электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.852

Коллаборативная фильтрация в рекомендательных системах. SVD-разложение

***Латкин И.И.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

научный руководитель: Гапанюк Ю.Е., к.т.н., доцент,
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»
gapyu@bmstu.ru

Введение

В современном мире существует огромное число самых разнообразных сервисов, предоставляющих какие-либо услуги. Для того, чтобы заинтересовать пользователя или рассказать ему о новых продуктах (товарах, новостях) система должна уметь «угадывать» предпочтения и *рекомендовать* какие-то продукты относительно его вкусов. Например, типичной системой, в которой необходимы рекомендации – сервисы электронной коммерции, рекомендации фильмов, музыки, путешествий и многие другие.

На текущий момент разработаны несколько подходов к формированию рекомендаций:

1. Неперсонализированные рекомендации
2. *Content-Based* рекомендации
3. Коллаборативная фильтрация (КФ)
   - 3.1. *User-User* КФ
   - 3.2. *Item-Item* КФ
   - 3.3. Методы сокращения размерности
     - 3.3.1. SVD-разложение
     - 3.3.2. PCA (Principal Component Analysis) – Анализ главных компонент
   - 3.4. Машины Больцмана

Неперсоналищированные рекомендации направлены на рекомендацию определенных продуктов всем пользователям вне зависимости от их предпочтений (например, «Лучшие товары 2015 года», «Наиболее часто приобретают следующие товары», «Топ 100 песен за всё время»). Минусом таких рекомендаций, очевидно,

является тот факт, что они не являются персонализированными – они для всех одинаковые.

Основной подход в *Content-Based* рекомендациях – анализ дополнительной мета-информации рекомендуемых продуктов (например, для фильмов это могут быть жанр, присутствие определенных актеров, режиссер, тематика фильма, описание, обладает ли Оскаром и т.п.). Тем самым, формирование рекомендаций для пользователя основано на анализе подобной информации и нахождении похожих продуктов на те, что уже были оценены пользователем. Минусом подобного подхода является необходимость проставления описанной мета-информации, а также тот факт, что рекомендации сильно зависят от этой мета-информации, которая не всегда может быть составлена достаточно точно.

Коллаборативная фильтрация основана на анализе поведения пользователей относительно продуктов (используя, так называемую, матрицу оценок R, где в строках расположены пользователи, а в столбцах - продукты). Пользователи голосуют за определенные продукты, проставляя им явный и неявный рейтинг. Суть User-User КФ состоит в том, что, используя матрицу R выявляются похожие друг на друга пользователи (которые проставляли похожие или противоположные оценки одинаковым продуктам). В Item-Item КФ всё то же самое, только, наоборот, для продуктов – по оценкам пользователей выявляются похожие друг на друга продукты и тогда пользователю будут рекомендоваться продукты похожие на те, что он оценивал (в положительную сторону) и не будут те, что были оценены в отрицательную сторону.

Матричное разложение. *SVD*-разложение.

Будем считать, что у нас существует матрица оценок пользователей, где в строках – пользователи, в столбцах объекты, которые необходимо рекомендовать (далее – объекты).

С данной матрицей связаны две основные проблемы. Первое – данная матрица имеет очень большой размер (тысячи пользователей и десятки и сотни тысяч объектов). Поэтому необходимо как-то уменьшить размер матрицы. Второе – из подобной матрицы довольно затруднительно определить предпочтения и вкусы конкретного пользователя, используя методы *User-User* и *Item-Item* коллаборативной фильтрации. Рассмотрим следующий пример. Пусть пользователь А оценил объект О1, а пользователь В оценил объект О2, но им обоим, в принципе нравится тематика объектов О1 и О2. Методы *User-User* и *Item-Item* коллаборативной фильтрации не дадут желаемый результат:

рекомендовать O2 пользователю A и O1 пользователю B. *User-User* фильтрация скажет то, что пользователь B не оценивал O1 – значит A и B не похожи, следовательно, O2 не будет рекомендован пользователю A. Точно также, *Item-Item* фильтрация не сможет соотнести объект O1 с O2, т.к. у пользователей A и B нет общих оценок на эти объекты. В такой затруднённой ситуации необходимы методы, с помощью которых можно было бы выявить и представить вкусы и предпочтения каждого пользователя.

