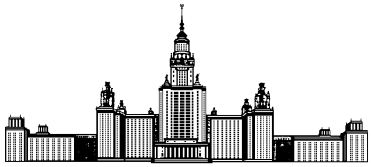
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

ДИПЛОМНАЯ РАБОТА СТУДЕНТА 517 ГРУППЫ

«Низкоранговые приближения в моделировании данных»

Выполнил:
студент 5 курса 517 группы
Ромов Петр Алексеевич

Научный руководитель:
к.ф-м.н.,
Дьяконов Александр Геннадьевич

Москва, 2014

Содержание

1 Введение

1.1 Задачи моделирования предпочтения

Рассмотрим задачу *коллаборативной фильтрации*. Пусть имеется конечное множество пользователей U и конечное множество объектов I. Пусть также имеется набор пар $\mathcal{R} = \{(u,i)\} \subset U \times I$, для которых известна вещественная *оценка предпочтения* r_{ui}. Требуется по обучающему набору $(\mathcal{R}, \{r_{ui}, (u,i) \in \mathcal{R}\})$ построить регрессор $\hat{r}(u,i)$, предсказывающий оценку предпочтения для произвольной пары $(u,i) \in U \times I$.

В случае, если каждый пользователь $u \in U$ и каждый объект $i \in I$ поддаются содержательному описанию признаковыми векторами: $\boldsymbol{x}(u) : U \to \mathbb{R}^{d_u}$, $\boldsymbol{x}(i) : I \to \mathbb{R}^{d_i}$, то задачу коллаборативной фильтрации можно свести к стандартной задаче обучения по прецедентам[34]: требуется обучить модель, которая по вектору признаков $\boldsymbol{x}(u,i) = [\boldsymbol{x}(u), \boldsymbol{x}(i)]$ предсказывает значение регрессионной переменной r_{ui}. Такой подход в рекомендательных системах называется *основанным на описаниях* (content-based).

Низкоранговые приближения. Существует альтернативный подход к решению задачи коллаборативной фильтрации: оценки предпочтения можно записать в виде матрицы $\boldsymbol{R} \in \mathbb{R}^{|U| \times |I|}$, в которой строки соответствуют пользователям, а столбцы — объектам. В матрице $\boldsymbol{R}$ известны некоторые элементы, остальные — пропущены. Построить правило $\hat{r}(u,i)$ означает научиться восстанавливать пропущенные значения матрицы $\boldsymbol{R}$. Для заполнения пропусков в матрице $\boldsymbol{R}$ используют предположение о ее низком ранге и строят приближение $\boldsymbol{R} \approx \boldsymbol{X}\boldsymbol{Y}^T$, где $\boldsymbol{X} \in \mathbb{R}^{|U| \times K}$, $\boldsymbol{Y} \in \mathbb{R}^{|I| \times K}$, K — предполагаемый ранг матрицы $\boldsymbol{R}$. Ограничение ранга в некотором смысле является регуляризацией, придающей низкоранговому приближению обобщающую способность. В таком подходе предсказанная оценка предпочтения является скалярным произведением соответствующих строчек матриц $\boldsymbol{X}$ и $\boldsymbol{Y}$:

$$\hat{r}(u,i) = \boldsymbol{x}_u^T \boldsymbol{y}_i, \quad \boldsymbol{x}_u \in \mathbb{R}^K, \, \boldsymbol{y}_i \in \mathbb{R}^K. \tag{1}$$

Учет признаковых описаний вместе с низкоранговым приближением. Подход с низкоранговым приближением использует исключительно информацию о парах (u,i) и соответствующих им значениях предпочтений. Известно [20, 17], что в рекомендательных системах большая часть информации содержится именно в обезличенных тройках $\{(u,i,r_{ui})\}$. Методы низкоранговых приближений дают значимо лучшую точность предсказания $\hat{r}(u,i)$ по сравнению с content-based методами. Тем не менее, учет известной информации о пользователе и объекте в дополнение к низкоранговому приближению дает лучшую точность и позволяет справиться с некоторыми проблемами рекомендательных систем (таких как *холодный старт*) [7]. Опубликовано большое число моделей для коллаборативной фильтрации, основанных на базовой идее низкорангового приближения (1), но с модификациями для учета внешней информации, например [16, 6, 31].

Модель *факторизационных машин* [23] призвана обобщить различные модификации модели (1). Путем задания соответствующего вектора признаков $\boldsymbol{x}(u,i) \in \mathbb{R}^d$ для пары пользователь-объект можно приводить предсказательную модель $\hat{y}(\boldsymbol{x})$ к различным факторизационным моделям. Отличие факторизационной машины от классических алгоритмов обучения по прецедентам — моделирование взаимодействий между компонентами вектора признаков, что приводит к тому, что индикаторные признаки во входном векторе модели $\boldsymbol{x}$ обрабатываются как в модели низкорангового приближения (1).

Работа с данными без явного отклика. В рекомендательных системах зачастую имеет место проблема отсутствия негативного предпочтения [14]. Это обусловлено тем, что веб-сервисам гораздо проще и надежнее накапливать акты о положительно впечатлении, произведенном на пользователя u объектом рекомендаций i (например, «репосты» в Twitter [13]), нежели узнавать у пользователей их предпочтений в какой-либо шкале (например, «число звезд» в Netflix [3]). Такие данные принято называть *данными без явного отклика* (implicit feedback datasets). Без привлечения дополнительных предположений, обучение модели предпочтения $\hat{r}(u,i)$ на таких данных бессмысленно — лучшая модель будет для любой пары (u,i) предсказывать положительную реакцию пользователя на объект. Очевидно, такая модель имеет плохую обобщающую способность.

Для работы с данными без явного отклика используют предположение о том, что пропуски в матрице оценок предпочтения $\boldsymbol{R}$ чаще происходят из-за негативного предпочтения, нежели положительного. Алгоритм IALS [14] настраивает приближение (1) на полную матрицу $\boldsymbol{R}$, в которой пропущенные значения заменены на 0, оптимизируя взвешенную сумму квадратов отклонений [29]. Известные пары $(u,i) \in \mathcal{R}$ имеют вес $c_{ui} \gg 1$, а пропущенные $(u,i) \in \mathcal{R}$ — $c_{ui} = 1$, что означает низкую уверенность в нулевой оценке предпочтения за пропущенную пару, и высокую степень уверенности за наблюдаемую оценку предпочтения.

Алгоритм iTALS [11] строит каноническое тензорное разложение тензора оценок, т.е. обучает предсказательную модель вида:

$$\hat{r}(u,i,c_1,c_2,\ldots,c_M) \approx r_{u,i,c_1,\ldots,c_M} \tag{2}$$

где $c_1 \in C_1, \ldots, c_M \in C_M$, а $C_1, \ldots, C_M$ — некоторые конечные множества, как правило они используются для описания контекста рекомендации [12].

Таким образом, алгоритм позволяет добавлять в модель дополнительные дискретные переменные и работать в условиях данных без явного отклика. Однако, модель (2), обучаемая алгоритмом iTALS не может учитывать произвольные числовые вектора признаков, как это позволяют делать факторизационные машины.

1.2 Цель работы

Целью работы является разработка предсказательной модели вида

$$\hat{r}(i_1, i_2, \ldots, i_M) \approx r_{i_1,\ldots,i_M}, \quad i_1 \in I_1, \ldots, i_M \in I_M, \tag{3}$$

где $I_1, \ldots, I_M$ — конечные множества объектов некоторой природы, у которых можно брать признаковые описания $\boldsymbol{x}_1(i_1) \in \mathbb{R}^{D_1}, \ldots, \boldsymbol{x}_M(i_M) \in \mathbb{R}^{D_M}$ в качестве дополнительной информации. Требуется обучать модель (3) на известных наборах $\{(r_{i_1 \ldots i_M}, i_1, \ldots, i_M)\}$, при этом учитывать потенциальную проблему данных без явного отклика.

Модель факторизационных машин не укладывается в данные требования, т.к. не может учитывать проблему данных без явного отклика, но зато может обрабатывать любые признаковые описания. Модель, обучаемая алгоритмом iTALS, не может обрабатывать признаковые описания, зато умеет работать с данными без явного отклика.

Множества $I_1, \ldots, I_M$ будем называть *факторами*, а элементы этих множеств — *значениями факторов*. Такое наименование позволит нам смотреть на алгоритмы без их привязки к предметной области рекомендательных систем. Далее будет приведено несколько приложений моделей подобного рода, не связанных напрямую со стандартной задачей рекомендательной системы.

1.3 Примеры приложений

Полуавтоматическая разметка контента. Веб-сервис Delicious[1] предоставляет пользователю возможность создавать собственную коллекцию закладок (ссылок на интернет-ресурсы), а также размечать закладки семантическими тегами.

Сервис накапливает информацию о взаимодействиях вида (пользователь, закладка) и (пользователь, закладка, тег). Помимо обозначенной выше задачи рекомендации закладок, имеет место задача предсказания релевантности тега в контексте закладки, созданной определенным пользователем.

Оценка знаний по небольшому числу вопросов. В конкурсе по анализу данных «What Do You Know?»[2] на платформе Kaggle, участникам предлагалось предсказать корректность ответа студента на вопрос во время прохождения тестирования.

В данной задаче имеются три фактора, каждый описывается набором признаков (в том числе категориальных):

- пользователь (известен только идентификатор);
- вопрос (номер вопроса, тип вопроса, идентификатор теста, набор тегов);
- контекст (дата и время ответа, скорость отправки ответа).

[1] `https://delicious.com`

[2] `http://www.kaggle.com/c/WhatDoYouKnow`

Обучающая выборка содержит четверки (пользователь, вопрос, контекст, корректность ответа).

Используя модель, предсказывающую корректность ответов на вопросы по известному набору троек (пользователь, вопрос, контекст), пользователю (студенту) можно задать небольшое число тестовых вопросов, посте получения ответов на которые, предсказать корректность ответов на все вопросы из теста (или из нескольких различных тестов), таким образом оценив уровень знаний студента и выявив пробелы в обучении.

Следует отметить, что лучшее решение конкурса использовало факторизационные машины.

1.4 Обозначения

Скалярные величины, обозначающие размеры или количество будем обозначать большими латинскими литерами: M, N.

Вектора и матрицы будем обозначать латинскими буквами жирным шрифтом, вектора — строчными литерами ($\boldsymbol{x} \in \mathbb{R}^N$), матрицы — прописными ($\boldsymbol{A} \in \mathbb{R}^{N \times M}$). Все вектора — столбцы (матрицы $N \times 1$), вектора-строки обозначаются путем явного транспонирования вектора-столбца: $\boldsymbol{x}^T \in \mathbb{R}^{1 \times N}$. Вектор, состоящий из всех единиц будем обозначать жирной единицей — $\mathbf{1}$, его длина должна быть понятна из контекста. Пример: $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{y}$ — квадратичная форма.

