УДК 004

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА МУЗЫКИ

Браневский А.Я., Бесчетников А.М., Наламвар Х.
Научный руководитель: Наламвар Х.

Национальный Исследовательский Томский политехнический университет,
634050, Россия, г. Томск, пр. Ленина, 30
E-mail: branevskij_aj@bw-sw.com

Введение

С ростом мировой экономики и соответствующего роста товарооборота, появляется потребность в совершенствовании «предложения» за счет алгоритмов, обеспечивающих подбор и рекомендацию товара целевому клиенту, который в нем заинтересован. Такой подход повышает конверсию и эффективность рекламы и интернет-магазинов, то есть увеличивает показатель перехода посетителей в покупателей.

Алгоритмы, основанные на коллаборативной фильтрации, используя статистику пользователя на веб-ресурсе (его оценки определенной продукции, покупки и т. д.) определяют на ее основе его вкусы и предпочтения. Все это дает возможность в подборе и рекомендации наилучшего предложения.

В наше время такие системы являются частью любого уважающего себя серьезного веб-сервиса (Amazon, Google, Yahoo и т. д.)

Алгоритм

Предположим, у нас есть некоторое количество триплетов (пользователь товар оценка) на их основании можно сформировать матрицу (рис. 1).

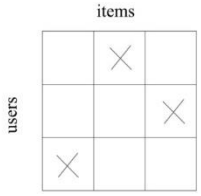


Рис. 1

Очевидно, что матрица будет сильно разреженной (т. к. есть неоцененные позиции и их большенство). Следовательно, восстановление таких позиций является нашей задачей (это и будут предсказываемые рекомендации). Восстановим применяя широкоизвестные подход метод наименьших квадратов.

Для начала введем обозначения:

$$Q_{ui} = \begin{cases} r & \text{if user u rate item i} \\ 0 & \text{if user u did not rate item i} \end{cases}$$

Рис. 2

где Q_{ui} – оценка пользователем u некоторого предмета *i*.

Также введем матрицу:

$$w_{ui} = \begin{cases} 0 & \text{if } q_{ui} = 0 \\ 1 & \text{else} \end{cases}$$

Рис. 3

где $W_{ui} = 1$ если пользователь u оценил предмет i, иначе 0.

Нашей задачей является найти 2 таких вектора X (вектор стоблец) и Y (вектор строка) которые при перемножении дадут матрицу максимально близкую к Q (в общем случае X Y могут быть матрицами, конечные формулы обобщены и на них в том числе). Запишем соотношения для y_i (рис. 4).

$$X * y_i = Q_i$$
$$W_i * X * y_i = W_i * Q_i$$

Рис. 4

Q_i это некоторое множество оценок всех пользователей предмета i, но т. к. некоторые пользователи не оценивают этот предмет i, можно домножить слева и справа на W_i. Но, такое соотношение не всегда будет иметь решение поэтому воспользуемся следующим подходом.

Найдем проекцию вектора Q_i на пространство всех линейных комбинаций вектора X, таким образом найдем приближенный вектор Q_{proji} для которого система будет гарантированно иметь решение так как Q_{proji} будет какой-то из линейных комбинаций вектора X.

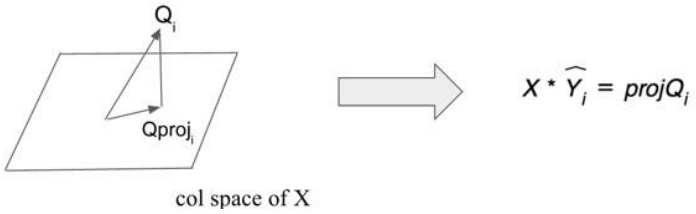
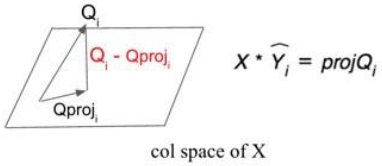


Рис. 5

Далее заметим, что разность $Q_i - Q_{proji}$ перпендикулярны любому вектору в col space X. Воспользуемся свойством скалярного произведения получим соотношения для X_u Y_i. В примере на рис 6. разобран случай получения Y_i X_u.



