Санкт–Петербургский государственный университет

ОМАРОВ Руслан Зулфигарович

Выпускная квалификационная работа

Применение методов байесовского обучения с подкреплением для решения некоторого класса задач рекомендации контента

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2015 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Исследование и проектирование систем управления и обработки сигналов»

Научный руководитель:
доцент, кафедра технологий программирования,
к.т.н. Блеканов Иван Станиславович

Рецензент:
старший преподаватель,
кафедра космических технологий и
прикладной астродинамики,
Давыденко Александр Александрович

Санкт-Петербург
2019 г.

Содержание

Введение . . . 3

Постановка задачи . . . 5

Обзор литературы . . . 6

Глава 1. Анализ существующих решений . . . 8

1.1. Коллаборативная фильтрация . . . 8

1.1.1 Тривиальный подход . . . 8

1.1.2 User-based . . . 9

1.1.3 Item-based . . . 9

1.1.4 Латентные модели . . . 11

1.2. Графовые методы . . . 13

Глава 2. Бандитские алгоритмы . . . 15

2.1. Бесконтекстные алгоритмы . . . 15

2.1.1 ε-greedy . . . 15

2.1.2 UCB1 . . . 16

2.2. Контекстные алгоритмы . . . 17

2.2.1 disjoint-linUCB . . . 17

2.2.2 hybrid-linUCB . . . 19

2.3. «Холодный старт» в задаче многоруких бандитов . . . 21

Глава 3. Метод offline оценки . . . 22

Глава 4. Тестовая задача . . . 25

Глава 5. Задача рекомендации групп . . . 29

5.1. Методы обработки данных . . . 29

5.1.1 Исходные данные . . . 30

5.1.2 Билинейное отображение . . . 31

5.1.3 Кластеризация . . . 32

5.2. Анализ полученных результатов . . . 33

Выводы . . . 37

Заключение . . . 38

Список литературы . . . 39

Введение

Крупные онлайн площадки, предоставляющие пользователям доступ к информации, а авторам и сообществам релевантную аудиторию, на данный момент являются неотъемлемой частью нашей жизни. Каждый день мы сталкиваемся с различными сайтами, социальными сетями. И все эти сервисы стремятся максимально угодить пользователям, предоставляя интересный для них контент. Внедрение элементов персонализации в подобные сервисы считается уже де-факто стандартом. Такие ресурсы изменяют свое содержание и наполнение (а иногда даже и функциональность), подстраиваясь к предпочтениям каждого конкретного пользователя.

Одним из решений задач персонализации контента являются рекомендательные системы. Рекомендательная система – это такой метод, который подбирает пары пользователь – ресурс, максимально удовлетворяющие интересы обеих сторон. Например, необходимо подобрать и показать посетителю такой товар в интернет магазине, который наверняка его заинтересует, и он (посетитель) скорее всего это купит. Или нужно показать пользователю на странице в социальной сети сообщество, которое будет удовлетворять его интересам, и к которому он возможно присоединится. Подобных примеров очень много. В связи с тем, что задачи рекомендации чего-либо кому-либо могут иметь разные постановки и конечные цели, существует множество различных подходов к решению данных проблем. При построении рекомендательной системы необходимо выделить следующие характеристики, которые позволяют описать ее:

1. **Предмет** – это то, что рекомендуется системой. Это могут быть сайты, музыка, объявления и т.д.

2. **Цель** – для чего это делается. К примеру, покупка, переход по ссылке на сайт, прослушивание определенной композиции.

3. **Источник** – тот, кто рекомендует. Это могут быть как другие пользователи ресурса, так и сам ресурс в лице его владельцев.

4. **Контекст** – при каком действии пользователя ему начинают что-

либо рекомендовать. Это может быть просмотр новостей, прослушивание музыки или же рекомендации начинаются сразу же при посещении главной страницы сайта.

5. **Персонализированность** – мера того, насколько сильно будет подстраиваться система под ваши интересы.

На сегодняшний день существует несколько подходов к построению рекомендательных систем: коллаборативная фильтрация, MAB (*Multi-Armed Bandits*), графовые методы и т.д. Далее будут рассмотрены некоторые вышеперечисленные методы, их достоинства и недостатки, в особенности в контексте проблемы «холодного старта» (*cold-start problem*). Так или иначе многие из этих подходов сводятся к задачам машинного обучения по имеющейся истории взаимодействий, выбирающих наиболее релевантный контент. Однако во многих ситуациях такой истории нет – например, для новых пользователей, и рекомендательная система не может дать качественный прогноз для такого пользователя. Это и называется проблемой «холодного старта». Однако необходимо и для данной группы пользователей и ресурсов строить адекватные рекомендации, чтобы максимально удовлетворить и тех, и других.

Постановка задачи

Цель работы – исследование методов контекстных многоруких бандитов (Contextual MAB) и их применения в решении задачи рекомендации групп в социальной сети Одноклассники, которая предоставила необходимые данные и вычислительные ресурсы. Чтобы достичь поставленной цели, необходимо решить следующие задачи:

1. Проанализировать существующие способы решения задачи в контексте проблемы «холодного старта» (коллаборативная фильтрация, графовые методы).

2. Изучить и разработать инструментарий для применения и оценки качества алгоритмов многоруких бандитов.

3. Проверить эффективность некоторых бандитских алгоритмов на тестовой задаче и определить оптимальный.

4. Применить выявленный в пункте 3 метод к задаче рекомендации групп в социальной сети Одноклассники.

Обзор литературы

В ходе проделанной работы был исследован ряд статей и научных трудов по данной тематике. Основным источником реализованных в работе алгоритмов стала статья «A contextual-bandit approach to personalized news article recommendation», Li L., Chu W., Langford J., Schapire R. E. [1]. В ней описаны основные алгоритмы контекстных многоруких бандитов, в частности disjoint-linUCB и hybrid-linUCB, а также приведены математические обоснования применимости данных методов.

Кроме того, для сравнения качества рекомендаций были дополнительно рассмотрены два бесконтекстных алгоритма многоруких бандитов ε-greedy и UCB1, которые подробно изучены в статье «Finite-time analysis of the multiarmed bandit problem», Auer P., Cesa-Bianchi N., Fischer P. [2]. В этой работе приведен псевдокод бесконтекстных алгоритмов, а также для них доказаны теоремы для оценки сожаления (regret) после n проведенных шагов.

Рассмотренные в данной работе алгоритмы являются методами обучения с подкреплением, что, в свою очередь, означает, что обучение происходит при взаимодействии модели со средой. Поскольку реальную среду с реальными пользователями смоделировать очень трудно, оценивание производилось с помощью метода offline оценки, описанного в работе «An unbiased offline evaluation of contextual bandit algorithms with generalized linear models», Li L., Chu W., Langford J., Moon T., Wang X. [3].

Также немаловажную роль в работе с данными является их компактное, но в то же время информативное представление. С этой целью использовался метод билинейного отображения, который переводит представления данных в новые векторные пространства. Этот метод представлен в статье «A case study of behavior-driven conjoint analysis on Yahoo! Front Page Today Module», Chu W., Park S.-T., Beaupre T., Motgi N. [4].

В исследовании также проведен анализ существующих решений задачи рекомендации. В частности, рассмотрены графовые модели и методы коллаборативной фильтрации, которые подробно представлены в www.machinelearning.ru [5] и «Musical recommendations and personalization

in a social network», Bugaychenko D., Dzuba A. [6].

Глава 1. Анализ существующих решений

На сегодняшний день существует несколько способов построения рекомендательных систем. В данной главе будут рассмотрены основные методы, их достоинства и недостатки.

1.1 Коллаборативная фильтрация

Существует множество различных видов алгоритмов коллаборативной фильтрации. Согласно [5] их можно разделить на две группы:

1. Корреляционные модели.
2. Латентные модели.