Обозначения для работы с многомерными матрицами (тензорами) взяты из наиболее представительной работы в области низкоранговых тензорных разложений [15].

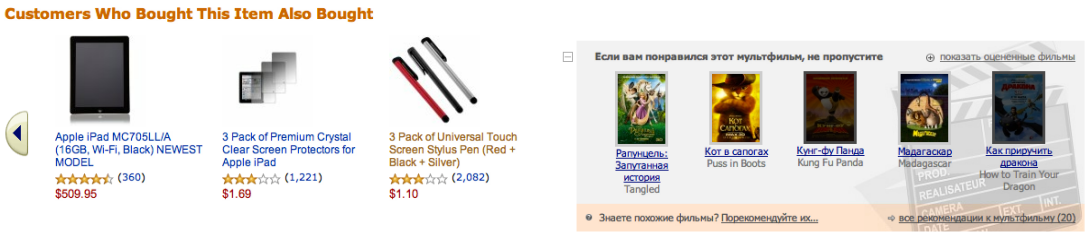
Покомпонентное произведение векторов и матриц будем обозначать: $\boldsymbol{x} \circ \boldsymbol{y} \circ \boldsymbol{z}$, $\boldsymbol{A} \circ \boldsymbol{B}$.

Число ненулевых элементов вектора или матрицы будем обозначать $N_z(\boldsymbol{x}), N_z(\boldsymbol{X})$.

1.5 Структура текста

В разделе 2 приводится описание предметной области рекомендательных систем и моделирования предпочтений пользователей интернет-сервисов, раскрывается роль низкоранговых приближений в рекомендательных системах. В разделе 3 приводится описание нескольких моделей для коллаборативной фильтрации, основанных на идее низкорангового приближения, краткое описание модели факторизационных машин и алгоритма тензорного разложения iTALS.

В разделе 4 вводится понятие тензорной машины, приводится алгоритм ALS обучения модели канонического разложения для тензорной машины. После чего, в разделе 5 приводятся примеры использования тензорной машины в реальных задачах.



(a) (b)

Рис. 1: Пример персональных блоков рекомендаций: интернет-магазин Amazon (a) предлагает пользователю обратить внимание на товары, которые могут быть релевантны с учетом недавно купленных; веб-сервис КиноПоиск (b) рекомендует пользователю посмотреть фильмы, подобранные исходя из его оценок просмотренных ранее фильмов.

2 Моделирование предпочтения в рекомендательных системах

2.1 Рекомендательные системы

Рекомендательные системы — одно из наиболее популярных приложений интеллектуального анализа данных и машинного обучения в сфере интернет-бизнеса. Рекомендательная система анализирует поведение пользователей интернет-сервиса, после чего может давать оценку предпочтения пользователем того или иного объекта рекомендаций. Объектами рекомендаций могут служить товары в интернет-магазине, набор разделов веб-сайта, медиа-контент, другие пользователи веб-сервиса. На основе предсказываемых рекомендательной системой предпочтений, поведение веб-сервиса к конкретному пользователю может меняться — такую функциональность принято называть *персонализацией*.

Наиболее очевидный пользователю пример персонализации — персональные блоки рекомендаций (см. рис. 1). На основании только что посмотренных пользователем объектов, либо используя данные предоставленные пользователем о его предпочтениях, веб-сервис ранжирует объекты по убыванию релевантности для конкретного пользователя.

Рекомендательные системы помогают пользователям ориентироваться в большом числе контента, размещенного на сайте. В некоторых случаях это необходимая функциональность. К примеру, на сервисе Яндекс.Музыка[3] по грубым оценкам размещено более 50 лет непрерывного аудио контента. Пользователь не сможет прослушать всю музыку, поэтому поиск нравящейся и при этом новой музыки пользователи осуществляют, используя советы от друзей, общественное радио-вещание. В задачи рекомендательной системы в данном случае входит построение персонального аудио-потока, интересного (релевантного) пользователю, на основании его взаимодействия с проигрывателем и поведения на веб-сервисе в целом.

[3] `http://music.yandex.ru`

Результатом работы рекомендательной системы с точки зрения бизнеса является увеличение целевых показателей эффективности. Такими показателями могут служить число проданных товаров, объем продаж, доход от продаж, длительность пользовательской сессии на сайте, степень вовлеченности пользователя, лояльность пользователей к сервису.

Рекомендательные системы не обязательно призваны рекомендовать некоторые объекты пользователям. Для увеличения эффективности промо-акций, интернет-магазины прибегают к помощи рекомендательных систем с целью выявления наиболее заинтересованных пользователей в одном из товаров. На основании покупок пользователей, их откликов на промо-письма, поведению на витринах интернет-магазина, рекомендательная система предсказывает степень заинтересованности каждого пользователя в конкретном товаре и промо-акция проводится только для наиболее заинтересованных.

В последние годы появилось множество компаний, занимающихся внедрением и развитием рекомендательных систем, наиболее известные: Gravity R&D[4], RichRelevance[5], Netflix[6], Spotify[7]. Ежегодно проводится конференция «The ACM Conference Series on Recommender System»[8] (сокращенно RecSys), на которой встречаются представители бизнесов, нуждающихся в технологии рекомендательной системы, а также исследователи в области интеллектуального анализа данных.

2.2 Построение рекомендательных систем

Можно выделить три основных подхода к построению рекомендательных систем (следуя [1]):

1. на основании признаковых описаний (content-based);
2. коллаборативная фильтрация (collaborative filtering);
3. гибридный подход.

Content-based. Подход на основании признаковых описаний предполагает, что про пользователей и про рекомендуемые объекты известно достаточно много информации. Например, все пользователи заполняют анкету, в которой указывают свою социально-демографическую информацию, интересы, и т.д. Про товары из интернет-магазина может быть известно их описание, предназначение, ценовая категория, бренд, и другие характеристики. По истории взаимодействия пользователей и объектов на сервисе можно построить обучающую выборку и свести предсказание предпочтения к хорошо изученной задаче обучения по прецедентам [34].

[4] `http://www.gravityrd.com`
[5] `http://www.richrelevance.com`
[6] `https://www.netflix.com`
[7] `http://www.spotify.com`
[8] `http://recsys.acm.org`

На практике, использование такого похода сильно ограничено, т.к. сбор описательной информации о пользователях и объектах очень дорогостоящая процедура, которую зачастую невозможно организовать не в ущерб качеству использования сервиса, что делает рекомендательную систему неоправданно дорогой.

Коллаборативная фильтрация. Коллаборативной фильтрацией называется предсказание степени предпочтения в условиях, когда рекомендательная система не обладает какой-либо описательной информацией о пользователях и объектах (либо не использует), строит прогноз исключительно на основании взаимодействия пользователей с объктами.

Огромным толчком в исследовании математических моделей коллаборативной фильтрации послужил конкурс Netflix Prize[3]. Компания Netflix занимается интернет-прокатом кинофильмов. Целью конкурса являлось улучшение качества предсказываемой оценки пользователя некоторому фильму. Набор данных конкурса содержал четверки (дата-время, пользователь, фильм, оценка). Оценка измерялась от 1 до 5.

Гибридный подход. Несмотря на то, что алгоритмы коллаборативной фильтрации на практике показывают высокие показатели эффективности, учет дополнительной информации может следать показатели еще выше. Одним из недостатков коллаборативной фильтрации по сравнению с методами, основанными на признаковом описании, является *проблема холодного старта* [28].

Гибридный подход использует композиции алгоритмов основанных на признаковых описаниях и результатов коллаборативной фильтрации.

2.3 Задача коллаборативной фильтрации

Как было сказано выше, коллаборативная фильтрация — подход к предсказанию предпочтения с использованием исключительно информации о связях пользователей и объектов рекомендации. Сформулируем задачу более строго.

Пусть U — множество пользователей (`users`), I — множество объектов (`items`), информация об известных предпочтениях представлена в виде набора троек:

$$\mathcal{D} = \{(u, i, r_{ui})\}_{(u,i)\in\mathcal{R}},$$

где $r_{ui} \in \mathbb{R}$ — вещественная степень предпочтения объекта $i \in I$ пользователем $u \in U$; $\mathcal{R} \subseteq U \times I$ — множество пар (пользователь, объект), про которые известна степень предпочтения.

Для дальнейшего удобства, введем также обозначение: $\mathcal{R}(u) = \{i : (u, i) \in \mathcal{R}\}$ — множество объектов, смежных с пользователем u, аналогично: $\mathcal{R}(i) = \{u : (u, i) \in \mathcal{R}\}$.

По известной информации $\mathcal{D}$ требуется уметь строить предсказание предпочтения $\hat{r}_{ui} \approx r_{ui}$ для новых пар $(u, i) \notin \mathcal{R}$.

Будем называть *матрицей оценок* матрицу $\boldsymbol{R} \in (\mathbb{R} \cup \emptyset)^{|U|\times|I|}$, строки которой соответствуют пользователям, столбцы — объектам, а элементы принимают значение r_{ui},

если $(u, i) \in \mathcal{R}$, иначе — пропуск $\emptyset$. На задачу коллаборативной фильтрации можно смотреть как на задачу заполнения пропущенных значений в матрице.

Помимо предсказания значения предпочтения, на практике могут быть интересны следующие задачи:

- построение списка рекомендаций из объектов, на которые не известна степень предпочтения (новые для пользователя):
  $\text{Recommended}_K(u) = \text{Top}_K \max_i \hat{r}_{ui} \mapsto \{(i_1, r_{ui_1}), (i_2, r_{ui_2}), \ldots, (i_K, r_{ui_K})\}$,
- определение степень похожести объектов и построение списков наиболее похожих:
  $\text{Similar}_K(i) \mapsto \{(i_1, s_{ii_1}), (i_2, s_{ii_2}), \ldots, (i_K, s_{ii_K})\}$, где s_{ij} — степень похожести между двумя объектами,
- обоснование списка рекомендаций: некоторое человеко-понятное пояснение, почему пользователю u был порекомендован объект i.

Подходы к решению задачи коллаборативной фильтрации условно можно разделить на две большие группы (следуя [1]):

1. основанные на эвристиках (memory/heuristic-based),
2. основанные на построении модели предпочтения (model-based).

Memory-based. К первой группе методов (memory-based) относятся алгоритмы, выражающие пропущенные значения непосредственно через элементы матрицы оценок.

