Домашнее задание №4: «Байес на страже SMS»

Дедлайн 1 (20 баллов): 30 марта, 23:59
Дедлайн 2 (10 баллов): 6 апреля, 23:59

Домашнее задание нужно написать на Python и сдать в виде одного файла. Правило именования файла: `name_surname_4.[py | ipnb]`. Например, если вас зовут Иван Петров, то имя файла должно быть: `ivan_petrov_4.py` или `ivan_petrov_4.ipnb`.

В данном домашнем задании предлагается реализовать наивный Байесов классификатор для определения спама в SMS-сообщении. По ссылке [1] находится датасет, размеченных сообщений. Первое слово строки это идентификатор класса spam или ham, далее через табуляцию следует сообщение.

1 Прежде чем строить классификатор нужно привести данные в удобный для классификации формат. Для этого воспользуемся страндартной моделью для текстов под названием Bag of words [2].
Реализуйте функцию `vectorize`, принимающую на вход список строк длины N и возвращающую матрицу размера (N, M), где M размер словаря для входных данных. В качестве словаря будем использовать все слова, которые встречаются в переданном массиве. В каждой строке матрицы на j-м позиции находится число x, которое означает, что j-е слово встретилось в сообщении x раз.

2 Наивный Байесов классификатор работает в предположении о независимости признаков объекта. В нашем случае это означает, что вероятность встретить некоторое слово в сообщении не зависит от наличия других слов в этом сообщении.
Т.к. мы векторизовали сообщения с учётом частот встречаемости слов, мы будем использовать мультиномиальную модель классификатора и оценки будут несколько отличаться от тех, которые использовались на лекции.

---

[1] `https://gist.github.com/ktisha/6951a0b85cf040cdeb46819e51fb62dd`
[2] `https://en.wikipedia.org/wiki/Bag-of-words_model`

Оценку для параметров для каждого из признаков можно записать следующим образом:

$$\theta_{yj} = \frac{\sum_{i=1}^{l}[y_i = y]x_{ij}}{\sum_{j \in V}\sum_{i=1}^{l}[y_i = y]x_{ij}}$$

x_{ij} – значение j-го признака объекта i
V – словарь входных данных.

Другими словами, числитель описывает сколько раз слово встречается в сообщениях класса y (включая повторы), а знаменатель – это суммарное количество слов во всех документах этого класса.

Реализуйте метод `fit`, который по переданной выборке вычисляет следующие параметры, которые понадобятся на этапе классификации:

1. оценка априорной вероятности классов $\hat{P}_y$
2. относительные частоты слов для каждого класса
3. суммарное количество слов для сообщений каждого класса
4. размер словаря выборки

```
class NaiveBayes:
    def __init(self, alpha):
        self.alpha = alpha
        ...

    def fit(self, X, y):
        ...

    def predict(self, X):
        ...

    def score(self, X, y):
        ...
```

4 Реализуйте метод `predict`, принимающий массив объектов `X` и возвращающий список соответствующих меток классов. Алгоритм классификации выглядит следующим образом:

$$a(x) = \arg\max_{y \in Y} P_y \prod_{j=1}^{|V|} p(x^j|y)$$

Однако при достаточно большой длине сообщения придётся перемножать большое количество очень маленьких чисел. Стандартный способ избежать арифметического переполнения снизу[3] – применение логарифма к выражению, стоящему под $\arg\max$. Таким образом формула для нашего алгоритма переписывается следующим образом:

$$a(x) = \arg\max_{y \in Y}[\log(P_y) + \sum_{j=1}^{|V|} log(p(x^j|y))]$$

[3] `https://en.wikipedia.org/wiki/Arithmetic_underflow`

Для решения проблемы неизвестных слов при классификации воспользуйтесь сглаживанием Лапласа для $p(x^j|y)$. В нашей модели данных аддитивное сглаживание[4] выражается в добавлении α в числитель и $\alpha * |V|$ в знаменатель выражения из пункта 2.

6 Реализуйте метод `score` для оценки работы классификатора, вычисляющий процент правильно классифицированных объектов.

[4] `https://en.wikipedia.org/wiki/Additive_smoothing`