Задача рекомендательной системы — найти для пользователя такие элементы заданного множества, которые с высокой вероятностью понравятся пользователю.

Например:

- Товары, которые пользователь захочет купить
- Фильмы, которые пользователю понравятся
- Статьи, которые пользователь дочитает до конца
- И так далее

Введем некоторые обозначения:

- Пользователи — множество U (users).
- Товары — множество I (items) — это могут быть любые категории товаров: и фильмы, и музыка, и техника. — зависит от задачи.
- Так же будем считать, что для некоторой пары пользователь-товар $u \in U \quad$ и $\quad i \in I$ известна оценка r, выражающая заинтересованность пользователя в этом товаре.

Заинтересованность может выражаться в разных единицах. В случае с музыкой, это может быть, как много раз пользователь слушал данный трек, как много раз дослушал до конца, лайкнул ли он его. В случае со статьями дочитал ли он его до конца, сколько раз кликнул, скорость проматывания страницы и так далее. Требуется по известным оценкам заинтересованности научиться находить для каждого пользователя u набор из k товаров $I(u)$, наиболее подходящих пользователю, то есть таких, для которых оценка r окажется максимальной. То есть в итоге задача сводится к тому, что для объекта вида x надо уметь предсказывать оценку r. Как и в любой задаче машинного обучения, требуется уточнить три пункта:

- Целевая переменная
- Признаки
- Функционал ошибки

Первый пункт был рассмотрен выше. Теперь время перейти к тому, какие признаки характеризуют пару пользователь-товар.

2. Признаки

2.1 Статистические признаки

Для начала рассмотрим простые типы факторов. Среди них могут быть:

- Количество покупок и просмотров данного товара
- количество покупок пользователя в данной категории
- Количество покупок пользователей

Если товар или пользователь уже набрали достаточно статистики, то часто такие признаки оказываются самыми главными при принятии решения, поскольку уже содержат в себе достаточно информации о предпочтениях. Так, используя эти признаки, можно рекомендовать популярные товары в той категории, что нравится пользователю.

2.2 Коллаборативная фильтрация

Методы коллаборативной фильтрации строят рекомендации для пользователя на основе схожести между пользователями и товарами. Различают несколько подходов:

- Memory-based
- Модели со скрытыми переменными
- Гибридный — объединение вышеперечисленных подходов.

Рассмотрим memory-based подход. В его основе лежит таблица R , $N \times M$, где N — количество пользователей, а M — количество товаров. На пересечении строки u и столбца i хранится значение r , если оно известно.

2.2.1 Baseline

Самое простое предсказание, которое можно сделать, имея таблицу R , посчитать среднее всех известных оценок и считать, что и остальным товарам пользователи будут ставить именно такие оценки:

$$\mu = \frac{\sum_U \sum_I r_{ui}}{|Y|}$$

где Y — множество известных оценок.

Но понятно, что такой подход имеет множество недостатков, и один из них заключается в том, что мы не учитываем предвзятость пользователей и популярность товаров. Для того, чтобы учесть данные характеристики, найдем среднюю оценку для каждого товара и для каждого пользователя:

$$\mu_i = \frac{\sum_U r_{ui}}{n_i}$$

$$\mu_u = \frac{\sum_I r_{ui}}{m_u}$$

где n_i и m_i - количество оценок у товара iii и пользователя uuu соответственно.

Посчитав сдвиги для пользователей и товаров относительно средней оценки, можно более точно вычислить, какую же оценку пользователь uuu поставит товару iii :

$$b_i = \mu_i - \mu$$

$$b_u = \mu_u - \mu$$

$$r_{ui} = \mu + b_i + b_u$$

Но пока мы никак не учитывали схожесть пользователей. Получается, что при подсчете средней оценки на неё влияли как пользователи со схожими вкусами, так и те, вкусы которых сильно разнятся. Поэтому можем немного модифицировать подсчет среднего: будем считать среднее для всех товаров для выбранного пользователя u_0 , при этом влияние других пользователей будет зависеть от их схожести с выбранным пользователем:

$$\mu_{u_0} = \frac{\sum_U sim(u_0,u) \sum_I r_{ui}}{|Y|}$$

$$|Y| = \sum_{ij} I(r_{ij}! = None)$$

Соответственно так же можно модифицировать подсчет средней оценки для товаров.

