

Курсовой проект по курсу дискретного анализа: Классификация документов

Выполнил студент группы М8О-312Б-22 МАИ *Юрков Евгений*.

Условие

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

Метод решения

Классификатор использует формулу Байеса для предсказаний:

$$P(c|d) \sim P(c) \prod_{1 \leq k \leq n_d} P(t_k|c),$$

где

- $P(c|d)$ - вероятность того, что документ d принадлежит классу c ,
- $P(c)$ - вероятность появления документа класса c ,
- $P(t_k|c)$ - вероятность того, что слово t_k появится в документе класса c .

Класс выбирается по формуле:

$$c_{map} = \arg \max_{c \in C} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c).$$

При перемножении вероятность может получиться очень маленькое число, при котором тип `double` может переполниться. Чтобы вычисления были более точными логарифмируем формулу:

$$c_{map} = \arg \max_{c \in C} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)].$$

Оценка вероятности класса равна отношению количества документов класса к общему количеству документов:

$$\hat{P}(c) = \frac{N_c}{N}$$

Оценка вероятности слова в классе рассчитывается по формуле:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}},$$

где

- T_{ct} - количество появлений слова t в документах класса c ,

- $\sum_{t' \in V} T_{ct'}$ - сумма появлений всех слов в документах класса c ,

Эта формула плохо справляется со словами, которые появляются только в своём классе. В таком случае для другого класса результат был бы равен нулю, тогда произведение в формуле Байеса стало бы так же равно нулю, хотя вероятности для других слов могли быть достаточно большими. Нужно чтобы вероятность таких слов была мала, но больше нуля. Для этого формула преобразуется следующим образом:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B'},$$

где B' - общее количество слов в словаре (без повторений).

Описание программы

Для предсказаний используется класс `NaiveBayes`, в котором реализованы функции:

- `void learn(const std::vector<std::string>& document, const std::vector<int>& classes)` - обучение модели на массиве предложений `document`, в котором каждое предложение соответствует своему классу.
- `std::vector<int> predict(const std::vector<std::string>& document) const` - осуществляет предсказание типа для каждого предложения из массива `document`.

Дневник отладки

1. Написан класс для предсказания двух классов: 0 или 1.
2. Исправлено вычисление вероятности для каждого слова.
3. Класс принимает заранее указанное число классов n .

Тест производительности

Алгоритм обучения использует разбиение предложения на массив из слов, подсчёт количества слов в предложении, а потом для каждого класса заполняет `unordered_map` вероятностей слов, то есть сложность предсказания $O(n \cdot t \cdot c)$, где n - количество документов, поданных для обучения, t - длина документа, c - количество предсказываемых классов.

Алгоритм предсказания рассчитывает формулу для каждого из классов и выбирает максимум, то есть сложность предсказания тоже $O(n \cdot t \cdot c)$.

Выводы

В ходе курсового проекта я реализовал программу для классификации документов с использованием наивного Байесовского классификатора. Я узнал теоретические основы его работы и укрепил навыки работы с текстовыми данными.

В ходе выполнения проекта сложности возникали при правильной организации структуры данных для классификации. Также была важной задача корректной работы с вероятностями, особенно в случае редких слов в тексте.

Полученные навыки полезны для дальнейшей работы с текстовыми данными и их обработкой, включая задачи анализа данных, создания систем рекомендаций или автоматического распределения текстов по категориям, например, для определения спама.