Одним из возможных решений данных проблем является *SVD*-разложение матрицы (*Singular Value Decomposition*).

Как известно из линейной алгебры, сингулярным разложением матрицы M порядка $n \times m$ является следующие разложение:

$$M = U \Sigma V^T,$$

где матрица U размера $n \times n$, V – размера $m \times m$, а Σ – размера $n \times m$.

Матрица Σ состоит из, так называемых, сингулярных чисел, лежащих на главной диагонали (все остальные элементы равны нулю), причем они расположены в невозрастающем порядке. Последнее свойство особенно интересно – можно оставить только первые k сингулярных чисел и отбросить оставшиеся, тем самым получив наилучшее k-приближение исходной матрицы M (это продемонстрировано на рисунке ниже).

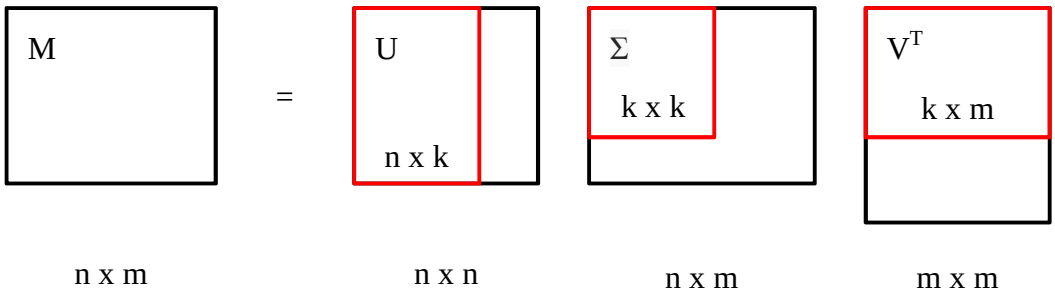

Рис. 1. SVD-разложение матрицы

Используя *SVD*-разложение матрицы оценок пользователей, можно получить две матрицы: матрицу U размера $n \times k$ и матрицу V размера $m \times k$ (значения сингулярных чисел матрицы Σ распределяются между матрицами U и V), где n – число пользователей, m – число объектов, а k – набор факторов. Данные факторы являются как раз характеристикой вкусов и предпочтений пользователя (для матрицы U) и характеристикой самих объектов, относительно вкусов и предпочтений пользователей (для матрицы V). Таким образом, имея данные матрицы для получения рейтинга r_{ui} пользователя *u* к

объекту *i* необходимо лишь вычислить скалярное произведение *u*-й строки матрицы U и *i*-ой строки матрицы V.

Однако, в реальной системе подобная матрица будет иметь относительно небольшое число ненулевых элементов (сильно разреженная) – это значит, что далеко не все пользователи оценили объекты, и, соответственно, не все объекты были оценены, т.е. большая часть оценок заранее неизвестна. Кроме того, алгоритм *SVD*-разложения является довольно трудоёмким, поэтому невозможно производить такие вычисления на больших объёмах данных часто.

Вероятностный подход к рекомендациям

Идея применения *SVD*-разложения для предсказания оценок является простой и изящной, однако, как было отмечено выше – очень сложной в вычислительном плане. Поэтому для решения этой проблемы можно применить методы машинного обучения и вероятностный подход к пониманию рекомендаций.

Как известно, формула Байеса имеет следующий вид:

$$p(\Theta|\mathbf{r}) = \frac{p(\mathbf{r}|\Theta)p(\Theta)}{p(\mathbf{r})},$$

где $p(\mathbf{r}|\Theta)$ называется функцией правдоподобия, т.е. такая функция, которая дает наиболее вероятные результаты случайной величины **r** при заранее фиксированных параметрах Θ, $p(\Theta)$ – априорной функцией распределения параметров Θ независимо от входных данных **r**, $p(\Theta|\mathbf{r})$ – апостериорная функция распределения вероятностей параметров Θ, которая дает наиболее вероятные значения Θ для входных данных **r**, $p(\mathbf{r})$ является нормализацией, чтобы полученное апостериорное распределение было действительно распределением вероятностей.