2.2.2 Сравнение пользователей и товаров

Возникает вопрос: как сравнивать схожесть пользователей. Будем исходить из следующего предположения: два пользователя считаются похожими, если они ставят одним и тем же товарам одинаковые оценки. Рассмотрим двух пользователей u и v . Обозначим через I_{uv} множество товаров i , для которых известны оценки обоих пользователей:

$$I_{uv} = \{i \in I | \exists r_{ui} \& \exists r_{vi}\}$$

Тогда сходство между двумя данными пользователями можно вычислить с помощью корреляции Пирсона:

$$sim(u,v) = \frac{\sum_{i \in I_{ui}} (r_{ui} - \mu_u)(r_{vi} - \mu_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \mu_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \mu_u)^2}}$$

Чтобы вычислить сходство между товарами i и j так же введем множество пользователей U_{ij} , для которых известны оценки для этих товаров.

$$U_{uj} = \{u \in U | \exists r_{ui} \& \exists r_{uj}\}$$

Тогда сходство двух товаров можно вычислить через корреляцию Пирсона:

$$sim(i,j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \mu_i)(r_{uj} - \mu_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \mu_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \mu_j)^2}}$$

Понятно, что можно применять и другие способы вычисления схожестей. Например, можно вычислить норме разности между векторами или скалярное произведение векторов.

Существует два метода, использующих сходство пользователей или товаров:

- Тривиальные рекомендации;
- Подход на основе сходств пользователей (user-based collaborative filtering);
- Подход на основе сходств товаров (item-based collaborative filtering).

2.2.3 Тривиальные рекомендации

Пусть пользователю u_0 понравился товар i_0 . Найдем множество пользователей, которым так же понравился этот товар:

$$U(i_0) = \{u \in U | \exists r_{ui_0}\}$$

Далее среди товаров, купленных пользователями из множества U(io)U(i_o)U(io) найдем те, которые наиболее похожи на исходный товар:

$I(i_0) = \{i \in I | sim(i_0, i) > \alpha \exists r(u, i), u \in U(i_0)\}$Пользователю рекомендуется k товаров из множества $I(i_0)$ с максимальным значением $sim(i_0, i)$.

У такого подхода есть ряд проблем. Так, например, все рекомендации будут тривиальные, то есть в основном будут рекомендоваться популярные товары. Так же не учитываются интересы конкретного пользователя u_0 . Но зато всегда можно будет что-то порекомендовать нетипичным пользователям, пусть это даже и будут только популярные товары.

2.2.4 Подход на основе сходств пользователей

Рассмотрим второй подход. Допустим, хотим сделать рекомендации для фиксированного пользователя u_0 Найдем множество $U(u_0)$ пользователей, похожих на данного:

$$U(u_0) = \{u \in U | sim(u_0, u) > \alpha\},$$

где α -- заданный порог схожести.

После этого для каждого товара вычислим, как часто пользователи из множества $U(u_0)$ покупали его:

$$\frac{p_i = |\{u \in U(u_0) | \exists r_{ui}\}|}{|U(u_0)|}$$

Пользователю рекомендуется k товаров с максимальными значениями p_i . Таким образом будет шанс рекомендовать не только самые популярные товары, но и те, что популярны среди данного типа пользователей. Так же в таком подходе учитываются интересы пользователя u_0 .

Проблема данного подхода в том, что он позволяет строить рекомендации только в том случае, если для данного пользователя существуют похожие на него. Если же пользователь новый или нетипичный, то подобрать что-либо не получится.

2.2.5 Подход на основе сходств товаров

Последний подход основан на сходстве товаров. Для него определяется множество товаров, похожих на те, которые интересовали пользователя u_0 :

$$I(u_0) = \{i \in I | \exists r_{u_0 i_0}, sim(i_0, i) > \alpha\}$$

Далее для каждого товара из найденного множества вычисляется его сходство с товарами из множества пользователя:

$p_i = max_{i_0 : \exists r_{u_0 i_0}} sim(i_0, i)$

Пользователю рекомендуется k товаров с наибольшими значениями p_i. В таком подходе решается проблема нетипичного пользователя, так как необязательно иметь пользователей со схожими интересами, и подход позволяет найти товары, похожие на интересные ему. Но в таком подходе есть и проблемы: есть вероятность, что вместо действительно интересных товаров будем рекомендовать популярные.

2.2.6 Проблематика

Все перечисленные выше подходы имеют и другие проблемы, кроме перечисленных:

- Не хватает теоретического обоснования;
- Необходимо хранить всю матрицу рейтингов R;
- Проблема холодного старта.

