Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовая работа по курсу «Дискретный анализ»
на тему: «Автоматическая классификация документов»

Студент: Н. П. Ежов
Преподаватель: С. А. Сорокин
Группа: М8О-307Б

Дата: 28.04.2022
Оценка:
Подпись:

Москва, 2022

Курсовая работа

Задача: Требуется разработать программу, получающую на вход сначала обучающие, а потом тестовые данные. При помощи наивного алгоритма Байеса программа должна обучится на первой части данных и классифицировать вторую часть.

Формат входных данных: Одно из самых частых применений наивного Байесовского классификатора является определение писем со спамом. Будем считать что 0 класс текста это спам, а 1-не спам. По итогу входные данные будут выглядеть следующим образом: идут n строк формата <класс>(0 - спам, 1- не спам) <текст письма> (оканчивающийся символом конца строки). После этого следует m строк вида <текст письма>, каждую из которых надо классифицировать.

Формат результата: Число 0, если тестовое сообщение является спамом или 1, если нет.

1 Описание

Требуется написать реализацию наивного Байесовского классификатора. Как сказано в [1], для работы классификатора: «Необходимо рассчитать оценку для каждого класса (спам/не спам) и выбрать ту, которая получилась максимальной.».
Для этого используем следующую формулу:

$$\arg\max [P(Q_k) \prod_{i=1}^{n} P(x_i|Q_k)]$$

$P(Q_k) = \frac{\text{число документов класса } Q_k}{\text{общее количество документов}}$
$P(x_i|Q_k) = \frac{\alpha+N_{ik}}{\alpha M+N_k}$ - вероятность принадлежности слова x_i к классу Q_k (со сглаживанием*)
N_k - количество слов, входящих в документ класса Q_k (сначала я думал, что N_k колво уникальных слов, но проверив формулу в статье на сайте [2] понял, что ошибся и внёс соответсвующие коррективы в программу)
M - количество уникальных слов из обучающей выборки
N_{ik} - количество вхождений слова x_i в документ класса Q_k
α - параметр для сглаживания
*Во время выполнения классификации может встретиться слово, которого в обучающей выборке не было. Чтобы оценка не было нулевой в любом случае, мы применяем сглаживание Лапласса, говоря проще, прибавляем ко всем вхождениям коэффициент α, который в нашей программе равен единице. Тогда даже те слова, которые в обучающей выборке не были, не будут давать нулевую вероятность.
Т.к. вероятности при большом объёме данных будут очень маленькими, их перемножение друг на друга может привести к арифмитическому переполнению снизу. Чтобы этого избежать, преобразуем формулу по свойству логарифма** и разобьём операции перемножения на операции сложения логарифмов

$$\log ab = \log a + \log b$$

**Логарифм – монотонно возрастающая функция. Как видно из первой формулы – мы ищем максимум. Логарифм от функции достигнет максимума в той же точке (по оси абсцисс), что и сама функция. Это упрощает вычисление, ибо меняется только численное значение. Подставляем и получаем:

$$\arg\max [\log P(Q_k) \sum_{i=1}^{n} \log P(x_i|Q_k)]$$

2 Исходный код

В программе байесовский классификатор сделан в виде отдельного класса, вынесенного в заголовочный файл. Внутри класса имеются методы для обучения на наборе входных данных и функция классификации переданной строки. Для упрощения представления обучающих данных была создана отдельная структура - CategoryItem, она содержит в себе лейбл класса и текст данного элемента выборки. Также были созданы несколько утилитарных функций, например, SplitIntoWords. Она принимает на вход строку, разбивает её на слова, после чего каждое слово переводит в нижний регистр и удаляет из него все не буквы.

```cpp
//
// BayersClassifier.h
//

//
// Created by nezhov on 27.04.22.
//

#pragma once
#include <map>
#include <vector>
#include <sstream>
#include <algorithm>
#include <numeric>
#include <cctype>
#include <set>
#include <iterator>
#include <cmath>
#include <string>
using namespace std;

void strTolower(string& arg);

vector<string> SplitIntoWords(const string& line);

struct CategoryItem{
    bool category;
    string text;
};

class BayersClassifier {
public:
    void learnClassifier(const vector<CategoryItem>& trainingSample);
    bool classify(const string& str);
private:
    double M = 0.;

    double P1 = 0.;
    double P2 = 0.;

    double N1 = 0.;
    double N2 = 0.;

    set<string> fClassWords;
    set<string> sClassWords;

    map<string, double> fClassCount;
    map<string, double> sClassCount;

