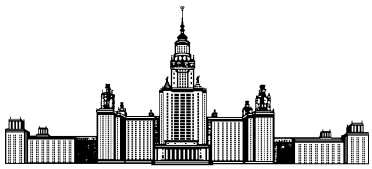
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

ДИПЛОМНАЯ РАБОТА СТУДЕНТКИ 517 ГРУППЫ

«Инкрементные матричные разложения в задачах коллаборативной фильтрации»

Выполнила:

студентка 5 курса 517 группы

Полежаева Елена Андреевна

Научный руководитель:

д.ф-м.н., доцент ВМиК МГУ

Воронцов Константин Вячеславович

Москва, 2011

Содержание

Аннотация

В современных приложениях методов коллаборативной фильтрации исходные данные могут иметь большие объёмы (миллионы клиентов и объектов) и поступать в реальном масштабе времени. В этом случае важно требование инкрементности: метод должен эффективно пересчитывать оценки сходства как при появлении новых клиентов и объектов, так и при обновлении значений в матрице исходных данных. Предлагается метод, который не требует хранения исходной матрицы и объединяет в себе оба типа инкрементности, используя инкрементное сингулярное разложение (ISVD) и инкрементное вычисление корреляции Пирсона (IPC).

Также предлагается алгоритм, основанный на методе градиентного спуска, позволяющий дополнять сжатые описания (профили) клиентов и объектов при решении задач коллаборативной фильтрации. Построен функционал, учитывающий особенность входных данных. Алгоритм удовлетворяет основным требованиям, предъявляемым к методам коллаборативной фильтрации. Алгоритм позволяет эффективно добавлять клиентов, объекты, модифицировать значения в ячейках исходной матрицы данных, совершая вычисления на основе известных значений. Результатом является лучшее заполнение пропущенных значений в ячейках исходной матрицы, чем были получены при использовании функционала, не учитывающего тип исходных данных.

1 Введение

1.1 Подходы к решению задач коллаборативной фильтрации

Задачи коллаборативной фильтрации стали актуальны в конце 80-х годов, когда появились потребности в эффективном использовании информации о поведении клиентов для решения бизнес — задач, таких как персонализация услуг и направленный маркетинг. В основе решения задач коллаборативной фильтрации лежит предположение о том, что схожие клиенты интересуются схожими объектами. Предпочтения схожих клиентов определяют возможные интересы данного клиента и позволяют составить список объектов, которые могут его заинтересовать.

Некоторые методы позволяют определять тематику интересов каждого клиента, а методы последних лет могут учитывать не только взаимосвязь клиентов и объектов (количество посещений того или иного сайта, рейтинги и т.п.), но и другие факторы, которые становятся очень важными при отсутствии большого количества информации в оценке. Под оценкой понимается информация, определяющая заполнение ячеек исходной матрицы «клиенты — объекты». Это может быть посещение сайта (очевидно, каждый клиент не может посетить весь огромный спектр сайтов, который присутствует в матрице), характеристика качества фильма и т.п. Итак, не все объекты оценены клиентом, и в матрице «клиенты — объекты» есть пропуски. Дополнительная информация в какой-то мере восполняет отсутствие значений в ячейках матрицы. Методы начинают учитывать такие факторы, как возраст клиента, его работа и т.д., или для объекта — жанр фильма, его режиссер и т.д.

Некоторые методы решения задач коллаборативной фильтрации, основанные на особенностях значений коэффициентов корреляции, сходствах векторов и статистических Байесовских методах, были предложены довольно давно (Breese et al., 1998 [8], Heckerman et al., 2000 [17]). В качестве результатов использованных методов клиенту выдавался упорядоченный список объектов (ресурсов), оцененных через вероятность. В последнее время определенный интерес получили методы, основанные на регуляризации (основоположники Srebro и Jaakkola, 2003 [22]).

Основной минус предыдущих работ, связанных с попытками решения задач коллаборативной фильтрации, связан с тем, что доступными данными являются только

выявленные интересы клиентов, и не используется никакая внешняя информация. В таком случае задача формулируется через частично заполненную матрицу предпочтений, в которой каждая строка — субъект (клиент), каждый столбец — объект (книга, фильм и т.п.), а элементы матрицы — оценки субъектов, данные объектам, например, рейтинги. Поскольку никакая внешняя информация ни о субъектах, ни об объектах не дана, заполнение пропусков в матрице происходит исходя из имеющихся элементов (рейтингов), представляющих собой факты покупки продукции, посещения или использования сайтов и т.п.

Для того чтобы предсказания были полезными, некоторые методы изучают факторы наличия связи между субъектами и объектами. Самое известное допущение заключается в том, что все предпочтения можно представить в виде небольшого количества признаков (features) как для субъектов, так и для объектов, так как именно небольшое количество признаков является определяющим для заполнения пропусков в исходной матрице. Задача сводится к нахождению матрицы небольшого ранга, которая аппроксимирует частично заполненную исходную матрицу (Srebro and Jaakkola, 2003 [22]). Выбранный ранг может рассматриваться в качестве регуляризации в признаковом пространстве, то есть допускается ограничение в размерности пространства исходных признаков с целью решения поставленной задачи и предотвращения переобучения. Примерами регуляризации являются введение штрафных функций за сложность модели, ограничение по норме векторного пространства. Задача становится сложной и невыпуклой, для решения которой известно только несколько эвристик, предложенных в [22].

Альтернативной формулировкой задачи стало штрафование исходной матрицы через сумму ее сингулярных значений (Srebro et al., 2005 [25]). Преимущество регуляризации, связанной с ограничением на сумму сингулярных значений матрицы, в том, что при достаточно большом параметре регуляризации, конечное решение будет иметь небольшой ранг (Fazel et al., 2001 [9], Bach, 2008 [5]). Важное ограничение методов, использующих регуляризацию, состоит в том, что они не учитывают часто доступную внешнюю информацию о клиентах и объектах (attributes), которая представлена в виде признаковых описаний. Интуитивно понятно, что такая информация очень полезна при формировании заключения о предпочтениях клиентов, особенно

для субъектов и объектов с маленьким числом связей (рейтингов). Например, нельзя рассматривать клиенты и объекты без наличия априорных рейтингов в классической формулировке задачи коллаборативной фильтрации, в то время как дополнительная информация о них могла бы дать предварительное заключение о предпочтениях.

Использование спектральной регуляризации (Jacob Abernethy et al., 2008 [3]) позволяет учитывать внешнюю информацию. Задача рассматривается уже не через матричное разложение, а через введение линейного оператора из пространства клиентов в пространство объектов, что эквивалентно введению билинейной формы между клиентами и объектами. В отличие от работы с матрицами, позволено работать с бесконечными размерностями, а произвольно выбранное пространство признаков может описываться некоторой функцией ядра. Выбор ядра — важная задача как для клиентов, так и для объектов. При удачном выборе ядра методы спектральной регуляризации представляют собой частные случаи существующих методов машинного обучения.

1.2 Постановка задачи

Методы коллаборативной фильтрации (CF) используются в рекомендующих системах и системах управления взаимоотношениями с клиентами (CRM) для автоматического формирования персональных предложений.

Целью рекомендующих систем является составление списка объектов, который соответствует интересам клиента, для того чтобы помочь клиенту выбрать объект из огромного множества вариантов. Такие системы имеют огромную важность в таких приложениях, как электронная торговля через Интеренет, сервисы, основанные на подписке, в которых подписка определяет, какая информация и на какое устройство будет послана абоненту уведомляющим приложением. Рекомендующие системы, которые позволяют составить клиенту индивидуальные предложения, значительно повышают вероятность того, что клиент совершит покупку. Индивидуальные рекомендации важны в магазинах, в которых много недорогих или одинаково стоящих товаров. Рекомендующие системы активно используются такими участниками электронной торговли, как Amazon и Netflix, а Netflix даже организовал конкурс для улучшения качества рекомендующей системы. Однако известно очень мало инфор-

мации об интересах клиента. Добавляются новые товары, клиенты, клиенты покупают новые товары, таким образом расширяя список своих интересов. Для создания хорошей рекомендующей системы требуется метод, который может эффективно добавлять клиентов, объекты, наращивать информацию о клиентах, получать хороший прогноз. Часто клиент ставит рейтинг объекту, например, фильму. Для улучшения качества прогноза можно воспользоваться знанием о типе входной информации.

Исходными данными является матрица Y, строки которой соответствуют n клиентам, столбцы — d объектам. В зависимости от приложения объектами могут быть товары, услуги, документы, и т. п. Каждая заполненная ячейка матрицы содержит информацию об использовании данным клиентом данного объекта. Это может быть отметка о посещении, выставленный клиентом рейтинг, заплаченная им сумма, и т. д. Задача состоит в том, чтобы для произвольного клиента спрогнозировать оценки предпочтительности объектов по всем незаполненным ячейкам в строке матрицы Y. Предполагается, что для этого выделяется множество клиентов со схожими предпочтениями (коллаборация).

Важно, чтобы метод удовлетворял требованиям, предъявляемым к методам CF:

- Инкрементность — эффективное добавление строк, столбцов и ячеек в Y;
- Строить модель, учитывая тип данных (порядковые / вещественные значения в Y);
- Возможность вычислений при разреженной Y;
- Находить сходство (корреляцию) между профилями (сжатыми описаниями) клиентов, профилями объектов, профилями клиентов и объектов.

Целью работы стала разработка метода, гибкого к добавлению клиентов, объектов, а также позволяющего спрогнозировать интересы клиента, учитывая особенность входных данных. При этом вычисления производятся на основе известных элементов Y.

Простые методы CF, основанные на поиске корреляций между клиентами (user — based) или объектами (item — based), неэффективны по времени и по памяти, поскольку они требуют хранения всей матрицы Y. Этих недостатков лишены методы сингулярного разложения SVD [6], неотрицательного матричного разложения

NNMF [20] и вероятностного латентного семантического анализа PLSA [12], позволяющие формировать сжатые описания (профили) клиентов и объектов. В современных динамических приложениях к методам CF предъявляется также требование инкрементности: метод должен эффективно пересчитывать хранимую информацию в случаях появления (1) нового клиента или объекта и (2) нового значения в ячейке матрицы Y. Обычно в работах по CF рассматривается только один из этих двух типов инкрементности.

В работах Brand [6], [7] эффективно добавляются объекты и элементы матрицы Y. В работе [28] предложен алгоритм, эффективно добавляющий только клиентов.

В данной работе рассматриваются разные инкрементные методы CF, и предлагается метод, который объединяет оба типа инкрементности. В первой части работы предлагается применять совместно формулы инкрементного вычисления корреляции Пирсона (IPC) [24] и инкрементного сингулярного разложения (ISVD) [7]. При изменении одной ячейки матрицы Y получена сложность нахождения сходства между клиентами при применении инкрементного вычисления корреляции Пирсона (IPC) [24]. Разбиение формулы на части и хранение промежуточных величин повышает эффективность вычисления оценок сходства клиентов и объектов. При добавлении клиента или объекта применяется инкрементное сингулярное разложение (ISVD) [7], в результате которого получается новое SVD для $Y_{n\times d} = U_{n\times r}S_{r\times r}(R^T)_{r\times d}$. Матрица US, содержащая профили клиентов, используется в формуле IPC. Эффективное вычисление новой строки матрицы U, отвечающей добавленному клиенту, позволяет применять IPC для US параллельно с получением ISVD. Оценим сложность вычислений. Пусть k — число клиентов, имеющих хотя бы с одним другим клиентом общий объект, n': $n' \ll n$ — число клиентов, с которыми данный клиент имеет хотя бы один общий объект, d': $d' \gg d$ — количество объектов, не оцененных данным клиентом, но оцененных по крайней мере одним из близких к нему клиентов. Метод не требует хранения всей матрицы Y. Объем хранимых данных $O(n + kn' + (n + d)r)$. Добавление клиента требует $O((nr + d)r)$ операций, добавление объекта — $O((nr + d)nr)$ операций, модификация ячейки матрицы Y — $O(n'd')$ операций. Таким образом, предложенный инкрементный метод CF достаточно эффективен для использования в современных приложениях.