Существует множество способов измерить схожесть пользователей и товаров. Так же придумали огромное множество гибридных методов, учитывающих преимущества подходов. Но без теоретического обоснования подходов сложно точно определить, какой подход где лучше использовать. Обычно, ответ на этот вопрос можно узнать только на практике.

Проблема холодного старта заключается в том, что непонятно, что рекомендовать новоприбывшему пользователю. От этой проблемы страдают большинство подходов в рекомендательной системе, так что эта проблема остается до сих пор актуальной.

Единственная действительно разрешимая проблема обозначена во втором пункте. Она решается при помощи подходов, основанных на моделях со скрытыми переменными. Обзор этих моделей не входит в данный курс, но по ссылкам в конце лекции вы сможете найти интересующие вас материалы.

Но помимо проблем у рассмотренных подходов есть и преимущества:

- Легко понять;
- Легко реализовать.

Благодаря этим сильным плюсам, зачастую memory-based подходы используют в качестве baseline.

2.3 Контентные модели

В коллаборативной фильтрации учитывается информация о вкусах пользователей и об их сходствах, но при этом никак не используется свойства самих пользователей и товаров. При этом может быть полезным находить товары, похожие описанием на те, которыми пользователи интересовались. Особенно видно польза такого подхода в рекомендательных системах, где пользователям предлагают видео, музыку или статьи.

Скорее всего пользователю, любящему электронную музыку, захочется послушать не только то, что нравится другим электронщикам, но и то, что по звучанию похоже на его любимых исполнителей.

При таком подходе все товары описываются векторами, называемые эмбедингами (embeddings). Затем измеряется сходство между вектором нового товара и векторами товаров из истории пользователя: можно вычислять как минимальное, так и среднее расстояние до векторов из истории. Или же можно обучить линейную модель, которая для данного пользователя предсказывает целевую переменную на основе представления о товаре:

$$\sum_{i \in I : \exists r_{ui}} (< \omega_u, q_i > - r_{ui})^2 \to min_{\omega_u}$$

и затем с помощью этой модели оценивать, насколько пользователю подойдут те или иные товары.

Можно обучить граф вычислений, который по всем известным характеристикам товара и интересам пользователя попытается предсказать целевую переменную.

Как видно, существует множество методов учета данных о пользователях и товарах, но никогда нельзя предсказать заранее, какой из них подойдет в данной задаче. Так же существует множество способов сбора такого рода данных. Самое простое, что можно сделать, это просить каждого нового пользователя заполнять анкету, где он должен немного рассказать о себе (пол, возраст, любые жанры музыки). Так же можно договариваться с другими системами и компаниями, которые уже владеют нужной вам информацией, в обмен на обещание, например, уменьшить цену на рекламу этих компаний на вашем сайте. Пример такой системы: Яндекс.Дзен.

Более сложная задача — сбор данных для товаров. Можно положиться на пользователей, которые будут ставить теги для каждого объекта, который они добавляют или который им нравится. Можно нанять экспертов, которые вам эти теги проставят. Так, например, поступил музыкальный сервис Pandora, взявший за основу рекомендательную систему Music Genome Project, которая в свою очередь наняла специалистов, которые определили для каждого трека значения более, чем для 450 характеристик, среди которых есть темп, частота, тональность, гармония и так далее.

3 Функционал ошибки

Как говорилось ранее, еще один важный пункт, который следует установить в любой задаче машинного обучения, это функционал ошибки. Существует довольно большое множество метрик качества рекомендательных систем. Различают онлайн-метрики и оффлайн-метрики.

Онлайн-метрика — метрика, ключевая с точки зрения бизнеса. Например, это может быть среднее время, проведенное пользователем на сайте, или средний чек пользователя. Затем выбирают оффлайн-метрику или линейную комбинацию оффлайн- метрик, которая лучше всего коррелирует с выбранной бизнес метрикой. Под онлайн-метрикой понимается показатель, который можно измерить только при запуске рекомендательной

системы на реальных пользователя, а под оффлайн-метрикой — показатель, который можно измерить для модели на исторических данных. Так же иногда пытаются найти промежуточную метрику, которая коррелирует с основной, но при этом быстро реагирует на изменения в работе рекомендательной системы, но эта тема не сегодняшней лекции.

3.1 Качество предсказаний

Рассмотрим несколько оффлайн-метрик. Поскольку модель обучается для предсказывания r_{ui}, логично оценивать качество решения именно этой задачи.

