Коллаборативная фильтрация на MapReduce

Васильев Руслан ВМК МГУ, 317 группа

26 мая 2021 г.

Алгоритм

Рассмотрим один из методов, применяемых в рекомендательных системах — *коллаборативную фильтрацию*. Пусть имеется набор известных троек (u, i, r_{ui}), где u — пользователь, i — объект, r_{ui} — оценка, которую пользователь ему поставил. Мы будем строить рекомендации на датасете MovieLens, поэтому далее под объектами понимаются фильмы.

Чтобы предсказать рейтинг $\hat{r}_{ui}$ для тех фильмов, которые пользователь u еще не смотрел, можно использовать информацию о фильмах, для которых оценки его известны:

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in I_u} \text{sim}(i, j) \cdot r_{uj}}{\sum\limits_{j \in I_u} \text{sim}(i, j)}, \tag{1}$$

где I_u — множество фильмов, которым пользователь поставил оценку, $\text{sim}(i, j)$ — мера сходства фильмов i и j:

$$\text{sim}(i, j) = \max\left\{\langle \mathbf{R}^{\mathbf{i}}_{\mathbf{ij}}, \mathbf{R}^{\mathbf{j}}_{\mathbf{ij}} \rangle, 0\right\}. \tag{2}$$

Векторы $\mathbf{R}^{\mathbf{k}}_{\mathbf{ij}}$, $k \in \{i, j\}$, определяются с помощью U_{ij} — множества пользователей, оценивших оба фильма i и j:

$$\mathbf{R}^{\mathbf{ij}}_{\mathbf{k}} = \frac{(r_{uk} - \bar{r}_u)_{u \in U_{ij}}}{\|(r_{uk} - \bar{r}_u)_{u \in U_{ij}}\|_2}, \quad \bar{r}_u = \frac{1}{|I_u|} \sum_{k \in I_u} r_{uk}. \tag{3}$$

Таким образом, описанный *item-oriented* алгоритм рекомендации сводится к подсчету косинусной меры близости и агрегации известных оценок.

Описание данных

Будем работать с датасетом MovieLens, производя тестирование на *small*-версии. Он состоит из двух файлов:

- `ratings.csv` — ≈ 100 000 строк, в каждой из которых записан пользователь, номер фильма, оценка в диапазоне от 0.5 до 5.0 и временная метка.
- `movies.csv` — ≈ 10 000 строк, каждая соответствует одному фильму (номер, название и список жанров).

Реализация

Произведем предобработку, подсчет вышеописанных формул и получение итогового файла с рекомендациями в 5 этапов, каждый из которых будет выполняться в парадигме MapReduce.

Шаг 1: группировка по пользователям

На данном этапе мы получаем для каждого u словарь со всеми его оценками и фильмами. Поскольку формулы (2), (3) не зависят от шкалы, в которой считается рейтинг, он переводится в целые числа от 1 до 10.

В маппере из `ratings.csv` выделяются тройки (u, i, r_{ui}), а на выходе редьюсера — мы получаем искомую агрегацию $(u, (items, ratings))$. Сложность по числу операций и памяти в обеих фазах — $O(UI\alpha)$, где U — количество пользователей в датасете, I — количтево фильмов, α — доля известных оценок к общему числу возможных (UI).

Понятно, что если операции выполняются параллельно на M редьюсерах и R маппперах, то в сложности для времени работы или памяти одного маппера (редьюсера) возникнет коэффициент $\frac{1}{M}$ $(\frac{1}{R})$.

Далее мы считаем, что распределение r_{ui} таково, что α эквивалентно средней доле пользователей, оценивших один фильм, и средней доле фильмов, оцененных одним пользователем.

Шаг 2: подсчет $\mathrm{sim}(i, j)$

Подадим на вход мапперу агрегированные по пользователю u фильмы $items$ и оценки $ratings$, на выходе продублируем наборы $(items, ratings)$ для каждого i из $items$, который и станет ключом для редьюсера. Причем сразу вычтем из каждого вектора рейтингов среднее значение. Таким образом, расход памяти на map-фазе составит $O(U(I\alpha)^2)$, по времени — всего потребуется произвести $UI\alpha$ итераций в циклах по $items$.

Редьюсеры, получая на вход для каждого ключа i в среднем αU наборов $(items, ratings)$, строят для всех j множество U_{ij} (если оно не пусто) и вычисляют (2), (3). На выходе записываются только $\mathrm{sim}(i, j) > 0$. С учетом выхода маппера, в процессе обработки сложность по памяти и времени на *reduce*-фазе также будет составлять $O(U(I\alpha)^2)$, но выход в худшем случае запишет в память I^2 элементов. Тем не менее исходная матрица (u, i, r_{ui}) — разреженная и в таком случае многие $\mathrm{sim}(i, j)$ также могут оказаться нулевыми. Мы можем оценить долю ненулевых значений $\mathrm{sim}(i, j)$, сделав несколько сильных предположений. Например, если считать, что все UI событий $(r_{ui} \neq 0)$ независимы, то