Также предлагается рассмотреть инкрементные методы сингулярного разложения, основанные на обобщенном алгоритме обучения Хебба [14]. Данный алгоритм использует шаги градиентного спуска, а также позволяет брать для вычислений только непустые ячейки исходной матрицы Y. Вводится функционал, позволяющий учесть информацию о том, что в ячейках Y находятся порядковые значения (например, рейтинги). Подбираются параметры для уменьшения суммы квадратов ошибок между значениями, полученными моделью, и реальными данными. При этом ограничений на сами параметры не накладывается. Таким образом, в работе рассмотрены случаи, когда элементами Y являются порядковые или вещественные данные.

В результате экспериментов выявлено, что возможно добавлять клиентов, не модифицируя профили (сжатые описания) объектов. Объем данных влияет на скорость сходимости алгоритма. При увеличении числа клиентов (строк Y) растет скорость сходимости алгоритма (от 1000 итераций при 600 клиентах до 40 — на 940-м клиенте).

Ниже описаны цели и проблемы, которые возникают при решении задач коллаборативной фильтрациии. Также формулируется задача коллаборативной фильтрации на основе матричных разложений. В разделе 2 подробно описывается подход, основанный на вычислительной формуле корреляции Пирсона и вводится понятие инкрементного сингулярного разложения. Предлагается совместный алгоритм инкрементного сингулярного разложения и вычислительной формулы корреляции Пирсона. Раздел 3 формулирует обобщенный алгоритм обучения Хебба, описывает подходы при решении задач с порядковыми данными в Y. Раздел 4 описывает эксперимент.

1.3 Основные обозначения

Введем основные обозначения, которые далее будут использоваться в работе.

n — число клиентов;

d — число объектов;

$Y_{n\times d}$ — разреженная матрица исходных данных;

$\widehat{Y}_{n\times d}$ — матрица небольшого ранга, аппросимирующая матрицу Y;

$U_{n\times L}$ — матрица, отвечающая профилям клиентов;

$R_{L\times d}$ — матрица, отвечающая профилям объектов;

L — число факторов;

$\Omega \subseteq \{1, ..., n\} \times \{1, ..., d\}$ — множество индексов непустых элементов Y.

1.4 Задача коллаборативной фильтрации. Структура исходных данных

Задачи коллаборативной фильтрации формулируются в трех основных аспектах:

1. вероятностный подход к решению задачи [21];
2. матричное разложение наименьшего ранга [25];
3. спектральная регуляризация — классическая теория, использующая тензорное произведение как средство представления ядер в гильбертовых пространствах [4].

Нас будут интересовать матричные разложения.

1.5 Цели, проблемы коллаборативной фильтрации

Пусть U — множество субъектов (клиентов — users), R — множество объектов (items), Y — характеристика оценки ресурсов клиентами (факт взаимодействия — рейтинги, частота использования ресурса и т.п.).

Тогда исходные данные можно представить в виде протокола — последовательности записей: $D = (u_i, r_i, y_i)_{i=1}^{l} \subset U \times R \times Y$, где l — количество записей в протоколе [21]. Протоколы представляются в виде матрицы размера $|U| \times |R|$ с помощью некоторой функции агрегирования $aggr$, зависящей от постановки задачи и описания данных: $F = [f_{ur}]$, где $f_{ur} = aggr((u_i, r_i, y_i) \subset D | u_i = u, r_i = r)$. Элементами матрицы могут быть рейтинги, частота посещения объекта клиентом и т.п.

Определим некоторые из целей коллаборативной фильтрации:

- прогнозирование незаполненных ячеек матрицы «клиенты — объекты»;
- оценка функций расстояния между клиентами $\rho(u_i, u_j)$, между объектами $\rho(r_i, r_j)$ и клиентами и объектами соответственно $\rho(u, r)$;
- выявление интересов клиентов;

- изучение активности использования тех или иных объектов в зависимости от внешней информации и т.д.

Очень часто требуется определить список объектов, «близких» клиенту, найти клиентов со схожими интересами, выдать объекты, каким-то образом родственные данному исходному объекту.

Итак, коллаборативная фильтрация оценивает предпочтения клиентов, сходство объектов и клиентов по взаимосвязи между собой. Данные о предпочтениях клиентов могут быть доступны как в явном, так и неявном виде. В первом случае объекты оцениваются клиентами. Клиенты могут выставлять рейтинги по разным шкалам, например, от 1 до 5. Во втором случае клиенты не оценивают объекты в явном виде. Рейтинги по объектам (например, ресурсам) определяются, например, условием, посетил клиент сайт или нет (это может делать сама система автоматически). Вследствие этого, соответствующим элементом матрицы «клиенты — объекты» становится 1 или 0.

Сам процесс коллаборативной фильтрации может быть разделен на 2 фазы: фаза моделирования и фаза рекомендации. Те алгоритмы, которые стремятся обойти первую фазу, относятся к алгоритмам, называющимся анамнестическими *(memory — based)*, например, алгоритм ближайших соседей.

Выделим 3 проблемы коллаборативной фильтрации:

1. проблема разреженности данных;
2. проблема масштаба вычислений;
3. проблема использования априорной информации.

Первая проблема, которая появляется при наличии в Y большого количества пропусков, может решаться посредством добавления других данных, например, содержания объектов (атрибутов), играющих значительную роль в методах спектральной регуляризации [2], а также кластеризации клиентов и/или объектов и уменьшении размерности исходной матрицы.

Нельзя избежать проблемы роста вычислений. В современных динамических приложениях данные постоянно обновляются — добавляются новые клиенты, объекты,

модифицируется информация в отдельных ячейках матрицы.

Одним из самых простых алгоритмических подходов, используемых в коллаборативной фильтрации, является алгоритм k — ближайших соседей (оцениваются наиболее близкие клиенты для исходного клиента). Для того чтобы заполнить пропуски в Y, применяется усреднение рейтингов «близких» клиентов по данному объекту. Также возможно вычисление взвешенных средних и корреляции для разных объектов и отдельных клиентов. Еще одним подходом является рассмотрение интересов клиента через признаковый вектор. Тогда объекты становятся признаками, а отвечающие им рейтинги — их признаковыми значениями. Следующая формула может быть применена для предсказания клиенту u рейтинга некоторого объекта r [15]:

$$p_{ur} = \bar{y_u} + k \sum_{v \in U} w(u,v)(y_{vr} - \bar{y_v})$$

где $w(u,v)$ — веса, которые тем больше, чем «ближе» клиенты u и v (наиболее коррелированные), $\bar{y_u}$ — среднее значение рейтинга, данное клиентом u, y_{vr} — рейтинг объекта r, данный клиентом v, k — нормировочный множитель, зависящий от способа выбора весов. Представляя признаковый вектор, мы сталкиваемся с проблемой разреженности, так как многие объекты клиентом не оценены. Если нет доступных рейтингов по объекту r, тогда значением компоненты вектора клиента, отвечающей данному объекту, может являться среднее значение рейтинга, данное клиентом u. Подсчеты весов могут производиться посредством вычисления косинусов углов между признаковыми векторами [15]:

$$w(u,v) = \frac{\sum\limits_{r \in R} y_{ur} y_{vr}}{\sqrt{\sum\limits_{r \in R_1} y_{ur}^2 \sum\limits_{r \in R_2} y_{vr}^2}}$$

Веса также могут рассматриваться в терминах корреляции Пирсона: 1 — положительная взаимосвязь, -1 — отрицательная, 0 — нет связи [15]:

$$w(u,v) = \frac{\sum\limits_{r \in R'} (y_{ur} - \overline{y}_u)(y_{vr} - \overline{y}_v)}{\sqrt{\sum\limits_{r \in R'} (y_{ur} - \overline{y}_u)^2 \sum\limits_{r \in R'} (y_{vr} - \overline{y}_v)^2}}$$

Из-за разреженности векторов существует мало объектов, которые были оценены многими клиентами, поэтому, используя корреляцию Пирсона, мы сталкиваемся с проблемой невидимых для нас транзитивных связей. Вместо того чтобы искать «пересечения» клиентов по объектам, можно брать их объединение и заполнять пропущенные значения вектора некоторыми предопределенными значениями. Пропущенные рейтинги также могут быть предсказаны как среднее значение рейтингов по объектам, которые «близки».

1.6 Коллаборативная фильтрация на основе матричных разложений

Для n клиентов и d объектов используется L — факторная модель, представленная матрицей $U_{n\times L}$, в которой каждая строка отвечает силе связи клиента с каждым фактором, и матрицей $R_{L\times d}$, столбцы которой отвечают силе связи объекта с каждым фактором. Матрицы, допускающие такое разложение, имеют ранг не более, чем L. Матрица предпочтений (рейтингов) Y аппроксимируется матрицей $\widehat{Y}$ меньшего ранга, где $\widehat{Y} = UR$. Иначе говоря, каждый элемент матрицы $\widehat{Y}$ выглядит следующим образом: $\widehat{y}_{ij} = \sum\limits_{l=1}^{L} u_{il}r_{lj}$. Это и есть представление данных через профили клиентов и объектов.

Также каждая ячейка Y может быть представлена в аналогичном виде, где в формулу добавляется s_l — важность l-ого фактора: $\widehat{y}_{ij} = \sum\limits_{l=1}^{L} u_{il}s_l r_{lj}$, где

l — фактор (например, тема); L — множество факторов;

s_l — важность l-ого фактора;

u_{il} — элемент профиля i-го клиента (сила связи i-го клиента и l-го фактора);

r_{lj} — элемент профиля j-го объекта (сила связи j-го объекта и l-го фактора).

