Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Дискретный анализ»

Студент: Т. Д. Голубев
Преподаватель: Н. К. Макаров
Группа: М8О-306Б-22
Дата:
Оценка:
Подпись:

Москва, 2025

Курсовой проект

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

1 Описание

Требуется написать реализацию наивного байесовского классификатора.

Наивный байесовский классификатор (Naive Bayes classifier) —- вероятностный классификатор на основе формулы Байеса со строгим (наивным) предположением о независимости признаков между собой при заданном классе, что сильно упрощает задачу классификации из-за оценки одномерных вероятностных плотностей вместо одной многомерной[1].

Формула Байеса:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

В контексте машинного обучения формула Байеса приобретает следующий вид:

$$P(y_k|X) = \frac{P(y_k)P(X|y_k)}{P(X)}$$

В конечном счёте правило классификации будет пропорционально выбору класса с максимальной апостериорной вероятностью:

$$y_k \propto \arg\max_{y_k} P(y_k) \prod_{i=1}^{n} P(X_i|y_k)$$

Чтобы получить вероятность $P(X_i|y_k)$ на практике, я вычисляю частоту слова X_i в классе y_k. Вероятность $P(y_k)$ – это отношение количества слов в классе y_k к количеству всех слов, использовавшихся для обучения модели.

Для того, чтобы вероятность не оказалась равной нулю, я использую сглаживание. При вычислении $P(X_i|y_k) = \frac{X_i}{X}$ я добавляю в числитель единицу.

Для задачи с подбором множества классов я отбираю их по порогу

$$\beta = 0.5 + \frac{1}{\max y_i},$$

где $\max y_i$ – это максимальное число слов в классе. Число 0.5 было выбрано эмпирически.

Отбор я осуществляю по отношению вероятности класса к максимальной вероятности:

$$\frac{P(X_i|y_k)}{\max P(X_i|y_k)} > \beta.$$

Если это условие выполняется, то класс включается в предсказание.

2 Исходный код

word_tools.h	
void ToLowerCase(std::string& str)	Преобразование строки в нижний регистр.
std::vector<std::string> Split(const std::string& s, const std::string& delimiters)	Разделить строку на слова.
std::vector<uint64_t> UIntSplit(const std::string& s, const std::string& delimeters)	Разделить строку на целые числа.
classifier.h	
void Fit(const std::vector<std::pair<TWord, TCategoryName»& data)	Обучение модели.
TCategoryName Predict(std::string& str)	Предсказать классы предложений.
void SaveToFile(const std::string& filename) const	Сохранить веса в файл.
void LoadFromFile(const std::string& filename)	Загрузить веса из файла.
static double Accuracy(const TConfusionMatrix& matrix)	Метрика accuracy.
static double Precision(const TConfusionMatrix& matrix, TCategoryName category)	Метрика точности.
static double Recall(const TConfusionMatrix& matrix, TCategoryName category)	Метрика полноты.
static double F1(const TConfusionMatrix& matrix, TCategoryName category)	F1 метрика.

```
using TWord = std::string;
using TCategoryName = uint64_t;
using TConfusionMatrix = std::vector<std::vector<uint64_t>>;

class TNaiveBayesClassifier {
private:
  using TCategory = std::unordered_map<TWord, uint64_t>;

  std::unordered_map<TCategoryName, TCategory> categoriesWeights;
  double beta = 0.5;
```

```
11 public:
12   TNaiveBayesClassifier() {}
13   void Fit(const std::vector<std::pair<TWord, std::vector<TCategoryName>>>& data);
14   std::vector<TCategoryName> Predict(std::string& str);
15   void SaveToFile(const std::string& filename) const;
16   void LoadFromFile(const std::string& filename);
17
18   std::unordered_map<TCategoryName, TCategory> GetCategoriesWeights() const {
19     return categoriesWeights;
20   }
21
22   uint64_t GetCategoriesNumber() const {
23     return categoriesWeights.size();
24   }
25
26   static double Accuracy(const TConfusionMatrix& matrix);
27
28   static double Precision(const TConfusionMatrix& matrix, TCategoryName category);
29
30   static double Recall(const TConfusionMatrix& matrix, TCategoryName category);
31
32   static double F1(const TConfusionMatrix& matrix, TCategoryName category);
33 };
```

3 Консоль

```
 cat-mood@nuclear-box:~/programming/mai-da-labs/kp/build$ ./kp_exe learn --input
learn.txt --output out.txt
 cat-mood@nuclear-box:~/programming/mai-da-labs/kp/build$ ./kp_exe classify
--input test.txt    --output ans1.txt --stats out.txt
```

4 Тест производительности

Тест производительности представляет из себя следующее: На вход подаётся 50 строк для обучения и 50 для предсказания. Строки для предсказания уже заранее определены в класс. На основе этого считаются метрики: accuracy, precision, recall, F1-мера.

```
cat-mood@nuclear-box:~/programming/mai-da-labs/kp/build$ ./kp_benchmark <test3.txt
Accuracy: 0.8
Precision: 1
Recall: 0.611111
F1: 0.758621
```

Как видно, модель имеет высокую точность, но слабую полноту. Это означает, что много ложно отрицательных срабатываний, то есть положительный класс не полный.

5 Выводы

Выполнив курсовой проект, я написал наивный байесовский классификатор. Поработал с такими метриками, как accuracy, precision, recall, F1-мера. Построил confusion matrix.

Список литературы

[1] Наивный байесовский классификатор. Основная идея, модификации и реализация с нуля на Python // Хабр URL: https://habr.com/ru/articles/802435/ (дата обращения: 25.12.2024).