можно оценить:

$$\begin{aligned}
\mathbb{P}(U_{ij} \neq \emptyset) &= \mathbb{P}\left(\sum_{u=1}^{U} \mathbb{1}(r_{ui} \neq 0) \cdot \mathbb{1}(r_{uj} \neq 0) \neq 0\right) \\
&= 1 - \mathbb{P}\left(\sum_{u=1}^{U} \mathbb{1}(r_{ui} \neq 0) \cdot \mathbb{1}(r_{uj} \neq 0) = 0\right) \\
&= 1 - \left[\mathbb{P}\left(\mathbb{1}(r_{ui} \neq 0) \cdot \mathbb{1}(r_{uj} \neq 0) = 0\right)\right]^U \\
&= 1 - \left[1 - \mathbb{P}\left(\mathbb{1}(r_{ui} \neq 0) \cdot \mathbb{1}(r_{uj} \neq 0) = 1\right)\right]^U \\
&= 1 - \left[1 - \mathbb{P}\left(\mathbb{1}(r_{ui} \neq 0) = 1\right)\left(\mathbb{1}(r_{uj} \neq 0) = 1\right)\right]^U \\
&= 1 - (1 - \alpha^2)^U.
\end{aligned}$$

Заметим, что при достаточно маленьких α и больших U

$$1 - (1 - \alpha^2)^U \approx 1 - (1 - \alpha^2 U) = \alpha^2 U,$$

что совпадает с коэффициентом при I^2 для сложности по памяти и времени в процессе работы маппера и редьюсера. На выходе редьюсер сохраняет только ненулевые (положительные) значения $\mathrm{sim}(i, j)$ — далее будем считать, что их доля по всем (i, j) равна β, т.е. мы сохранили βI^2 значений $\mathrm{sim}(i, j)$, а остальные считаем равными нулю.

Шаг 3: агрегация множителей

Несложно заметить, что если бы $sim(i, j)$ и r_{ui} были были бы плотными, то (1) соответствовало бы их матричному произведению r_{ui} и нормированной $sim(i, j)$. Можем посчитать это выражение по тому же принципу.

Сначала мы соберем для всех k фильмов пары $(r_{uk}, sim(k, i))$. Заметим, что мы берем только имеющие смысл пары (ненулевые). Для этого в маппере мы считаем, во-первых, `ratings.csv` и будем выдавать $(k, (\langle r \rangle, u, r_{uk}))$, а, во-вторых, выход предыдущего шага который преобразуется в $(k, (\langle s \rangle, i, \mathrm{sim}(k, i))$, где в треугольных скобках проставлен тег. Получается, на *map*-фазе сложность по времени и памяти составит $\alpha UI + \beta I^2$.

Редьюсер же, собрав все наборы по ключу k, будет выдавать декартово произведение — все получившиеся $(u, i, r_{uk}, \mathrm{sim}(k, i))$. Количество таких четверок можем оценить как $I \cdot \alpha I \cdot \beta I = \alpha\beta I^3$ — этому же и будет пропорциональна сложность фазы по времени и памяти.

Шаг 4: подсчет $\hat{r}_{ui}$

На предыдущем этапе мы собрали все, что требуется для (1) — осталось распределить и перемножить. Маппер на данном этапе тождественный (но, формально, он отображает $(u, i, r_{uk}, \mathrm{sim}(k, i)) \mapsto (u, i), (r_{uk}, \mathrm{sim}(k, i)))$. Редьюсер же, собрав по ключу (k, i) все множители (1), считает эту взвешенную сумму. На обоих этапах по-прежнему

сложность по времени и памяти $\alpha\beta I^3$, но на выход редьюсер уже запишет только $\hat{r_{ui}}$ — сложность по памяти будет равна $O(UI)$.

Однако как мы понять, что оценка $\hat{r}_{ui}$ соответствует фильму, который пользователь еще не смотрел? Для этого на самом деле во время выполнения предыдущего шага на этапе редьюсера вместо $r_u i$ с ключом (u, i) в набор записывалось «-inf», из-за которой предсказания на этапе редьюсера 4-го шага становятся отрицательными и не дальше не выдаются.

Шаг 5: топ-100 лучших фильмов по версии Hadoop

На последнем task'е среди всех полученных $\hat{r}_{ui}$ нужно выбрать 100 максимальных и приписать их нужному пользователю, заменив номера фильмов на названия. Поскольку размер `movies.csv` является небольшим (I записей) относительно $\hat{r}_{ui}$ и всего, что происходило ранее, передадим его каждому мапперу вместе в выходом предыдущего шага. Маппер тогда только заменит номер фильма на название, выдавая $(u, \text{title}(i), \hat{r}_{ui})$. Далее мы воспользумся параметрами для сортировки — сначала по пользователю, затем по рейтингу и наконец по названию фильма — и в редьюсере сможем только конкатенировать первые 100 фильмов для каждого первичного ключа u. Время работы и память маппера и редьюсера — $O(UI)$, выход редьюсера займет $O(U)$ памяти.