1.7 Алгоритмы, допускающие ограничение на норму следа матрицы в задачах коллаборативной фильтрации

Итак, вернемся к точной матричной формулировке задачи коллаборативной фильтрации. Матрица оценок объектов (рейтингов) Y аппроксимируется матрицей $\widehat{Y}$ меньшего ранга. Матрица $\widehat{Y}$, минимизирующая квадрат суммы расстояний до пол-

ностью заполненной матрицы Y, эффективно находится через главные сингулярные числа Y. Однако в коллаборативной фильтрации матрица Y обычно не заполнена полностью, нахождение матрицы $\widehat{Y}$ путем сингулярного разложения усложняется, так как при минимизации функционала:

$$\|Y - UR\|^2 = \sum_{i,j} \left(y_{ij} - \sum_l u_{il} r_{lj} \right)^2 \to \min_{U,R}$$

приходится брать сумму не по всем индексам i и j. Нахождение аппроксимации становится очень трудной задачей с множеством локальных минимумов. Некоторые эвристики предлагают Srebro and Jaakkola, 2003 [22]. Эту проблему очень интересно решают Alex Kleeman, Nick Hendersen и Sylvie Denuit в статье «Matrix Factorization for Collaborative Prediction» [19]. Предложено использовать инкрементное сингулярное разложение (постепенно наращивается количество вектор — столбцов значений, этим и объясняется такое название). В своей работе авторы опираются на результаты [22], поэтому обратимся вначале к источнику. Даже если матрица Y полностью известна, нахождение матрицы $\widehat{Y}$ — невыпуклая оптимизационная задача с многочисленными локальными минимумами. Уже описанный подход к решению данной задачи — использование немногочисленности признаков и неотрицательности компонент матриц. Его исследуют Lee и Seung, 1999 [20]. Srebro в своих последних работах, начиная с 2005 года, приходит к тому, что лучше ограничивать не размерность матриц U и R, а их нормы [25]. При таком подходе количество признаков не ограничивается и может быть очень большим, зато минимизация норм матриц (при соответствующем введении нормы, например, Фробениуса) приводит к тому, что элементы не могут быть очень большими, а значит, влияние признака (например, темы) на клиента или объект уменьшается. Например, в рейтингах фильмов большую роль играют признаки, характеризующие жестокость, чуть меньшую, отвечающие за драматичность или комичность, не так важно оформление сцен или музыка (эти признаки могут иметь очень маленькие значения). Математически, ограничение норм U и R эквивалентно ограничению суммы сингулярных значений матрицы $\widehat{Y}$. Задача упрощается, так как становится выпуклой оптимизационной задачей. Исходно задача является невыпуклой, даже если минимизируется сумма квадратов ошибок, но в специальных случаях, когда минимизируется сумма квадратов ошибок полностью

заполненной матрицы, все локальные минимумы являются глобальными [22]. Аппроксимация матрицы $Y_{n\times d}$ матрицей $\widehat{Y} = UR$ ранга L становится линейной задачей прогнозирования, если одна из двух матриц, U или R, известна (то есть остается найти только одну матрицу). В коллаборативной фильтрации ни одна из матриц не известна, поэтому требуется найти такие признаковые вектора (строки U), отвечающие каждой строке в Y, которые одновременно осуществляют хорошее линейное предсказание относительно всех объектов (столбцы Y), каждая из которых имеет свой линейный предиктор (столбцы R). Столбцы R определяют значения в соответствующих столбцах матрицы Y, опираясь на значения признаковых векторов в U.

2 Вычислительная формула корреляции Пирсона и инкрементное сингулярное разложение

2.1 Вычислительная формула корреляции Пирсона

Классический алгоритм коллаботративной фильтрации генерирует предложения, которые основаны на интересах подмножества клиентов, наиболее «близких» искомому клиенту. Для того чтобы найти сходство между клиентами, были предложены различные меры сходства. В коллаборативной фильтрации корреляция Пирсона является одним из наиболее распространённых способов оценивания сходства $\mathrm{sim}(u, v)$ между клиентами u, v. Обычно матрица $Y = (y_{ur})$ является сильно разреженной, и для произвольного клиента u большинство элементов в строке пустые, $y_{ur} = \emptyset$, что обязательно должно учитываться при вычислении корреляции Пирсона:

$$\mathrm{sim}(u, v) = \frac{\sum\limits_{r\in R_{uv}} (y_{ur} - \bar{y}_u)(y_{vr} - \bar{y}_v)}{\sqrt{\sum\limits_{r\in R_{uv}} (y_{ur} - \bar{y}_u)^2 \sum\limits_{r\in R_{uv}} (y_{vr} - \bar{y}_v)^2}}, \tag{1}$$

где $R_{uv} = \{r\colon y_{ur} \neq \emptyset,\ y_{vr} \neq \emptyset\}$ — подмножество объектов, которые оценили клиенты u и v, причём, как правило, $|R_{uv}| \ll d$; y_{ur} — рейтинг, данный клиентом u объекту r; $\bar{y}_u, \bar{y}_v$ — средние рейтинги клиентов u и v.

2.1.1 Основная идея

Когда клиент u добавляет новый рейтинг или изменяет имеющийся, должны изменяться и значения сходства между этим клиентом и остальными. Инкрементное вычисление корреляции Пирсона [24] основано на представлении (1) в виде функции от трёх переменных B, C, D, которые вычисляются аддитивно по элементам матрицы Y:

$$A = \mathrm{sim}(u, v) = B/\sqrt{CD};$$
$$B = \sum_{r \in R_{uv}} (y_{ur} - \bar{y}_u)(y_{vr} - \bar{y}_v),$$
$$C = \sum_{r \in R_{uv}} (y_{ur} - \bar{y}_u)^2, \quad D = \sum_{r \in R_{uv}} (y_{vr} - \bar{y}_v)^2.$$

Значения переменных A, B, C, D можно быстро пересчитать как после заполнения ранее пустой ячейки матрицы Y, так и после изменения значения ячейки, по общей формуле

$$A_{\mathrm{new}} = B_{\mathrm{new}}/\sqrt{C_{\mathrm{new}} D_{\mathrm{new}}},$$
$$B_{\mathrm{new}} = B + b, \quad C_{\mathrm{new}} = C + c, \quad D_{\mathrm{new}} = D + d,$$

где величины b, c, d, называемые *инкрементами*, вычисляются по-разному в случаях добавления и изменения ячейки. Рассмотрим эти случаи по отдельности.

Добавление нового рейтинга. Требуется найти сходство между клиентами u и v, когда клиент v добавляет новый рейтинг для объекта r. Возможны два случая:

- клиент u уже оценил r; тогда B, C, D изменяются в соответствии с $\bar{y}'_v$ — новым средним рейтингом v, добавленным рейтингом v для объекта r и новым количеством взаимно оцененных объектов клиентами u и v:

$$b = (y_{vr} - \bar{y}'_v)(y_{ur} - \bar{y}_u) - \sum_{r \in R_{uv}} \delta\bar{y}_v (y_{ur} - \bar{y}_u),$$
$$c = (y_{vr} - \bar{y}'_v)^2 + \sum_{r \in R_{uv}} \delta\bar{y}_v^2 - 2\sum_{r \in R_{uv}} \delta\bar{y}_v (y_{vr} - \bar{y}_v),$$
$$d = (y_{ur} - \bar{y}_u)^2,$$

где

$\bar{y}_u$, $\bar{y}_v$ — средний рейтинг клиентов u и v;

$\bar{y}'_v = \frac{y_{vr}}{m_v+1} + \frac{m_v}{m_v+1}\bar{y}_v$ — новое значение среднего рейтинга активного клиента v;

m_v — число объектов, оцененных клиентом v;

$\delta\bar{y}_v = \bar{y}'_v - \bar{y}_v$ — разность текущего и предыдущего значений среднего рейтинга клиента v;

- клиент u не оценивал r; тогда B, C изменяются в соответствии с новым средним рейтингом v (в этом случае формулы для инкрементов выписываются аналогично):

$$b = -\sum_{r \in R_{uv}} \delta\bar{y}_v (y_{ur} - \bar{y}_u),$$
$$c = \sum_{r \in R_{uv}} \delta\bar{y}_v^2 - 2\sum_{r \in R_{uv}} \delta\bar{y}_v (y_{vr} - \bar{y}_v),$$
$$d = 0.$$

Изменение рейтинга. Требуется найти сходство между клиентами u и v, когда клиент v модифицирует рейтинг для объекта r. Возможны два случая:

- клиент u уже оценил r, тогда B, C изменяются в соответствии с новым средним рейтингом v и добавленным рейтингом v для объекта r:

$$b = \delta\bar{y}_{vr}(y_{ur} - \bar{y}_u) - \sum_{r \in R_{uv}} \delta\bar{y}_v (y_{ur} - \bar{y}_u),$$
$$c = \delta y_{vr}^2 + 2\delta y_{vr}(y_{vr} - \bar{y}'_v) + \sum_{r \in R_{uv}} (\delta\bar{y}'_v)^2 - 2\sum_{r \in R_{uv}} \delta\bar{y}_v (y_{vr} - \bar{y}_v),$$
$$d = 0.$$

- клиент u не оценивал r , тогда B, C изменяются в соответствии с новым средним рейтингом v:

$$b = -\sum_{r \in R_{uv}} \delta\bar{y}_v (y_{ur} - \bar{y}_u),$$
$$c = \sum_{r \in R_{uv}} (\delta\bar{y}'_v)^2 - 2\sum_{r \in R_{uv}} \delta\bar{y}_v (y_{vr} - \bar{y}_v),$$
$$d = 0.$$

2.1.2 Сохранение параметров

Следующие величины требуется хранить и изменять при работе метода:

- B, C, D для всех пар клиентов u, v;
- m_u — количество объектов, оцененных каждым клиентом u;
- $\bar{y}_u$ — средний рейтинг для каждого клиента u;
- $\bar{y}'_v$ — новое значение среднего рейтинга активного клиента v:
  $\bar{y}'_v = \frac{y_{vr}}{m_v+1} + \frac{m_v}{m_v+1}\bar{y}_v$ при добавлении рейтинга,
  $\bar{y}'_v = \frac{\delta\bar{y}_v}{m_v} + \bar{y}_v$ при модификации рейтинга;
- $\delta\bar{y}_u = \bar{y}'_u - \bar{y}_u$ — разность текущего и предыдущего среднего значения рейтинга клиента u;
- $\sum\limits_{r\in R_{uv}} y_{ur}$, $\sum\limits_{r\in R_{uv}} y_{vr}$ — сумма рейтингов по объектам, оцененным каждой парой клиентов u, v.

2.1.3 Алгоритм «Инкрементные вычисления корреляции Пирсона»

В зависимости от того, добавляется или модифицируется значение в ячейке Y, можно написать следующий алгоритм. В нем явно просматриваются 4 случая.

Алгоритм «Инкрементные вычисления корреляции Пирсона»

Вход: (v, r);

Выход: модифицированное значение сходства $\mathrm{sim}(v, u)$;

1: **если** добавление рейтинга **то**
2: **для всех** $u \in U$
3: **если** u уже оценил **то**
4: ДобавлениеОц (v, r, u);
5: **иначе если** u не оценил **то**
6: ДобавлениеНеОц (v, r, u)
7: **end если**

8: **end для**

9: **иначе если** модификация рейтинга **то**

10: **для всех** $u \in U$

11: **если** u уже оценил **то**

12: МодифицированиеОц (v, r, u);

13: **иначе если** u не оценил **то**

14: МодифицированиеНеОц (v, r, u)

15: **end если**

16: **end для**

17: **end если**

Приведем пример работы алгоритма в случае добавления рейтинга, оцененного u:

ДобавлениеОц (v, r, u)

$$\overline{y}'_v = \frac{y_{vr}}{m+1} + \frac{m}{m+1}\overline{y}_v;$$

$$\overline{y}'_v = \overline{y}_v + h\overline{y}_v;$$

$$b = (y_{vr} - \bar{y}'_v)(y_{ur} - \bar{y}_u) - \sum_{r \in R_{uv}} \delta\bar{y}_v(y_{ur} - \bar{y}_u);$$

$$c = (y_{vr} - \bar{y}'_v)^2 + \sum_{r \in R_{uv}} \delta\bar{y}_v^2 - 2\sum_{r \in R_{uv}} \delta\bar{y}_v(y_{vr} - \bar{y}_v);$$

$$d = (y_{ur} - \bar{y}_u)^2;$$

$$B = B + b;$$

$$C = C + c;$$

$$D = D + d;$$

$$A = \frac{B}{\sqrt{C}\sqrt{D}} = \mathrm{sim}(u, v);$$

$$R_{uv} = R_{uv} \bigcup r;$$

$$d' = d' + 1;$$

$$\sum_{r \in R_{uv}} y_{ur}, \sum_{r \in R_{uv}} y_{vr}.$$

{Сохраняется сумма рейтингов}

Другие 3 функции пишутся абсолютно аналогично.

2.1.4 Сложность алгоритма

Такое разбиение формулы на части при хранении промежуточных величин повышает эффективность вычисления оценок сходства клиентов и объектов. Важными являются следующие параметры: n' — число клиентов, с которыми данный клиент имеет хотя бы один общий объект, как правило, $n' \ll n$ ($n' > 0$ — условие, позволяющее оценивать сходство); d' — число объектов, не оцененных данным клиентом, но оцененных по крайней мере одним из близких к нему клиентов, как правило, $d' \ll d$ ($d' > 0$ — условие того, что хотя бы один объект может быть рекомендован данному клиенту).