Сделаем предположение, что оценка пользователя является случайной величиной и имеет нормальной распределение:

$$r_{ui} \sim \mathcal{N}(r_{ui}|\mu + b_u + b_i + \mathbf{p}_u \cdot \mathbf{q}_i, \beta^{-1}).$$

Рассмотрим подробнее, что из себя представляет предсказание оценки пользователя. Пользователи в системе могут быть разными: одни могут ставить постоянно высокие оценки, другие – низкие, третьи – по-своему, а также сами объекты могут быть попросту лучше или хуже других, поэтому от всех таких отклонений стоит избавиться, введя, так называемые, базовые предикторы пользователя b_u и базовые предикторы объекта b_i, а также просто средний рейтинг по всей базе μ. Следовательно, остается лишь относительная оценка каждого пользователя, и эта оценка и есть скалярное произведение

вектора предпочтений пользователя $\mathbf{p}_u$ и вектора свойств объекта $\mathbf{q}_i$, что приводит к оценке, описанной выше:

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{p}_u \cdot \mathbf{q}_i.$$

Будем считать, что все оценки пользователей являются независимыми случайными величинами, поэтому функцию правдоподобия можно записать в виде:

$$p(\mathbf{r}|\Theta, \beta) = \prod_{u=1\ldots N,\ i=1\ldots M} \mathcal{N}(r_{ui}|\hat{r}_{ui}, \beta^{-1}),$$

где $\mathbf{r}$ – матрица всех оценок пользователей, а $\boldsymbol{\Theta}$ – вектор всех факторов ($\mathbf{p}_u$, $\mathbf{q}_i$, b_u, b_i для каждого пользователя u и для каждого объекта i), имеющего размер (N+M)(k+1), β^{-1} – дисперсия.

Рассмотрим вместо данной плотности распределения логарифм от нее (мы можем это сделать, т.к. логарифм является непрерывной монотонной функцией), а также подставим формулу для нормального распределения:

$$\begin{aligned}\ln p(\mathbf{r}|\boldsymbol{\Theta}, \beta) &= \ln \prod_{u=1\ldots N,\ i=1\ldots M} \frac{\beta^{1/2}}{\sqrt{2\pi}} \exp\{-\frac{(r_{ui} - \hat{r}_{ui})^2}{2}\beta\} \\ &= \sum_{u=1\ldots N,\ i=1\ldots M} \ln\frac{\beta^{1/2}}{\sqrt{2\pi}} + \sum_{u=1\ldots N,\ i=1\ldots M} -\frac{(r_{ui} - \hat{r}_{ui})^2}{2}\beta \\ &= NM\ln\frac{\beta^{1/2}}{\sqrt{2\pi}} - \frac{\beta}{2}\sum_{u=1\ldots N,\ i=1\ldots M}(r_{ui} - \hat{r}_{ui})^2 \end{aligned} \quad (1)$$

Следует заметить, что при максимизации данной функции правдоподобия можно отбросить первый член, т.к. он не зависит от максимизируемых параметров $\boldsymbol{\Theta}$. Кроме того, максимизация данной функции равносильна минимизации отрицательной функции правдоподобия.

Введём теперь априорное распределение параметров $\boldsymbol{\Theta}$:

$$p(\boldsymbol{\Theta}|\alpha) = \mathcal{N}(\boldsymbol{\Theta}|0, \alpha^{-1}\mathbf{I}) = (\frac{\alpha}{2\pi})^{(N+M)(k+1)} \exp\{-\frac{\alpha}{2}\boldsymbol{\Theta}^T\boldsymbol{\Theta}\}, \quad (2)$$

где матрица $\mathbf{I}$ – размера (N+M)(k+1), т.к. именно столько имеется факторов (N пользователей и M объектов по k факторов каждый, а также N + M базовых предикторов для пользователей и объектов соответственно).

