

Лекция 1

Рекомендательные системы

Е. А. Соколов
ФКН ВШЭ

23 января 2020 г.

Задача рекомендательной системы — выбрать для пользователя элементы из некоторого множества, на которых он с наибольшей вероятностью совершит интересные нам действия. Это могут быть:

- товары, которые пользователь захочет купить;
- музыка, которую пользователь захочет дослушать до конца;
- статьи, которые пользователь дочитает до конца;
- видео, которые пользователь досмотрит до конца;
- и многое другое.

Мы будем рассуждать в терминах пользователей (users, U) и товаров (items, I), но все методы подходят для рекомендаций любых объектов. Будем считать, что для некоторых пар пользователей $u \in U$ и товаров $i \in I$ известны оценки r_{ui} , которые отражают степень заинтересованности пользователя в товаре. Вычисление таких оценок — отдельная тема. Например, в интернет-магазине заинтересованность может складываться из покупок товара и просмотров его страницы, причём покупки должны учитываться с большим весом. В социальной сети заинтересованность в материале может складываться из времени просмотра, кликов и явного отклика (лайки, репосты); это всё тоже должно суммироваться с различными весами. Не будем сейчас останавливаться на этом вопросе, а перейдём к основной задаче.

Требуется по известным рейтингам r_{ui} научиться строить для каждого пользователя u набор из k товаров $I(u)$, наиболее подходящих данному пользователю — то есть таких, для которых рейтинг r_{ui} окажется максимальным.

Получается понятная задача: для объекта x_{ui} «пользователь-товар» нужно предсказать значение целевой переменной r_{ui} . Здесь требуют уточнения три пункта: целевая переменная, признаки и функционал ошибки. Первое мы затронули выше, третье обсудим позже, а сейчас поговорим о том, какими признаками можно охарактеризовать пару «пользователь-товар».

1 Признаки в рекомендательных системах

§1.1 Коллаборативная фильтрация

Как понять, что пользователю может понравиться товар? Первый вариант — поискать похожих на него пользователей и посмотреть, что нравится им; также можно поискать товары, похожие на те, которые этот пользователь уже покупал. Методы коллаборативной фильтрации строят рекомендации для пользователя на основе похожестей между пользователями и товарами. Мы рассмотрим два подхода к определению сходства.

1.1.1 Memory-based

Два пользователя похожи, если они ставят товарам одинаковые оценки. Рассмотрим двух пользователей u и v . Обозначим через I_{uv} множество товаров i , для которых известны оценки обоих пользователей:

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

Тогда сходство двух данных пользователей можно вычислить через корреляцию Пирсона:

$$w_{uv} = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}},$$

где \bar{r}_u и \bar{r}_v — средние рейтинги пользователей по множеству товаров I_{uv} .

Чтобы вычислять сходства между товарами i и j , введём множество пользователей U_{ij} , для которых известны рейтинги этих товаров:

$$U_{ij} = \{u \in U \mid \exists r_{ui} \ \& \ \exists r_{uj}\}.$$

Тогда сходство двух данных товаров можно вычислить через корреляцию Пирсона:

$$w_{ij} = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}},$$

где \bar{r}_i и \bar{r}_j — средние рейтинги товаров по множеству пользователей U_{ij} . Отметим, что существуют и другие способы вычисления похожестей — например, можно вычислять скалярные произведения между векторами рейтингов двух товаров.

Мы научились вычислять сходства товаров и пользователей — разберём теперь несколько способов определения товаров, которые стоит рекомендовать пользователю u_0 . В подходе на основе сходств пользователей (user-based collaborative filtering) определяется множество $U(u_0)$ пользователей, похожих на данного:

$$U(u_0) = \{v \in U \mid w_{u_0v} > \alpha\}.$$

После этого для каждого товара вычисляется, как часто он покупался пользователями из $U(u_0)$:

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

Пользователю рекомендуются k товаров с наибольшими значениями p_i . Данный подход позволяет строить рекомендации, если для данного пользователя найдутся похожие. Если же пользователь является нетипичным, то подобрать что-либо не получится.

Также существует подход на основе сходств товаров (item-based collaborative filtering). В нём определяется множество товаров, похожих на те, которые интересовали данного пользователя:

$$I(u_0) = \{i \in I \mid \exists r_{u_0 i_0}, w_{i_0 i} > \alpha\}.$$

Затем для каждого товара из этого множества вычисляется его сходство с пользователем:

$$p_i = \max_{i_0: \exists r_{u_0 i_0}} w_{i_0 i}.$$

Пользователю рекомендуются k товаров с наибольшими значениями p_i . Даже если пользователь нетипичный, то данный подход может найти товары, похожие на интересные ему — и для этого необязательно иметь пользователя со схожими интересами.