Корреляционные модели основаны на хранении целиком матрицы R — матрицы интересов/рейтингов, где по строкам расположены пользователи, а по столбцам предметы рекомендации. Весь алгоритм заключается в специальном «пробегании» по строкам и столбцам этой матрицы и определении схожести пользователей или объектов. Более формально:
U — множество пользователей (users);
I — множество объектов-предметов рекомендации (items);
R — матрица размерности $|U| \times |I|$ интересов/рейтингов;

1.1.1 Тривиальный подход

Можно определить алгоритм подбора подходящих объектов данному пользователю [5] следующим образом:

1. Пользователь заходит на страницу какого-либо объекта i_0.
2. Определяется множество $U(i_0)$:

$$U(i_0) = \{u \in U |\ r_{u,i_0} \neq 0, u \neq u_0\}$$

Это множество пользователей, которые взаимодействовали с товаром i_0.

3. Определяется множество $I(i_0)$:

$$I(i_0) = \{i \in I|\ \text{sim}(i, i_0) > \delta\}$$

Это множество объектов, схожих с объектом i_0. Мера схожести определяется функцией sim. Далее выбирается определенное число максимально схожих с i_0 объектов и выдается пользователю в качестве рекомендации.

Данный подход является достаточно простым и наивным и основывается лишь на одном действии пользователя и ни на чем другом.

Два следующих подхода отличаются от описанного выше и отталкиваются уже от самих пользователей и объектов, а не только от их действий.

1.1.2 User-based

Теперь будем исходить из интересов пользователей. Для данного конкретного пользователя u_0 определяем объекты, которыми он уже когда-то интересовался. Затем с помощью заранее определенной функции близости sim находим множество $U(u_0)$ пользователей, схожих с u_0. Далее определяем множество $I(u_0)$:

$$I(u_0) = \left\{ i \in I|\ B(i) = \frac{|U(u_0) \cap U(i)|}{|U(u_0) \cup U(i)|} > 0 \right\}$$

где $U(i) = \{u \in U|\ r_{u,i} \neq 0\}$. Смысл данного множества состоит в том, что в нем находятся те объекты, которые были оценены пользователями из $U(u_0)$. И большую значимость в этом множестве будут иметь те объекты, которые нравились максимально схожим с u_0 пользователям. Затем данное множество объектов ранжируется по значениям $B(i)$, и осуществляется рекомендация максимальных из них.

1.1.3 Item-based

Следующий подход отталкивается не только от текущего действия пользователя, но и от всей его истории на данном ресурсе. Для этого вы-

полняются следующие действия

1. Для пользователя u_0 определяется множество $I(u_0)$:

$$I(u_0) = \{i \in I | \ \exists i_0 : \ r_{u_0,i_0} \neq 0, \ B(i) = \mathrm{sim}(i, i_0) > \alpha\}$$

   В данное множество входят те объекты i, которые достаточно схожи (это определяется с помощью параметра α) с объектами i_0, которыми уже когда-то интересовался пользователь u_0.

2. Определяется множество $I(u_0)$, сортируется по убыванию значения функции схожести sim, и затем рекомендуются N первых объектов из этого упорядоченного множества.

Отдельного внимания в вышеупомянутых методах заслуживают параметры алгоритмов (α, δ) и функция близости sim. Значения α и δ определяются исследователем и зависят от задачи: чем больше эти значения, тем меньше объектов или пользователей будут считаться близкими. Среди функций близости также возможны варианты. Наиболее распространенные из них:

1. косинусная мера

$$\mathrm{sim}(u, u_0) = \frac{\sum\limits_{i \in I(u,u_0)} r_{u_0,i} \cdot r_{u,i}}{\sqrt{\sum\limits_{i \in I(u,u_0)} r_{u_0,i}^2 \sum\limits_{i \in I(u,u_0)} r_{u,i}^2}}$$

2. корреляция Пирсона

$$\mathrm{sim}(u, u_0) = \frac{\sum\limits_{i \in I(u,u_0)} (r_{u_0,i} - \overline{r}_{u_0}) \cdot (r_{u,i} - \overline{r}_u)}{\sqrt{\sum\limits_{i \in I(u,u_0)} (r_{u_0,i} - \overline{r}_{u_0})^2 \sum\limits_{i \in I(u,u_0)} (r_{u,i} - \overline{r}_u)^2}}$$

Можно выделить следующие недостатки алгоритмов, основанных на корреляционных моделях:

1. Необходимо хранить целиком всю матрицу рейтингов R.

2. Проблема «холодного старта» — нечего рекомендовать совсем новому пользователю, который еще ничего не успел оценить/купить на данном ресурсе. При этом новые объекты не будут никому рекомендоваться, поскольку с ними никто не взаимодействовал.

Также в отдельных случаях можно отметить невозможность построения адекватной рекомендации для нетипичных пользователей и тривиальность рекомендаций: чаще всего пользователям будут показаны самые популярные объекты.

Однако стоит отметить среди преимуществ подобных методов их быструю и простую реализацию, а также понятную интерпретацию рекомендаций.

1.1.4 Латентные модели

Другой тип моделей, которые также описаны в [5] основаны на выявлении скрытых зависимостей между объектами и пользователями. Теперь для каждого пользователя или объекта будет рассматриваться не вектор его взаимодействий с соответствующими элементами рекомендательной системы, а так называемый профиль — вектор, который описывает объект или пользователя в каком-то пространстве, размерность которого, как правило, много меньше размерностей исходной матрицы R. Это сразу же позволяет избавиться от такого существенного недостатка корреляционных моделей, как необходимость хранения матрицы большой размерности — теперь вместо неё будут полученные профили. При этом может пострадать интерпретируемость рекомендаций ввиду перехода к новому векторному представлению. Однако при удачном построении вышеупомянутого перехода можно получить хорошо интерпретируемые компоненты новых векторных представлений элементов.

Более формально основную идею латентных моделей можно записать следующим образом. По данным D взаимодействий пользователей и объектов строятся векторы:

1. $(p_{t,u})_{t \in G}$ — профили пользователей $u \in U$, $|G| \ll |I|$;

2. $(g_{t,i})_{t \in H}$ — профили объектов $i \in I$, $|H| \ll |U|$.

В зависимости от способа построения данных векторов их компоненты могут иметь различный смысл. Например, можно воспользоваться кластеризацией и разделить объекты и пользователей на некоторое количество классов. Тогда $p_{t,u}$ и $g_{t,i}$ будут означать степень принадлежности элемента к классу t. И затем уже исследуются взаимодействия между не отдельным пользователем и объектом, а классами.

Также для выделения скрытых (латентных) зависимостей можно воспользоваться методом SVD-разложения. Идея этого метода состоит в поиске представления исходной матрицы рейтингов R в виде произведения трех матриц:

$$R = P^{\mathrm{T}} \Delta Q$$

где $P = (p_{t,u})_{|T| \times |U|}$ — матрица профилей пользователей, $Q = (q_{t,i})_{|T| \times |I|}$ — матрица профилей объектов, $|T|$ — множество интересов (здесь предполагается, что $G = H = T$), $\Delta = \mathrm{diag}(\pi_1, \ldots, \pi_t)$ — диагональная матрица весов различных тем. Поиск такого разложения осуществляется с помощью метода стохастического градиентного спуска (SGD). Он является особенно эффективным задачах больших размерностей, когда обычный градиентный спуск оказывается чересчур ресурсоёмким: каждый обычный градиентный шаг требует суммирования m слагаемых, где m — размер обучающей выборки, но при значениях m в несколько миллионов (а в крупных рекомендательных системах именно так и есть) вычисления будут занимать достаточно большое время. В SGD на каждом шаге значения градиента вычисляется не на каждом объекте выборки, а на одном, который каждый раз выбирается случайным образом. При этом доказано [7], что SGD сходится к оптимальному решению. При этом минимизируется функционал ошибки:

$$||R - P^{\mathrm{T}} \Delta Q|| \to \min_{P,Q}$$

или при разреженности матрицы R:

$$\sum_{(u,i)\in R} (r_{u,i} - \overline{r}_u - \overline{r}_i - \sum_{t\in T} p_{t,u}q_{t,i})^2 \to \min_{P,Q},$$

где $\overline{r}_u$ и $\overline{r}_i$ — средние рейтинги пользователя u и объекта i соответственно.