Метод не требует хранения всей матрицы Y. Модификация ячейки матрицы Y требует $O(n'd')$ операций для поддержания матрицы сходств клиентов (не более, чем n' сходств надо модифицировать, и не больше, чем d' объектов просмотреть для формирования рекомендаций клиенту). Для сравнения, в классическом методе CF сложность вычислений будет порядка n^2, так как надо дважды в цикле «обойти» всех клиентов (рассмотреть все пары клиентов), и потом еще объекты, оцененные обоими клиентами.

2.2 Инкрементное сингулярное разложение

Второй подход связан с сингулярным разложением, которое позволяет представить матрицу Y в виде произведения двух ортогональных матриц U и R и диагональной матрицы S, так что верны равенства $USR^T = Y,\ U^TYR = S$. Диагональные элементы S называются сингулярными числами, а столбцы U и R — левыми и правыми сингулярными векторами, соответственно. Оставим только r наибольших сингулярных чисел в S, у U и R оставим только столбцы, отвечающие этим сингулярным числам. Результат после прореживания матриц $U'S'R'^T \approx Y$ есть наилучшая аппроксимация ранга r матрицы Y в смысле наименьших квадратов. Нас будет интересовать именно такое разложение.

2.2.1 Постановка задачи ISVD

Пусть $USR^T = Y$ — разреженное SVD ранга r матрицы $Y_{n\times d}$, где $U^TU = R^TR = I$. Задача состоит в том, чтобы модифицировать U, S, R так, чтобы получить SVD для новой матрицы $Y + AB^T$, где A и B имеют c столбцов ($A_{n\times c}$,$B_{d\times c}$).

$c = 1$ — частный случай. Тогда A и B являются столбцами, обозначим их a и b.

Модификация данных, удаление строк и столбцов, добавление строк и столбцов — случаи уменьшения или увеличения ранга на 1. Пусть есть два вектор столбца a и b и известное сингулярное разложение матрицы $Y = USR^T$, где b — бинарный вектор, определяющий, какие столбцы должны быть модифицированы, а a — вектор, состоящий из элементов добавления и удаления значений (c), модификации значений в выбранных строках и столбцах (d), или изменения среднего значения (изменение центра) (m), которое должно быть вычтено из всех столбцов [7]. Требуется найти SVD матрицы $Y + ab^T$.

Таблица 1 . Операции над последним столбцом или над всеми столбцами как модификации ранга 1 для SVD матрицы $Y = USR^T$ и получение нового SVD для $Y + ab^T = U'S'R'^T$

Операция	Известное разложение	Требуемое разложение	a	b
Добавление столбца	$US[R^T \quad 0] = [Y \quad 0]$	$U'S'R'^T = [Y \quad c]$	c	$[0, ..., 0, 1]^T$
Удаление столбца	$USR^T = [Y \quad c]$	$U'S'R'^T = Y$	$-c$	$[0, ..., 0, 1]^T$
Модификация значений столбца	$USR^T = [Y \quad c]$	$U'S'R'^T = [Y \quad d]$	$d - c$	$[0, ..., 0, 1]^T$
Изменение центра	$USR^T = Y$	$U'S'R'^T = Y - m1^T$	$-m$	$1^T = [1, ..., 1, 1]^T$

2.2.2 Основные вычисления

Покажем, как можно совершить такие модификации в подпространствах низкой размерности, заданные известным SVD. Идея состоит в том, что новое SVD может быть выражено через известные подпространства (слабо пополненные) и матрицу,

близкую к диагональной, но которая может быть диагонализирована через левое и правое вращения. Применяя вращения к подпространствам, получаем новое SVD [7].

Эффективные преобразования ранга 1 позволяют модифицировать, а также удалять строки и столбцы Y без участия всей матрицы U или R. Требуется найти SVD следующей блочной матрицы:

$$Y + AB^T = [U, A] \begin{bmatrix} S & 0 \\ 0 & I \end{bmatrix} [R, B]^T. \tag{2}$$

Интересен случай, когда $rank(Y + AB^T) \leq r + c \leq \min(n, d)$. Будем использовать искусственно введенный метод.

Пусть P — ортогональный базис пространства столбцов $\tilde{A} = (I - UU^T)A = A - UU^TA$. Применим QR-разложение к $\tilde{A}$: $\tilde{A} = PR_A$. P — ортонормированная, а R_A — верхняя треугольная матрица, $R_A = P^T(I - UU^T)A$.

Итак, запишем QR-разложение [1]:

$$[U, P] \begin{bmatrix} I & U^TA \\ 0 & R_A \end{bmatrix} = [U, A]. \tag{3}$$

Разложение может быть эффективно вычислено, в частности, с помощью модифицированной ортогонализации Грама–Шмидта [13].

Аналогично, пусть Q — ортогональный базис $B - RR^TB$: $QR_B = (I_RR^T)B$.

Представим матрицу $Y + AB^T$ в виде произведения трёх матриц:

$$\begin{aligned} Y + AB^T &= [U, P] \begin{bmatrix} I & U^TA \\ 0 & R_A \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & R^TB \\ 0 & R_B \end{bmatrix}^T [R, Q]^T \\ &= [U, P] \underbrace{\left(\begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U^TA \\ R_A \end{bmatrix} \begin{bmatrix} R^TB \\ R_B \end{bmatrix}^T \right)}_{K} [R, Q]^T; \\ K &= \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} (U^TA)(B^TR) & (U^TA)R_B^T \\ R_A(B^TR) & R_AR_B^T \end{bmatrix}. \end{aligned}$$

Пусть $U'S'R'^T$ — SVD ранга $(r+c)$ матрицы K. Диагонализируя K, $U'^T K R' = S'$, получаем новые матрицы U', S', R'. Тогда модификация ранга r SVD до ранга $(r+c)$ SVD выглядит так:

$$Y + AB^T = \big([U\ P]U'\big)S'\big([R\ Q]R'\big)^T. \tag{4}$$

Заметим, что матрица Y не требуется.

Докажем, что выше приведенные равенства верны.

Изменения ранга 1 можно производить следующим образом: для измененного SVD (a — вектор размерности n, b — вектор размерности d) используем модификацию Грама-Шмидта [13]:

$$z = U^T a; \qquad p = a - Uz;$$
$$w = R^T b; \qquad q = b - Rw;$$

Тогда верно:

$$\begin{bmatrix} U^T A \\ R_A \end{bmatrix} \begin{bmatrix} R^T B \\ R_B \end{bmatrix}^T = \begin{bmatrix} z \\ \|p\| \end{bmatrix} \begin{bmatrix} w \\ \|q\| \end{bmatrix}^T; \tag{5}$$

$$K = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} zw^T & z\|q\| \\ \|p\|w^T & \|p\|\|q\| \end{bmatrix}.$$

2.2.3 Примеры эффективных преобразований ранга 1

1. Например, пусть требуется изменить первый столбец Y на x, тогда $b = [0, ..., 0, 1]^T$, w^T — первая строка R, $a = x - USw$ — x минус первый столбец Y, $z = U^T x - Sw$, $p = x - U(U^T x)$ и т. д.

2. Пусть требуется изменить всего один элемент в Y, тогда вычисления очень просты: $a = x - US, \quad w = [0, .., 0, \delta y_{ij}, 0, ..., 0]$, где $\delta y_{ij} = y'_{ij} - y_{ij}$.

$$z = U^T a = \begin{bmatrix} u_{1i}^T \delta y_{ij} \\ ... \\ u_{ri}^T \delta y_{ij} \end{bmatrix}; \quad p = \delta y_{ij} \begin{bmatrix} -u_{11}u_{i1} - ... \\ ... \\ 1 - u_{i1}^2 - ... \\ ... \\ -u_{n1}u_{i1} - ... \end{bmatrix}$$

3. Для того чтобы добавить столбец размера n к матрице Y, т.е. к SVD, добавим нулевой столбец к начальному SVD, т.е. нулевую строку к матрице R, затем изменим этот столбец соответствующим образом на c. Надо найти новое SVD с изменением ранга 1: $U'S'R'^T = [Y \quad 0] + c[0, .., 0, 1]$. В этом случае $w = 0$, а $\|q\| = 1$. Т.е. остается диагонализировать только матрицу:

$$K = \begin{bmatrix} S & z \\ 0 & \|p\| \end{bmatrix} \tag{6}$$

Это можно сделать за $O(r^2)$ операций [16].

4. Если нужно убрать столбец x, надо взять $x = 0$, который обнулится за счет вектора b, который «укажет» этот столбец. Проводим простые вычисления:

$$a = 0 - USw; \quad z = -U^T(USw) = -Sw; \quad p = -USw + USw = 0; \quad q = b - R(R^T b)$$

$$\begin{aligned} q^T q &= [b^T - (b^T R)R^T][b - R(R^T b)] = b^T b - (b^T R)R^T b - b^T R(R^T b) + (b^T R)(R^T b) = \\ &= b^T b - (b^T R)R^T b = 1 - w^T w \end{aligned} \tag{7}$$

$$\|q\| = \sqrt{1 - w^T w}$$

Тогда (5) упрощается так:

$$K = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \left(I - \begin{bmatrix} w \\ 0 \end{bmatrix} \begin{bmatrix} w \\ \sqrt{1 - w^T w} \end{bmatrix}^T \right)$$

Заметим, что чтобы удалить i-й столбец Y, надо лишь знать i-ю строку R.

2.2.4 Модификация матриц разложения

Вместо того, чтобы оперировать с огромными матрицами, вводится SVD разложение, состоящее из пяти матриц:

$$U_{n \times r} U'_{r \times r} S_{r \times r} R'^T_{r \times r} R^T_{r \times d}, \tag{8}$$

UU', RR', U, U' — ортонормированные матрицы. Изменения пространств матриц U и R так, чтобы S была диагональной матрицей, совершаются с помощью небольших матриц U' и R' , поэтому вычисления производятся быстро.

Модификация левого подпространства Пусть K и p определены, как и раньше, а C, D — матрицы размера $(r+1)\times(r+1)$, которые диагонализируют K: $CS'D = K$. Из (4) вытекает, что левое преобразование таково: $U_{new}U'_{new} = [U_{old} \quad p]U'_{old}C$. Если K имеет ранг r (т.е. *ранг не увеличился*), то C выглядит следующим образом:

$$C = \begin{bmatrix} C_{1:r,1:r} & 0 \\ 0 & 1 \end{bmatrix}, \text{а левое преобразование (8) имеет вид:} \quad U' \longleftarrow U'C_{1:r,1:r}.$$

При *увеличении ранга* преобразование иное:

$$U' = \begin{bmatrix} U' & 0 \\ 0 & 1 \end{bmatrix}, \quad U \longleftarrow [U \quad p].$$

Такие преобразования сохраняют ортогональность U, так как $U^T p = 0$ по определению.

Модификация правого подпространства Модификации правого подпространства сложнее, так как к матрице R добавляются новые строки, при этом ортогональность столбцов RR' должна сохраняться. Из (4) преобразования выглядят так:

$$R_{new}R'_{new} = \begin{bmatrix} R_{old}R'_{old} & 0 \\ 0 & 1 \end{bmatrix} D. \tag{9}$$

Чтобы сделать такое преобразование, удобно посчитать и модифицировать небольшую псевдообратную матрицу R'^{+} . Когда при добавлении строк *ранг увеличивается*, то преобразования правого подпространства таковы:

$$R'_{new} = \begin{bmatrix} R'_{old} & 0 \\ 0 & 1 \end{bmatrix} D; \quad R'^{+}_{new} = D^T \begin{bmatrix} R'^{+}_{old} & 0 \\ 0 & 1 \end{bmatrix}; \quad R_{new} = \begin{bmatrix} R_{old} & 0 \\ 0 & 1 \end{bmatrix}.$$

Это верно, так как $\begin{bmatrix} RR' & 0 \\ 0 & 1 \end{bmatrix} D = \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R' & 0 \\ 0 & 1 \end{bmatrix} D$, и D ортогональна.

