Faiss / Annoy

МОЕ ПОГРУЖЕНИЕ В SIMILARITY SEARCH

Зачем нам вообще KNN

Есть 3 основных подхода в рекомендациях

SVD РАЗЛОЕНИЕ

Получаем 2 матрицы:
Описательные матрицы клиентов и товаров.
Скалярное произведение строки клинтов на столбец товара, степень близости клиента к данному товару.

ITEM-BASED METHOD

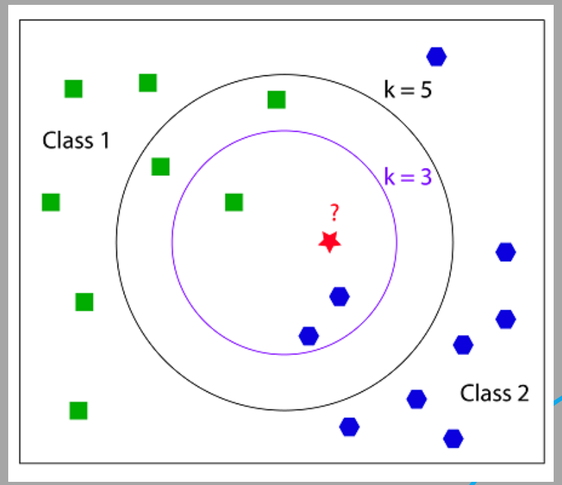
На основании используемых товаров клиента, подбираем к каждому наиболее близкие.

USER-BASED METHOD

Основываясь на поведении клиента ищем похожих на него. Рекомендуем те товары которые используют похожие клиенты.

Недостатки KNN

- Обучение состоит из запоминания всего датасета
- Поиск ближайших соседей по всем объектам
- Пересчет метрики каждый с каждым (Brute Force)



КАК СЛЕДСТВИЕ РАБОТАЕТ ОЧЕНЬ МЕДЛЕННО НА БОЛЬШИХ ОБЪЕМАХ

FAISS

И С ЧЕМ ЕГО ЕДЯТ?

ПОИСК БЛИЖАЙШИХ СОСЕДЕЙ

Тот же KNN, но хранит искомые объекты так, что по ним удобно искать

НАПИСАН НА C++

Имеет обертку на Python.
Способен работать в огромной размерности (млрд объектов)

СОЗДАН FACEBOOK

Применяется в Facebook для поиска похожих страниц в поисковой выдаче.

Где выигрвывает Faiss

ОСНОВНАЯ ИДЕЯ

Создать такую упорядоченную структуру данных, в которой можно осуществлять поиск быстрее

ПРЕДКЛАСТЕРИЗАЦИЯ

Разбиение всех данных на N кластеров
Поиск ближайшего кластера
Поиск ближайшего внутри кластера

ИЗМЕНЕНИЕ СТРУКТУРЫ ХРАНЕНИЯ

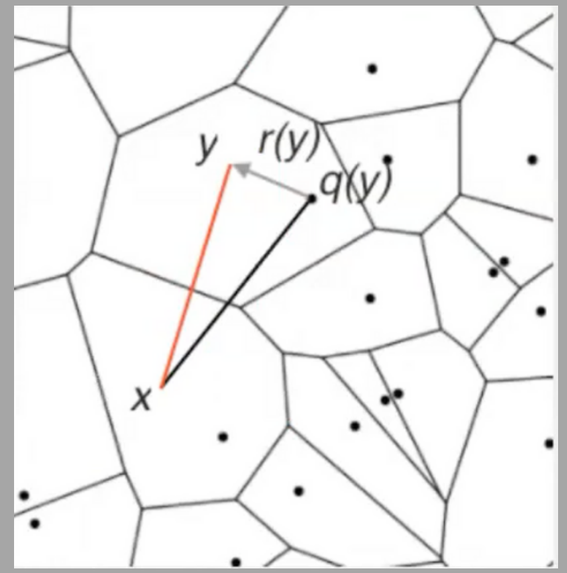
Переход от хранения таблицы:
object-> cluster_ID
к cluster_ID->[list of objects]