Отдельно стоит отметить класс неотрицательных матричных разложений (NNMF [8]), при которых все элементы матриц P и Q не меньше нуля. Они являются более предпочтительными, чем обычные SVD-разложения, поскольку элементы полученных матриц гораздо проще интерпретировать. Например, каждая компонента профиля $(p_{t,u})_{t\in T}$ может рассматриваться как степень привлекательности t категории объектов для пользователя u.

К преимуществам данного типа моделей можно отнести низкий объем хранимых данных и интерпретируемость профилей элементов. Кроме того, возможно частичное решение проблемы «холодного старта» построением усредненного профиля элемента по каким-либо первичным признакам. Таковыми могут быть, например, пол, возраст, регион или жанр литературного произведения и его автор. Среди недостатков можно отметить неединственность SVD-разложения матрицы.

1.2 Графовые методы

Данный подход описан в [6] и основывается на построении «графа вкусов» (taste graph). Этот граф можно представить в виде набора следующих элементов:

1. V — конечное непустое множество вершин; T_V — конечное непустое множество типов вершин; $\tau_V : V \to T_V$ — отображение, которое каждой вершине ставит в соответствие её тип.

2. $\theta \in V$ — балансировочная вершина.

3. E — конечное непустое множество ребер между вершинами; T_E — конечное непустое множество типов ребер; $\tau_E : E \to T_E$ — отображение, которое каждому ребру ставит в соответствие его тип.

4. $R : E \to V \times V$ — отображение, которое каждому ребру ставит в соответствие его начало и конец.

5. $\omega_E : E \to [0, 1]$ — функция весов для ребер.

Рекомендации с использованием такой модели строятся на основе метода случайного блуждания (random walk) для графов. Допустим, необходимо рекомендовать какому-либо пользователю объекты, которые его наверняка заинтересуют. Тогда в данном графе нужно найти вершину v, которая соответствует этому пользователю. Далее начинается случайное блуждание из этой вершины: в каждый момент времени, находясь в какой-либо вершине random walk совершает случайный шаг в какую-либо смежную вершину. После N шагов это блуждание останавливается и извлекается путь, который был совершен по этому графу. Затем уже из этого пути выбираются вершины определенного типа, они ранжируются в том порядке, в котором осуществлялся обход, и далее осуществляется рекомендация. При этом вероятности перехода в какую-либо смежную вершину на определенном шаге зависит от веса ребра, соединяющего эти вершины.

Основным преимуществом данного типа моделей является наглядность рекомендаций и простота реализации. При этом задача «холодного старта» решается построением приближения элемента рекомендательной системы и помещения этой «усредненной» вершины в taste graph. Однако качество такого решения сильно зависит от способа построения этого приближения и начального объема информации, которая имеется у исследователя о пользователе/объекте.

Глава 2. Бандитские алгоритмы

Проблема в случае многоруких бандитов ставится следующим образом: представим себе, что перед нами стоит N различных игровых автоматов. Дергая за ручки этих автоматов, необходимо максимизировать суммарную прибыль, которую принесут эти действия. Задача заключается в нахождении оптимального способа выбора ручки на очередном шаге. Математически это можно записать так:
$\mathcal{A}$ — множество доступных действий («ручек»);
$x_t \in R^d$ — контекст (информация об объектах и/или пользователях) на определенном шаге. Он определяется средой, в которой рассматриваются многорукие бандиты;
$p_{t,a,x}$ — ожидаемые выплаты для ручки a, которые можно получить в момент времени t при заданном контексте x_t;
r_t — реальная награда, наблюдаемая на шаге t.

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, p_{t,a,x} \ ,$$

$$R_T = \sum_t r_t \to \max, \text{ при } t = 1, 2, \ldots$$

2.1 Бесконтекстные алгоритмы

Данные методы основываются исключительно на статистических данных, полученных в результате своих предыдущих действий.

2.1.1 ε-greedy

Алгоритм ε-greedy [2] работает следующим образом: на каждом шаге для очередного пользователя выбираем с вероятностью $1-\varepsilon$ ручку с максимальной на данный момент средней наградой и с вероятностью ε выбираем случайную

Algorithm 1 ε-greedy

while True do
  $\varepsilon \leftarrow random[0, 1]$
  if $\varepsilon > 1 - \varepsilon_0$ **then**
    $a_t = random(\mathcal{A})$
  else
    $a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \overline{x}_a$
  Show to user a_t and get reward r_t
  Update $\overline{x}_{a_t}$

где $random[0, 1]$ и $random(\mathcal{A})$ — случайное вещественное число из отрезка $[0, 1]$ и случайный элемент множества $\mathcal{A}$ соответственно.

2.1.2 UCB1

Алгоритм UCB1 [2] устроен несколько сложнее: каждый раз необходимо вычислять приоритет всех ручек по формуле:

$$p_a = \overline{x}_a + \sqrt{\frac{2 \ln n}{n_a}},$$

где $\overline{x}_a$ — это средняя награда ручки a, n — общее количество шагов, n_a — количество показов ручки a. Далее выбирается ручка с максимальным приоритетом

Algorithm 2 UCB1

Initialization: play each arm once
while True do
  $a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left(\overline{x}_a + \alpha \sqrt{\frac{2 \ln n}{n_a}} \right)$
  Show to user a_t and get reward r_t
  Update $\overline{x}_{a_t}, n_{a_t}, n$

где α — параметр алгоритма, который определяет соотношение использование/исследование (exploit/exploration): чем больше значение параметра, тем больше алгоритм будет стараться показать пользователю малоисследованные ручки (те, у которых мало показов). Идея алгоритма

состоит в том, что чаще всего будут показываться ручки с максимальной средней наградой, но некоторая доля показов, определяемая параметром α, будет «исследовательской». Исследование необходимо для выявления популярных и интересных объектов. И даже если у какой-либо ручки очень маленький $\overline{x}_a$ при большом количестве показов n_a (объект оказался никому не интересным или очень специфичным), то через некоторое время числитель подкоренного выражения станет больше знаменателя, и второе слагаемое в правой части равенства формулы (в строке 3) станет достаточно большим, и ручка будет снова показана пользователю. При относительном постоянстве множества объектов/ручек $\mathcal{A}$ имеет смысл уменьшать значение α при увеличении общего количества показов n.

2.2 Контекстные алгоритмы

В отличие от бесконтекстных алгоритмов, которые используют для рекомендации только накопленные со временем статистические данные, контекстные алгоритмы вдобавок к этой информации пытаются выявить зависимости между пользователями и объектами, основанные на их признаковых представлениях.

2.2.1 disjoint-linUCB

Алгоритм disjoint-linUCB [1] представляет собой объединение методов линейной регрессии и UCB1. Основное предположение данного типа алгоритмов заключается в том, что математическое ожидание награды $r_{t,a}$ является линейной функцией от признаков пользователей:

$$E\left[r_{t,a}|x_{t,a}\right] = x_{t,a}^{\mathrm{T}}\hat{\theta}_a,$$

где $x_{t,a}$ — контекст пользователя на шаге t, соответствующий ручке a (в данном случае для всех ручек он будет один и тот же на определенном шаге); θ_a^* — неизвестный вектор параметров для ручки a, который определяется с помощью гребневой регрессии.