Ярким примером memory-based алгоритма коллаборативной фильтрации является взвешивание предпочтения по пользователям (user-based) и по объектам (item-based).

$$\hat{r}_{ui} = \bar{r}_u + \frac{1}{\sum_{u' \in \mathcal{R}(i)} |\text{sim}(u, u')|} \sum_{u' \in \mathcal{R}(i)} \text{sim}(u, u')(r_{u',i} - \bar{r}_{u'}), \qquad \text{User-based}$$

$$\hat{r}_{ui} = \bar{r}_i + \frac{1}{\sum_{i' \in \mathcal{R}(u)} |\text{sim}(i, i')|} \sum_{i' \in \mathcal{R}(u)} \text{sim}(i, i')(r_{u,i'} - \bar{r}_{i'}), \qquad \text{Item-based}$$

где $\bar{r}_u = \frac{1}{|\mathcal{R}(u)|} \sum_{i \in \mathcal{R}(u)} r_{ui}$, $\bar{r}_i = \frac{1}{|\mathcal{R}(i)|} \sum_{u \in \mathcal{R}(i)} r_{ui}$ — средние значения предпочтений по пользователям и объектам, $\text{sim}(u, u')$, $\text{sim}(i, i')$ — заранее заданные метрики схожести пользователей и объектов.

Мера схожести $\text{sim}(u, u')$ (и аналогичная для объектов) вычисляется по матрице оценок $\boldsymbol{R}$, либо с использованием дополнительной информации о пользователях (объектах). Мера схожести является важным параметром алгоритма. Наиболее общеупотребимые простые метрики схожести — корреляция Пирсона и косинусное расстояние

соответствующих строк (столбцов) матрицы оценок:

$$\mathrm{sim}(u,u') = \frac{\sum_{i\in\mathcal{R}(u)\cap\mathcal{R}(u')}(r_{u,i}-\bar{r}_u)(r_{u',i}-\bar{r}_{u'})}{\sqrt{\sum_{i\in\mathcal{R}(u)\cap\mathcal{R}(u')}(r_{u,i}-\bar{r}_u)^2\sum_{i\in\mathcal{R}(u)\cap\mathcal{R}(u')}(r_{u',i}-\bar{r}_{u'})^2}} \qquad \text{Pearson}$$

$$\mathrm{sim}(u,u') = \frac{\sum_{i\in\mathcal{R}(u)\cap\mathcal{R}(u')} r_{u,i}r_{u',i}}{\sqrt{\sum_{i\in\mathcal{R}(u)} r_{u,i}^2}\sqrt{\sum_{i\in\mathcal{R}(u')} r_{u',i}^2}} \qquad \text{Cosine}$$

Наиболее используемые и реализованные в виде библиотек с открытым исходным кодом memory-based алгоритмы описаны в статьях [27, 18, 32, 9].

Подобные алгоритмы хороши для единоразового вычисления рекомендаций на распределенном кластере, хорошо ложатся на вычислительную архитектуру MapReduce, однако плохо подходят для оперативного обновления рекомендаций. Настройка меры схожести для задачи из конкретной предметной области является скорее искусством, нежели налаженной технологией.

Одна из наиболее серьезных проблем memory-based методов — неадекватность предсказаний в условиях сильной разреженности матрицы оценок $\boldsymbol{R}$ (в смысле пропущенных значений), приводящая к невозможности подсчета метрик похожести (в случае, если множество $\mathcal{R}(u)\cap\mathcal{R}(u')$ — пусто). Сильная разреженность матрицы оценок может быть следствием проблемы холодного старта, однако в некоторых областях она сильно разрежена всегда и не может стать достаточно плотной (например, в случае интернет-магазинов, пользователь оставляет информацию о предпочтениях в среднем 2–5 объектам).

Подобные алгоритмы не будут рассматриваться далее.

Model-based. Алгоритмы из второй группы (model-based) стремятся выбрать функцию $\hat{r}(u,i;\theta)$ из некоторого семейства моделей, параметризованного $\theta\in\Theta$. Выбор происходит, например, путем минимизации регуляризованного эмпирического риска:

$$\mathcal{L}(\theta) = \sum_{(u,i)\in\mathcal{R}} l(\hat{r}(u,i;\theta), r_{ui}) + \lambda\Omega(\theta) \to \min_{\theta\in\Theta}, \tag{4}$$

где $l(\hat{r},r)$ — функция потерь регрессии, λ — сила регуляризации, $\Omega(\theta)$ — регуляризатор на множестве параметров Θ.

2.4 Модель предпочтения SVD

Одна из наиболее известных моделей предпочтения — SVD [16], определяется следующим образом:

$$\hat{r}(u,i;\theta) = \boldsymbol{p}_u^T\boldsymbol{q}_i, \quad \theta = (\{\boldsymbol{p}_u\}_{u\in U}, \{\boldsymbol{q}_i\}_{i\in I}), \tag{5}$$

где $\boldsymbol{p}_u \in \mathbb{R}^R$ — *вектор скрытых (латентных) предпочтений* пользователя, $\boldsymbol{q}_i \in \mathbb{R}^R$ — аналогичный вектор для объекта, R — размерность латентных векторов, будем называть ее *рангом* модели SVD. Модель SVD обучается путем оптимизации квадратичных потерь $l(\hat{r}, r) = (\hat{r} - r)^2$ с квадратичной регуляризацией: $\Omega(\theta) = \sum_{u \in U} \|\boldsymbol{p}_u\|^2 + \sum_{i \in I} \|\boldsymbol{q}_i\|^2$.

Название модели SVD не следует путать с *сингулярным разложением* (Singular Value Decomposition). Такое название модели вошло в оборот в сообществе исследователей рекомендательных систем с момента публикации Саймоном Фанком статьи в блоге[9], описывающей его решение Netflix Prize [19]. Идея подобной модели была быстро подхвачена и распространена исследователями в различных модификациях (см. обзор в статье [16]), т.к. имела достаточно хорошую обобщающую способность (как следствие — результат в конкурсе) по сравнению с memory-based методами, популярными на тот момент. Как будет показано далее, данная модель имеет мало общего с сингулярным разложением.

В своей статье, Саймон Фанк оптимизировал модель SVD методом стохастического градиентного спуска. Несколько позже был предложен более эффективный и допускающий распараллеливание по пользователям и объектам алгоритм ALS (Alternating Least Squares) [33]. Его идея заключается в том, что при фиксированных латентных векторах объектов $\{\boldsymbol{q}_i\}_{i \in I}$, оптимизация (4) только по латентным векторам пользователей $\{\boldsymbol{p}_u\}_{u \in U}$ разбивается на независимые регуляризованные задачи наименьших квадратов по пользователям:

$$\mathcal{L}_u(\boldsymbol{p}_u) = \sum_{i \in \mathcal{R}(u)} (\boldsymbol{p}_u^T \boldsymbol{q}_i - r_{ui})^2 + \lambda \|\boldsymbol{p}_u\|^2 \to \min_{\boldsymbol{p}_u}, \quad \forall u \in U. \tag{6}$$

Для каждого пользователя $u \in U$, оптимальный латентный вектор $\boldsymbol{p}_u^\star$ вычисляется как решение СЛАУ с матрицей размера $R \times R$:

$$\boldsymbol{p}_u^\star = \left(\sum_{i \in \mathcal{R}(u)} \boldsymbol{q}_i \boldsymbol{q}_i^T + \lambda \boldsymbol{I} \right)^{-1} \left(\sum_{i \in \mathcal{R}(u)} r_{ui} \boldsymbol{q}_i \right). \tag{7}$$

При фиксированных латентных векторах пользователей, оптимальные латентные вектора объектов вычисляются аналогичным образом. Алгоритм ALS заключается в циклическом пересчете латентных векторов пользователей и объектов.

2.5 Обратная связь в рекомендательных системах

Обратной связью (feedback) пользователя на некоторый объект в рекомендательных системах принято называть событие, по которому можно судить о предпочтении пользователя к объекту. Вот несколько примеров обратной связи от пользователя:

- проставление оценки объекту по бальной шкале (количество звезд);

[9] `http://sifter.org/~simon/journal/20061211.html`

- нажатие на кнопку «нравится» (лайк) / «не нравится» (дизлайк);
- посещение страницы с описанием объекта, переход по ссылке на объект (клик);
- посещение страницы с описанием объекта более одного раза (заинтересованность);
- добавление в корзину / покупка объекта в случае, если это товар.

Именно по обратной связи пользователя на различные объекты, рекомендательная система формирует матрицу оценок предпочтений $\boldsymbol{R}$, к которой затем применяются алгоритмы коллаборативной фильтрации. Преобразование обратной связи в числовое значение предпочтения — непростая и очень важная задача в настройке рекомендательных систем. Как правило, при выборе схемы оценки предпочтения оптимизируется метрика, непосредственно связанная с ключевыми показателями эффективности (KPI) бизнеса. Техники подбора схемы оценки предпочтения выходят за рамки данной работы.

По видам обратной связи, задачи моделирования предпочтения в рекомендательных системах принято разделять на два вида:

1. с явной обратной связью (explicit feedback);
2. с неявной обратной связью (implicit feedback).

Так, например, рекомендации по оценкам из пятибальной шкалы — пример задачи с явной обратной связью. Рекомендательные системы, руководствующиеся актами покупок, посещением страниц — примеры задач с неявной обратной связью.

В случае неявной обратной связи имеется неопределенность в том, положительно или отрицательно влияют конкретный акт обратной связи на степень предпочтения. Покупка товара в интернет-магазине может означать достижение ползователем своей потребительской цели (положительное предпочтение), но в то же время покупатель мог после получения товара в нем разочароваться и правильно было бы засчитать негативную степень предпочтения. Очевидно, что посещения страниц пользователями веб-сервиса могут происходить при совершенно разной степени заинтересованности пользователя в контенте.

Стоит отметить достаточно типичную ситуацию, когда рекомендательной системе подаются на вход исключительно положительные примеры взаимодействия пользователей и объектов. Например, веб-сервис Twitter[10] не имеет функциональности, позволяющей пользователю выразить свое низкое предпочтение контенту, а присутствует только лишь способ «поощрить» тот или иной контент, распространив его своим подписчикам посредством функции «репост». Подобная обратная связь пользователя очень надежно (по сравнению с остальными) указывает на положительную степень предпочтения. Надежность «репостов» в сервисе Twitter подкреплена ответственностью пользователей перед своими подписчиками.

В работе [30] выдвигается три типа предположений о матрице оценок $\boldsymbol{R}$, используемых в алгоритмах коллаборативной фильтрации:

[10] `https://twitter.com`

1. все пропуски в матрице $\boldsymbol{R}$ произошли случайно (MAR, missing-at-random);

2. все пропуски являются следствием негативного предпочтения (AMAN, all-missing-are-irrelevant);

3. пропуски в матрице $\boldsymbol{R}$ произошли не случайно (MNAR, missing-not-at-random).

Под «не случайностью» выше имеется в виду вероятностное предположение о смещенности распределения предпочтения пропущенных оценок в негативную сторону.