По формуле Байеса имеем следующее выражение для апостериорной вероятности оценок пользователей:

$$p(\boldsymbol{\Theta}|\mathbf{r}, \alpha, \beta) \propto p(\mathbf{r}|\boldsymbol{\Theta}, \beta)p(\boldsymbol{\Theta}|\alpha) \quad (3)$$

Теперь можно найти наиболее вероятное значение вектора $\boldsymbol{\Theta}$ по имеющимся входным данным (оценкам) $\mathbf{r}$, т.е. максимизируя апостериорную вероятность. Подставляя (1) и (2) в (3), и, вычисляя отрицательный логарифм выражения, получим:

$$\ln p(\boldsymbol{\Theta}|\mathbf{r}, \alpha, \beta) = \frac{\beta}{2} \sum_{u=1\ldots N,\ i=1\ldots M} (r_{ui} - \hat{r}_{ui})^2 + \frac{\alpha}{2}\boldsymbol{\Theta}^T\boldsymbol{\Theta}$$

или, подставляя значения вектора $\boldsymbol{\Theta}$:

$$\ln p(\mathbf{p}_u, \mathbf{q}_i, b_u, b_i, \mu|\mathbf{r}, \alpha, \beta) = \frac{\beta}{2} \sum_{u=1\ldots N,\ i=1\ldots M} (r_{ui} - \tilde{r}_{ui})^2 + \frac{\alpha}{2}\left(\sum_{u=1\ldots N} \|\mathbf{p}_u\| + \sum_{i=1\ldots M} \|\mathbf{q}_i\| + \sum_{u=1\ldots N} b_u^2 + \sum_{i=1\ldots M} b_i^2 \right)$$

Таким образом, максимизация апостериорной функции распределения сводится к минимизации суммы квадратов отклонений с параметром регуляризации

$$\lambda = \frac{\alpha}{\beta}$$

Обучение *SVD* модели

Можно обучать такую линейную регрессию разными способами, например, стохастическим градиентным спуском (*SGD – Stochastic Gradient Descent*), обычный градиентный спуск не совсем подходит, т.к. он будет слишком медленным для данной задачи (из-за большого объема входных данных). Правила обновления после вычисления производных апостериорной функции распределения по всем параметрам будут выглядеть следующим образом (для каждой оценки всех пользователей):

$$\begin{aligned} p_{uj} &:= p_{uj} + \gamma(e_{ui}q_{ij} - \lambda p_{uj}), \forall j = 1 \ldots k \\ q_{ij} &:= q_{ij} + \gamma(e_{ui}p_{uj} - \lambda q_{ij}), \forall j = 1 \ldots k \\ b_u &:= b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &:= b_i + \gamma(e_{ui} - \lambda b_i) \\ \mu &:= \mu + \gamma e_{ui} \end{aligned} ,$$

где $e_{ui} = (r_{ui} - \mathbf{p}_u \cdot \mathbf{q}_i - b_u - b_i - \mu)$ – разница между реальным значением оценки и предсказанным, γ – скорость обучения.

В случае, если в системе нужны нечисловые оценки рейтинга, а бинарные (лайк/дизлайк, к примеру), то вместо оценки $\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{p}_u \cdot \mathbf{q}_i$ можно использовать логистический сигмоид:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\hat{r}_{ui} = \sigma(\mu + b_u + b_i + \mathbf{p}_u \cdot \mathbf{q}_i).$$

Известно, что данный сигмоид обладает следующим свойством:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)).$$

Поэтому, нетрудно показать, что после подстановки оценки $\hat{r}$ в выражение для апостериорной вероятности и нахождения производных правила обновления параметров не изменятся.

Реализация программного модуля

В качестве дипломной работы была разработана библиотека, реализующая алгоритмы формирования рекомендаций, в том числе и *SVD*-разложение. На базе данной библиотеки был построен сервис для рекомендации музыки, где оценки пользователей являются бинарными (лайк/дизлайк), поэтому в качестве предиктора используется логистический сигмоид как было показано выше. Кроме того, системой анализируется характер прослушивания аудиозаписей с целью выявления неявных оценок, в случае, если пользователь не поставит свою оценку самостоятельно.