3.1.1 Предсказание рейтингов

Если модель предсказывает рейтинг, время нахождения на странице или любую другую вещественную величину, то качество рекомендаций может быть измерено с помощью MSE , $RMSE$, MAE или другие регрессивные метрики.

3.1.2 Предсказание событий

Если модель предсказывает вероятно сть клика на статью, покупки товара, лайка или наступления любого другого события, то качество можно измерить с помощью метрик качества классификации: доля правильных ответов, точность, полнота, F-мера, $AUC - ROC$, $log - loss$ и так далее.

Если вспомнить, что мы показываем пользователю только kkk товаров, получивших самые высокие предсказания модели, то можно понять, что вас интересует качество только для этих товаров. Если через $R_u(k)$обозначить предсказание для пользователя и kkk товаров, а через L_u - товары, для которых действительно произошло интересующее нас событие, то можно ввести следующие метрики:

- Наличие верной рекомендации:
- $\boxed{hitrate@k = [R_u(k) \cap L_u \neq \emptyset]}$
- Точность: $\boxed{precision@k = \frac{|R_u(k) \cap L_u|}{|R_u(k)|}}$
- Полнота: $\boxed{recall@k = \frac{|R_u(k) \cap L_u|}{|L_u|}}$

3.1.3 Качество ранжирования

В большинстве случаев неважно, насколько хорошо модель предсказывает целевую переменную (рейтинг, вероятность лайка, вероятность дочитывания). Важно чтобы она давала более релевантным товарам более высокие предсказания. То есть модель должна правильно ранжировать (сортировать) товары.

Например, одной из популярных метрик является average precision at k - модификация precision at k.

$$AP@K = \frac{1}{K}\sum_{k=1}^{K} p@k * rel(k)$$

где $p@k$, и $AP@k$ оценивают качество ранжирования для отдельно взятого объекта, тогда как обычно рекомендательные системы работают с миллионами подобных объектов. Метрика $M\ AP@k$ позволяет вычислить среднюю оценку $AP@k$по всем пользователям:

$$MAP@K = \frac{1}{|U|}\sum_{u=1}^{|U|}(AP@K)_u$$

3.1.4 Недостатки

Сложно говорить о том, насколько хорошо работает рекомендательная система,

основываясь только на качестве предсказаний. Возможно, пользователь купил бы

товары, предсказанные моделью, и без её помощи. Поскольку нельзя узнать, повлияло ли предсказание модели на поведение пользователя, имеет смысл анализировать

и другие метрики качества, которые могут косвенно говорить о пользе предсказаний

модели.

3.2 Покрытие

3.2.1 Покрытие пользователей

Важно следить за долей пользователей, для которых рекомендательная система

не может ничего порекомендовать. Такие ситуации могут происходить, например,

из-за отсутствия определенных признаков в модели или из-за низкой уверенности

модели. В таком случае необходимо отслеживать проблемы с покрытием в модели

рекомендаций.

3.2.2 Покрытие товаров

Так же полезно обращать внимание на то, какие товары рекомендует модель. Может оказаться, что доля рекомендуемых товаров мала и покрывает только популярные предложения. Самое простое, что можно сделать, это посчитать долю каталога,

то есть сколько товаров было порекомендовано хотя бы один раз.

Так же можно оценить общее разнообразнее рекомендаций. Пусть p(i)p(i)p(i) — доля показа товара i∈Ii \in Ii∈I среди всех показов для данной рекомендательной системы. Тогда

разнообразие можно определить как энтропию такого распределения:

$$H(p) = -\sum_{i \in I} p(i) log p(i)$$

3.3 Новизна

Новизна — это доля товаров, которые пользователь видит впервые. В идеале, это товары, который пользователь видит впервые не только на нашем сайте, но он так же не видел их на других ресурсах.

Для улучшения этой метрики, можно, например, удалить из модели все объекты, которые пользователь видел раньше. Так же можно каждый объект домножить на вес, обратный его популярности. В этом случае, чем меньше популярность товара, тем выше его вес, и тем вероятнее мы его покажем пользователю.

3.4 Прозорливость

Прозорливость — умение модели порекомендовать товары, отличные от всех, что пользователь видел ранее. Например, если пользователь на сайте смотрит только боевики, то модель поощряется, если она посоветует ему фильм хороший с точки зрения клиента, но другого жанра.