объектов каждого из классов, $p_y(x) = p(x|y)$ — плотности распределения классов (функции правдоподобия). $\alpha : X \to Y$ — алгоритм

Таблица 1.

Спам	Не спам
Ноутбуки по выгодной цене	Завтра состоится конференция
Распродажа! Закажи велосипед и получи наушники в подарок	Закажи коньки и велосипед

Таблица 2.

	Слова	Попадания в класс «Спам»	Попадания в класс «Не спам»	Вероятность попадания в «Спам»	Вероятность попадания в «Не спам»
Слова обучающей выборки	ноутбуки	1	0		
	выгодной	1	0		
	цене	1	0		
	распродажа	1	0		
	закажи	1	1	(1+1)/(13+9)	(1+1)/(13+6)
	велосипед	1	1	(1+1)/(13+9)	(1+1)/(13+6)
	получи	1	0		
	наушники	1	0		
	подарок	1	0		
	завтра	0	1		
	состоится	0	1		
	конференция	0	1		
	коньки	0	1	(1+0)/(13+9)	(1+1)/(13+6)
	пару	0	0	(1+0)/(13+9)	(1+0)/(13+6)
	сайте	0	0	(1+0)/(13+9)	(1+0)/(13+6)
	представлены	0	0	(1+0)/(13+9)	(1+0)/(13+6)
	одну	0	0	(1+0)/(13+9)	(1+0)/(13+6)

фильтрации. β_{yk} — потери при отнесении объекта класса y классу k ($\beta_{yy}=0,\ \beta_{yk}>0$ при $y\neq k$).

При определении спама: $y=1$ — спам, $y=0$ — не спам, $\beta_{01}>\beta_{10}$ (т.е. потеря при пропуске спама является меньшей потерей, чем ложное обнаружение).

Если полагать, что потери определяются только истинной классификацией объекта, а не тем, к какому классу он был ошибочно присвоен, то $\beta_{yk}\equiv\beta_y,\ \forall y,k\in Y$.

При априорных вероятностях $P(y)$ и функции правдоподобия $p_y(x)$, $\beta_{yk}\equiv\beta_y,\ \forall y,k\in Y$, $\beta_{yy}=0$, средний риск минимизируется благодаря алгоритму $\alpha(x)=\arg\max\limits_{y\in Y}\beta_y P_y p_y(x)$.

Согласно определению условной вероятности, имеем: $p(x,y)=p_y(x)P_y=P(y|x)p(x)$. Условная вероятность $P(y|x)$ является апостериорной вероятностью класса y для объекта x. Для её вычисления применим формулу Байеса:

$$P(y|x)=\frac{p(x,y)}{p(x)}=\frac{p_y(x)P_y}{\sum\limits_{k\in Y}p_k(x)P_k}.$$

С помощью апостериорной вероятности алгоритм $\alpha(x)$ примет вид: $\alpha(x)=\arg\max\limits_{y\in Y}\beta_y P(y|x)$.

При условии равнозначности классов $\beta_y\equiv 1$ речь идёт о максимальной $P(y|x)$. Если классы равновероятны, то объект x относится к классу с наибольшей плотностью распределения, $\alpha(x)=\arg\max\limits_{y\in Y}p_y(x)$.

Допустим, почтовой системе, которая прошла обучение на определённом числе входящих писем (относящихся к двум классам: спам и не спам), необходимо отнести следующее послание к одному из рассматриваемых при обучении классов.

Считается, что в письме слова не зависят друг от друга. Использование, так называемого «наивного», байесовского алгоритма связано с предположением

Таблица 3.

Спам	Не спам
8.01E-10	4.47E-09

```
C:\Python39\python.exe C:/Projects/SpamLearn/main.py
Веса: спам - 8.018095597354795e-10, не спам - 4.474914940788836e-09
Не спам
```

Рис. 1. Результат работы программы

```
      # библиотека из которой берётся список знаков пунктуации
from string import punctuation
# библиотека из которой берётся список "стоп-слов"
from stop_words import get_stop_words

spam_line = ['Ноутбуки по выгодной цене',
       'Распродажа! Закажи велосипед и получи наушники в
подарок']
# список с "не спам-сообщениями"
not_spam_line = ['Завтра состоится конференция',
         'Закажи коньки и велосипед']

# проверочное сообщение на спам
search_spam_line = 'На сайте представлены коньки. Закажи одну
пару и велосипед'
```

Рис. 2

независимости и равновозможности всех рассматриваемых параметров. Следует заметить, что эти, не совсем корректные на практике, предположения оправдывают себя при практическом применении. Отсюда и вытекает «наивность» алгоритма.

$P(y|x)$ вычисляется по формуле Байеса для каждого класса (с помощью создания частотных таблиц для всех объектов (относительно искомого результата), из которых создаются таблицы правдоподобия). Класс с наибольшей $P(y|x)$ и является искомым.

2. Задача определения спама

Приведём пример реализации байесовского алгоритма.

Пусть системе в качестве обучающей выборки предложены следующие сообщения (см. табл. 1):

В качестве проверяемого сообщения на спам выберем следующее послание: «На сайте представлены коньки. Закажи одну пару и велосипед».

Выполним математические расчёты и проведём проверку с помощью программного кода.

Для подсчёта вероятностей воспользуемся формулой

$$\frac{\beta + n}{\beta V + N},$$

где β — параметр сглаживания (положим его равным 1), n — число попаданий слова в документ класса, N — число слов документа класса, V — объём обучающей выборки.