Алгоритмы коллаборативной фильтрации, использующие гипотезы AMAN и MNAR исторически принято называть «применимыми к данным с неявной обратной связью» (for implicit feedback datasets).

Идея учета гипотез AMAN/MNAR породила группу алгоритмов коллаборативной фильтрации, настраивающих латентные факторы пользователей и объектов не только на известные элементы матрицы $\boldsymbol{R}$, но и на пропущенные [14, 21, 24, 11]. Пропущенные элементы, по предположению, имеют негативную оценку предпочтения, но при этом влияют на параметры модели с меньшим весом, нежели известные.

2.6 Учет контекста в рекомендательных системах

Часто, для построения релевантного списка рекомендаций, рекомендательной системе нужна информация о контексте, в котором веб-сервис запрашивает рекомендацию для пользователя. Вот некоторые примеры такой контекстной информации:

- географическое положение путешественника (например, в случае рекомендации достопримечательностей [8]);

- состояние корзины посетителя интернет-магазина, наличие заказа в оформлении;

- сезон, время, погода и другие факторы.

В простых случаях (как, например с географическим положением), достаточно фильтровать объекты рекомендаций некоторым правилом. В более сложных случаях (как, например, погода) трудно построить однозначно разумные правила фильтрации объектов.

В алгоритмах коллаборативной фильтрации с использованием контекстной информации требуется научиться предсказывать оценку предпочтения $\hat{r}_{uic} \approx r_{uic}$ для пользователя $u \in U$, объекта $i \in I$ и контекста $c \in C$, где C — заранее заданный набор контекстов.

Одним из основных подходов к моделированию предпочтения с учетом контекстной информации являются факторизационные модели. В работе [25] предлагается использовать факторизационные машины (будут описаны далее), контекстную информацию представлять в виде дополнительных признаков для факторизационной машины. В работе [11] предлагается кодировать контекст в дополнительные координаты тензора

оценок, осуществлять низкоранговое каноническое разложение тензора оценок (предложенный в работе алгоритм iTALS далее будет рассмотрен подробно), а в работе [10] — попарное тензорное разложение (Pairwise-Interaction Tensor Factorization). Более подробный обзор алгоритмов коллаборативной фильтрации с учетом контекста можно найти в работе [2].

3 Факторизационные модели

3.1 Низкоранговые приближения

Рассмотрим матрицу $\boldsymbol{R} \in \mathbb{R}^{n \times m}$. По определению ранга матрицы, если $\operatorname{rank}(\boldsymbol{R}) \leq k$, то найдутся $\boldsymbol{X} \in \mathbb{R}^{n \times k}, \boldsymbol{Y} \in \mathbb{R}^{m \times k}$, такие что

$$\boldsymbol{R} = \boldsymbol{X}\boldsymbol{Y}^T.$$

В случае, если $\operatorname{rank}(\boldsymbol{R}) > k$, то *приближением* ранга k матрицы $\boldsymbol{R}$ будем называть

$$\boldsymbol{X}\boldsymbol{Y}^T = \tilde{\boldsymbol{R}} \approx \boldsymbol{R},$$

где $\boldsymbol{X} \in \mathbb{R}^{n \times k}, \boldsymbol{Y} \in \mathbb{R}^{m \times k}$ — матрицы-факторы приближения. Приближение ранга k позволяет представить в памяти компьютера матрицу размера $n \times m$, используя объем памяти $O(k(n+m))$, вместо $O(nm)$. Рассмотрим два способа получения приближения ранга k.

Сингулярное разложение. Оптимальное приближение ранга k с точки зрения суммы квадратичных отклонений всех элементов матрицы $\boldsymbol{R}$ находится из решения оптимизационной задачи

$$\sum_{i=1}^{n}\sum_{j=1}^{m}(r_{ij} - \boldsymbol{x}_i^T\boldsymbol{y}_j)^2 = \|\boldsymbol{R} - \boldsymbol{X}\boldsymbol{Y}^T\|_{\text{fro}}^2 \to \min_{\boldsymbol{X},\boldsymbol{Y}} \tag{8}$$

при помощи сингулярного разложения (SVD). Известно [29], что все локальные оптимумы оптимизационной задачи (8) являются глобальными, матрицы $\boldsymbol{X}, \boldsymbol{Y}$ определены с точностью до невырожденного линейного преобразования $\boldsymbol{A} \in \mathbb{R}^{k \times k}$, т.к.:

$$(\boldsymbol{X}\boldsymbol{A})(\boldsymbol{Y}\boldsymbol{A}^{-T})^T = \boldsymbol{X}(\boldsymbol{A}\boldsymbol{A}^{-1})\boldsymbol{Y}^T = \boldsymbol{X}\boldsymbol{Y}^T.$$

Взвешенное низкоранговое приближение. В рассматриваемых нами задачах матрица $\boldsymbol{R}$ не известна полностью и содержит пропуски. В таком случае имеет смысл от задачи (8) перейти к приближению путем оптимизации взвешенных квадратичных отклонений:

$$\sum_{i,j} w_{ij}(r_{ij} - \boldsymbol{x}_i^T\boldsymbol{y}_j)^2 = \|\boldsymbol{W} \odot (\boldsymbol{R} - \boldsymbol{X}\boldsymbol{Y}^T)\| \to \min_{\boldsymbol{X},\boldsymbol{Y}}. \tag{9}$$

Здесь $\boldsymbol{W} \in \mathbb{R}^{n \times m}$ — матрица весов приближения. Рассмотренный в разделе 2.4 алгоритм обучения модели SVD оптимизирует взвешенное приближение с весами

$$w_{ui} = \begin{cases} 0, & (u,i) \notin \mathcal{R} \\ 1, & (u,i) \in \mathcal{R}. \end{cases}$$

Из исследования взвешенных низкоранговых приближений в работе [29] известно, что в случае $\operatorname{rank}(\boldsymbol{W}) = 1$ задача (9) может быть найдена методом сингулярного разложения и, также как в невзвешенном приближении, все локальные оптимумы являются глобальными. Однако, если $\operatorname{rank}(\boldsymbol{W}) > 1$, то задача (9) может иметь сколько угодно много локальных оптимумов, не являющихся глобальными.

Именно случай $\operatorname{rank}(\boldsymbol{W}) > 1$ представляет практический интерес. С учетом сложности глобальной оптимизации функционала (9), ограничиваются поиском локального оптимума, иногда запуская из различных начальных приближений матриц $\boldsymbol{X}, \boldsymbol{Y}$.

3.2 Задача заполнения матрицы

В работе [4] доказано следующее теоретическое утверждение:

Теорема 1. *Пусть имеется матрица $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$, у которой известны значения элементов с индексами из множества Ω, известен ранг матрицы $\operatorname{rank}\boldsymbol{M} = r$. Если индексы известных элементов Ω взяты наугад и выполняется неравенство*

$$m \geq C n^{5/4} r \log n$$

где $m = |\Omega|$ — число известных элементов матрицы, $n = \max\{n_1, n_2\}$, то с вероятностью

$$p = 1 - cn^{-3} \log n$$

все пропущенные значения матрицы $\boldsymbol{M}$ можно восстановить в точности и единственным образом.

Т.е. в предположении достаточно низкого ранга, матрица большого размера может быть восстановлена полностью из случайно выбранных элементов. Данное утверждение можно рассматривать как мотивацию использования низкоранговых приближений для моделирования матриц, содержащих пропуски.

3.3 Факторизационные машины

Модель факторизационной машины предложена в работе [22] как универсальная модель, позволяющая иммитировать различные модели для коллаборативной фильтрации путем задания различных признаковых описаний известным оценкам предпочтения. Модель была успешно применена Стефаном Рендлом для решения задачи «What Do You Know?» на платформе Kaggle (автор решения победил в конкурсе).

Модель факторизационной машины порядка 2 определяется следующим образом:

$$\hat{y}(\boldsymbol{x};\Theta) = w_0 + \boldsymbol{w}^T\boldsymbol{x} + \sum_{p=1}^{P}\sum_{p'=p+1}^{P} x_p x_{p'} \boldsymbol{v}_p^T \boldsymbol{v}_{p'}, \tag{10}$$

набор параметров модели $\Theta = \{w_0, \boldsymbol{w}, \boldsymbol{V}\}$, $\boldsymbol{w} \in \mathbb{R}^P$, $\boldsymbol{V} \in \mathbb{R}^{R\times P}$.

Следующее выражение эквивалентно (10), но позволяет считать рекомендации за $O(RN_z(\boldsymbol{x}))$ операций:

$$\hat{y}(\boldsymbol{x};\Theta) = w_0 + \boldsymbol{w}^T\boldsymbol{x} + \frac{1}{2}\sum_{r=1}^{R}\left\{\left(\sum_{p=1}^{P} x_p v_{pr}\right)^2 - \sum_{p=1}^{P}(v_{pr}x_p)^2\right\}. \tag{11}$$

Заметим, что модель (10) обладает свойством *мультилинейности* по всем параметрам, т.е. ее можно представить в виде:

$$\hat{y}(\boldsymbol{x};\Theta) = g_\theta(\boldsymbol{x};\Theta_{\setminus\theta}) + \theta \cdot h_\theta(\boldsymbol{x};\Theta_{\setminus\theta}). \tag{12}$$

Обучение факторизационных машин. Пусть $S = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ — обучающая выборка. Обучение факторизационной машины происходит путем оптимизации регуляризованных квадратичных потерь:

$$\mathcal{L}(\Theta) = \sum_{(\boldsymbol{x},y)\in S}(\hat{y}(\boldsymbol{x}_n;\Theta) - y_n)^2 + \sum_{\theta\in\Theta}\lambda_\theta\theta^2 \to \min_{\Theta} \tag{13}$$

В работе [23] предлагается три алгоритма обучения факторизационных машин:

1. стохастический градиентный спуск (SGD);
2. стохастический градиентный спуск с адаптивной настройкой гиперпараметров (SGDA);
3. покоординатный спуск (ALS).

В работе [23] предлагается также алгоритм вывода методом марковских цепей монте-карло (MCMC) в вероятностной модели, основанной на (10). Известно, что алгоритм MCMC не требует настройки гиперпараметров и как правило дает результат значительно лучший, по сравнению с оценками, полученными из точечно-настроенной модели. Однако, данный подход сильно ограничен в своей применимости, т.к. для построения одного предсказания требует прохода по всей обучающей выборке, что неприемлемо в рекомендательных системах.