Если предположить, что имеется матрица D_a размерности m на d (d

— размер вектора признаков — контекста), соответствующая m тренировочным примерам, которые наблюдались к моменту t для ручки a, то тогда получим формулу для вектора параметров

$$\hat{\theta}_a = (D_a^{\mathrm{T}} D_a + I_d)^{-1} D_a^{\mathrm{T}} c_a,$$

где c_a — вектор ответов размерности m; I_d — единичная матрица размерности d. Тогда, согласно [1, 9], можно показать, что с вероятностью $1 - \delta$ будет верно неравенство

$$|x_{t,a}^{\mathrm{T}} \hat{\theta}_a - E\left[r_{t,a} | x_{t,a}\right] \leq \alpha \sqrt{x_{t,a}^{\mathrm{T}} (D_a^{\mathrm{T}} D_a + I_d)^{-1} x_{t,a}}$$

для всех $0 < \delta < 1$. Причём $A_a \stackrel{\text{def}}{=} D_a^{\mathrm{T}} D_a + I_d$. Следовательно, получаем формулу выбора ручки на очередном шаге

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left(x_{t,a}^{\mathrm{T}} \hat{\theta}_a + \alpha \sqrt{x_{t,a}^{\mathrm{T}} A_a^{-1} x_{t,a}} \right).$$

Далее представлен псевдокод алгоритма.

Algorithm 3 disjoint-linUCB, Part 1

Input: $\alpha > 0$
while True do
    Get $x_{t,a}$ from the environment
    for all $a \in \mathcal{A}$ **do**
        if a is new **then**
            $A_a = I_d$
            $b_a = 0_{d \times 1}$ (d-dimensional zero vector)
        $\hat{\theta}_a \leftarrow A_a^{-1} b_a$
        $p_{t,a} \leftarrow x_{t,a}^{\mathrm{T}} \hat{\theta}_a + \alpha \sqrt{x_{t,a}^{\mathrm{T}} A_a^{-1} x_{t,a}}$
    $a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, p_{t,a}$
    Show to user a_t and get reward r_t
    $A_{a_t} \leftarrow A_{a_t} + x_{t,a_t} x_{t,a_t}^{\mathrm{T}}$
    $b_{a_t} \leftarrow b_{a_t} + r_{t,a} x_{t,a_t}$

В строках 5—7 происходит инициализация начальных данных для ручки a, если она только что попала во множество $\mathcal{A}$ или в начале работы алгоритма. Как упоминалось в начале данного пункта, алгоритмы linUCB сочетают в себе подходы и линейной регрессии, и UCB1, и в данном случае первое слагаемое в формуле вычисления $p_{t,a}$ (строка 9) относится к регрессионной части подхода, а второе — к UCB1.

2.2.2 hybrid-linUCB

Данный алгоритм является продолжением идей disjoint-linUCB и, помимо информации о пользователях, использует дополнительно контексты объектов. Согласно [1] предполагается следующая оценка математического ожидания клика пользователя:

$$E\left[r_{t,a}|x_{t,a}\right] = z_{t,a}^{\mathrm{T}}\hat{\beta} + x_{t,a}^{\mathrm{T}}\hat{\theta}_a,$$

где $z_{t,a} \in R^k$ — признаковое описание конкретной комбинации пользователь-объект; $\hat{\beta}$ — вектор параметров, общий для всех ручек $a \in \mathcal{A}$. Вектор $z_{t,a}$ вычисляется как тензорное произведение (outer product) контекстов пользователя и объекта. Затем полученная матрица выпрямляется в вектор (например, все строки записываются в одну строку). Таким образом, $z_{t,a}$ можно интерпретировать как меру «привлекательности» объекта для пользователя и наоборот. Вектор $\hat{\beta}$ содержит веса, описывающие важности компонент $z_{t,a}$. Данный алгоритм является более трудоёмким по сравнению с disjoint-linUCB. Однако эффективная реализация позволит свести к минимуму время обработки одной итерации алгоритма и сделать возможным применение метода на практике в реальных условиях.

Algorithm 4 hybrid-linUCB, Part 1

1: **Input:** $\alpha > 0$
2: $A_0 \leftarrow I_k$
3: $b_0 \leftarrow 0_{k\times 1}$ (k-dimensional zero vector)
4: **while True do**
5: Get $x_{t,a}, z_{t,a}$ from the environment
6: $\hat{\beta} \leftarrow A_0^{-1} b_0$
7: **for all** $a \in \mathcal{A}$ **do**
8: **if** a is new **then**
9: $A_a = I_d$
10: $B_a = 0_{d\times k}$ (d-by-k zero matrix)
11: $b_a = 0_{d\times 1}$ (d-dimensional zero vector)
12: $\hat{\theta}_a \leftarrow A_a^{-1}(b_a - B_a\hat{\beta})$
13: $tmp_1 = z_{t,a}^{\mathrm{T}} A_0^{-1} z_{t,a} - 2 z_{t,a}^{\mathrm{T}} A_0^{-1} B_a^{\mathrm{T}} A_a^{-1} x_{t,a}$
14: $tmp_2 = x_{t,a}^{\mathrm{T}} A_a^{-1} x_{t,a} + x_{t,a}^{\mathrm{T}} A_a^{-1} B_a A_0^{-1} B_a^{\mathrm{T}} A_a^{-1} x_{t,a}$
15: $s_{t,a} \leftarrow tmp_1 + tmp_2$
16: $p_{t,a} \leftarrow z_{t,a}^{\mathrm{T}}\hat{\beta} + x_{t,a}^{\mathrm{T}}\hat{\theta}_a + \alpha\sqrt{s_{t,a}}$
17: $a_t = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, p_{t,a}$
18: Show to user a_t and get reward r_t
19: $A_0 \leftarrow A_0 + B_{t,a_t}^{\mathrm{T}} A_{t,a_t}^{-1} B_{t,a_t}$
20: $b_0 \leftarrow b_0 + B_{t,a_t}^{\mathrm{T}} A_{t,a_t}^{-1} b_{t,a_t}$
21: $A_{a_t} \leftarrow A_{a_t} + x_{t,a_t} x_{t,a_t}^{\mathrm{T}}$
22: $B_{a_t} \leftarrow B_{a_t} + x_{t,a_t} z_{t,a_t}^{\mathrm{T}}$
23: $b_{a_t} \leftarrow b_{a_t} + r_{t,a} x_{t,a_t}$
24: $A_{a_t} \leftarrow A_{a_t} + z_{t,a_t} z_{t,a_t}^{\mathrm{T}} - B_{t,a_t}^{\mathrm{T}} A_{t,a_t}^{-1} B_{t,a_t}$
25: $b_{a_t} \leftarrow b_{a_t} + r_{t,a} z_{t,a_t} - B_{t,a_t}^{\mathrm{T}} A_{t,a_t}^{-1} b_{t,a_t}$

Разделение формулы для вычисления $s_{t,a}$ на 2 части в строках 13 — 15 сделано исключительно ради удобства чтения.

2.3 «Холодный старт» в задаче многоруких бандитов

Как уже упоминалось ранее, проблема «холодного старта» является одной из основных задач, возникающих при построении рекомендательных систем: что рекомендовать новым пользователям и кому показывать новые объекты.

В бесконтекстных алгоритмах (пункты 2.1.1-2) эта задача решается с помощью накопленной статистической информации. В частности, новому пользователю чаще всего будут показаны наиболее популярные объекты (с большим значением $\overline{x}_a$) и с некоторой вероятностью, которая определяется значениями параметров ε и α для ε-greedy и UCB1 соответственно, новые объекты. В случае, когда в систему попал новый объект $item$, алгоритм UCB1 из-за особенностей определения показываемых ручек (в формуле $\overline{x}_a + \alpha\sqrt{\frac{2\ln n}{n_a}}$ второе слагаемое будет стремится к бесконечности из-за малости n_a для новых объектов) будет некоторое время показывать именно $item$ для сбора информации о нем. При этом, ε-greedy будет попрежнему использовать наиболее популярные ручки, но с вероятностью ε/k, где $k = |\mathcal{A}|$, покажет очередному пользователю $item$.

