Работа с текстовыми и категориальными данными

Кантонистова Елена

elena.kantonistova@yandex.ru

1 ноября 2017

План лекции

1. Работа с категориальными признаками

2. Работа с текстами

Категориальные признаки

Категориальные признаки - это признаки, которые нельзя воспринимать как числа.
Примеры: названия городов, названия фильмов, марки машин и т.д.

Категориальные признаки необходимо как-либо закодировать числами, чтобы с ними могли работать алгоритмы машинного обучения.

Нумерация категориальных признаков

На первый взгляд, кажется, что можно закодировать категориальные признаки подряд идущими числами, например:

$$\textit{Москва} \rightarrow 1,$$

$$\textit{Санкт-Петербург} \rightarrow 2,$$

$$\textit{Новосибирск} \rightarrow 3,$$

$$\textit{Казань} \rightarrow 4,$$

$$\ldots .$$

Однако, порядок нумерации никак не обоснован, т.е. неравенство 2 (Санкт-Петербург) $>$ 1 (Москва) не несёт никакого смысла. А алгоритм будет сравнивать числа, что приведет к непредсказуемым результатам.

One-Hot Encoding

Данный метод (OHE) преобразует каждое категориальное значение в строку из 0 и 1. Например, если у нас в предыдущем примере в стоблце городов встречается все 4 различных города, то в этом случае One Hot Encoding подразумевает создание 4 признаков, все из которых равны нулю за исключением одного. На позицию, соответствующую численному значению признака мы помещаем 1:

Москва - 1, 0, 0, 0
Санкт-Петербург - 0, 1, 0, 0
Новосибирск - 0, 0, 1, 0
Казань - 0, 0, 0, 1
Тем самым, один столбец "Город"превратится в четыре столбца с 0 и 1.

Плюсы и минусы OHE

- Плюс: алгоритмы машинного обучения получают на вход числа. Проблема нумерации решена. Можно применять машинное обучение!
- Минус: если в столбце с категориальным признаком много различных значений, то после OHE у нас в матрице появится очень много новых числовых столбцов, и алгоритму придется иметь дело с матрицей огромных размеров, что очень ресурсозатратно.

Разреженные матрицы

После применения OHE мы получим матрицу, в которой очень много нулей. Такая матрица называется разреженной (sparse matrix). OHE по умолчанию в результате своей работы вернет нам разреженную матрицу, чтобы не хранить избыточные данные.
Большинство алгоритмов умеет работать с таким форматом данных.

Кодирование данных с помощью OHE. Summary.

Чтобы закодировать категориальные данные с помощью one-hot encoding, необходимо:

- Занумеровать исходные данные последовательно идущими числами (sklearn LabelEncoder)
- Применить к столбцу полученных номеров OHE (sklearn OneHotEncoder)

Хэширование

Реальные данные могут оказаться гораздо более динамичными, и мы не всегда можем рассчитывать, что категориальные признаки не будут принимать новых значений. Все это сильно затрудняет использование уже обученных моделей на новых данных. Кроме того, LabelEncoder подразумевает предварительный анализ всей выборки и хранение построенных отображений в памяти, что затрудняет работу в режиме больших данных.

Hashing trick

Для решения этих проблем существует более простой подход к векторизации категориальных признаков, основанный на хэшировании (hashing trick). Этот метод использует некоторую функцию (хэш-функцию), которая переводит значения категориальных признаков в числа.
Идея метода состоит в группировке значений признака по разным параметрам - значения, попадающие в одну группу, получают одинаковый числовой номер.

Hashing trick

В данном методе мы можем задать число новых числовых столбцов, которые хотим получить после хэширования, то есть, по сути, число различных способов группировки категориальных признаков.

Плюсы и минусы хэширования

- Плюс: получаем заранее заданное число новых столбцов, тем самым экономим память и не работаем с огромными объемами данных
- Минус: непонятно, какое количество новых столбцов создать, чтобы не потерять много информации после хеширования.

Счетчики

Мы можем закодировать каждый признак частотой его встречаемости в столбце, т.е. например, если длина таблицы 1000 строк, а город Москва встречается в ней 100 раз, то Москву мы кодируем числом $\frac{100}{1000} = 0.1$.

Этот способ кодирования более экономный, чем предыдущие, однако мы, очевидно, теряем много информации при таком кодировании.

Счетчики

Можно придумывать и другие счетчики. Например, частоту встречания пары признаков в таблице и т.д.