СЖАТИЕ ИСХОДНОГО ВЕКТОРА

Разбиваем исходный вектор на подвектора и каждый подвектор кодируем новым значением

ПРЕДКЛАСТЕРИЗАЦИЯ

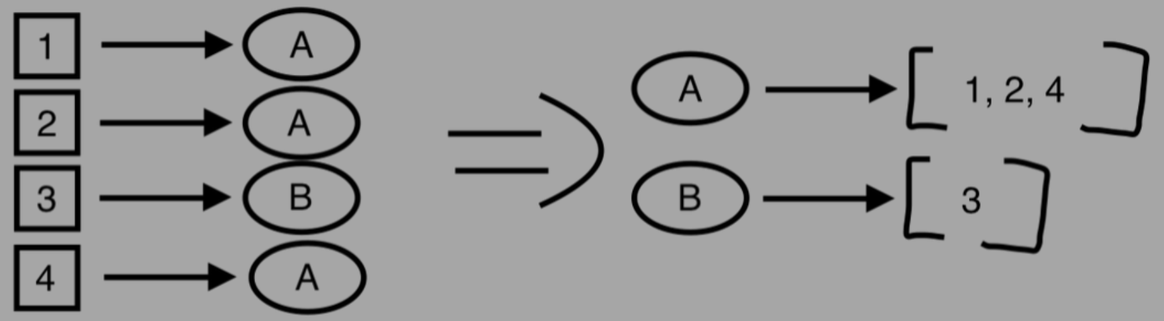
- Все объекты разбиваются на N кластеров
- У кластеров высчитываются центроиды
- У нового обьекта идет поиск ближайшего кластера
- Детальный поиск K ближайших соседей внутри M ближайших кластеров



ИЗМЕНЕНИЕ СТРУКТУРЫ ХРАНЕНИЯ

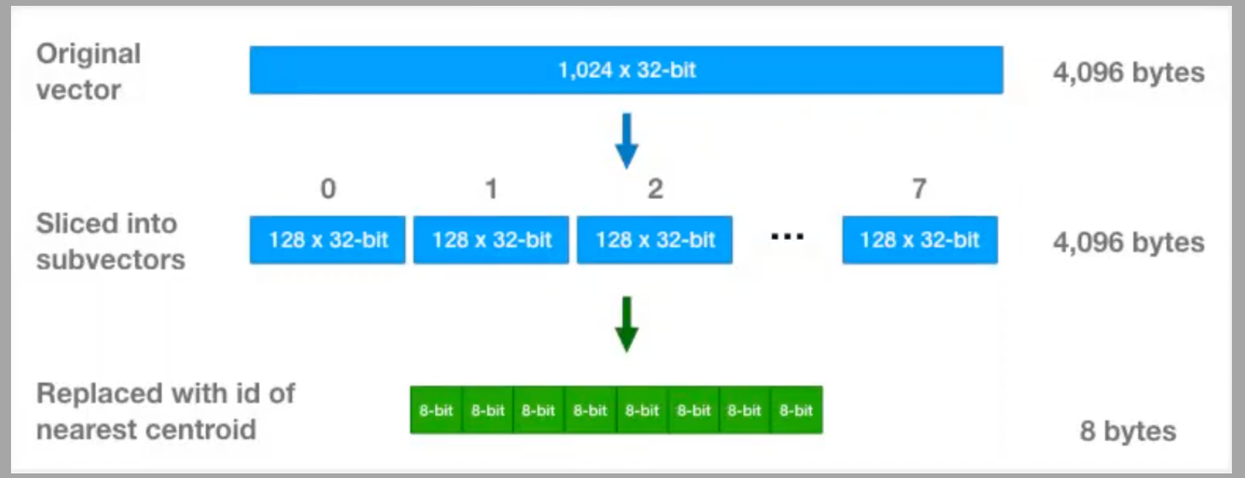
Переход от хранения таблицы:
object-> cluster_ID
k cluster_ID->[list of objects]