Контекстные алгоритмы, в свою очередь, для новых элементов используют начальные данные о них. Для пользователя, например, это могут быть его пол, возраст, регион, род деятельности и т.д. По этим данным строятся некоторые вектора контекстов $x_{t,a}$ и $z_{t,a}$ и осуществляется действие алгоритма. В этих контекстах будут только те компоненты, которые определяются по начальным данным, о которых говорилось выше. Остальные будут нулевыми. При этом будет собираться статистика и выявляться зависимость между объектами и пользователями, которая хранится во время исполнения алгоритма внутри различных матриц и векторов. Далее если встретится очередной новый пользователь $user$ с контекстом, похожим на тот, который уже видела система, то рекомендация будет строится уже с учетом информации, полученной во время предыдущего взаимодействия пользователя, похожего на $user$.

Глава 3. Метод offline оценки

Алгоритмы, описанные в главе 2, являются алгоритмами обучения с подкреплением. Это означает, что сам процесс обучения происходит во время взаимодействия алгоритма — «агента» со «средой». В случае рекомендательных систем среду сложно смоделировать, поскольку невозможно вручную учесть все взаимосвязи между объектами и пользователями, что существенно может сказаться на качестве модели. Однако необученную модель в промышленных проектах нельзя использовать для взаимодействия с реальными пользователями — это первое время может принести большие убытки. Для решения данной проблемы существует метод offline оценки качества бандитских алгоритмов [3], который основан на использовании истории предыдущих показов.

Algorithm 5 Offline Evaluation

Input: policy π; stream of events S;
$h_0 \leftarrow \emptyset$ (match history)
$R_0 \leftarrow 0$ (total rewards)
$t \leftarrow 0$ (counter)
for event $e \in S$ **do**
    Get (x_t, Z, a, r_a) from event
    if $\pi(h_{t-1}, (x_t, Z)) = a$ **then**
        Update π with using (x_t, Z, a, r_a)
        Add e to history h_{t-1}
        $R_t \leftarrow R_{t-1} + r_a$
        $t \leftarrow t + 1$
Return: R_t/t =0

где Z — множество векторов контекстов $z_{t,a}$. На самом деле, каждое событие содержит в себе набор актуальных объектов, которые необходимо показать пользователю, в виде контекстов $z_{t,a}$. Оценивание производится следующим образом:

1. Поступает очередное событие $e \in S$, которое состоит из контекста пользователя x_t, контекстов актуальных объектов/ручек Z, показанной ручки a, полученной при этом показе награды r_t (1/0 — клик/не клик).

2. Алгоритм π, опираясь на историю предыдущих показов h_{t-1}, старается предсказать ручку, на которую с максимальной вероятностью кликнет текущий пользователь.

3. Если предсказанная ручка совпала с той, что реально была показана в событии e, то происходит обновление алгоритма π, истории h_t, общей награды R_t и счетчика шагов t.

4. После того, как поток событий S закончится, метод оценки возвращает среднюю награду R_t/t за один шаг, которую получал алгоритм π.

Важно отметить, что оценка метода должна производится на событиях, поступающих из равномерного распределения [3], иначе алгоритм будет запоминать не реальные связи между объектами и пользователями, а только те закономерности, на основе которых построен рекомендер, из которого и брались эти события. Например, если в оценочных данных будут присутствовать ситуации, в которых всем мужчинам от 20 до 30 лет рекомендуются только статьи/группы про автомобили, то и бандитский алгоритм будет данной группе пользователей рекомендовать только эти объекты. При этом никакой информации об интересах конкретных пользователей алгоритм не получит.

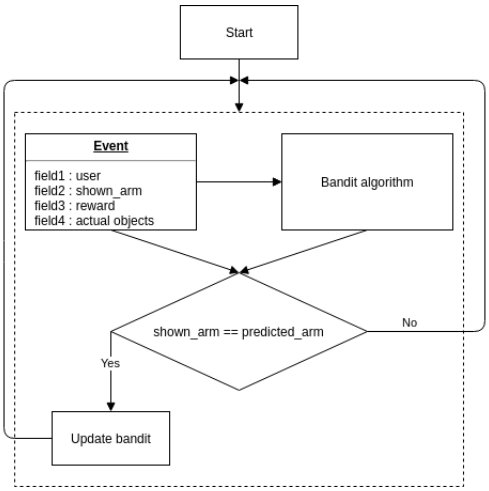

Рис. 1: Блок-схема метода offline оценки.

Для обновления данных в контекстных алгоритмах вместо дискретной награды r_t, которая принимает лишь значения 0 или 1, можно использовать любые другие коэффициенты r_1 для клика и r_0 для не клика. Например, r_1 может быть положительной величиной, а r_0 — отрицательной. Это позволит исследователю регулировать с помощью абсолютных значений данных коэффициентов значимость положительных или отрицательных событий, увеличивая при этом возможности алгоритмов для настройки под конкретные задачи. В частности, можно задать достаточно большое по модулю значение для r_0 по сравнению с r_1, тем самым увеличивая значимость безрезультатных показов.

Глава 4. Тестовая задача

В качестве тестовой задачи была взята проблема рекомендации новостей на портале Yahoo. В 2010 году компания Yahoo! проводила исследование по применению бандитских алгоритмов для рекомендации новостей посетителям главной страницы Yahoo! Today Module. Для этого были использованы данные о посещении страницы в период с 1 по 10 мая 2009 года [10].

Задача ставилась следующим образом: определить новость из имеющегося набора актуальных статей, на которую очередной посетитель страницы вероятнее всего кликнет, и поместить ее на первое место как на рисунке 2. При этом в методе offline оценки участвовали только те события, в которых совпадают новость, помещенная алгоритмом на первое место, и реально помещенная на страницу новость.

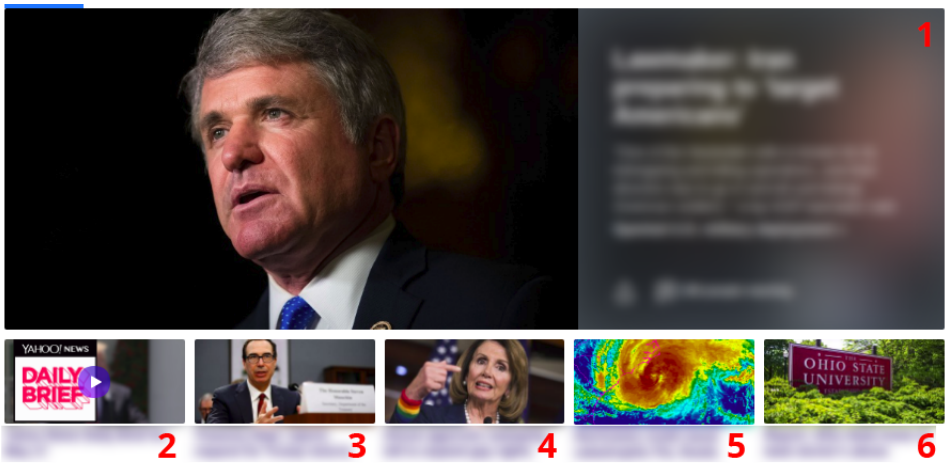

Рис. 2: Новостная страница Yahoo!

Каждое событие описывается кортежем следующего вида: временная отметка, идентификатор новости, полученная награда (1/0 — клик/не клик), информация о пользователе, информация об актуальных новостях (рисунок 3).