Рассмотрим алгоритм ALS. Его идея заключается в покоординатной оптимизации (13):

$$\theta^\star = \arg\min_\theta \left(\sum_{(\boldsymbol{x},y)\in S} (\hat{y}(\boldsymbol{x};\Theta) - y)^2 + \sum_\theta \lambda_\theta \theta^2 \right) = \tag{14}$$

$$= \arg\min_\theta \left(\sum_{(\boldsymbol{x},y)\in S} (g_\theta(\boldsymbol{x};\Theta_{\setminus\theta}) + \theta \cdot h_\theta(\boldsymbol{x};\Theta_{\setminus\theta}))^2 + \sum_\theta \lambda_\theta \theta^2 \right) = \tag{15}$$

$$= \frac{\theta \sum_{n=1}^N h_\theta^2(\boldsymbol{x}_n) + \sum_{n=1}^N h_\theta(\boldsymbol{x}_n) e_n}{\lambda_\theta + \sum_{n=1}^N h_\theta(\boldsymbol{x}_n)^2}, \tag{16}$$

где $e_n = y_n - \hat{y}(\boldsymbol{x}_n;\Theta)$ — ошибка с предыдущего шага.

3.4 Алгоритм факторизации тензора iTALS

Предложенный в работе [11] алгоритм строит каноническое тензорное разложение ранга K, т.е. представляет каждый элемент тензора $\boldsymbol{T}$ размера $S_1 \times \cdots \times S_D$ в виде:

$$\hat{\boldsymbol{T}}_{i_1,\ldots,i_D} = \mathbf{1}^T \boldsymbol{M}^{(1)}_{:,i_1} \circ \boldsymbol{M}^{(2)}_{:,i_2} \circ \cdots \circ \boldsymbol{M}^{(D)}_{:,i_D}, \tag{17}$$

где $\{\boldsymbol{M}^{(d)}\}_{d=1}^D$ — факторы канонического разложения, $\boldsymbol{M}^{(d)} \in \mathbb{R}^{K\times S_d}$.

Подробнее про каноническое тензорное разложение и его свойства см. в работе [15].

Матрицы $\{\boldsymbol{M}^{(d)}\}_{d=1}^D$ находятся путем минимизации взвешенного квадратичного отклонения элементов тензора:

$$\mathcal{L}(\boldsymbol{M}^{(1)},\ldots,\boldsymbol{M}^{(D)}) = \sum_{i_1=1}^{S_1} \cdots \sum_{i_D=1}^{S_D} W_{i_1,\ldots,i_D} (\hat{T}_{i_1,\ldots,i_D} - T_{i_1,\ldots,i_D})^2, \tag{18}$$

где $\boldsymbol{W}$ — тензор весов, имеющего размер тензора $\boldsymbol{T}$.

Вносится предположение том, что тензор $\boldsymbol{W} - \mathbf{1}$ имеет малое число ненулевых элементов $N_z(\boldsymbol{W})$, а $T_{i_1,\ldots,i_D} = 0$, если $W_{i_1,\ldots,i_D} = 0$. Такое предположение позволяет естественным образом внести в разложение предположение гипотезы MNAR (см. раздел 2.5).

Вычислительная сложность одной итерации алгоритма iTALS — $O(K^3 \sum_{d=1}^D S_d + K^2 N_z(\boldsymbol{W}))$.

4 Тензорные машины

Классическая задача обучения по прецедентам [34] заключается в восстановлении скрытой зависимости $f : \mathcal{X} \to \mathcal{Y}$ по набору известных пар $(x_n, y_n), x_n \in \mathcal{X}, y_n = f(x_n)$. В большинстве задач рассматривают зависимость вещественной функции $f(\boldsymbol{x}) \in \mathbb{R}$ от вектора-признака $\boldsymbol{x} \in \mathbb{R}^D$.

Факторизационные машины (см. раздел 3.3), позволяют работать с моделями для коллаборативной фильтрации, оставаясь при этом в рамках классической формулировки задачи обучения по прецедентам. Для работы с факторными данными, в признаковое описание прецедентов $\boldsymbol{x}$ добавляются индикаторные переменные. Так, например, модель для задачи коллаборативной фильтрации — предсказание оценки предпочтения для пользователя u к объекту i, в рамках факторизационных машин получается путем кодирования пары (u, i) в вещественный вектор признаков, состоящий из двух наборов индикаторов:

$$\boldsymbol{x}(u,i) = (\underbrace{[u=1],[u=2],\ldots,[u=|U|]}_{\text{признаки пользователя}},\underbrace{[i=1],[i=2],\ldots,[i=|I|]}_{\text{признаки объекта}}). \tag{19}$$

Алгоритмы коллаборативной фильтрации, учитывающие гипотезу MNAR (например, iTALS, описанный в разделе 3.4), настраивают предсказательную модель не на известные пары $(u, i) \in \mathcal{R}$ из обучающей выборки, а на все возможные пары $(u, i) \in U \times I$. Классическая формулировка задачи обучения по прецедентам не позволяет работать с подобными моделями, поскольку из нее не ясно, как вектор признаков разбивается на факторы, и какие значения каждый фактор может принимать.

Введению понятия *прогнозирующего тензора* есть две предпосылки:

- дать свободу выражения всей известной информации о значениях факторов (объектов, пользователей и пр.) в виде вещественных вектор-признаков (как это делается в факторизационных машинах);
- дать возможность формулировать алгоритмы, учитывающие гипотезу MNAR и настраивающие модель на все комбинации значений факторов (как это делается в алгоритме iTALS).

В данном разделе будут формализованы основные понятия, касающиеся прогнозирующих тензоров, приведены примеры конкретных моделей. Далее будет предложена модель тензорной машины и алгоритм ее обучения, который как и iTALS будет иметь возможность учитывать гипотезу MNAR.

4.1 Понятие прогнозирующего тензора

Факторы. *Фактором* назовем пространство $\mathcal{X}$ признаковых описаний объектов некоторого типа. Предполагается, что признаковое описание может быть представлено вещественнозначным вектором размера $\dim \mathcal{X}$.

Пример факторов: пользователь, предмет рекомендаций, контекст; фактор имеет размер $|\mathcal{X}|$ (число значений) $\times \dim \mathcal{X}$ (размерность признаков), может быть задан вещественной матрицей $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{X}| \times \dim \mathcal{X}}$ соответствующего размера.

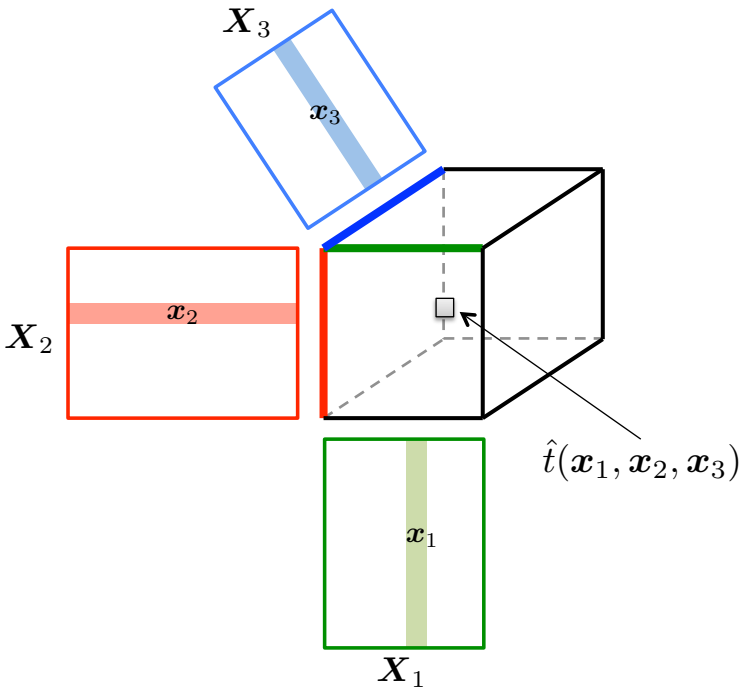


Рис. 2: Иллюстрация прогнозирующего тензора

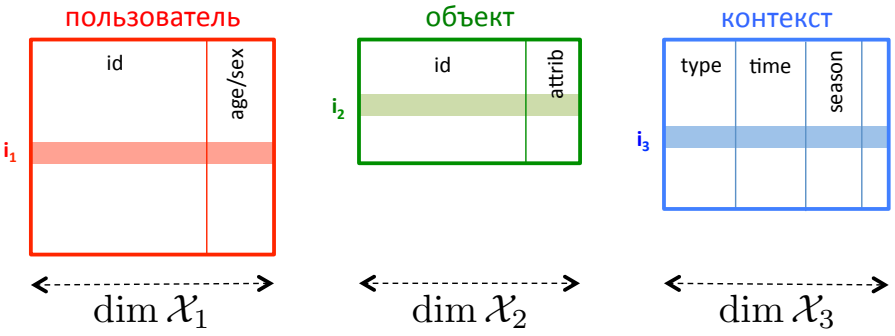


Рис. 3: Иллюстрация базового набора значений факторов

Тензоры. Будем называть *прогнозирующим тензором* отображение декартового произведения факторов в множество ответов:

$$\hat{t} : \mathcal{X}_1 \times \cdots \times \mathcal{X}_F \to \mathcal{Y}.$$

Здесь и далее будем рассматривать $\mathcal{Y} = \mathbb{R}$, т.е. задачу регрессии, причем с квадратичными потерями. Переход к нетривиальным $\mathcal{Y}$ можно производить при помощи различных функций активации, как это делается в обобщенных линейных моделях, но это отдельная задача, выходящая за рамки дипломной работы.

Значения факторов. Фиксированный набор значений факторов нужен для проведения процедуры обучения с использованием значения по-умолчанию в пропущенных элементах (гипотеза MNAR, см. раздел 2.5). Для вывода выражений обучения представления тензора удобно рассматривать фактор как имеющий фиксированный набор известных векторов, хотя после обучения модель тензора можно использовать без фиксированного набора значений его факторов.

Пример ситуации, когда при обучении могут использоваться не все возможные на практике значения факторов: фактор имеет множественный признак (представленный векторным описанием как набор индикаторов элементов множества). Мы естественно не будем работать со всеми возможными подмножествами — значениями множественного признака, поскольку их комбинаторно много. Такая ситуация на практике возникает, например, если добавить в фактор номера товаров, содержащиеся у пользователя в корзине.

4.2 Модели низкорангового представления тензора

Модель нулевого ранга. Модель нулевого ранга не содержит в себе элементов матричного разложения, представляет собой простую модель линейной регрессии для конкатенации признаков всех факторов:

$$\hat{t}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_F) = w_0 + \sum_{f=1}^{F} \boldsymbol{w}_f^T \boldsymbol{x}_f.$$

Модель канонического разложения (тензорная машина). По аналогии с тензорным разложением, описанным в [11]:

$$\hat{t}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_F) = \mathbf{1}^T (\boldsymbol{V}_1 \boldsymbol{x}_1) \circ \cdots \circ (\boldsymbol{V}_F \boldsymbol{x}_F) = \\ = \sum_{r=1}^{R} \left\{ \prod_{f=1}^{F} \boldsymbol{v}_{f,r}^T \boldsymbol{x}_f \right\} = \sum_{r=1}^{R} \left\{ \prod_{f=1}^{F} \sum_{p=1}^{\dim \mathcal{X}_f} v_{f,rp} x_{f,p} \right\}. \quad (20)$$

Такую модель далее будем именовать моделью *тензорной машины*. Для модели тензорной машины будет предложен алгоритм обучения.