Тексты

Теперь мы будем решать задачу с другой постановкой. Теперь нам дана не таблица с различными данными, а текст или несколько текстов.
Например, можно решать задачу определения тональности. Допустим, нам даны отзывы людей на различные фильмы, и нам необходимо обучить алгоритм определять, положительный отзыв или отрицательный.

Векторизация текста

Каждый отзыв необходимо векторизовать, то есть, перевести в числа.

CountVectorizer

- Создадим словарь всех слов, встречающихся в тексте (все отзывы)
- Тогда для каждого отзыва вектором является набор чисел, где каждое число - это количество раз, которое каждое слово из словаря вошло в данный отзыв

TF-IDF

Ещё один способ работы с текстовыми данными — TF-IDF (**T**erm **F**requency–**I**nverse **D**ocument **F**requency). Рассмотрим коллекцию текстов D. Для каждого уникального слова t из документа $d \in D$ вычислим следующие величины:

1. Term Frequency – количество вхождений слова в отношении к общему числу слов в тексте:
$$\mathrm{tf}(t, d) = \frac{n_{td}}{\sum_{t \in d} n_{td}},$$
где n_{td} — количество вхождений слова t в текст d.
2. Inverse Document Frequency
$$\mathrm{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|},$$
где $|\{d \in D : t \in d\}|$ – количество текстов в коллекции, содержащих слово t.

Тогда для каждой пары (слово, текст) (t, d) вычислим величину:

$$\text{tf-idf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D).$$

Отметим, что значение $\mathrm{tf}(t, d)$ корректируется для часто встречающихся общеупотребимых слов при помощи значения $\mathrm{idf}(t, D)$.

Признаковым описанием одного объекта $d \in D$ будет вектор $(\text{tf-idf}(t, d, D))_{t \in V}$, где V – словарь всех слов, встречающихся в коллекции D.

Word2Vec. Что это?

Word2Vec — это технология от гугл, которая заточена на статистическую обработку больших массивов текстовой информации. W2V собирает статистику по совместному появлению слов в фразах, после чего методами нейронных сетей решает задачу снижения размерности и выдает на выходе компактные векторные представления слов, в максимальной степени отражающие отношения этих слов в обрабатываемых текстах.

Word2Vec

Идея

Нахождение связей между контекстами слов основано на предположении, что слова, находящиеся в похожих контекстах, имеют тенденцию значить похожие вещи, т.е. быть семантически близкими. Формально задача стоит так: максимизировать косинусное расстояние между векторами слов (скалярное произведение векторов), которые появляются рядом друг с другом, и минимизировать косинусное расстояние между векторами слов, которые не появляются друг рядом с другом. Рядом друг с другом в данном случае значит в близких контекстах.

Применение. Поиск опечаток.

Enter word or sentence (EXIT to break): преключение

Word: преключение Position in vocabulary: 124515

Word Cosine distance
— приключение 0.748698
преключения 0.726111
приключения 0.692828
приключеия 0.670168
прключение 0.666706
приключеня 0.663286
прключения 0.660438
приключени 0.659609

Транслитерация

Enter word or sentence (EXIT to break): avito

Word: avito Position in vocabulary: 1999

Word Cosine distance
— awito 0.693721
авито 0.675299
fvito 0.661414
авита 0.659454
irr 0.642429
овито 0.606189
avito 0.598056

Семантически близкие слова

```
Enter word or sentence (EXIT to break): кофе
— коффе 0.734483
чая 0.690234
чай 0.688656
капучино 0.666638
кофн 0.636362
какао 0.619801
эспрессо 0.599390
кофя 0.595211
цикорий 0.594247
кофэ 0.593993
копучино 0.587324
шоколад 0.585655
капучинно 0.580286
кардамоном 0.566781
латте 0.563224
```

Важность слов в запросе

Enter word or sentence (EXIT to break): купить пиццу в москве

Importance купить = 0.159387
Importance пиццу = 1
Importance в = 0.403579
Importance москве = 0.455351

Информация о word2vec и примерах взята по ссылке https://habrahabr.ru/post/249215/.

Предобработка текстов

Перед векторизацией текста бывает полезно сделать его некоторую предобработку:

- Понизить регистр всех букв (Python -> python)
- Удалить стоп-слова (this, a/an, the...)
- Сделать любую другую предобработку, подходящую для конкретной задачи