Данные представляют собой векторы размерности 6 следующей структуры: первая компонента всех векторов постоянна и равна 1, остальные представляют собой вероятности принадлежности элемента (пользователя или новости) к одному из пяти кластеров. Эти кластеры получены из

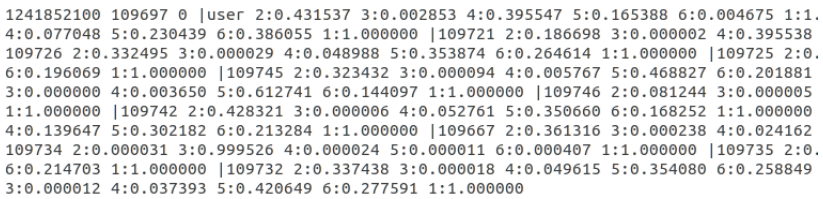

Рис. 3: Пример исходных данных.

исходных данных, представляющих собой различные демографические и социальные показатели, с помощью билинейного преобразования [4] и Гауссовской кластеризации [11].

В задаче были протестированы все упомянутые в пункте 5 алгоритмы. Особый интерес представляли результаты работы контекстных алгоритмов, поскольку ожидалось, что вероятность клика определенного пользователя на конкретную новость сильно зависит от интересов пользователя и информации, представленной в новости.

Основной метрикой качества в данной задаче является относительный CTR (relative Click-Through Rate, rCTR). Для начала определим, как вычисляется CTR:

$$\text{CTR} = \frac{n_{clicks}}{n_{shows}} \cdot 100\%,$$

где n_{clicks} — общее число кликов на новости, n_{shows} — общее число показов новостей. Тогда относительный CTR:

$$\text{rCTR} = \frac{\text{CTR}_{policy1}}{\text{CTR}_{policy2}},$$

где $\text{CTR}_{policy1}$ и $\text{CTR}_{policy2}$ — соответсвенно CTR алгоритмов (политик) 1 и 2. Мерой качества в этой задаче является rCTR алгоритмов относительно случайной политики, для которой предполагается, что $\text{rCTR} = 1$.

Полученные результаты представлены на рисунках 4, 5. Как и ожидалось, контекстные алгоритмы показали себя лучше бесконтекстных.

Также для данной задачи был проведен анализ сходимости контекстных алгоритмов по метрике rCTR в зависимости от общего количества показов новостей. Результаты представлены на рисунке 6. Как видно из графика, устойчивая сходимость достигается после 300000 показов ново-

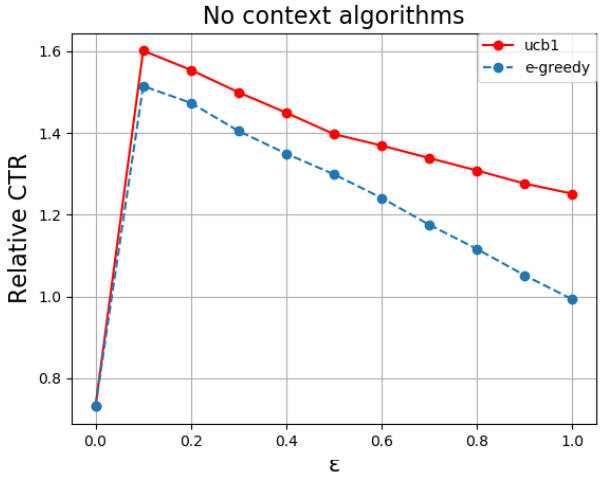

Рис. 4: Бесконтекстные алгоритмы.

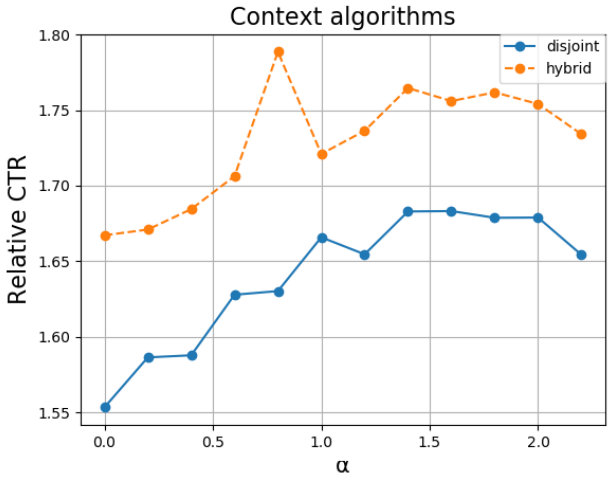

Рис. 5: Контекстные алгоритмы.

стей, что соответствует 7500 показов на каждую новость в среднем (всего в этот день было показано 39 различных новостей). Весь поток состоял из более чем 38000000 событий на 247 различных новостей.

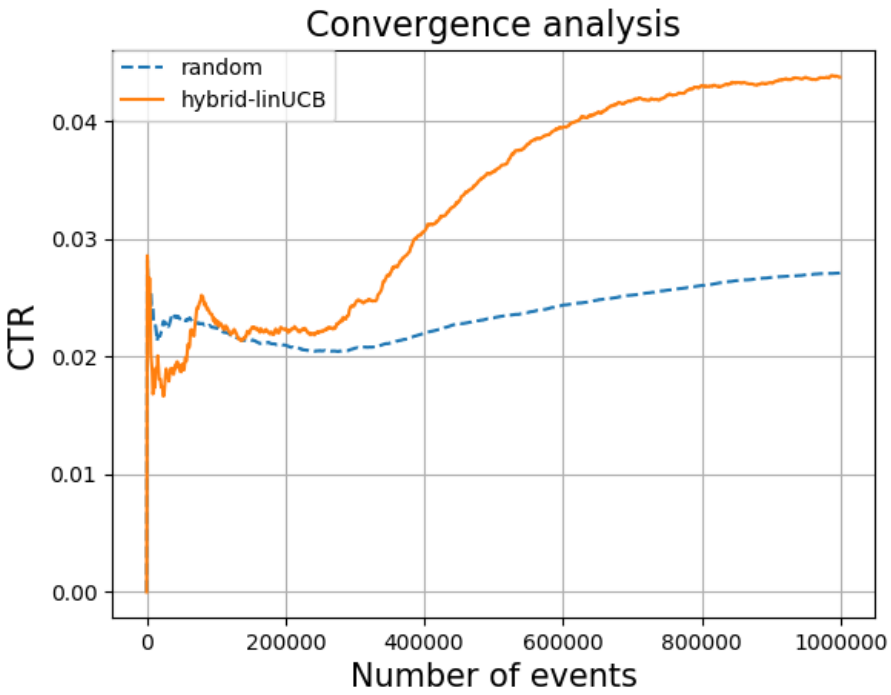

Рис. 6: Анализ сходимости CTR.

Глава 5. Задача рекомендации групп

В качестве основной задачи рассматривалась проблема рекомендации групп пользователям в социальной сети Одноклассники (далее ОК). Необходимо сразу отметить отличие этой задачи от тестовой:

1. В отличие от новостей, группы представляют собой более сложные объекты, время жизни которых на веб-ресурсе гораздо больше, а векторное представление сложнее.

2. Множество различных групп (41042 группы) намного больше множества различных новостей (247 новостей).

3. Пользователи на сайте Yahoo! также имеют куда более упрощенное представление из-за дополнительных возможностей (следовательно, и дополнительной информации) взаимодействия участников ОК.

4. В этой задаче необходима предобработка данных, тогда как в задаче рекомендации новостей все эти действия уже проделаны авторами обучающей выборки.

5.1 Методы обработки данных

Контекстные алгоритмы сильно зависят от длин контекстов пользователя и объекта (каждая размерность матриц, используемых в методе зависит линейно от размеров контекстов, т. е. количество элементов одной матрицы уже имеет квадратичную зависимость от длин этих векторов), т. к. на каждом шаге рекомендации необходимо вычислять некоторое количество обратных матриц. И выгоднее всего в данной ситуации использовать контексты как можно меньших размерностей, но при этом сохранять информативность. С этой целью применяется ряд последовательных преобразований начальных данных пользователей и объектов, которые подробнее описаны далее.