Модель разложения с парными взаимодействиями между факторами (Pairwise). По аналогии с тензорным разложением PITF (Pairwise-Interaction Tensor Factorization), описанным в [26]:

$$\hat{t}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_F) = \sum_{f=1}^{F} \sum_{f'=f+1}^{F} (\boldsymbol{V}_f \boldsymbol{x}_f)^T (\boldsymbol{V}_{f'} \boldsymbol{x}_{f'}) =$$

$$= \sum_{f=1}^{F} \sum_{f'=f+1}^{F} \left\{ \sum_{p=1}^{\dim \mathcal{X}_f} \sum_{p'=1}^{\dim \mathcal{X}_{f'}} x_{f,p} x_{f',p'} \cdot \boldsymbol{v}_{f,p}^T \boldsymbol{v}_{f',p'} \right\} =$$

$$= \frac{1}{2} \sum_{r=1}^{R} \left\{ \left(\sum_{f=1}^{F} \sum_{p=1}^{\dim \mathcal{X}_f} x_{f,p} v_{f,rp} \right)^2 - \sum_{f=1}^{F} \left(\sum_{p=1}^{\dim \mathcal{X}_f} x_{f,p} v_{f,rp} \right)^2 \right\}. \quad (21)$$

Модель разложения со всеми парными взаимодействиями (Full Pairwise). В дополнение к межфакторным взаимодействиям введем внутрифакторные взаимодействия:

$$\hat{t}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_F) = \sum_{f=1}^{F} \sum_{f'=f}^{F} (\boldsymbol{V}_f \boldsymbol{x}_f)^T (\boldsymbol{V}_{f'} \boldsymbol{x}_{f'}) =$$

$$= \sum_{f=1}^{F} \sum_{f'=f+1}^{F} \left\{ \sum_{p=1}^{\dim \mathcal{X}_f} \sum_{p'=1}^{\dim \mathcal{X}_{f'}} x_{f,p} x_{f',p'} \cdot \boldsymbol{v}_{f,p}^T \boldsymbol{v}_{f',p'} \right\} =$$

$$= \frac{1}{2} \sum_{r=1}^{R} \left\{ \left(\sum_{f=1}^{F} \sum_{p=1}^{\dim \mathcal{X}_f} x_{f,p} v_{f,rp} \right)^2 - \sum_{f=1}^{F} \left(\sum_{p=1}^{\dim \mathcal{X}_f} x_{f,p} v_{f,rp} \right)^2 \right\}. \quad (22)$$

В итоге получается модель, в точности повторяющая факторизационные машины, которым в качестве вектора признаков подается конкатенация из векторов значений факторов:

$$\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_F].$$

Модель канонического разложения со сдвигом.

$$\hat{t}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_F) = w_0 + \sum_{f=1}^{F} \boldsymbol{w}_f^T \boldsymbol{x}_f + \mathbf{1}^T (\boldsymbol{V}_1 \boldsymbol{x}_1) \circ \cdots \circ (\boldsymbol{V}_F \boldsymbol{x}_F) \quad (23)$$

Модель наиболее применимая и интересная, но рассмотрена будет только каноническая часть, т.к. линейную часть при реализации можно легко добавить путем добавления строчки из единиц в матрицы $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_F$, а оптимизация параметра w_0 в алгоритме ALS вычисляется элементарно.

4.3 Связь с классическими тензорными разложениями

Многие известные тензорные разложения, используемые в задачах коллаборативной фильтрации, являются частным случаем разложения Таккера. Пусть имеется тензор $\mathcal{G}$ размера $S_1 \times \cdots \times S_D$. Разложение Таккера имеет вид (следуя [15]):

$$\mathcal{T} = \mathcal{G} \times_1 \boldsymbol{M}_1 \times_2 \boldsymbol{M}_2 \cdots \times_D \boldsymbol{M}_D = \sum_{i_1=1}^{S_1} \cdots \sum_{i_D=1}^{S_D} g_{i_1 \ldots i_D} \boldsymbol{m}_{i_1} \circ \cdots \circ \boldsymbol{m}_{i_D}. \tag{24}$$

Здесь символом "$\circ$"обозначено не покомпонентное произведение, а внешнее (outer) векторное произведение, $\boldsymbol{m}_{i_1} \circ \cdots \circ \boldsymbol{m}_{i_D}$ означает тензор ранга 1.

Разложение Таккера предполагает нахождение и дальнейшее использование тензора $\mathcal{G}$, что неприемлемо на практике в силу его размеров. Частные случаи разложения Таккера, такие как каноническое разложение (CANDECOMP/PARAFAC, скелетное разложение), парное разложение (PITF) используют заранее заданный тензор $\mathcal{G}$ простой формы (например, квадратный единичный тензор).

Основной идея в построении модели для тензорной машины — замена факторов $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_D$ на линейные преобразования исходного признакового пространства факторов:

$$\boldsymbol{m}_{i_1} = \boldsymbol{V}_1 \boldsymbol{x}_{i_1}, \ \ldots, \ \boldsymbol{m}_{i_D} = \boldsymbol{V}_D \boldsymbol{x}_{i_D}. \tag{25}$$

При этом настраиваются не факторы $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_D$, а линейные преобразования $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_D$.

4.4 Обучение тензорных машин

Тензорной машиной называется модель прогнозирующего тензора вида (20).

Будем настраивать параметры модели, минимизируя регуляризованные квадратичные потери:

$$\mathcal{L}(\Theta) = \sum_{x_1 \in \mathcal{X}_1} \cdots \sum_{x_F \in \mathcal{X}_F} c(x_1, \ldots, x_F) \left\{ \hat{t}(x_1, \ldots, x_F; \Theta) - t(x_1, \ldots, x_F) \right\}^2 + \Omega(\Theta) \to \min_{\Theta}, \tag{26}$$

Для краткости далее будем обозначать: $x = (x_1, \ldots, x_F)$.

Оптимизация $\boldsymbol{v}_{f,p}$. Для любого фактора $f = 1, \ldots, F$ предсказание модели можно представить в виде:

$$\hat{t}(x) = \boldsymbol{q}_f(x_{\setminus f})^T \boldsymbol{V}_f \boldsymbol{x}_f = \sum_{p=1}^{P_f} x_{f,p} \cdot \boldsymbol{q}_f(x_{\setminus f})^T \boldsymbol{v}_{f,p} \tag{27}$$

Это представление позволит удобно выписать выражения для оптимизации.

Выпишем функционал оптимизации относительно параметра $\boldsymbol{v}_{f,p}$, выбросив части, не зависящие от него:

$$\mathcal{L}(\boldsymbol{v}_{f,p}) = \sum_{x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_F} c(x) \left\{ \boldsymbol{q}_f(x_{\setminus f})^T \sum_{p=1}^{P_f} x_{f,p} \cdot \boldsymbol{v}_{f,p} - t(x) \right\}^2 + \lambda \boldsymbol{v}_{f,p}^T \boldsymbol{v}_{f,p} \tag{28}$$

Продифференцируем и приравняем к нулю производную:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_{f,p}} = 2 \sum_x c(x) \left\{ x_{f,p} \cdot \boldsymbol{q}_f(x)^T \boldsymbol{v}_{f,p} + \boldsymbol{q}_f(x)^T \sum_{p' \neq p} x_{f,p'} \cdot \boldsymbol{v}_{f,p'} - t(x) \right\} \cdot x_{f,p} \boldsymbol{q}_f(x) + \\ + 2\lambda \boldsymbol{v}_{f,p} = 0 \tag{29}$$

Из уравнения (29) получаем, что оптимальное значение $\boldsymbol{v}_{f,p}$ получается как решение СЛАУ:

$$\boldsymbol{v}_{f,p}^\star = (\boldsymbol{A}_{f,p} + \lambda \boldsymbol{I})^{-1} \boldsymbol{b}_{f,p} \tag{30}$$

Матрица $\boldsymbol{A}_{f,p}$ и вектор $\boldsymbol{b}_{f,p}$ вычисляются по обучающей выборке $\mathcal{X}_{\text{learn}}$, а также параметрам t_0, c_0 по аналогии с алгоритмом, описанным в [11]. Быстрое вычисление матриц для одного фактора производится путеп предпосчета $\boldsymbol{Q}_f \boldsymbol{Q}_f^T = \sum_{x \in \mathcal{X}} \boldsymbol{q}_f(x) \boldsymbol{q}_f(x)^T$:

$$\boldsymbol{Q}_f \boldsymbol{Q}_f^T = \sum_{x \in \mathcal{X}} \boldsymbol{q}_f(x) \boldsymbol{q}_f(x)^T = \\ \sum_{\boldsymbol{x}_1 \in \mathcal{X}_1} \cdots \setminus_f \cdots \sum_{\boldsymbol{x}_F \in \mathcal{X}_F} \left\{ (\boldsymbol{V}_1 \boldsymbol{x}_1) \circ \ldots \setminus_f \ldots (\boldsymbol{V}_F \boldsymbol{x}_F) \right\} \left\{ (\boldsymbol{V}_1 \boldsymbol{x}_1) \circ \ldots \setminus_f \ldots (\boldsymbol{V}_F \boldsymbol{x}_F) \right\}^T = \\ \sum_{\boldsymbol{x}_1 \in \mathcal{X}_1} \cdots \setminus_f \cdots \sum_{\boldsymbol{x}_F \in \mathcal{X}_F} \left\{ (\boldsymbol{V}_1 \boldsymbol{x}_1)(\boldsymbol{V}_1 \boldsymbol{x}_1)^T \right\} \circ \ldots \setminus_f \cdots \circ \left\{ (\boldsymbol{V}_F \boldsymbol{x}_F)(\boldsymbol{V}_F \boldsymbol{x}_F)^T \right\} = \\ \left\{ \sum_{\boldsymbol{x}_1 \in \mathcal{X}_1} (\boldsymbol{V}_1 \boldsymbol{x}_1)(\boldsymbol{V}_1 \boldsymbol{x}_1)^T \right\} \circ \ldots \setminus_f \cdots \circ \left\{ \sum_{\boldsymbol{x}_F \in \mathcal{X}_F} (\boldsymbol{V}_F \boldsymbol{x}_F)(\boldsymbol{V}_F \boldsymbol{x}_F)^T \right\} \tag{31}$$

Обозначим P_f — среднее число ненулевых элементов среди значений фактора $\mathcal{X}_f$, $P = P_1 + \cdots + P_F$. Сложность одной итерации алгоритма 4.1 — $O(R^2 |\mathcal{X}_{\text{know}}| P + R^3(|\mathcal{X}_1| + \cdots + |\mathcal{X}_F| P))$.