Внесём данные в таблицу 2:

```
# функция для форматирования сообщения (принимает строку,
возвращает кортеж)
def clear_line(line_clearing: str) -> tuple:
  # вся строка переводится в нижний регистр
  line_clearing = line_clearing.lower()
  # проходимся по списку знаков пунктуации
  for i in punctuation:
    # заменяем знак пунктуации на "пустоту"
    line_clearing = line_clearing.replace(i, '')
  # разбиваем строку на список слов
  list_words = line_clearing.split()
  # проходимся по списку "стоп-слов"
  for i in get_stop_words('ru'):
    # если слово попалось в списке готовых слов
    if i in list_words:
      # удаляем слово из списка
      list_words.remove(i)
  # возвращаем кортеж готового списка
  return tuple(list_words)

# функция проверки на спам (принимает строку и словарь,
возвращает булевое значение)
def check_spam(line_check: str, table_info: dict) -> bool:
  # веса "спам" и "не спам"
  result = [1, 1]
  # количество слов из обучающей выборки,
  # количество слов входящих в "спам" и в "не спам"
  count = [0, 0, 0]
  ### заполнение count ###
  # проход по всем элементам "спам"
  for i in table_info['spam']:
    # добавление количества "встречаний" к общему счётчику
    count[0] += table_info['spam'][i]
    # добавление количества "встреваний" к локальному счётчику
"спам"
    count[1] += table_info['spam'][i]
  # проход по элементам "не спам"
  for i in table_info['not_spam']:
    # если элемента нет в "спам"
    if i not in table_info['spam']:
      # добавление количества "встречаний" к общему счётчику
      count[0] += table_info['not_spam'][i]
    # добавление количества "встречаний" к локальному счётчику
"не спам"
    count[2] += table_info['not_spam'][i]
  # проход по всех итоговым словам проверочной строки после
форматирования
  for i in clear_line(line_check):
    # если слова нет в "спам"
    if i not in table_info['spam']:
      # добавляем слово со значением 0
```

Рис. 3

```
result[1]))
  # если веса "спам" больше весов "не спам"
  if result[0] > result[1]:
    # возвращаем true (спам)
    return True
  # иначе, если "не спам" больше "спам"
  else:
    # возвращаем false (не спам)
    return False

# функция для "обучения" (принимает список "спам" и список "не
спам", возвращает словарь)
def learn_spam(spam: list, not_spam: list) -> dict:
  # создаём "чистый лист" словаря
  dict_words = {'spam': {}, 'not_spam': {}, 'count_in_spam': 0,
'count_in_not_spam': 0}
  # буферные списки для слов
  spam_words, not_spam_words = [], []
  # проход по списку "спам"
  for i in spam:
    # объединение результата форматирования в один список
    spam_words.extend(clear_line(i))
  # проход по буферному списку "спам"
  for i in spam_words:
    # добавляем в результативный словарь в словарь "спам"
    # слово и в качестве значения - количество повторений
    dict_words['spam'][i] = spam_words.count(i)
  # проход по списку "не спам"
  for i in not_spam:
    # объединение результата форматирования в один список
    not_spam_words.extend(clear_line(i))
  # проход по буферному списку "не спам"
  for i in not_spam_words:
    # добавляем в результативный словарь в словарь "не спам"
    # слово и в качестве значения - количество повторений
      dict_words['not_spam'][i] = not_spam_words.count(i)
    # добавление в результативный список длинн обучающих списков
    dict_words['count_in_spam'], dict_words['count_in_not_spam'] =
  len(spam), len(not_spam)
    # возвращаем "обученный" словарь
    return dict_words

  # вызываем функцию проверки на спам, передавая в неё проверяемую
  строку и
  # "обученный" списками "спам" и "не спам" словарь
  if check_spam(search_spam_line, learn_spam(spam_line,
  not_spam_line)):
    # если функция вернула "true"
    print('Спам')
  else:
    # если функция вернула "false"
    print('Не спам')
```

Рис. 4

Получаем следующий результат для класса «Спам»:

$$\frac{2}{4}\cdot\frac{1}{22}\cdot\frac{1}{22}\cdot\frac{1}{22}\cdot\frac{2}{22}\cdot\frac{1}{22}\cdot\frac{1}{22}\cdot\frac{2}{22}=\frac{1}{1247178944}=$$

$$=8{,}01809560E-10.$$

Для «Не спам»:

$$\frac{2}{4}\cdot\frac{1}{19}\cdot\frac{1}{19}\cdot\frac{2}{19}\cdot\frac{2}{19}\cdot\frac{1}{19}\cdot\frac{1}{19}\cdot\frac{2}{19}=\frac{16}{3575486956}=$$

$$=4{,}47491494E-9.$$

Реализуем задачу на языке программирования Python.

В результате работы программы получаем веса, равные (см. табл. 3, рис. 1):

Так как веса класса «Спам» меньше, чем веса класса «Не спам», можно сделать вывод, что сообщение не является спамом, что подтверждается программой.

Листинг программы — рис. 2, 3, 4.

Заключение

В работе описано применение «наивного» байесовского классификатора к фильтрации спам-сообщений. Изложен соответствующий математический аппарат и предложено детальное описание алгоритма на конкретном примере. Приведена программа для решения поставленной задачи на языке программирования Python.

ЛИТЕРАТУРА

1. Гмурман В.Е. Теория вероятностей и математическая статистика: учебное пособие для вузов. 11 изд. М.: Высшая школа, 2005. 479 с.
2. Высокоуровневый язык программирования Python [Электронный ресурс]. — Режим доступа: https://www.python.org/ (дата обращения: 5.01.2022)
3. Barber D. Bayesian reasoning and machine learning. Cambridge University Press, 2012. 642 p.
4. Охлупина О.В., Прокопенко А.А., Згонникова А.О. О ёмкости модели классификации // Учёные записки Брянского государственного университета. Брянск: БГУ, 2021 (4). С. 22–27.

Журнал «Современная наука: актуальные проблемы теории и практики»