Когда ранг *не увеличивается*, то последний столбец D представляет собой неиспользуемую размерность подпространства и должен быть удален. Это требует другой оптимизации. Расщепляем D на части:

$$D \in \mathbb{R}_{(r+1)\times r} = \begin{bmatrix} W_{r\times r} \\ w_{1\times r} \end{bmatrix}$$

W — линейное преобразование, которое применяется к R' , w — вектор-строка, являющаяся подпространством проекции нового вектора данных. Итак, результирующее преобразование:

$$R'_{new} = R'_{old}W; \quad R'^{+}_{new} = W^{+}R'^{+}_{old}; \quad R_{new} = \begin{bmatrix} R_{old} \\ wR'^{+}_{new} \end{bmatrix}$$

Данные преобразования и их верность проверяется прямой подстановкой в (9). W^+ удобно и быстро находить (порядок $O(r^2)$), только перемножая матрицу и вектор и вектор и вектор, используя равенство: $W^+ = W^T + \frac{w^T}{1-\|w\|^2}(wW^T)$, которое является специальным случаем для подматриц ортонормированной матрицы (для доказательства этого факта используется формула Шермана-Вудбури-Моррисона) [7].

2.2.5 Алгоритм

Алгоритм, позволяющий найти сингулярное разложение при модификациях в матрице Y:

Алгоритм «Инкрементное сингулярное разложение»

Вход: $Y_{(n\times d)}$, a — вектор размерности n, b — вектор размерности d ;

Выход: $U'S'R'^T = Y + ab^T$

1: SVD начальной матрицы $USR^T = Y$;

2: Оставить r наибольших сингулярных значений в S, у U и R оставить столбцы, которые отвечают этим наибольшим r сингулярным числам $U'S'R'^T \approx Y$;

3: $z = U^T a; \quad p = a - Uz; \quad p' = \|p\| = \sqrt{p^T p} = \sqrt{a^T p}; \quad P = \frac{p}{p'}$;

4: $w = R^T b; \quad q = b - Rw; \quad q' = \|q\| = \sqrt{q^T q} = \sqrt{b^T p}; \quad Q = \frac{q}{q'}$;

5: $K = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} zw^T & z\|q\| \\ \|p\|w^T & \|p\|\|q\| \end{bmatrix}$;

6: SVD матрицы $CS'D = K$;

7: Модификация левого подпространства в зависимости от изменения ранга;

8: Модификация правого подпространства в зависимости от изменения ранга;

9: Результат $Y + ab^T = U_{n\times r}U'_{r\times r}S_{r\times r}R'^T_{(r\times r)}R^T_{(r\times d)}$

2.2.6 ISVD при добавлении клиента

При добавлении нового клиента возможно улучшение по времени: эффективное вычисление новой строки матрицы U, отвечающей добавленному клиенту, позволяет применять формулу вычисления корреляции Пирсона для US параллельно с получением ISVD. Используя результаты [11], можно при добавлении нового клиента u найти псевдовектор для матрицы U, зная только вектор оценок объектов клиентом u (новую строку Y), а также матрицы предыдущего разложения SVD:

$$u_{new} = y_{new} R S^{-1},$$

где u_{new} — новая вектор-строка матрицы U, y_{new} — вектор рейтингов нового клиента (новая строка в матрице Y). Таким образом, имея новое U, можем найти сходство клиентов через профили, применяя IPC к US, а параллельно находить новое SVD.

В результате добавление клиента требует $O((nr + d)r)$ операций.

2.3 Общий алгоритм ISVD и IPC

Для того чтобы дать новый топ рекомендуемых объектов для клиента, будем использовать сходство клиентов, основанное на ISVD и ICF. Чтобы выделить топ объектов для прогноза клиенту, будем использовать значение средней абсолютной ошибки (MAE — Mean Absolute Error):

$$p_{ur} = \overline{y}_u + \frac{\sum\limits_{v \in U} sim(u, v)(y_{vr} - \overline{y}'_u)}{\sum\limits_{v \in U} \|sim(u, v)\|}. \tag{10}$$

Далее сортируем объекты по значениям рейтинга и получаем топ объектов.

Приведенные выше рассуждения позволяют написать

«Общий алгоритм инкрементного сингулярного разложения (ISVD) и использования формулы инкрементного вычисления корреляции Пирсона (IPC)»:

Вход: $Y \in \mathbb{R}^{n \times d}$; $a \in \mathbb{R}^n$; $b \in \mathbb{R}^d$;

Выход: p_{ur};

1: **если** модификация одной ячейки матрицы **то**

2: используем IPC для Y, сохраняем в массив $\delta y_{ij} = y'_{ij} - y_{ij}$ (разность текущего и предыдущего значения при изменении одной ячейки матрицы Y) , при этом возможно параллельно с IPC найти SVD с помощью ISVD;

3: **иначе если** новый клиент **то**

4: $u_{\text{new}} = y_{\text{new}} R S^{-1}$. Находим $\text{sim}(u, v)$ для US, при этом возможно параллельно найти SVD с помощью ISVD;

5: **иначе если** новый объект **то**

6: Применяем ISVD к матрице Y, затем находим $\text{sim}(u, v)$ для $UU'S$;

7: **end если**

8: Применяем (10).

Предложенный метод позволяет эффективно хранить информацию, а также эффективно находить рекомендации для клиентов через оценку их сходства, используя икрементное вычисление корреляции Пирсона и инкрементное сингулярное разложение. При этом объем хранимых данных в случае исходной матрицы Y размера $n \times d$ ранга r составляет $O(n + kn' + (n + d)r)$, где n' — число клиентов, с которыми данный клиент имеет хотя бы один общий объект (обычно $n' \ll n$); k — число клиентов, имеющих хотя бы с одним другим клиентом общий объект. Добавление клиента требует $O((nd + n)r)$ операций, добавление объекта — $O((nd + d)r)$ операций, модификация ячейки матрицы Y — $O(n'd')$ операций.

3 Обобщенный алгоритм обучения Хебба (Generalized Hebbian Algorithm — GHA)

Описанный ниже подход широко применялся в конкурсе Netflix. Для решения задачи используем обобщенный алгоритм обучения Хебба (Generalized Hebbian Algorithm (GHA)) [14]. Данный алгоритм используется для получения SVD, когда имеется разреженная матрица данных Y, необязательно симметрическая. Польза подхода GHA заключается в том, что алгоритм позволяет получить вектора сингулярного разложения, обращаясь только к подмножеству непустых элементов матрицы Y: $\Omega \subseteq \{1, ..., n\} \times \{1, ..., d\}$. Операция модификации векторов сингулярного

разложения использует параметр обучения и очень проста для вычисления. Операция модификации вектора не меняет его длину (принимая ее как исходный известный параметр), таким образом, каждый шаг модификации происходит с одинаковой скоростью. При этом размерность самой исходной матрицы может меняться. Данный подход несколько отличается от классического подхода получения SVD, так как он использует итерационные шаги метода градиентного спуска. Получение разложения на собственные вектора прямоугольной матрицы Y требует ее симметричности, чтобы она могла характеризоваться как матрица корреляций исходных данных. Сингулярное разложение исходной матрицы запишем в следующем виде: $Y = USR^T$. Матрицы U и R могут рассматриваться как множество ортогональных базисных векторов с отвечающими им пространствами, а сингулярные числа матрицы S, как характеристика размерности для каждой пары сингулярных векторов. При этом U и R так преобразованы, что сингулярные числа в S представляют собой невозрастающую последовательность, что позволяет оставить только r наибольших сингулярных значений, оставив соответствующие столбцы матриц U и R.

SVD исходной матрицы Y тесно связано с собственным разложением матриц YY^T и Y^TY, для которых сингулярные вектора U и R — собственные вектора, а сингулярные числа S — квадратные корни из соответствующих собственных значений.

3.1 Основная идея метода

Sanger [26] в 1989 году обобщил метод Oja and Karhunen 1985 года [23], который позволял находить лишь первый собственный вектор, на возможность нахождения первых r сингулярных векторов, используя GHA. В основе алгоритма лежит простое правило обучения Хебба:

$$u_n(t+1) = u_n(t) + \eta(u_n^T y_j)y_j,$$

где u_n — n-й столбец матрицы U;

η — параметр (темп) обучения;

y_j — j-й столбец исходной матрицы Y;

t — временной шаг.

Sanger предложил следующую формулу [27]:

$$u_{ij}(t+1) = u_{ij}(t) + \eta(a_i(t)y_j(t) - a_i(t)\sum_{k\leq i} u_{kj}(t)a_k(t)),$$

где $u_{ij}(t+1)$ — элемент собственного вектора; $y_j(t)$ — входной вектор; $a_i(t)$ — активация (скалярное произведение $u_i y_j$ i-го собственного вектора и j-го элемента входного вектора), η — темп обучения.

Текущий собственный вектор модифицируется добавлением входного вектора, умноженного на активацию, минус проекция входного вектора на все собственные вектора, включая текущий, умноженные на активацию. Такое включение текущего собственного вектора в сумму обеспечивает поддержание нормализации собственных векторов. В векторном виде, исключая a (темп обучения подразумевается, но явно не выражен) получаем формулу:

$$\delta u_i = u_i y \left(y - \sum_{j<i} (y u_j) u_j \right). \tag{11}$$

Дельта характеризует модификацию вектора. Вычитаемый элемент отвечает за удаление из модификации собственного вектора проекции на уже полученные сингулярные вектора, сохраняя ортогональность. Если бы требовалось найти только первый собственный вектор, то формула была бы совсем простой:

$$\delta u_i = u_i y(y).$$

Итак, пусть Y — вся матрица данных, а l — число объектов обучения. Тогда

$$\delta u = \frac{1}{l} u Y(Y).$$

Такое упрощение действительно при стабильном векторе u, т.е. когда вектор u сильно не меняется при модификации. Для неквадратной несимметрической матрицы Y стандартный GHA имеет вид:

$$\delta u = \frac{1}{l} u Y Y^T (Y Y^T), \tag{12}$$

$$\delta r = \frac{1}{l} r Y^T Y (Y^T Y), \tag{13}$$

где u и r — левые и правые сингулярные вектора.

Хотелось бы избежать одновременного использования всех данных в обучении. Для этого запишем следующие выражения, которые вытекают из основной формулы сингулярного разложения [14]:

$$su = rY^T = \sum_{y \in \Omega} (rb_y)a_y, \tag{14}$$

$$sr = uY = \sum_{y \in \Omega} (ua_y)b_y. \tag{15}$$

Здесь s — сингулярное значение, $y \in \Omega$ — непустые элементы в исходной матрице Y, a_y и b_y — вектора данных Y, причем векторное произведение a_y и b_y дает Y. Получаем из (12), (13), (14),(15) несколькими подстановками формул друг в друга:

$$\delta u = \frac{s}{l} r Y^T Y Y^T,$$

$$\delta r = \frac{s}{l} u Y Y^T Y,$$

$$\delta u = \frac{s^2}{l} u Y Y^T,$$

$$\delta r = \frac{s^2}{l} r Y^T Y,$$

$$\delta u = \frac{s^3}{l} r Y^T,$$

$$\delta r = \frac{s^3}{l} u Y.$$

В результате имеем:

$$\delta u = \sigma^3 (rb)a,$$

$$\delta r = \sigma^3 (ua)b.$$

Тогда преобразование (11) перепишется следующим образом:

$$\delta u_i = r_i b \left(a - \sum_{j<i} (au_j)u_j \right),$$

$$\delta r_i = u_i a \left(b - \sum_{j<i} (br_j)r_j \right).$$

Таким образом находятся левые и правые сингулярные вектора. Сингулярные значения исключены из конечной формулы, но всегда могут быть найдены делением δu или δr на s^3.