Примерная архитектура сервиса представлена на рис. 2:

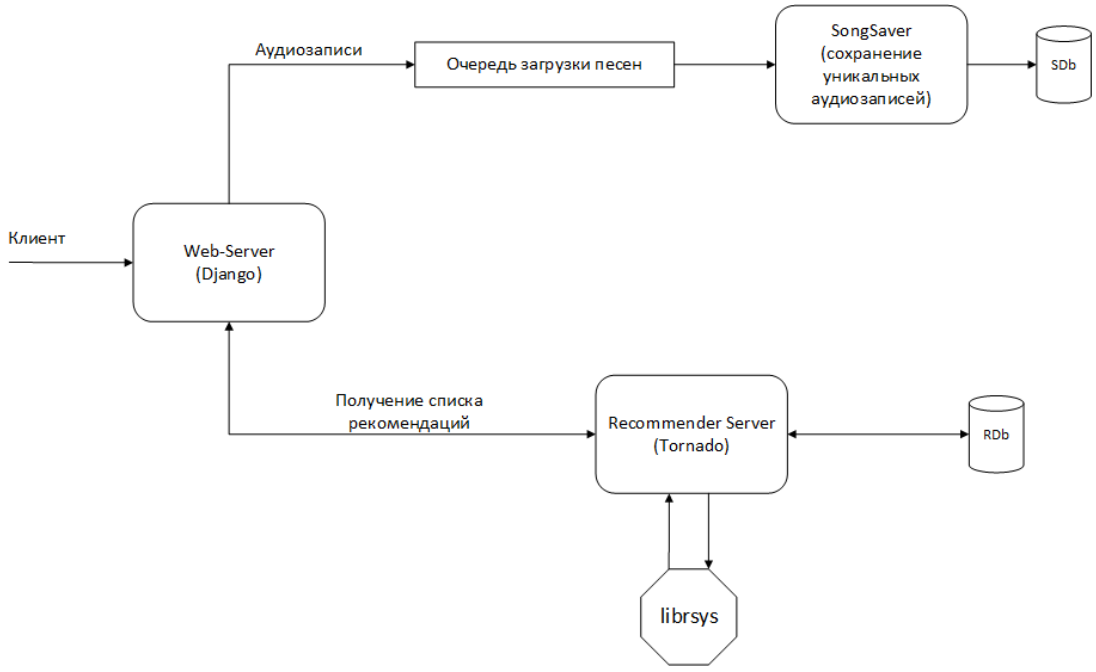

Рис. 2. Архитектура сервиса рекомендаций музыки

В системе присутствует два веб-сервера (первый – для общения с пользователем системы, второй (Recommender Server) – для обработки запросов на формирование списка рекомендаций). Сервер рекомендаций использует библиотеку librsys, написанную на C++ с байндингами в Python.

SongSaver создан для управления сохранения приходящих новых аудиозаписей с учётом их уникальности.

Оффлайн-часть рекомендательной системы (часть, которая запускается с некоторой периодичностью) реализована с помощью *BGD (Batch Gradient Descent)*, т.е. движение функции против градиента идет не по одной оценке, а по нескольким сразу («пачке» – *batch*). Количество факторов, которое используется для каждого пользователя и аудиозаписи равно восьми. Это количество выбрано в результате анализа результатов обучения на тестовой выборке.

Достоинством такой архитектуры является модульность (компоненты, входящие в систему, являются слабо связанными между собой и, как следствие, легко масштабируемы).

Заключение

В результате работы было показано, что *SVD*-разложение является довольно простым способом анализа оценок пользователей и формирования рекомендаций, а также, что функционал линейной регрессии вместе с регуляризатором является естественным следствием довольно простого предположения, что оценки пользователей – случайные величины с нормальным распределением.

Список литературы

1. Christopher M. Bishop. Pattern Recognition and Machine Learning. 1st ed. New York: Springer-Verlag, 2007. 703 p.
2. Matthew Brand. Fast online SVD revisions for lightweight recommender systems. Режим доступа: http://www.merl.com/publications/docs/TR2003-14.pdf (дата обращения: 29.03.2015).
3. Recommendation Engine Introduction. Режим доступа: http://dataaspirant.com/2015/01/24/recommendation-engine-part-1 (дата обращения: 29.03.2015).
4. Konstan J., Ekstard M. Introduction to Recommender Systems. Режим доступа: https://www.coursera.org/learn/recommender-systems/outline (дата обращения: 29.03.2015).