5.1.1 Исходные данные

Исходные данные в этой задаче были предоставлены сотрудниками ОК и представляли собой поток событий рекомендаций групп со структурой такой же, как и в задаче Yahoo. При этом вектора контекстов пользователей и групп строились следующим образом:

1. Для каждого пользователя и группы был получен вектор размерности 25 на основе метода SVD-разложения матрицы пользователь-документ-отклик (в данном случае лайки в ОК) описанного в [12].

2. Кроме того, в контекст пользователя и группы, при попадании их пары в бандитский алгоритм для рекомендации, включался набор счётчиков, который показывал историю взаимодействия данного пользователя с этой группой. Например, такими счетчиками были количество лайков, поставленных пользователем, на стене группы, количество просмотренных фото в этом сообществе, количество комментариев и прочее.

3. еще одной компонентой в обоих контекстах пары пользователь-группа было скалярное произведение их LDA представлений. Модель LDA строилась на текстовой коллекции русского языка, используя методы, описанные в [13]

Итоговый исходный контекст пользователя/группы представлял собой объединение всех трех вышеописанных сущностей:

$$\text{context}_u = (\text{SVD}_u, counters_{u,g}, \langle \text{LDA}_u, \text{LDA}_g \rangle) \,,$$

$$\text{context}_g = (\text{SVD}_g, counters_{u,g}, \langle \text{LDA}_u, \text{LDA}_g \rangle) \,,$$

где u и g — какие-либо пользователь и группа соответственно; $\langle x, y \rangle$ — скалярное произведение векторов x и y.

5.1.2 Билинейное отображение

Сначала к векторам, состоящим из SVD-описаний пользователей и объектов, а также счетчиков, характеризующих их взаимодействие, применялся метод главных компонент (PCA) с сохранением 90% дисперсии начальных данных. При этом получились вектора размерности 24. Для дальнейшего увеличения сходимости бандитских алгоритмов и уменьшения размерности контекстов было использовано билинейное отображение [4]. Этот метод переводит вектора контекстов в новое векторное пространство таким образом, что новый вектор пользователя описывает предпочтения особенностей (признаков) объекта. Математически это можно представить следующим образом:

$$s_{i,j} = \sum_{a=1}^{C} \sum_{b=1}^{D} x_{i,b} \omega_{a,b} y_{j,a},$$

или в матричной форме

$$s_{i,j} = x_i^{\mathrm{T}} W z_j,$$

где C и D — размеры контекстов объекта и пользователя соответственно; W — матрица весов размерности $D \times C$, которая определяется с помощью решения задачи логистической регрессии. Для этой задачи определяется следующая логистическая функция

$$p(r_{i,j}|s_{i,j}) = \frac{1}{1 + \exp(-r_{i,j} s_{i,j})},$$

где $p(r_{i,j}|s_{i,j})$ — вероятность получения награды $r_{i,j}$ для комбинации i-го пользователя и j-го объекта. Искомая матрица получается путем решения оптимизационной задачи

$$\min_W \left(\frac{1}{2c} \sum_{a=1}^{C} \sum_{b=1}^{D} \omega_{a,b}^2 - \sum_{i,j \in \mathcal{O}} \log p(r_{i,j}|s_{i,j}) \right).$$

Минимум данного выражения можно находить с помощью градиентного спуска, но в случае выборки больших размеров (а здесь это именно так —

размер выборки составляет несколько миллионов примеров) лучше всего воспользоваться методом стохастического градиентного спуска (SGD). От обычного градиентного спуска он отличается тем, что на каждом шаге для вычисления очередного приближения какого-либо весового коэффициента рассматривается не вся выборка целиком, а один ее случайный элемент. Формула градиентного шага будет иметь следующий вид:

$$\omega_{a,b} \leftarrow \omega_{a,b} - \left(\frac{1}{c}\omega_{a,b} - r_{i,j}x_{i,b}y_{j,a}(1 - p(r_{i,j}|s_{i,j})) \right),$$

где начальные значения $\omega_{a,b}$ определяются из условия $\{\omega_{a,b}\} \sim \mathcal{N}(0, c)$. Параметр c определяется с помощью кросс-валидации (cross-validation).

Тогда, после нахождения оптимальной матрицы W, новые признаковые описания пользователей и объектов можно получить по следующим формулам:

$$\hat{x}_i = x_i^{\mathrm{T}} W,$$

$$\hat{y}_j = W y_j.$$

5.1.3 Кластеризация

После применения билинейного отображения исходные данные стали более информативными, однако размерности контекстов (24 компоненты, которые были получены при PCA и еще одна постоянная компонента равная 1) по-прежнему большие для эффективного по памяти и скорости применения бандитских алгоритмов. Поэтому было принято решение разделить объекты и пользователей на кластеры. Поскольку, как объект, так и пользователь могут относится не к одному кластеру, а сразу к нескольким, для кластеризации использовался метод GMM из пакета sklearn.mixture, т. к. он позволяет получить не только дискретные ответы, в каком кластере находится элемент, но и степени принадлежности этого объекта ко всем кластерам.

Количество кластеров при этом выбиралось с использованием метрики силуэт (пакет sklearn.metrics), т. к. изначально неизвестны истинные метки кластеров элементов. Результаты представлены на рисунке 5. Коли-

чество компонент равное 2 и 3 не рассматривалось из-за нецелесообразности и неинформативности таких отображений.

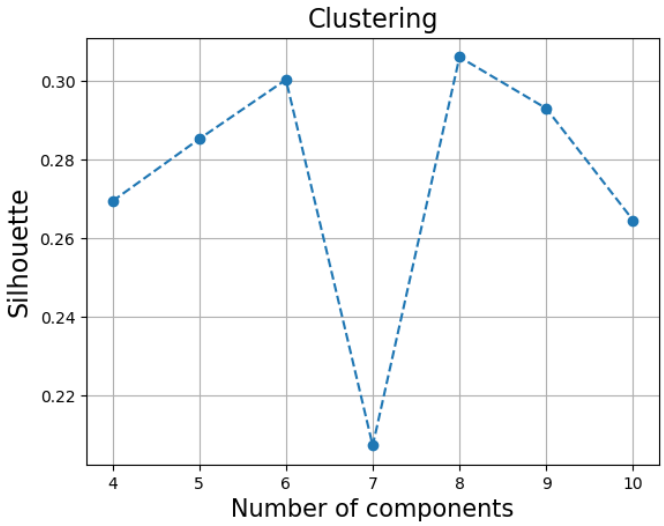

Рис. 7: Качество кластеризации.

Значения силуэта лежат в отрезке $[-1, 1]$, где значения, близкие к -1 означают, что кластеризация разрозненная, и кластеры не выделяются; значения около 0 — кластеры пересекаются и накладываются друг на друга; значения около 1 — кластеры расположены далеко друг от друга и хорошо разделены. Как видно из графика (рисунок 7), оптимальным оказалось число компонент равное 8.

После применения кластеризации признаками, описывающими пользователей и объекты, уже будут степени принадлежности элемента к кластеру. Полная схема обработки данных представлена на рисунке 8.

Рис. 8: Схема обработки исходных данных.

5.2 Анализ полученных результатов

Для анализа эффективности реализованных алгоритмов методом offline оценки были использованы данные о показах групп пользователям в ОК в

период с 1 января 2019 по 23 апреля 2019. Всего было 12760895 событий со случайными показами. Каждое событие представляло собой кортеж следующего вида:

1. Временная отметка.
2. Контекст пользователя.
3. Пять контекстов групп, показанных пользователю, которые считались актуальными на тот момент.
4. Группа, которая была показана пользователю на определенном месте (аналог места под номером один для новостей в задаче Yahoo!).
5. Награда, полученная за этот показ (клик/не клик).

