Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Дискретный анализ»

Студент: И. Д. Черненко
Преподаватель: А. Н. Ридли
Группа: М8О-306Б

Дата:
Оценка:
Подпись:

Москва, 2021

Классификация документов

Задача: Разработать наивный байесовский классификатор, с помощью которого провести бинарную классификацию документов.

1 Описание

Наивный байесовский классификатор — простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими (наивными) предположениями о независимости.

В зависимости от точной природы вероятностной модели, наивные байесовские классификаторы могут обучаться очень эффективно. Во многих практических приложениях для оценки параметров для наивных байесовых моделей используют метод максимального правдоподобия; другими словами, можно работать с наивной байесовской моделью, не веря в байесовскую вероятность и не используя байесовские методы.

Несмотря на наивный вид и, несомненно, очень упрощенные условия, наивные байесовские классификаторы часто работают намного лучше во многих сложных жизненных ситуациях.

Достоинством наивного байесовского классификатора является малое количество данных, необходимых для обучения, оценки параметров и классификации.[1].

2 Исходный код

Для хранения информации о каждом документе, на котором обучается классификатор, я создал структуру document, в которой хранится номер класса и вектор из всех слов данного документа. Также каждой из двух категорий сопоставляется условная вероятность появления каждого считанного слова, счётчик для каждого слова, вероятность того, что в данном классе не встретится новое слово, а также априорная вероятность. Также мы считаем общее количество слов и документов в данном классе.

Программа работает так: сначала считываются все размеченные документы, на которых будет обучаться алгоритм. После этого, подсчитываются необходимые данные для расчёта нужных вероятностей и для применения формулы Байеса. Для этого мы создаём unordered_set для подсчёта уникальных слов. После этого, для каждого класса считается вероятность, что новый документ будет и не будет принадлежать данному классу, а затем для каждого слова из входных данных ищется вероятность его появления в каждом из двух классов. На этом обучение классификатора заканчивается.

Предсказание класса нового документа происходит путём выбора наибольшей среди вероятностей того, что документ принадлежит первому или второму классу. Вероятность принадлежности классу складывается из априорной вероятности принадлежности классу и суммы вероятностей принадлежности данному классу всех слов текста.

```cpp
#include <iostream>
#include <cmath>
#include <string>
#include <vector>
#include <sstream>
#include <iterator>
#include <unordered_map>
#include <unordered_set>

struct document {
    std::vector<std::string> text_vec;
    int class_n;
};

struct category {
    std::unordered_map<std::string, long double> cond_prob;
    std::unordered_map<std::string, size_t> word_count;
    long double prob_word_not_found;
    long double prior_prob;
    size_t total_words = 0;
    size_t total_docs = 0;
};

class NaiveBayesClassifier {
public:
    NaiveBayesClassifier()= default;;
    void Train(size_t train_docs_number);
    int Test(const std::vector<std::string>& test_doc);
private:
    std::vector<document> train_docs;
    std::unordered_map<int, category> categories;
};

std::vector<std::string> Parse(std::string& text);
void NaiveBayesClassifier::Train(const size_t train_docs_number) {
    for (size_t i = 0; i < train_docs_number; i++) {
        document doc;
        std::string input_text;
        std::cin >> doc.class_n;
        std::cin.ignore();
        std::getline(std::cin, input_text);
        doc.text_vec = Parse(input_text);
        train_docs.push_back(doc);
        categories[doc.class_n].total_docs++;
        categories[doc.class_n].total_words += doc.text_vec.size();
        for (const auto& word : doc.text_vec) {
            categories[doc.class_n].word_count[word]++;
        }
    }
```

```
50
51      std::unordered_set<std::string> unique_words;
52      for (int i = 0; i <= 1; i++) {
53          for (auto & iter : categories[i].word_count) {
54              unique_words.insert(iter.first);
55          }
56      }
57      size_t unique_words_number = unique_words.size();
58
59      for (int i = 0; i <= 1; i++) {
60          categories[i].prior_prob = (long double)categories[i].total_docs /
                train_docs_number;
61          categories[i].prob_word_not_found = 1.0 /
62                 (long double)(categories[i].total_words + unique_words_number);
63      }
64
65      for (const auto & unique_word : unique_words) {
66          for (auto i = 0; i <= 1; i++) {
67              auto iter = categories[i].word_count.find(unique_word);
68              if (iter == categories[i].word_count.end()) {
69                  categories[i].cond_prob[unique_word] = categories[i].prob_word_not_found
                        ;
70              } else {
71                  categories[i].cond_prob[unique_word] =
72                                  (long double)(1.0 + categories[i].word_count[unique_word
                                      ])
73                                  / (categories[i].total_words + unique_words_number);
74              }
75          }
76      }
77  }
78
79  int NaiveBayesClassifier::Test(const std::vector<std::string>& test_doc) {
80      long double predicted_prob_for_class[2];
81      for (int i = 0; i <= 1; i++) {
82          predicted_prob_for_class[i] = categories[i].prior_prob;
83          for (auto word : test_doc) {
84              auto iter = categories[i].cond_prob.find(word);
85              if (iter == categories[i].cond_prob.end()) {
86                  predicted_prob_for_class[i] += categories[i].prob_word_not_found;
87              } else {
88                  predicted_prob_for_class[i] += categories[i].cond_prob[word];
89              }
90          }
91      }
92      if (predicted_prob_for_class[0] >= predicted_prob_for_class[1]) {
93          return 0;
94      } else {
95          return 1;
```

```
    }
}

std::vector<std::string> Parse(std::string& text) {
    for (char & i : text) {
        if (std::isupper(i)) {
            i = std::tolower(i);
        } else if (std::ispunct(i) && i != '\'') {
            i = ' ';
        }
    }
    std::istringstream ss{text};
    using string_iteratir = std::istream_iterator<std::string>;
    std::vector<std::string> container{string_iteratir{ss}, string_iteratir{}};
    return container;
}

int main() {
    size_t train_docs_number, test_docs_number;
    std::cin >> train_docs_number;
    std::cin >> test_docs_number;
    NaiveBayesClassifier nbc;
    nbc.Train(train_docs_number);
    std::vector<std::string> test_doc;
    std::string input_text;
    for (size_t i = 0; i < test_docs_number; i++) {
        if (!std::getline(std::cin, input_text)) {
            return 0;
        }
        test_doc = Parse(input_text);
        std::cout << nbc.Test(test_doc) << std::endl;
    }

    return 0;
}
```