Поиск соседей будет только внутри искомыхх кластеров, не будут обходиться обекты других кластеров



СЖАТИЕ ИСХОДНОГО ВЕКТОРА

Разбиваем исходный вектор на подвектора и каждый подвектор кодируем новым значением



ОБУЧЕНИЕ И ПРИМЕНЕНИЕ

- Обучать структуру данных
- Добавлять новые объекты
- Поиск ближайших соседей

```
In [8]: import faiss

In [17]: index = faiss.index_factory(128, "IVF256,PQ32", faiss.METRIC_INNER_PRODUCT)
         index.train(X_dense)
         index.add(X_dense)
         index.nprobe = 10
```

```
faiss_result = index.search(row_dense, num_neighbours)
neighbors = faiss_result[1]
```

СОХРАНЯТЬ И ЧИТАТЬ

- Сохранять обученную и заполненную структуру данных
- Читать и искать без обучения

```
In [19]: faiss.write_index(index, '../tmp/u2u/faiss.idx')

In [20]: ! ls -lah ../tmp/u2u

         total 434512
         drwxr-xr-x  6 a18339743  staff   192B Feb 11 18:54 .
         drwxr-xr-x  7 a18339743  staff   224B Feb 11 18:49 ..
         -rw-r--r--  1 a18339743  staff    99M Feb 11 18:49 X_stored.pkl
         -rw-r--r--  1 a18339743  staff   2.0M Feb 17 09:54 faiss.idx
         -rw-r--r--  1 a18339743  staff    90M Feb 11 18:49 knn.pkl
         -rw-r--r--  1 a18339743  staff    21M Feb 11 18:49 svd.pkl

In [21]: new_index = faiss.read_index('../tmp/u2u/faiss.idx')
         new_index.is_trained

Out[21]: True
```

Annoy

ИЛИ ЧТО БЫ МНЕ ПОСЛУШАТЬ?

ПОИСК БЛИЖАЙШИХ СОСЕДЕЙ

Схож с Faiss, но кластеризует объекты некоторым случайным образом и использует подход RandomForest