4.5 Реализация алгоритма обучения

Алгоритм 4.1 реализован на языке Python с использованием следующих сторонних библиотек:

- NumPy
- SciPy.sparse (разреженные матрицы)

Algorithm 4.1 Обучение канонической тензорной машины методом ALS

Вход:

- $\mathcal{X}_{\text{learn}} = \{(\boldsymbol{x}_{1,n}, \ldots, \boldsymbol{x}_{F,n})\}_{n=1}^{N}$ — обучающие элементы тензора;
- $\{(t(x), c(x)),\ x \in \mathcal{X}_{\text{learn}}\}$ — значения обучающих элементов тензора и их веса;
- t_0, c_0 — значения, которыми будут заполнены неизвестные элементы тензора;
- R — ранг модели;
- $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_F$ — матрицы значений факторов;
- число итераций n_iters.

Выход: Параметры модели $\Theta = (\boldsymbol{V}_1, \ldots, \boldsymbol{V}_F)$.
for $k = 1, \ldots,$ n_iters **do**
  for $f = 1, \ldots, F$ **do**
    Предпосчитать $\boldsymbol{Q}_f \boldsymbol{Q}_f^T$ *используя выражения для быстрого подсчета (31)*
    for $p = 1, \ldots, \dim \mathcal{X}_f$ (в случае категориальных признаков — параллельно) **do**
      Обновить $\boldsymbol{v}_{f,p}$*, используя выражение (30) и предпосчитанные* $\boldsymbol{Q}_f \boldsymbol{Q}_f^T$

- IPython.parallel (распараллеливание оптимизационных задач)

Запуски проводились на 32-х ядерной машине на базе процессора Intel Xeon CPU 2.2GHz. Выполнение производилось в среде IPython[11].

5 Применение тензорных машин

5.1 Персонализация веб-сервиса социальных закладок Delicious

Веб сервис Delicious (`delicious.com`) предоставляет пользователям возможность создавать собственную коллекцию закладок (интернет-ссылок), размечать закладки в коллекции семантическими тегами, делиться закладками с другими пользователями и заводить взаимные социальные связи.

На рис. 4 показан интерфейс добавления закладки в коллекцию. Можно видеть основные принципы работы с сервисом: зарегистрированный пользователь добавляет URL, по URL определяется заголовок страницы, который можно заменить на любой другой; далее пользователь может поставить теги новой закладке, в то время как он это делает, сервис уже предлагает ему выбор из наиболее подходящих тегов.

Набор данных Delicious Bookmarks был представлен на конференции RecSys'11 [5]. В наборе присутствуют следующие данные:

[11] `http://ipython.org/`

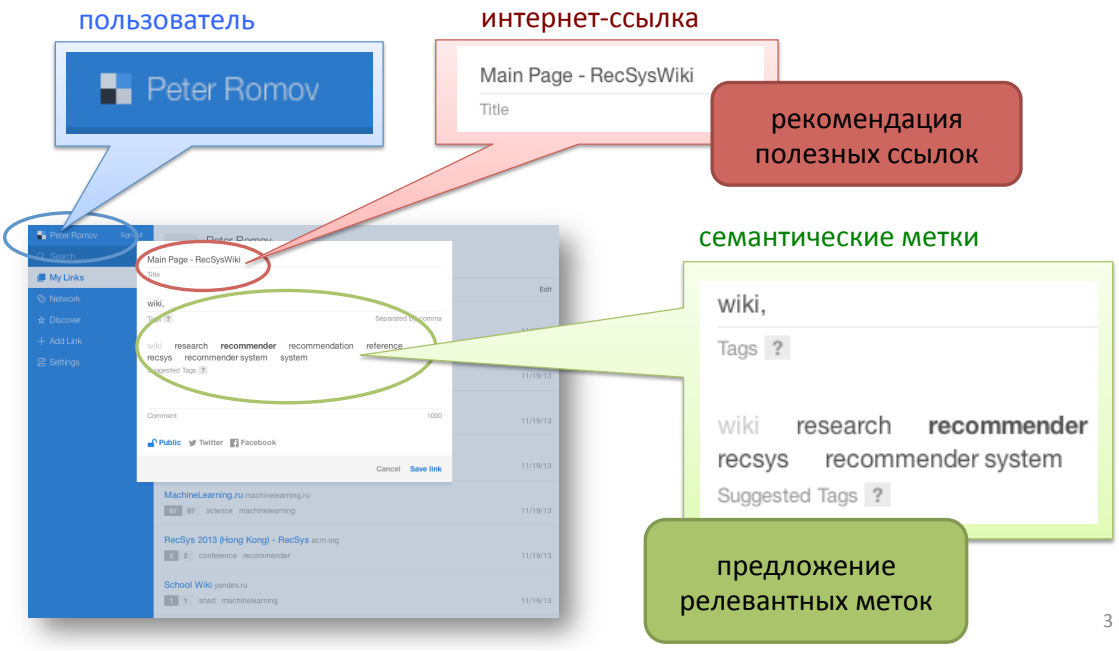


Рис. 4: Персонализация веб-сервиса для хранения закладок Delicious.

- 104'799 закладок (`bookmark`);
- 1'867 пользователей (`user`);
- 53'388 тегов (`item`)
- 104'799 связей вида «пользователь добавил закладку в свой список» (`user bookmarks`)
- 437'593 связей вида «пользователь поставил закладке некоторый тег» (`tag assignment`)
- 7'668 связей вида «пользователь u_1 является другом пользователя u_2» (`user contacts`),

Каждая закладка имеет URL и заголовок.

Для демонстрации работы с тензорными машинами, а также сравнения качества моделей с другими методами, попробуем решить две задачи моделирования предпочтения:

1. оценка предпочтения закладки b пользователем u (персональные рекомендации закладок);
2. оценка релевантности тега t к закладке b, которую добавил в коллекцию пользователь u (автоматическая аннотация закладок тегами).

Построение факторов По набору данных были построены факторы для пользователей (`user`), закладок (`bookmark`) и тегов (`tag`). На рис. 5 схематично изображены матрицы значений факторов.

Опишем факторы и их признаки:

- фактор `user` (соответствует пользователю сервиса):
  - признак `id` — идентификатор пользователя, категориальный;
  - признак `friends` — индикаторы дружбы пользователя с остальными, вектор смежности пользователя с другими;
- фактор `bookmark` (соответствует закладке):
  - признак `id` — идентификатор закладки, категориальный;
  - признак `zone` — доменная зона из URL (пример: `.edu`, `.com`, `.cn`, `.us`), категориальный;
  - признак `title-bow` — вектор частот слов из заголовка, частоты от 0 до 1, словарь слов был построен путем выбора 2000 наиболее частотных слов (примеры: «webpage», «academy», «developer», «internet»);
  - признак `tags` — индикаторы тегов, к которым была отнесена закладка хотя бы одним пользователем; признак не используется в задаче автоматической разметке тегами;
- фактор `tag` (соответствует тегу), фактор используется только в задаче автоматической разметки тегами:
  - признак `id` — идентификатор тега, категориальный.

Оценка качества рекомендаций. Для оценки качества были выбраны пользователи, имеющие достаточное количество закладок (не менее 20); закладки, используемые не менее 10 пользователями. У каждого пользователя выбирались 3 тестовых закладки (релевантные), которые не участвовали в обучении, а также генерировались 97 случайных закладок (нерелевантные), из тех что отобраны для участия в оценке качества. На таких наборах из 100 закладок тестировалась ранжирующая рекомендательная модель. В качестве оценок берем AUC-Recall@k (площадь под кривой Recall@k), а также средние значения (по пользователям) Recall@k для $k = 5, 10, 20$.

Оценки предпочтения. Будем считать, что если пользователь добавил закладку в свою коллекцию, то оценка предпочтения такой пары — 1, иначе — 0. Отметим, что в данных есть только примеры добавления закладок в коллекцию, отрицательных примеров нет.

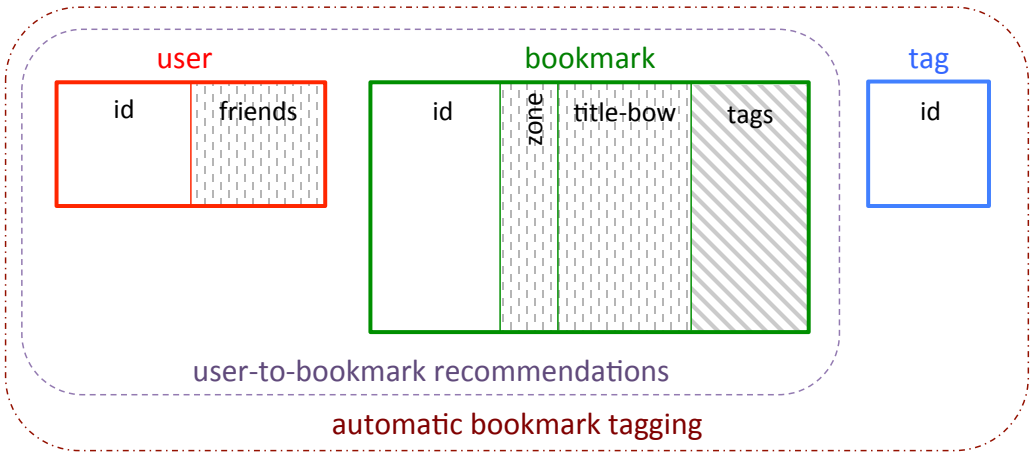


Рис. 5: Факторы для персонализации веб-сервиса Delicious при помощи тензорных машин.

Ранг приближения. Зафиксируем параметры обучения модели: $c_0 = 0.05$, $\lambda = 0.1$. Будем настраивать и оценивать качество моделей с различными рангами и различными наборами признаков. На рис. 6 показана получившаяся зависимость качества ранжирования от ранга модели.

Вес пропущенных элементов c_0. Зафиксируем модель ранга $R = 8$. Будем использовать все признаки в факторе. На рис. 7 представлена зависимость качества ранжирования от параметра обучения c_0.

Сравнение моделей. В таблице 1 приведены оценки качества ранжирования для различных моделей:

- random — случайное ранжирование;
- popularity — ранжирование закладок по убыванию популярности;
- iTALS — модель, оптимизированная алгоритмом, предложенным в [11] (эмулируется предложенным алгоритмом обучения тензорной машины, когда все факторы имеют только один признак-идентификатор);
- TM — модель тензорной машины, обозначения id/tags/feats говорят о том, какие признаки были использованы в факторе: id — только идентификатор, tags — только теги (для фактора закладок), feats — все признаки, перечисленные выше.