3 Консоль

```
/mnt/d/Study/da_kp/cmake-build-debug/da_kp
4 2
0
omega fight charlie
0
omega
0
omega delta delta
1
and look at
i look at you and suddenly
fight omega charlie
1
0
/mnt/d/Study/da_kp/cmake-build-debug/da_kp
4 2
0
omega charlie fight omega
1
at suddenly
0
charley fight alpha
1
look and
fight fight alpha alpha and and
charley charlie look
0
0
```

4 Выводы

Наивный байесовски классификатор несмотря на то, что имеет весьма простую математическую основу является одним из самых популярных алгоритмов. К его положительным сторонам можно отнести: лёкгость в реализации, требуется малы объем данных для обучения. К отрицательным же сторонам омжно отнести: одним из ограничений НБА является допущение о независимости признаков, в реальности наборы полностью независимых признаков встречаются крайне редко и если в тестовом наборе данных присутствует некоторое значение категорийного признака, которое не встречалось в обучающем наборе данных, тогда модель присвоит нулевую вероятность этому значению и не сможет сделать прогноз. НБК можно применить в следующих случаях [2]:

- обработка данных в режиме реального времени;
- многоклассовая классификация;
- классификация текстов, фильтрация спама;
- создание рекомендательных систем.

Таким образом, данный алгоритм нашел себе множество применений и сегодня является эталоном для алгоритмов классификации.

Список литературы

[1] *Wiki*
URL: https://en.wikipedia.org/wiki/Naive_Bayes_classifier (дата обращения: 10.03.2020)

[2] *DataReview*
URL: http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/ (дата обращения: 14.03.2020)

[3] *ifmo*
URL: https://neerc.ifmo.ru/wiki/index.php?title=Байесовская_Классификация (дата обращения: 14.03.2020)

[4] *medium*
URL: https://medium.com/@dr.sunhongyu/machine-learning-c-naive-bayes-classifier-example-dbe7b88a999b (дата обращения: 14.03.2020)

[5] *towarddatascience*
URL: https://towardsdatascience.com/naive-bayes-implementation-from-scratch-using-c-51c958094041 (дата обращения: 14.03.2020)