$$X^T * (Q_i - projQ_i) = 0$$
$$X^T * (Q_i - X * \widehat{Y_i}) = 0$$
$$X^T * Q_i = X^T * X * \widehat{Y_i}$$
$$\widehat{Y_i} = (X^T * X)^{-1} * X^T * Q_i$$
$$\widehat{Y_i} = (X^T * W_i * X)^{-1} * X^T * W_i * Q_i$$

$$\widehat{Y_i} = (X^T * W_i * X)^{-1} * X^T * W_i * Q_i$$
$$\widehat{X_u} = (Y * W_u * Y^T)^{-1} * Q_u * W_u * Y^T$$

Рис. 6

На малых объемах данных такой способ будет показывать хороший результат, но при больших объемах могут возникнуть проблемы. Воспользуемся возможностями фреймворка обработки больших данных SPARK. Архитектура его такова, что в процессе вычисления информация разбивается на блоки и каждый обсчитывается на отдельном кластере, далее вся информация сливается на основной кластер и там идет возможная свертка результатов.

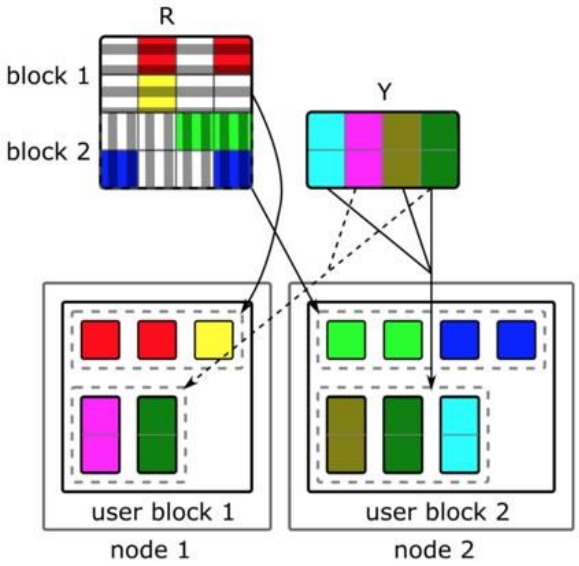


Рис. 7

На картинке выше показан пример разбиения на блоки вычисления вектора X. Для определенной его компоненты достаточно знать некоторые компоненты вектора Y(тк часть из них откидывается при перемножении с матрицей W) и некоторые компоненты матрицы Q. Пользуясь таким подходом можно существенно ускорить вычисления.

Проблемы

Масштабируемость

С увеличением количества пользователей в системе, появляется проблема масштабируемости. Например, имея 10 миллионов покупателей O(M) и миллион предметов O(N), алгоритм коллаборативной фильтрации со сложностью равной O(MN) уже слишком сложен для расчётов. Также возникают проблемы с пересчетом оценок, т. к. стандартная версия алгоритма не предполагает онлайновые обновления.

Синонимия

Стандартный алгоритм никак не анализирует названия продуктов. Например, фильмы для детей и детские фильмы будут анализироваться как разные объекты. Хотя этого можно избежать, обрабатывая такие случаи в препроцессинге.

Нечестные оценки

Каждый пользователь может выставлять оценку любому продукту, которая может не совпадать с его объективной оценкой. Это вносит определенный шум в данные и может сказаться на итоговых результатах. Также оценки могут быть искажены из-за форсированного продвижения продукции определенной фирмой (например, накрутка рейтинга). Это также вносит шум в данные.

Белые вороны

Всегда существуют пользователи, вкусы которых не совпадают с юольшинством. Собственно, из-за этого рекомендации им могут быть затруднены. Однако такие люди имеют проблемы с получением рекомендаций и в реальной жизни.

Заключение

Вышеописанный алгоритм был реализован на языке программирования Scala и протестирован на больших объемах данных (в качестве исходных данных использовалась база известного веб-сервиса Last.fm). Примеры рекомендаций можно увидеть в презентации.