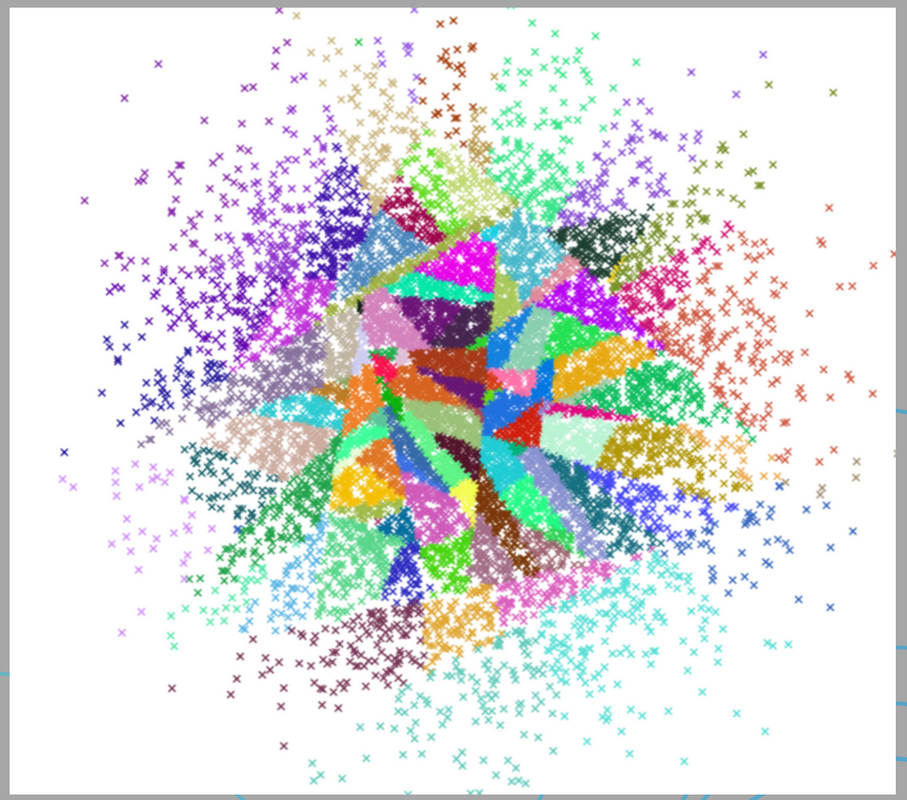
НАПИСАН НА C++

Имеет обертку на Python.
Способен работать в большой размерности (десятки млн объектов)

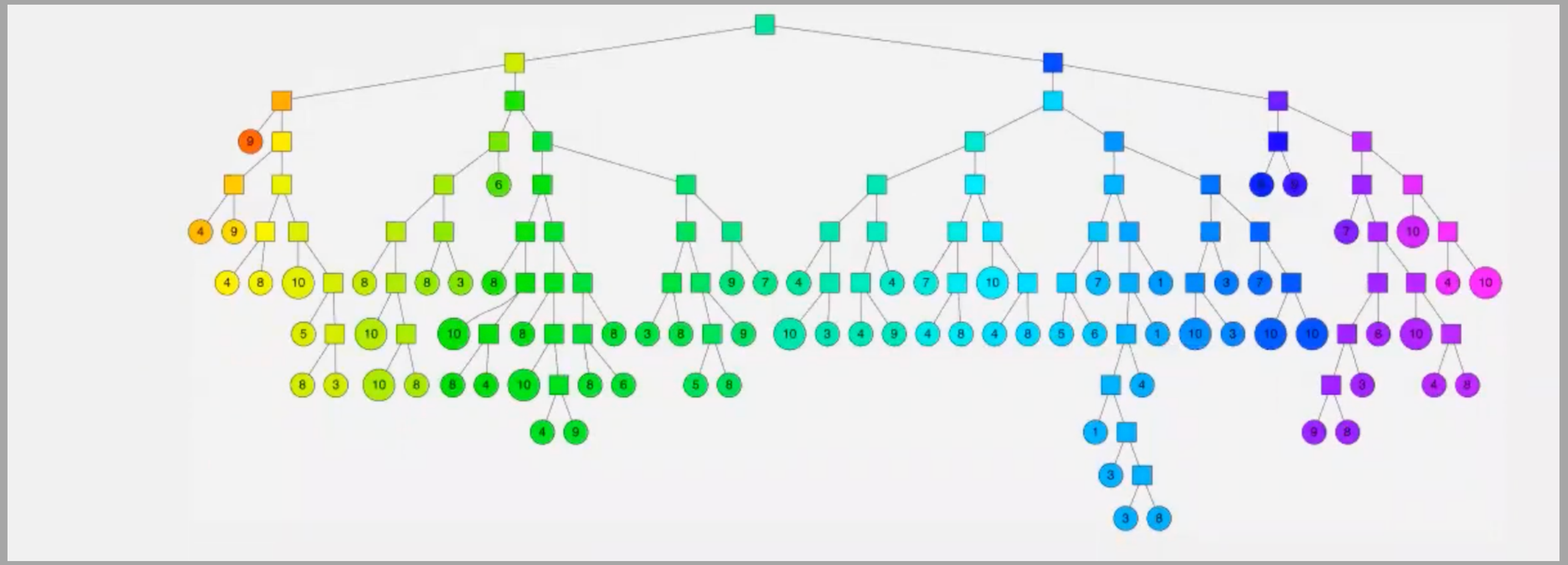
СОЗДАН SPOTIFY

Используется для музыкальных рекомендаций в стриминговой платформе.