3.2 Обобщение метода на матрицы с пропусками

Simon Funk в своем алгоритме для конкурса Netflix [10] использует подход, основанный на GHA и предложенном подходе Gorell, 2006 [14], описанном выше. Вводится параметр регуляризации, и метод обобщается на задачу с сильно разреженной матрицей.

Имеем тройки (U, R, Y): U — множество клиентов, R — множество объектов, Y — оценки, данные клиентом объекту, например, рейтинги. Целью является оценить среднеквадратичную ошибку [28]:

$$RMSE = \sqrt{\mathbb{E}\{(Y - \hat{Y})^2\}},$$

где $\hat{Y}$ — оценка, полученная алгоритмом. Распределение неизвестно, имеются только тройки вида $\Omega' = (u_1, r_1, y_1), ..., (u_t, r_t, y_t)$. Считаем, что клиент мог оценить объект только один раз. Определим множество $\Omega \subseteq \{1, ..., n\} \times \{1, ..., d\}$ — индексов непустых элементов Y. Итак, имеем $Y_{n \times d}$ — разреженная матрица, в которой известные элементы представлены парами $(i, j) \in \Omega$, а пустые ячейки определяются условием, что $(i, j) \notin \Omega$.

Рейтинг клиента u объекту r обозначим через y_{ur} . Можно оценить ошибку на выбранном множестве (validation set). Тогда $\Psi' = [1, ..., n] \times [1, ..., d] \times Y$— выбранное множество. $\Psi \subseteq \{1, ..., n\} \times \{1, ..., d\}$ — индексы непустых элементов Y из множества Ψ', $\Omega \bigcap \Psi = \emptyset$ — по построению.

Тогда среднеквадратичная ошибка имеет вид:

$$\widetilde{RMSE} = \sqrt{\frac{1}{|\Psi|} \sum_{(u,r) \in \Psi} (y_{ur} - \hat{y}_{ur})^2},$$

где $\hat{y}_{ur}$ — рейтинг клиента u, данный объекту r, исходя из предложенной модели. Используется регуляризация для матричной факторизации и обучение методом градиентного спуска. Аппроксимируем Y двумя матрицами (средняя — диагональная матрица сингулярных значений, входящая в классическое SVD, не выделяется отдельно в данном разложении): $Y \approx UR, U^{n \times L}, R^{L \times d}$, где U характеризует отношения между клиентами и признаками, R — между объектами и признаками, L — число признаков для факторизации. R и U осуществляют следующие отображения:

$$R : \mathbb{R}^d \longrightarrow \mathbb{R}^L, \quad U : \mathbb{R}^L \longrightarrow \mathbb{R}^n.$$

Таким образом, число параметров, описывающих Y, может быть сокращено до $|\Omega| = NL+LD$. Заметим, что элементами U и R могут быть и действительные числа, в то время как в Y входят только целые (если речь идет, например, о количестве посещений клиентом какого-нибудь сайта). Очевидно, что заполнение огромного числа пропусков нулями не приведет к хорошему результату.

3.2.1 Получение пропущенных значений в матрице через матричную факторизацию (MF)

Пусть $u_{il} \in U_{n\times L}$ — элемент матрицы, который соответствует клиенту i и признаку l, $r_{lj} \in R_{L\times d}$— элемент матрицы, который соответствует объекту j и признаку l. Тогда можно записать:

$$\hat{y}_{ij} = \sum_{l=1}^{L} u_{il} r_{lj} = u_i r_j, \tag{16}$$

где u_i — строка U, а r_j — столбец R.

Требуется минимизировать следующий функционал:

$$\sum_{(i,j)\in\Omega} (y_{ij} - u_i r_j)^2 \to \min_{u,r}. \tag{17}$$

Запишем ошибку в полученном моделью рейтинге $\hat{y}_{ij}$:

$$\varepsilon_{ij} = y_{ij} - \hat{y}_{ij}, (i,j) \in \Omega;$$

Тогда среднеквадратичная ошибка определяется так:

$$RMSE = \sqrt{\frac{\sum\limits_{(i,j)\in\Omega} \varepsilon_{ij}^2}{|\Omega|}}$$

В (17) оптимальные матрицы U и R минимизируют сумму квадратов ошибок только по известным элементам в матрице Y.

Используется метод градиентного спуска. При этом предполагается, что каждый шаг метода уменьшает квадрат ошибки при предсказании одного рейтинга, т.е. минимизируется ε_{ij}^2.

Минимизация RMSE может рассматриваться как взвешенная аппроксимация низкого ранга исходной матрицы Y. Взвешенная аппроксимация низкого ранга представляется в виде: $SSE_W = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{d} w_{ij}\varepsilon_{ij}^2$, где w_{ij} — определенные неотрицательные веса.

В CF w_{ij} равен 1 для известных рейтингов и 0 — для неизвестных. В [22] доказано, что если ранг матрицы весов $||w_{ij}||$ равен 1, то локальный минимум будет являться глобальным минимумом. Однако в CF ранг матрицы больше 1, поэтому утверждение несправедливо, что доказано авторами [22] через контрпримеры.

Пусть имеем для метода градиентного спуска: y_{ij} — известный рейтинг на элементе обучающей выборки и $\hat{y}_{ij}$ — известная аппроксимация данного рейтинга.

Найдем градиент ошибки ε'_{ij}, где $\varepsilon'_{ij} = \frac{1}{2}\varepsilon^2_{ij}$:

$$\frac{\partial(\varepsilon'_{ij})}{\partial u_{il}} = -\varepsilon_{ij} r_{lj}, \quad \frac{\partial(\varepsilon'_{ij})}{\partial r_{lj}} = -\varepsilon_{ij} u_{il};$$

Веса вычисляем следующим образом:

$$\tilde{u}_{il} = u_{il} + \eta \varepsilon_{ij} r_{lj}, \tag{18}$$

$$\tilde{r}_{lj} = r_{lj} + \eta \varepsilon_{ij} u_{il}, \tag{19}$$

где η — темп обучения. Таким образом, минимизируется сумма квадратов ошибок, и получается лучшая аппроксимация y_{ij}.

Когда процесс обучения закончен, то неизвестные рейтинги легко находятся через (16), т.е. модель, использующая пару матриц (U, R), предлагает метод, который позволяет оценить рейтинги по всем объектам для каждого клиента.

3.2.2 Получение пропущенных значений в матрице, используя регуляризацию (RISMF— Regularization Incremental Simultaneous Matrix Factorization)

Рассмотрим усложнение предыдущего подхода. Оно актуально, так как метод MF прошлого раздела может привести к переобучению для клиентов, которые оценили меньше L объектов. Считается, что все признаковые вектора клиента u_i линейно независимы. Приведенные рассуждения ведут к идее переобучения, если η и степень изменения R малы.

Для того чтобы избежать эффекта переобучения, вводится регуляризация, которая штрафует квадрат евклидовой нормы весов. Такая регуляризация часто применяется в гребневой регрессии, нейронных сетях или методе опорных векторов (SVM).

Штрафование весов приводит к следующей оптимизационной проблеме:

$$\varepsilon'_{ij} = \frac{\varepsilon_{ij}^2 + \lambda_1 u_i u_i^T + \lambda_2 r_j^T r_j}{2}, \tag{20}$$

В этом случае функционал имеет вид:

$$\sum_{(i,j)\in\Omega} (y_{ij} - u_i r_j)^2 + \lambda_1 \sum_{i=1}^{n} \|u_i\|^2 + \lambda_2 \sum_{j=1}^{d} \|r_j\|^2 \to \min_{u,r}, \tag{21}$$

где $\lambda_1, \lambda_2 \geq 0$ — параметры регуляризации.

Заметим, что минимизация данного функционала неэквивалентна минимизации (17), если $\lambda \neq 0$. Если $\lambda = 0$, то возвращаемся к предыдущему методу ISMF (Incremental Simultaneous Matrix Factorization).

Аналогично MF найдем градиент ошибки ε'_{ij}:

$$\frac{\partial(\varepsilon'_{ij})}{\partial u_{il}} = -\varepsilon_{ij} r_{lj} + \lambda_1 u_{il}, \tag{22}$$

$$\frac{\partial(\varepsilon'_{ij})}{\partial r_{lj}} = -\varepsilon_{ij} u_{il} + \lambda_2 r_{lj}; \tag{23}$$

Веса вычисляем следующим образом:

$$\tilde{u}_{il} = u_{il} + \eta(\varepsilon_{ij} r_{lj} - \lambda_1 u_{il}), \tag{24}$$

$$\tilde{r}_{lj} = r_{lj} + \eta(\varepsilon_{ij} u_{il} - \lambda_2 r_{lj}), \tag{25}$$

3.2.3 Алгоритм получения пропущенных значений в матрице, использующий регуляризацию (RISMF)

Запишем общий алгоритм получения пропущенных значений в исходной матрице (рейтингов), используя рассуждения, описанные в предыдущих двух пунктах.

Останов обучения в алгоритме будет осуществляться, когда RMSE не будет умешьшаться на последних двух эпохах, так как происходит минимизация для выбранного множества (validation set). Начальная инициализация матриц U и R может быть произвольной. Если матрицы U и R инициализируются константой, то их ранг равен 1, что эквивалентно тому, что $L = 1$. Модификация весов не изменяет ранг матриц. Можно выбрать небольшие произвольные значения элементов матриц из интервала, например, $[-0.1, 0.1]$. Начальная инициализация не влияет на результат

работы метода [10]. В отличие от подхода Simon Funk [10], признаки модифицируются вместе [28], а матрицы инициализируются произвольными значениями. Simon Funk инициализировал элементы матриц константными значениями 0.1, а модификация вектора признакового описания объекта r_j происходила после модификаци вектора признакового описания клиента u_i и зависела от полученного на данном шаге значения вектора признакового описания клиента.

Алгоритм 1 «Предсказание неизвестных рейтингов в исходной матрице (метод RIMF)»

Вход: Ω — обучающая выборка, η — темп обучения, λ — параметр регуляризации;

Выход: $(\tilde{U}, \tilde{R})$ — матрицы признаков клиентов и объектов;

1: Разбиваем произвольным образом Ω на 2 множества Ω_1 и Ω_2 (validation set);
2: Начальная инициализация матриц U и R небольшими произвольными значениями
3: **цикл** // *пока не достигли заключительного состояния (Одна эпоха)*
4: **для всех** $(i, j) \in \Omega_1$
5: Находим ε'_{ij} из (20)
6: Находим градиент ε'_{ij} из (22) и (23)
7: **для всех** l признаков
8: обновить u_i — i-ю строку U и r_j — j-й столбец R из (29) и (30)
9: **end для**
10: **end для**
11: Найти RMSE на Ω_2
12: **если** RMSE лучше, чем на предыдущих эпохах **то**
13: $\tilde{U} = U, \tilde{R} = R$
14: **end если**
15: Заключительное состояние: RMSE не уменьшается на последних двух эпохах
16: **end цикл**

3.2.4 Неотрицательная и положительня MF

В описанном алгоритме RISMF элементы матриц U и R могут принимать как положительные, так и отрицательные значения. В [20] описано NNMF (Non-negative Matrix Factorization), и генерируются неотрицательные значения признаков. В [18] генерируются матрицы только с положительными значениями признаков. В некоторых случаях,например, просто для лучшего восприятия того, что означает какой-то признак, неотрицательное представление кажется более понятным. В этом случае (29) и (30) можно заменить так:

$$\tilde{u}_{il} = \max\{0, u_{il} + \eta(\varepsilon_{ij} r_{lj} - \lambda_1 u_{il})\}, \tag{26}$$

$$\tilde{r}_{lj} = max\{0, r_{lj} + \eta(\varepsilon_{ij} u_{il} - \lambda_2 r_{lj})\}, \tag{27}$$

3.2.5 Алгоритм, позволяющий добавлять строки и элементы в исходную матрицу

Алгоритм RISMF имеет серьезный недостаток: при проходении итераций в цикле по клиентам меняются признаковые вектора объектов. Если изменения значительны, то модификации признаковых векторов клиентов, полученные в начале эпохи, могут стать непригодными в конце той же эпохи. Предлагается пересчитать признаковые вектора клиентов после обучения.