```

```
50      void CountWords(const string& line, set <string>& curUnique, map<string, double>&
            curCount);
51
52      double calculateP(const string &arg, double Nk, const map<string, double> &
            ClassCount, double Pq);
53  };
```

```
1   //
2   // BayersClassifier.cpp
3   //
4
5   //
6   // Created by nezhov on 27.04.22.
7   //
8
9   #include "BayersClassifier.h"
10
11  void strTolower(string& arg){
12      transform(cbegin(arg), cend(arg), begin(arg),
13                    [](unsigned char c)-> unsigned char{ return tolower(c); });
14      arg.erase(remove_if(arg.begin(),
15                          arg.end(),
16                          [](unsigned char c){return !('a' <= c && c <= 'z');}),
17                arg.end());
18  }
19
20  vector<string> SplitIntoWords(const string& line) {
21      istringstream words_input(line);
22      vector<string> res {istream_iterator<string>(words_input), istream_iterator<string
            >()};
23      for(string& str: res){
24          strTolower(str);
25      }
26      return res;
27  }
28
29
30  void BayersClassifier::learnClassifier(const vector<CategoryItem> &trainingSample) {
31      fClassWords.clear();
32      sClassWords.clear();
33
34      fClassCount.clear();
35      sClassCount.clear();
36
37
38
39      double subSum1 = 0.;
40      double subSum2 = 0.;
41      for(const auto & [category, line]: trainingSample){
42          if(category){
```

```
43              CountWords(line, fClassWords, fClassCount);
44              ++subSum2;
45          }
46          else{
47              CountWords(line, sClassWords, sClassCount);
48              ++subSum1;
49          }
50      }
51
52      N1 = accumulate(fClassCount.begin(), fClassCount.end(), 0.,
53                      [](double value, const pair<string, double> &valArg){
54          return value + valArg.second;
55      });
56      N2 = accumulate(sClassCount.begin(), sClassCount.end(), 0.,
57                      [](double value, const pair<string, double> &valArg){
58                          return value + valArg.second;
59                      });;
60
61      P1 = log(subSum1/(subSum1+subSum2));
62      P2 = log(subSum2/(subSum1+subSum2));
63
64      M = static_cast<double>(fClassWords.size()) + static_cast<double>(sClassWords.size
            ());
65  }
66
67  bool BayersClassifier::classify(const string &str) {
68      return calculateP(str, N1, fClassCount, P1) > calculateP(str, N2, sClassCount, P2);
69  }
70
71  double BayersClassifier::calculateP(const string &arg, double Nk, const map<string,
        double> &ClassCount, double Pq) {
72      double subM = M + Nk;
73      double alpha = 1.;
74
75      double subRes = 0.;
76      for(const string& word: SplitIntoWords(arg)){
77          try{
78              double Nik = 0.;
79              Nik = ClassCount.at(word);
80              subRes += log((alpha + Nik) / subM);
81          }catch(out_of_range &e) {
82              subRes += log((alpha) / subM);
83          }
84      }
85      return Pq + subRes;
86  }
87
88  void BayersClassifier::CountWords(const string &line, set<string> &curUnique, map<
        string, double> &curCount) {
```

```
89      for(const string& word: SplitIntoWords(line)){
90          curUnique.insert(word);
91          ++curCount[word];
92      }
93  }
```

BayersClassifier.h	
void learnClassifier(const vector<CategoryItem>& trainingSample)	Функция, которая обучает классификатор.
bool classify(const string& str)	Функция для классификации переданной строки. Возвращает false или true в зависимости от того, вероятность принадлежности к какому классу у переданного текста больше

3 Консоль

```
nezhov@killswitch:~/CLionProjects/Bayes_Classifier/cmake-build-debug$ cat test.txt
10 5
1
You have 5 notifications .
0
-70% sale on BLACK FRIDAY
0
You have won 1000000$ !!!
1
Congratulations we won first round !
0
Notification that you won 1000000$
1
Will you go on shopping during black friday ?
0
You can buy a car right now !
0
You have -100% sale ! You can get air conditioner for free right now !
1
How much is 500$ in rubles ?
1
Hi ! We want to buy a car .
Is 500$ a lot ?
How much is 1000000$ in rubles ?
You got -70% sale on air conditioner !!!
Our new black car costs about 1000000$ ! We never had a car before . We will
buy it next friday .
Congratulations ! This is notification that you can still get from -70% to
-100% sale !!!
nezhov@killswitch:~/CLionProjects/Bayes_Classifier/cmake-build-debug$ ./Bayes_Classifi
<test.txt >res.txt
nezhov@killswitch:~/CLionProjects/Bayes_Classifier/cmake-build-debug$ cat res.txt
1
1
0
1
0
```

4 Выводы

Выполнив курсовой работу я изучил работу наивного байесовского классификатора и понял, что, несмотря на свою простосту, это может быть мощный инструмент для бинарной классификации данных. Предположу, что с увеличением классов классификатор будет работать хуже, т.к. вероятности могут стать *ближе* друг к другу, из-за чего сравнивать их будет проблематично. С другой стороны, формула была переведена в логарифмы, что должно упростить работу с классами в количестве больше двух. В любом случае, чистую формулу напрямую лучше не использовать, т.к. почти мгновенно достигается арифмитическое переполнение снизу.

Список литературы

[1] *Наивный Байес, или о том, как математика позволяет фильтровать спам.* URL: `https://habr.com/ru/post/415963/` (дата обращения: 27.04.2022).

[2] *Наивный байесовский классификатор.* URL: `http://bazhenov.me/blog/2012/06/11/naive-bayes.html` (дата обращения: 27.04.2022).