Московский авиационный институт (национальный исследовательский университет)

Факультет компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Курсовая работа по курсу «Дискретный анализ»

Студент: А. А. Каримов
Преподаватель: С. А. Сорокин
Группа: М8О-306Б-22

Дата:
Оценка:
Подпись:

Москва, 2024

Курсовой проект

Задача: Необходимо реализовать наивный байесовский классификатор, который будет обучен на первой части входных данных и классифицировать им вторую часть.

Формат ввода

Вам даны данные в следующем формате: <training> <test> <class_1 >< $data_1$ >< $class_2$ >< $data_2$ > ... < $class_training$ >< $data_training$ >< $query_1$ >< $query_2$ > ... < $query_test$ > : ., 01, , ., .

Тексты могут содержать любые ascii символы.

Задача реализована как интерактивная, поэтому следующий документ для классификации будет передан, только после ответа на предыдущий.

Формат вывода

В ответ на каждый тестовый запрос выведите единственное число 0 или 1 — предполагаемый класс документа.

1 Описание

Наивный байесовский классификатор (Naive Bayes classifier) — вероятностный классификатор на основе формулы Байеса со строгим (наивным) предположением о независимости признаков между собой при заданном классе. Использоваться будет формула Байеса в виде:

$$P(class|question) = \frac{P(question|class)P(class)}{P(question)},$$

где $P(class)$ - вероятность встретить вопрос класса $class$ среди train-выборки
$P(question)$ - вероятность вопроса $question$ в выборке
$P(question|class)$ - вероятность встретить вопрос $question$ среди всех вопросов класса $class$
$P(class|question)$ - вероятность, что вопрос $question$ принадлежит классу $class$

Предположение независимости признаков: Если в обычной речи слова сильно зависят от контекста, то мы делаем предположение о том, что слова в вопросе друг от друга не зависят. Это даёт нам право представить формулу Байеса в следующем виде:

$$P(class|question) = \frac{P(w_1|class)P(w_2|class)...P(w_n|class)}{P(w_1, w_2, ..., w_n)} = \frac{\prod_{i=1}^{n} P(w_i|class)}{P(w_1, w_2, ..., w_n)},$$

где w_i - это i-ое слово в вопросе question.

Поскольку знаменатель данной дроби зависит только от признаков, а не от класса, его можно опустить и продолжать вычисления по следующей формуле:

$$P(class|question) \approx P(class) \prod_{i=1}^{n} P(w_i|class)$$

Логарифмирование функции: Мы будем по вышеописанной формуле перемножать множество вероятностей $P(w_i|class)$, из-за чего значение функции может быть настолько малым, что мы получим большую погрешность вычислений. Во избежание этого прологарифмируем функцию. Благодаря монотонности log параметры, при которых достигаются оптимальные значения, останутся теми же. Итого получим новую формулу расчета:

$$\log P(class|question) \approx \log P(class) + \sum_{i=1}^{n} \log P(w_i|class)$$

Расчет использующихся вероятностей: Расчет вероятностей будем проводить на основе их частотности:

$$P(class_i) = \frac{amountClass_i}{totalAmountClasses},$$

где $P(class_i)$ - вероятность класса $class_i$
$amountClass_i$ - количество классов $class_i$ в train-выборке
$totalAmountClasses$ - количество всего различных классов в train-выборке

$$P(w_i|class) = \frac{WordCountInClass_i}{UniqueWordCount},$$

где $WordCountInClass_i$ - количество вхождений слова w_i в вопросы класса $class$
$UniqueWordCount$ - количество уникальных слов среди всех классов.

2 Описание алгоритма

Этап обучения:

На вход нам дается файл, на основе которого с помощью словаря мы считаем частоты вхождений для каждого встреченного слова. Для каждого типа документа добавляем расчитанную статистику.

```
void BayesClassificator::GaussianNaiveBayes::fit(classType type, const text_t& text) {
    if(type == classType::DOCUMENT) {
        docWordCount += text.size();
        for(std::string word : text) {
            ++freqDoc[word];
        }
        ++docsEntries;
    } else {
        notDocWordCount += text.size();
        for(std::string word : text) {
            ++freqNotDoc[word];
        }
        ++notDocEntries;
    }
}
}
```

Этап классификации:

Для каждого типа документа считаем вероятность по вышеописанной формуле, используя статистики рассчитанные на этапе обучения. Из двух полученных значений выбираем большее, как наиболее вероятный тип документа.

```
BayesClassificator::classType BayesClassificator::GaussianNaiveBayes::predict(const
    text_t& text) {
    // calc doc probability
    double probDoc = log(docsEntries / static_cast<double>(docsEntries + notDocEntries)
        );

    for(const std::string& word : text) {
        probDoc += log(freqDoc[word] + 1e-6);
        probDoc -= log(docWordCount);
    }

    // calc notDoc prob
    double probNotDoc = (notDocEntries / static_cast<double>(docsEntries +
        notDocEntries));

    for(const std::string& word : text) {
        probNotDoc += log(freqNotDoc[word] + 1e-6);
        probNotDoc -= log(notDocWordCount);
    }

```

```
18      return (probDoc >= probNotDoc) ? BayesClassificator::classType::DOCUMENT :
            BayesClassificator::classType::NOT_DOCUMENT;
19  }
```

3 Консоль

```
karseny99@karseny99:/mnt/study/DA/lab4/src$  g++ -std=c++20 contest.cpp &&
./a.out <1.in
0
1
```

4 Выводы

В ходе выполнения курсового проекта я смог реализовать наивный Байесовский классификатор. До этого я уже встречался с этим классификатором в статье Пола Грэма (2003). В те времена в электронную почту не были встроены умные системы фильтрации спама, поэтому Пол применил Байесовскую фильтрацию к своим приходящим письмам. На примерах он показывал, как определенные словесные конструкции на фоне огромного текста сильно повышают вероятность на то, что письмо спам. При первом прочтении я мало что понял, но сейчас, познакомившись с курсом машинного обучения, с некоторой теорией я смог лучше разобраться в применениях вероятностной формулы Байеса.

Список литературы

[1] Блог *разработчика программного обеспечения.*
URL: http://bazhenov.me/blog/2012/06/11/naive-bayes.html

[2] Блог *разработчика программного обеспечения.*
URL: https://habr.com/ru/articles/802435/

[3] Блог *разработчика программного обеспечения.*
URL: https://www.bazhenov.me/blog/2012/07/21/classification-performance-evaluation

[4] Блог *разработчика программного обеспечения.*
URL: https://habr.com/ru/articles/661119/

[5] *Лекции по Машинному обучнию*