Алгоритм 2 «Получение новых признаковых векторов клиентов («переобучение»)

Вход: Ω — обучающая выборка, η — темп обучения, λ — параметр регуляризации;

Выход: $(\tilde{U}, \tilde{R})$ — матрицы признаков клиентов и объектов;

1: Разбиваем произвольным образом Ω на 2 множества настройки Ω_1 и Ω_2 (validation set);

2: Первый шаг обучения: Вызов Алгоритма 1, сохранение результата в (U_1, R_1);

3: $R = R_1$, элементы U задаются произвольными значениями;

4: Второй шаг обучения: Вызов Алгоритма 1 со следующими требованиями:

5: Обойти шаг инициализации весов, берем полученные выше U и R

6: Матрица R не меняется, нет вычислений (30), вычисляем только (29)

7: Сохранить оптимальное число эпох в *EpochOpt*

8: Вернуть результат Алгоритма 1, вызванного выше

Заметим, что **Алгоритм 2** позволяет добавлять новых клиентов и рейтинги для уже имеющихся клиентов без переобучения всей модели, что очень важно для современных рекомендующих систем. Не применяется вся процедура обучения: для повторного поиска признаковых векторов клиентов используется оптимальное число эпох *EpochOpt*. Только один раз второй шаг обучения происходит не за *EpochOpt*. При добавлении нового рейтинга матрица R становится «устаревшей», и надо повторить первый шаг обучения.

Заметим, что данный алгоритм не позволяет добавлять объекты. Но объекты хорошо позволяет добавлять метод, описанный в разделе **2.2**.

3.3 Порядковые данные в исходной матрице

Опишем метод, учитывающий то, что значения в матрице исходных данных порядковые. Минимизируется следующий функционал, где Y дано, а U, R, β требуется найти:

$$\sum_{(i,j)\in\Omega} (\beta_{y_{ij}} - u_i r_j)^2 \to \min_{u,r,\beta}, \tag{28}$$

где $\beta_{y_{ij}}$ — параметр, аппроксимирующий порядковые значения в Y; u_i — i-я строка в U; r_j — j-й столбец в R; $[y_{ij} = m]$ равно 1, если $y_{ij} = m$, и равно 0, если $y_{ij} \neq m$.

Введем новую переменную $\overline{y}_m$, которая упростит записи следующих ниже формул:

$$\overline{y}_m = \sum_{(i,j)\in\Omega} [y_{ij} = m].$$

$\beta_{y_{ij}}$ находится аналитически:

$$\beta_{y_{ij}} = \frac{1}{\overline{y}_m} \sum_{(i,j)\in\Omega} [y_{ij} = m] u_i r_j.$$

Ошибка по каждому элементу Y имеет вид:

$$\varepsilon_{ij} = \beta_{y_{ij}} - u_i r_j.$$

В (28) оптимальные матрицы U и R минимизируют сумму квадратов ошибок для непустых элементов Y.

Аналогично (21) введем двойственные переменные, дополнительные условия нормировки и запишем оптимизационную задачу:

$$\sum_{(i,j)\in\Omega}(\beta_{y_{ij}} - u_i r_j)^2 + \lambda_1 \sum_{i=1}^{n}(\|u_i\|^2 - 1) + \lambda_2 \sum_{j=1}^{d}(\|r_j\|^2 - 1) \to \min_{u,r,\beta},$$

где $\lambda_1, \lambda_2 \geq 0$ — параметры регуляризации.

Используем метод градиентного спуска для нахождения новых весов:

$$\varepsilon_{ij} = \beta_{y_{ij}} - u_i r_j,$$

$$\tilde{u}_{il} = (1 - \eta\lambda_1)u_{il} + \eta \sum_j r_{lj}\varepsilon_{ij}, \tag{29}$$

$$\tilde{r}_{lj} = (1 - \eta\lambda_2)r_{lj} + \eta \sum_i u_{il}\varepsilon_{ij}, \tag{30}$$

$$\beta_{y_{ij}} = \frac{1}{\overline{y}_m} \sum_{(i,j)\in\Omega} [y_{ij} = m]u_i r_j,$$

где η — темп обучения.

Ниже в экспериментах используются L_1 и L_2 нормы. Нормировка по строкам U и по столбцам R позволяет избежать достижения минимума в нуле. Элементы U и R, а следовательно, и $\beta_{y_{ij}}$ находятся в промежутке от 0 до 1. Таким образом, получив $\beta_{y_{ij}}$, задача заключается в том, чтобы ввести функцию, которая отобразит элементы от 0 до 1 в искомые порядковые значения Y. При этом $\beta_{y_{ij}}$ монотонно возрастают, что позволяет ввести ступенчатую функцию. В экспериментах используется следующая функция для каждого элемента $\widehat{y}_{ij} = u_i r_j$ (при условии, что значения в Y от 1 до 5):

$$y_{ij} = \begin{cases} 1, & \text{если } \widehat{y}_{ij} \leq \frac{\beta_1+\beta_2}{2}, \\ 2, & \text{если } \frac{\beta_1+\beta_2}{2} < \widehat{y}_{ij} \leq \frac{\beta_2+\beta_3}{2}, \\ \dots & \\ 5, & \text{если } \frac{\beta_4+\beta_5}{2} \leq \widehat{y}_{ij}. \end{cases} \tag{31}$$

После применения функции находим RMSE:

$$RMSE = \sqrt{\frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \varepsilon_{ij}^2}.$$

Алгоритм поиска U, R представлен ниже.

Вход: $Y = \{y_{ij}\} \in \mathbb{N}^{n \times d}, (i,j) \in \Omega;\ \lambda_1;\ \lambda_2;\ \eta;$

Выход: U, R;

1: Инициализация: $\beta_{y_{ij}} = m;$ U, R инициализируются небольшими значениями из интервала $[0, 0.7]$;

2: $\overline{y}_m = \sum\limits_{(i,j)\in\Omega} [y_{ij} = m];$

3: **для** $q = 1, \ldots, EpochNumber$

4: **для всех** $\{i,j\} \in \Omega$

5: $\varepsilon_{ij} = \beta_{y_{ij}} - u_i r_j,$

6: **для всех** $l = 1, \ldots, L$

7: $\tilde{u}_{il} = (1 - \eta\lambda_1)u_{il} + \eta \sum\limits_j r_{lj}\varepsilon_{ij},$

8: $\tilde{r}_{lj} = (1 - \eta\lambda_2)r_{lj} + \eta \sum\limits_i u_{il}\varepsilon_{ij},$

9: **end для**

10: $\beta_{y_{ij}} = \frac{1}{\overline{y}_m} \sum\limits_{(i,j)\in\Omega} [y_{ij} = m] u_i r_j,$

11: **end для**

12: Найти RMSE. Добавить новые (i,j).

13: **end для**

14: Пока не достигли заключительного состояния: $RMSE$ не уменьшается на последних двух эпохах.

Результатом работы алгоритма является заполненная матрица Y.

4 Описание эксперимента

На основе приведенного выше алгоритма был выполнен эксперимент, в котором были объединены оба типа инкрементности (добавление клиентов и объектов, а также модификация ячеек исходной матрицы).

При добавлении нового клиента GHA не требует полной перенастройки алгоритма, что очень важно при работе с большими данными. Эксперименты на модельных и реальных данных (MovieLens — 943 клиента, 1682 объекта, 100000 заполненных

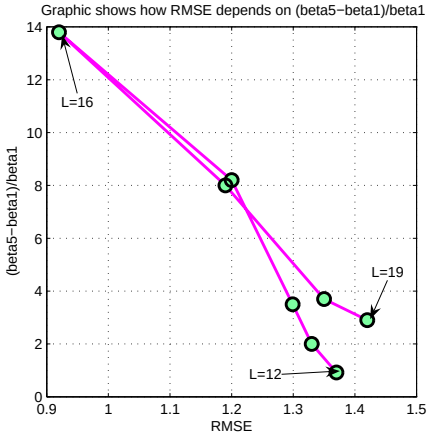


Рис. 1: Зависимость RMSE от отношения $\frac{\beta_5-\beta_1}{\beta_1}$.

ячеек Y) показали, что объем данных и временной фактор влияют на скорость сходимости алгоритма.

При увеличении числа клиентов (строк Y) растет скорость сходимости алгоритма (от 1000 итераций при 600 клиентах до 40 — на 940-м клиенте) (рис.5 см. Приложения).

При увеличении объема данных ошибка уменьшается. Наилучший результат алгоритма, учитывающего информацию о типе исходных данных, достигнут на 16 признаках, RMSE $= 0.92$, $\frac{\beta_5-\beta_1}{\beta_1} = 13.8$, $\lambda_1 = \lambda_2 = 0.18$, $\eta = 0.25$ (см. рис. 1). Средняя ошибка, полученная при работе алгоритма, лучше (RMSE $= 0.92$ при 5 итерациях), чем результат Takacs [28], не учитывающего того условия, что данные порядковые: RMSE $= 0.93$ при 7 итерациях (см. рис. 2).

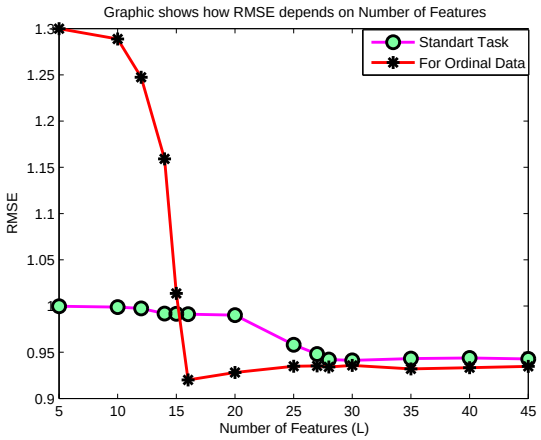


Рис. 2: Зависимость RMSE от числа факторов (L)

Результат работы алгоритма при различных модификациях (добавления значений в ранее пустые ячейки, добавление строк, столбцов Y) показан на рис. (3). На графике видно, что результат работы алгоритма сильно зависит от:

- Типа модификаций;
- Эпохи, в которую происходят изменения.

При выбранном варианте останова (RMSE не уменьшается на последних двух эпохах) при добавлении отдельных элементов ошибка резко не возрастает, так как, добавляя (i, j), возможно что в выборке i-й клиент и/или j-й объект уже есть, а это значит, что u_i и/или r_j уже модифицировались на предыдущих эпохах алгоритма (см. шаг 7, 8 **Алгоритма поиска** U**,** R).