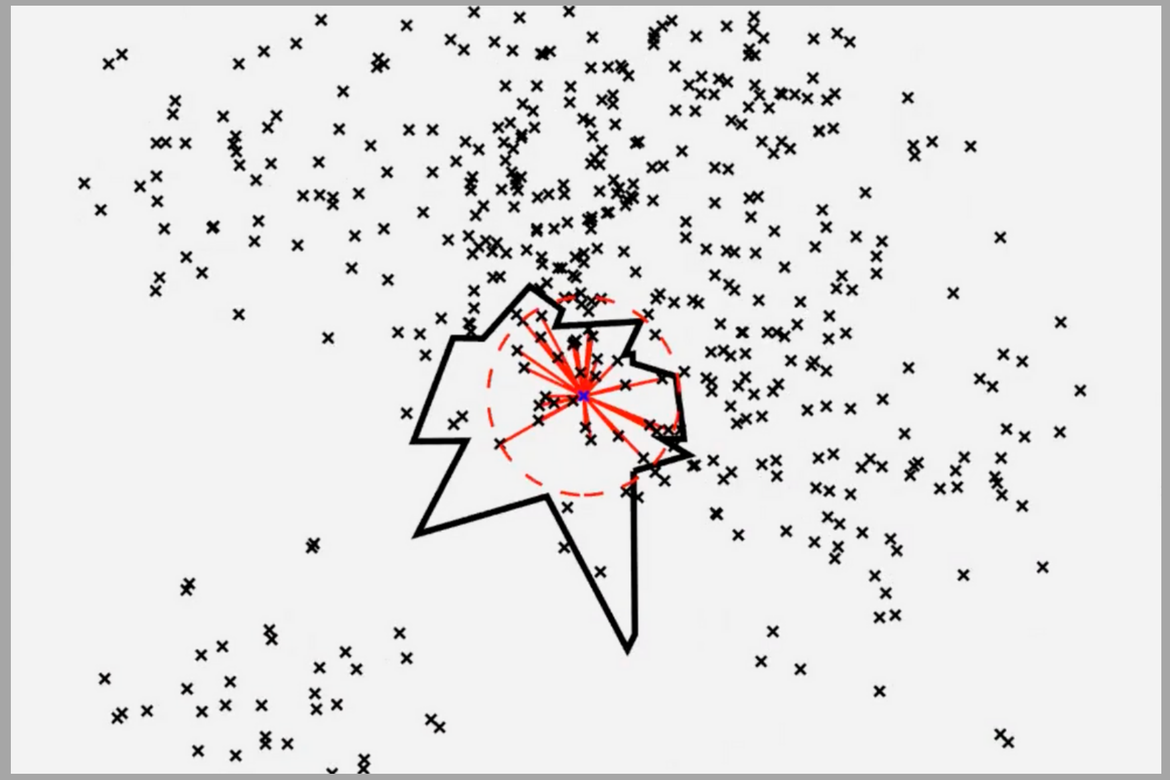
- Объекты обучающей матрицы кластеризуются по определенному алгоритму с некоторой случайностью



- На основе полученной кластеризации строится дерево решений
- В узлах разделяющие плоскости



- При построении N различных случайных кластеризаций, деревья объединяются в ансамбль.
- По каждому дереву осуществляется поиск N ближайших
- Соседи по всем деревьям объединяются, исключаются дубликаты и выводится список из N ближайших
- В итоге разделяющая поверхность приближается к поверхности которую бы построил KNN



Выводы и сравнение

KNN

Идеальная разделяющая поверхность
Невозможно использовать при больших объемах из-за полного перебора.

FAISS

Работает с огромными объемами данных (млрд точек)
Есть возможность заускать на GPU
Средняя точность
Скорость обучения чуть выше, чем у Annoy

ANNOY

Работает с большим объемами данных (сотни млн точек)
Точность выше среднего
Скорость поиска чуть выше чем у Faiss

Ресурсы и ссылки

PYDATA WORKSHOP

https://www.youtube.com/watch?v=XbhA2Kxeuuc

BASICS OF SIMILARITY SEARCH

https://www.youtube.com/watch?v=oUTY703R1-Y

FAISS GITHUB

https://github.com/facebookresearch/faiss/wiki

ANNOY GITHUB

https://github.com/spotify/annoy