1.1.2 Модели со скрытыми переменными

Все описанные выше подходы требуют хранения разреженной матрицы $R = \{r_{ui}\}$, которая может быть достаточно большой. Более того, они весьма эвристичны и зависят от выбора способа вычисления сходства, способа генерации товаро-кандидатов, способа их ранжирования. Альтернативой являются подходы на основе моделей со скрытыми переменными (latent factor models).

Мы будем пытаться построить для каждого пользователя u и товара i векторы $p_u \in \mathbb{R}^d$ и $q_i \in \mathbb{R}^d$, которые будут характеризовать «категории интересов». Например, каждую компоненту такого вектора можно интерпретировать как степень принадлежности данного товара к определённой категории или степень заинтересованности данного пользователя в этой категории. Разумеется, никак не будет гарантироваться, что эти компоненты соответствуют каким-то осмысленным категориям, если только мы специально не потребуем этого от модели. По сути, векторы пользователей и товаров являются представлениями (embeddings), позволяющими свести эти сущности в одно векторное пространство.

Сходство пользователя и товара будем вычислять через скалярное произведение их представлений:

$$r_{ui} \approx \langle p_u, q_i \rangle.$$

Также через скалярное произведение можно вычислять сходство двух товаров или двух пользователей.

Мы можем записать функционал ошибки, исходя из способа вычисления сходства:

$$\sum_{(u,i) \in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 \rightarrow \min_{P,Q} \quad (1.1)$$

Суммирование здесь ведётся по всем парам пользователей и товаров, для которых известен рейтинг r_{ui} . Заметим, что если R' — матрица R с центрированными строками и столбцами, то данная задача сводится к низкоранговому матричному разложению:

$$\|R' - P^T Q\|^2 \rightarrow \min_{P, Q}$$

Здесь представления пользователей и товаров записаны в столбцах матриц P и Q . Существуют модификации, в которых к скалярным произведениям добавляется масштабирующий множитель $\alpha \in \mathbb{R}$:

$$\|R' - \alpha P^T Q\|^2 \rightarrow \min_{P, Q, \alpha}$$

Данный функционал можно регуляризовать:

$$\sum_{(u,i) \in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 + \lambda \sum_{u \in U} \|p_u\|^2 + \mu \sum_{i \in I} \|q_i\|^2 \rightarrow \min_{P, Q} \quad (1.2)$$

Описанная модель носит название Latent Factor Model (LFM).

Отметим, что использование среднеквадратичной ошибки не всегда имеет смысл — в рекомендациях требуется выдать более высокие предсказания для товаров, которые более интересны пользователю, но вовсе не требуется точно предсказывать рейтинги. Впрочем, среднеквадратичную ошибку удобно оптимизировать; более того, именно она использовалась в качестве функционала в конкурсе Netflix Prize, который во многом определил развитие рекомендательных систем и в котором было предложено много популярных сейчас методов.

Существует два основных подхода к решению задачи (1.1). Первый — стохастический градиентный спуск, который на каждом шаге случайно выбирает пару $(u, i) \in R$:

$$\begin{aligned} p_{uk} &:= p_{uk} + \eta q_{ik} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle), \\ q_{ik} &:= q_{ik} + \eta p_{uk} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle). \end{aligned}$$

Второй подход основан на особенностях функционала (1.1) и называется ALS (alternating least squares). Можно показать, что этот функционал не является выпуклым в совокупности по P и Q , но при это становится выпуклым, если зафиксировать либо P , либо Q . Более того, оптимальное значение P при фиксированном Q (и наоборот) можно выписать аналитически, — но оно будет содержать обращение матрицы:

$$\begin{aligned} p_u &= \left(\sum_{i: \exists r_{ui}} q_i q_i^T \right)^{-1} \sum_{i: \exists r_{ui}} r_{ui} q_i; \\ q_i &= \left(\sum_{u: \exists r_{ui}} p_u p_u^T \right)^{-1} \sum_{u: \exists r_{ui}} r_{ui} p_u; \end{aligned}$$

(здесь через p_u и q_i мы обозначили столбцы матриц P и Q).

Чтобы избежать сложной операции обращения, будем фиксировать всё, кроме одной строки p_k матрицы P или одной строки q_k матрицы Q . В этом случае можно найти оптимальное значение для p_k и q_k :

$$p_k = \frac{q_k(R - \sum_{s \neq k} p_s q_s^T)^T}{q_k q_k^T},$$

$$q_k = \frac{p_k(R - \sum_{s \neq k} p_s q_s^T)}{p_k p_k^T}.$$

Данный подход носит название Hierarchical alternating least squares (HALS) [1].

Список литературы

- [1] *Gillis, Nicolas and Glineur, François* (2012). Accelerated Multiplicative Updates and Hierarchical Als Algorithms for Nonnegative Matrix Factorization. // *Neural Comput.*, 24, 4, p. 1085–1105.