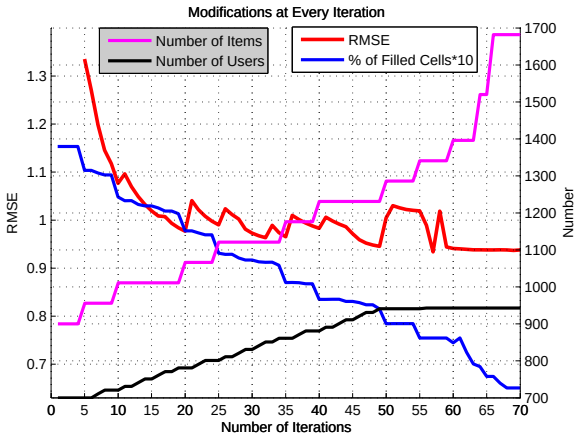


Рис. 3: Поведение RMSE при модификациях на каждой итерации

При добавлении строк и столбцов на графике видно возрастание RMSE. Добавление объектов, больше влияет на увеличение среднеквадратичной ошибки, чем добавление клиентов. Даже если модификации происходят на каждой итерации (добавляются известные значения в Y), алгоритм сходится. Несмотря на то, что добавляются новые клиенты и объекты, а следовательно, и новые значения в ячейки Y, RMSE также сильно меняется при любых модификациях даже на больших итерациях. Это можно объяснить, посмотрев на линию, отвечающую за долю заполненных ячеек на каждой итерации. Кривая убывает: клиенты и объекты добавляются, но клиенты мало оценили объекты, и объекты тоже мало оценены клиентами.

График 1 отражает важность разницы между самым маленьким и самым большим значением β. Самая большая разница между β_1 и β_5 достигается именно при наименьшей RMSE, что закономерно, так как это позволяет точнее получить элементы Y, используя полученные значения $\beta_{y_{ij}}$ и функцию преобразования (31).

Измерения времени выполнения также показывают, что предложенный метод может быть полезен в современных динамических приложениях CF.

5 Заключение

В результате проделанной работы были изучены разные инкрементные методы в CF. Каждый из них имеет свои достоинства и недостатки. Были рассмотрены формула инкрементного вычисления корреляции Пирсона (IPC), инкрементное сингулярное разложение (ISVD), обобщенный алгоритм обучения Хебба (GHA).

Результаты, выносимые на защиту

- Предложен метод, который объединяет разные типы инкрементности, позволяет добавлять новые элементы матрицы, а также строки и столбцы.
- Введен функционал, учитывающий тип входных данных. Проведенные эксперименты показали, что прогноз становится точнее относительно прогноза метода, не учитывающего особенность исходной информации.
- Получены оценки сложности для сходства профилей клиентов и объектов через корреляцию Пирсона и инкрементное сингулярное разложение.

Список литературы

[1] *Гантмахер Ф. Р.* Теория матриц. — М.: Физматлит, 2004. — 560 с.

[2] *Abernethy J., Bach F., Evgeniou T.* Low-rank matrix factorization with attributes: Tech. Rep. cs/0611124: arXiv, 2006.

[3] *Abernethy J., Bach F., Evgeniou T.* A new approach to collaborative filtering: Operator estimation with spectral regularization // *Journal of Machine Learning Research.* — 2008. — Vol. 10. — Pp. 803–826.

[4] *Aronszajn N.* Theory of reproducing kernels // *Trans. Am. Math. Soc.* — 1950. — Vol. 68. — Pp. 337 – 404.

[5] *Bach F. R., Shen X.* Consistency of trace norm minimization // *Journal of Machine Learning Research.* — 2007. — Vol. 9. — Pp. 1019–1048.

[6] *Brand M.* Fast online svd revisions for lightweight recommender systems // In SIAM International Conference on Data Mining. — 2003. — P. 37–46.

[7] *Brand M.* Fast low-rank modifications of the thin singular value decomposition // *Linear Algebra and Its Applications.* — 2006. — Vol. 415, no. 1. — Pp. 20–30.

[8] *Breese J., Heckerman D., Kadie C.* Empirical Analysis of Predictive Algorithms for Collaborative Filtering // Proc. of Conference on Uncertainty in Artificial Intelligence. — Madison, USA: Morgan Kaufmann Publisher, 1998. — July. — Pp. 43 – 52.

[9] *Fazel M., Hindi H., Boyd S. P.* A rank minimization heuristic with application to minimum order system approximation // In Proceedings of the 2001 American Control Conference. — 2001. — Pp. 4734–4739.

[10] *Funk S.* Incremental svd method. http://sifter.org/ simon/journal/20061211.html.

[11] *Furnas G. W., Deerwester S., Dumais S. T.* Information retrieval using a singular value decomposition model of latent semantic structure // Proceedings of the 11th annual international ACM SIGIR conference on Research and development in

information retrieval. — SIGIR '88. — New York, NY, USA: ACM, 1988. — Pp. 465–480.

[12] *Gaussier E., Goutte C.* Relation between plsa and nmf and implications // Research and Development in Information Retrieval. — 2005. — Pp. 601–602.

[13] *Golub G. H., Van Loan C. F.* Matrix computations (3rd ed.). — Baltimore, MD, USA: Johns Hopkins University Press, 1996. — 48 pp.

[14] *Gorrell G.* Generalized hebbian algorithm for incremental singular value decomposition in natural language processing // Proceedings of Interspeech. — 2006.

[15] *Grcar M.* User profiling: Collaborative filtering // *Knowledge Creation Diffusion Utilization.* — 2004. — Vol. 10. — P. 803 – 826.

[16] *Gu M., Eisenstat S. C.* A stable and fast algorithm for updating the singular value decomposition: Tech. rep.: 1994.

[17] *Heckerman D., Chickering D. M.* A comparison of scientific and engineering criteria for bayesian model selection: Tech. rep.: Statistics and Computing, 1996.

[18] *Hofmann T.* Latent semantic models for collaborative filtering // *ACM Trans. Inf. Syst.* — 2004. — Vol. 22, no. 1. — Pp. 89–115.

[19] *Kleeman A., Hendersen N., Denuit S.* Matrix factorization for collaborative prediction. — 2006.

[20] *Lee D. D., Seung H. S.* Learning the parts of objects by nonnegative matrix factorization // *Nature.* — 1999. — Vol. 401. — Pp. 788–791.

[21] *Leksin V. A.* Symmetrization and overfi tting in probabilistic latent semantic analysis // *Pattern Recognition and Image Analysis.* — 2009. — Vol. 19, no. 4. — Pp. 565–574.

[22] *Nati N. S., Jaakkola T.* Weighted low-rank approximations // In 20th International Conference on Machine Learning. — AAAI Press, 2003. — Pp. 720–727.

[23] *Oja E., Karhunen J.* On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix // *Journal of Mathematical Analysis and Applications.* — 1985. — Vol. 106. — Pp. 69–84.

[24] *Papagelis M., Rousidis I., Plexousakis D.* Incremental collaborative filtering for highly-scalable recommendation algorithms // International Syposium on Methodologies for Intelligent Systems. — 2005. — Pp. 553–561.

[25] *Rennie J. D. M., Srebro N.* Fast maximum margin matrix factorization for collaborative prediction // In Proceedings of the 22nd International Conference on Machine Learning (ICML. — ACM, 2005. — Pp. 713–719.

[26] *Sanger T. D.* Optimal unsupervised learning in a single-layer linear feedforward neural network // *Neural Networks.* — 1989. — Vol. 2, no. 6. — Pp. 459–473.

[27] *Sanger T. D.* Two iterative algorithms for computing the singular value decomposition from input/output samples // NIPS / Ed. by J. D. Cowan, G. Tesauro, J. Alspector. — Morgan Kaufmann, 1993. — Pp. 144–151.

[28] *Takacs G., Pilaszy I., Nemeth B.* Scalable collaborative filtering approaches for large recommender systems // *Journal of Machine Learning Research (Special Topic on Mining and Learning with Graphs and Relations).* — 2009. — Vol. 10. — Pp. 623–656.

6 Приложения

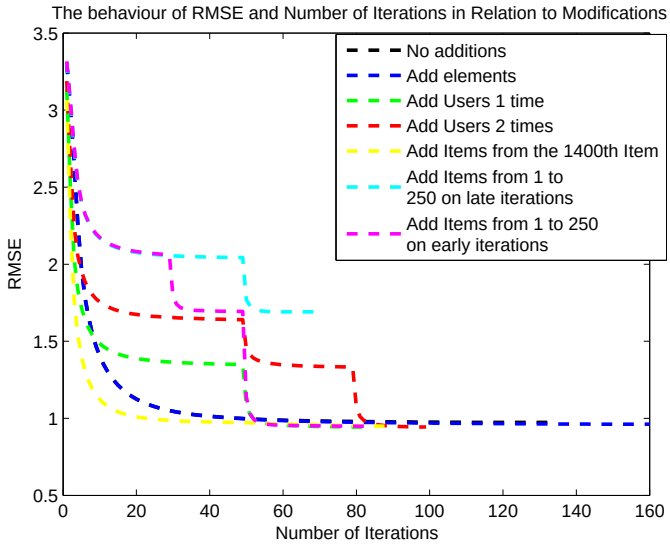


Рис. 4: Зависимость RMSE и числа итераций от модификаций

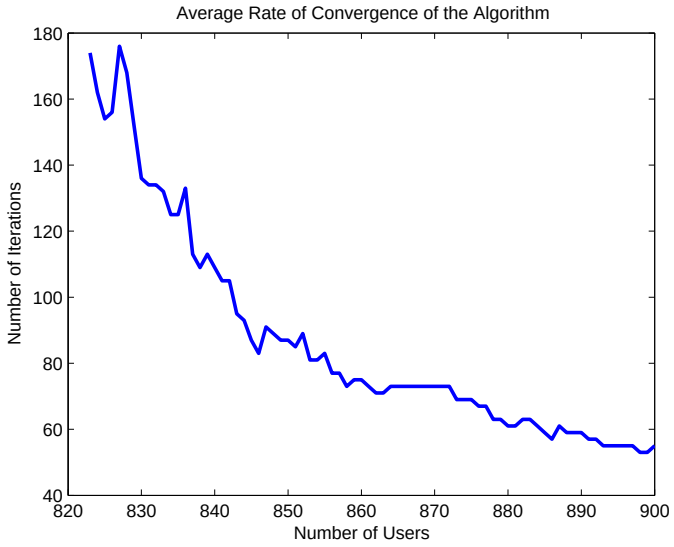


Рис. 5: Увеличение числа клиентов

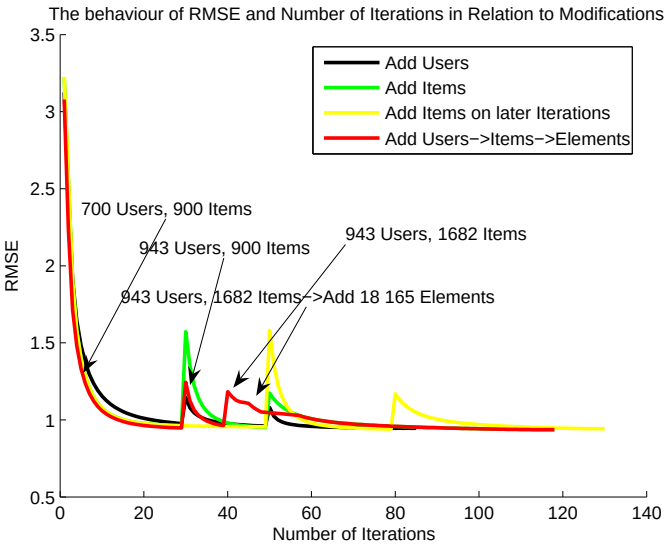


Рис. 6: Зависимость RMSE от модификаций