Параметр регуляризации подбирался по тестовой выборке из набора $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$.

Из таблицы 1 хорошо видно, что использование различных дополнительных признаков в факторе дает значительный прирост качества ранжирования.

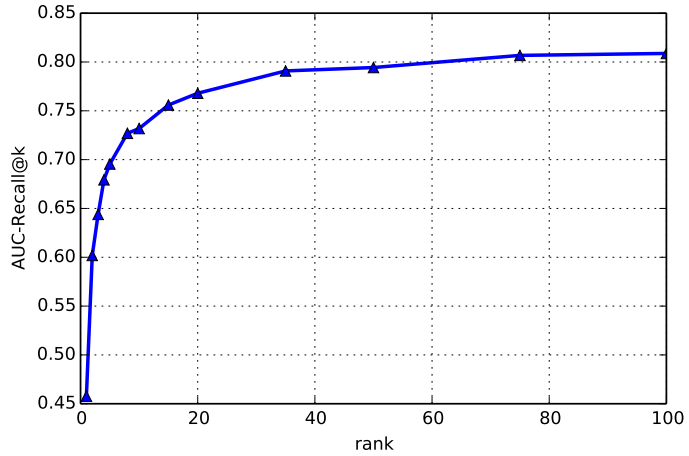


Рис. 6: Влияние ранга модели на качество ранжирования. При обучении использовались все признаки факторов `user` и `bookmark`, $\lambda = 0.1$.

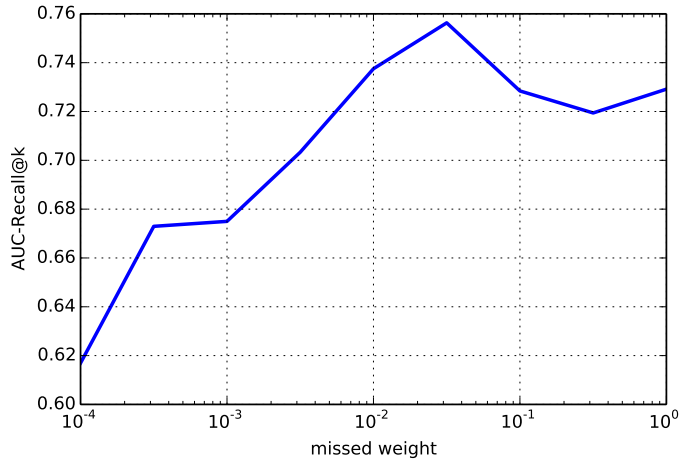


Рис. 7: Влияние параметра обучения c_0 на качество ранжирования. При обучении использовались все признаки факторов `user` и `bookmark`, $\lambda = 0.1$.

Method	AUC-Recall@k	Recall@5	Recall@10	Recall@20
random	0.49636	0.00444	0.00716	0.01876
popularity	0.59508	0.02456	0.04197	0.06789
iTALS (no feats)	0.68373	0.01194	0.03514	0.15183
TM: user id + bookmark tags	0.71973	0.05595	0.11088	0.20471
TM: user id + bookmark feats	0.75338	0.12760	0.25486	0.37803
TM: user feats + bookmark feats	**0.75592**	**0.14364**	**0.28250**	**0.40464**

Таблица 1: Оценки качества ранжирования закладок в рекомендательной системе для веб-сервиса Delicious.

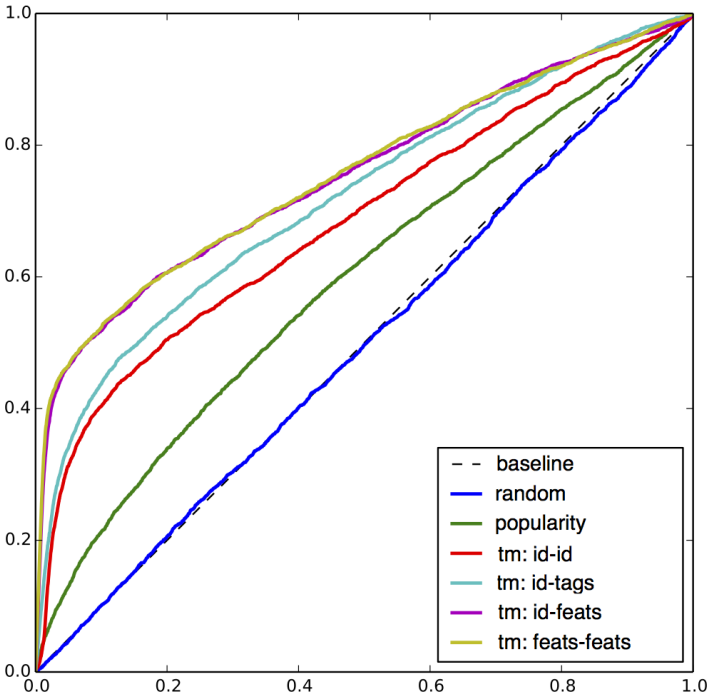


Рис. 8: Кривые AUC-Recall@k для различных рекомендательных моделей в Delicious Dataset.

Method	c_{def}	RMSE	MAE
FM Session–Track	—	0.324110	0.240907
FM Session–Track(taxonomy)	—	**0.317298**	**0.239264**
FM Session–Track(taxonomy)–Genre	—	0.412428	0.386374
TM Session–Track	0	0.320103	**0.236280**
TM Session–Track(taxonomy)	0	**0.319800**	0.236794
TM Session–Track(taxonomy)–Genre	0	0.397406	0.374351
TM Session–Track(taxonomy)	1e-2	0.323549	0.242878
TM Session–Track(taxonomy+tags)	1e-2	**0.320103**	**0.236321**
TM Session–Track(taxonomy)–Genre	1e-3	0.379479	0.245278

Таблица 2: Метрики RMSE и MAE на тестовой выборке качества регрессионных моделей, предсказывающих долю прослушанного трека пользователем до «пропуска»; 1 — пользователь прослушал весь трек целиком, 0 — пользователь пропустил трек не начав слушать, 0.5 — пользователь прослушал половину трека и «пропустил».

5.2 Персональное радио на сервисе Яндекс.Музыка

Набор данных содержит записи за 10 дней о поведении пользователей на сервисе Яндекс.Музыка. Пользователь описывается номером своей сессии. Трек описывается своим номером, номером альбома, номером артиста и номерами тегов, соответствующих альбому и артисту. Требуется предсказать регрессионную переменную, соотествующую доле прослушанной части трека.

В таблице 2 приведено сравнение с моделью факторизационных машин. Точность предсказания тензорной машины и

6 Заключение

6.1 Проделанная работа

В рамках дипломной работы были решены следующие задачи:

- проведен обзор техник построения рекомендательных систем, выявлена потребность в алгоритме коллаборативной фильтрации, учитывающем произвольные признаковые описания факторов, способном работать для данных с неявной обратной связью (implicit feedback);
- проведен обзор алгоритмов коллаборативной фильтрации;
- предложено понятие прогнозирующего тензора как адаптация задачи обучения по прецедентам на случай многофакторных задач;
- предложен и реализован эффективный алгоритм обучения тензорной машины с каноническим разложением на основе алгоритма ALS (Alternative Least Squares);

- применимость тензорных машин продемонстрирована на реальных данных с вебсервисов Delicious и Яндекс.Музыка.

6.2 Выводы

Модели низкоранговых приближений имеют огромный потенциал для решения задач коллаборативной фильтрации, а также из других предметных областей, в которых имеют место взаимодействия между объектами разных типов.

Тензорные машины обобщают зарекомендовавшие себя на практике факторизационные модели для коллаборативной фильтрации, являются удобной конструкцией для экспериментов с низкоранговыми разложениями, позволяют менять структуру модели путем изменения признакого пространства.

Тензорная машина с каноническим разложением может быть эффективно обучена при помощи алгоритма ALS (Alternative Least Squares). Польза от добавления признаков в факторы показана на реальных данных в задаче моделирования пользовательского предпочтения.

Список литературы

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[3] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[4] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[5] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.

[6] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *The Journal of Machine Learning Research*, 13(1):3619–3622, 2012.

[7] Tianqi Chen, Zhao Zheng, Qiuxia Lu, Weinan Zhang, and Yong Yu. Feature-based matrix factorization. *arXiv preprint arXiv:1109.2271*, 2011.

[8] Keith Cheverst, Nigel Davies, Keith Mitchell, Adrian Friday, and Christos Efstratiou. Developing a context-aware electronic tourist guide: some issues and experiences. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24. ACM, 2000.

[9] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[10] Zeno Gantner, Steffen Rendle, and Lars Schmidt-Thieme. Factorization models for context-/time-aware movie recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, pages 14–19. ACM, 2010.

[11] Balázs Hidasi and Domonkos Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer, 2012.

[12] Balázs Hidasi and Domonkos Tikk. Context-aware item-to-item recommendation within the factorization framework. In *Proceedings of the 3rd Workshop on Context-awareness in Retrieval and Recommendation*, pages 19–25. ACM, 2013.

[13] Liangjie Hong, Aziz S Doumith, and Brian D Davison. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 557–566. ACM, 2013.

[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[15] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[16] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[18] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SDM*, volume 5, pages 1–5. SIAM, 2005.

[19] Gregory Piatetsky. Interview with simon funk. *ACM SIGKDD Explorations Newsletter*, 9(1):38–40, 2007.

[20] István Pilászy and Domonkos Tikk. Recommending new movies: even a few ratings are more valuable than metadata. In *Proceedings of the third ACM conference on Recommender systems*, pages 93–100. ACM, 2009.

[21] István Pilászy, Dávid Zibriczky, and Domonkos Tikk. Fast als-based matrix factorization for explicit and implicit feedback datasets. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 71–78. ACM, 2010.

[22] Steffen Rendle. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE Computer Society, 2010.

[23] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[25] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th ACM SIGIR Conference on Reasearch and Development in Information Retrieval*. ACM, 2011.

[26] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM, 2010.

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[28] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[29] Nathan Srebro, Tommi Jaakkola, et al. Weighted low-rank approximations. In *ICML*, volume 3, pages 720–727, 2003.

[30] Harald Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–722. ACM, 2010.

[31] Chengjie Sun, Lei Lin, Yuan Chen, and Bingquan Liu. Expanding user features with social relationships in social recommender systems. In *Natural Language Processing and Chinese Computing*, pages 247–254. Springer, 2013.

[32] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

[33] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. Springer, 2008.

[34] К. В. Воронцов. Математические методы обучения по прецедентам (теория обучения машин). *Москва*, 2011.