Если сравнить объем имеющихся данных с тем, который был доступен в тестовой задаче (Глава 4), то можно заметить явное несоотвествие: 12760895 событий на 41042 групп и 38000000 показов на 247 новостей. Исходя из этого, предположение о том, что события в этой задаче полностью поступали из равномерного распределения, будет не совсем верно: хоть и все события были случайны, но ввиду большого количества групп и малого числа показов случилось так, что у некоторых групп количество показов пользователям в десятки раз больше, чем у других. Поэтому было принято решение об использовании другой метрики — сожаления на каждом шаге (regret). Она напрямую связана с CTR, поскольку вычисляется через награды, получаемые на каждом шаге от бандитского алгоритма и случайной политики. Связь с CTR можно записать следующим образом: алгоритм будет показывать эффективный CTR только, если его regret будет уменьшаться с каждым шагом. Формула для вычисления regret имеет вид:

$$regret_{\mathrm{T}} = \mathrm{T} \cdot \hat{r}_{\mathrm{T}} - \sum_{t=0}^{\mathrm{T}} r_t,$$

где T — текущий момент времени; $\hat{r}_{\mathrm{T}}$ — максимальная средняя награда ручки, которую получила на момент времени T случайная политика; r_t —

награда, полученная бандитским алгоритмом на t шаге. Смысл этой формулы заключается в вычислении разности между абсолютной наградой, которую можно было бы наблюдать каждый раз просто показывая пользователю ручку с максимальной средней наградой, и абсолютной наградой, которую принес бандитский алгоритм на момент времени T. При этом получается, что если regret уменьшается с каждым шагом, то бандитский алгоритм действует оптимальнее, чем наивная стратегия (т. е. постоянно показывается ручка с максимальной средней наградой). В частности, можно считать, что алгоритм будет действовать эффективнее, чем случайная политика по той причине, что, в предположении равномерного распределения событий, средняя награда всех ручек будет одинаковой для случайной политики.

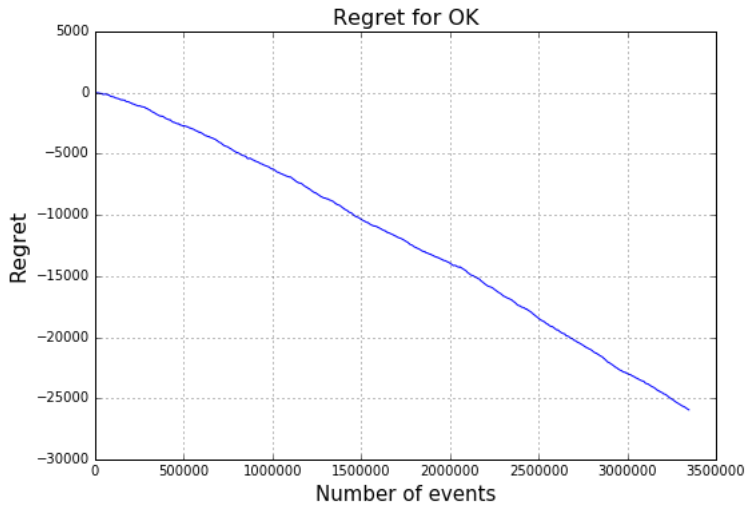

Рис. 9: Regret в задаче OK.

График regret в зависимости от времени представлен на рисунке 9. Для сравнения приведен график regret для задачи Yahoo! (рисунок 10). Ожидаемые перегибы на графике, отмеченные звездочками, соответствуют переходам от потока событий одного дня к другому. При этом набор актуальных новостей несколько изменяется: на смену некоторых старых новостей приходят новые. Как и ожидается, сначала алгоритм действует не оптимально и regret начинает расти, но после того, как собрано достаточное количество информации о новых объектах, сожаление продолжает

уменьшаться.

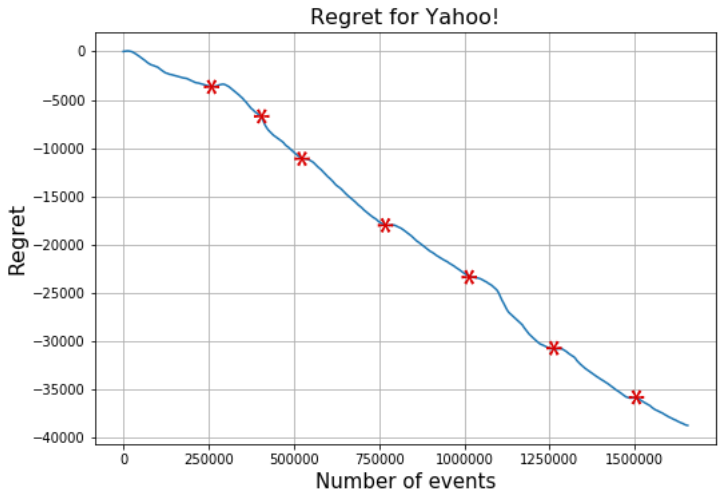

Рис. 10: Regret в задаче Yahoo!

Весь применяемый программный комплекс был реализован на языке программирования Python 3.6 с использованием возможностей пакетов numpy, sklearn, matplotlib, pandas. Подробнее ознакомиться с разработанными инструментами можно в репозитории.

Выводы

В рамках данной работы была поставлена цель: исследовать эффективность применения алгоритмов на основе байесовских бандитов для рекомендации групп в социальной сети Одноклассники. Для достижения цели получены следующие результаты:

1. Сделан обзор существующих решений задачи рекомендации контента.

2. Исследован ряд бандитских алгоритмов: ε-greedy, UCB1, disjoint-linUCB, hybrid-linUCB.

3. Разработан программный комплекс на языке Python 3.6 для реализации и проверки качества работы вышеупомянутых алгоритмов.

4. Выявлен лучший среди реализованных алгоритмов на примере задачи Yahoo. Как и ожидалось, таковым оказался hybrid-linUCB.

5. Также на примере задачи Yahoo проведен анализ сходимости алгоритма hybrid-linUCB. В ходе анализа получены оценки на среднее количество показов, необходимых для одной новости, чтобы метрика CTR стабилизировалась.

6. Разработан инструментарий на языке Python 3.6 для обработки данных перед непосредственным применением их в контекстных алгоритмах (билинейное отображение с применением встроенных PCA и кластеризации GMM).

7. Проведена оценка качества hybrid-linUCB для рекомендации групп в ОК. Алгоритм показал приемлемые результаты на основе имеющихся данных. Однако для еще большего увеличения эффективности требуется дополнительный объем данных.

Суммируя вышеупомянутые результаты, можно сделать вывод, что применение контекстных алгоритмов в данной задаче оправдано, но требует больших объемов данных и вычислительных мощностей.

Заключение

В ходе исследования были подробно изучены некоторые виды алгоритмов многоруких бандитов. Результатами работы являются полученные выводы о сходимости и эффективности этих методов, а также инструментарий для работы с ними.

Важно отметить, что основными недостатками бандитских методов обучения являются высокая чувствительность к исходным данным и вычислительным возможностям рекомендательной системы. В частности, время работы контекстных алгоритмов квадратично зависит от размеров признаковых описаний элементов системы. Именно с этой целью и был разработан инструментарий для предобработки и снижения размерности данных.

Кроме того, важным моментом при решении задач с бандитскими методами является определение множества актуальных объектов. Ведь чем больше это множество, тем больше времени потребуется на обработку одного события, поскольку нужно вычислить ожидаемые выплаты $p_{t,a}$ для каждого объекта из этого множества.

Однако эти методы обладают рядом достоинств. Например, их можно обучать параллельно с тем, пока работает какой-либо другой алгоритм рекомендации, не теряя при этом эффективности на еще необученном алгоритме. Также бандитские алгоритмы могут хорошо справляться с проблемой «холодного старта» по методике, описанной в пункте 2.3 главы 2.

Подводя итоги, можно заключить, что применение бандитских алгоритмов в данной задаче приемлемо, но требует большого внимания к реализации, в частности, к техническим средствам и способу представления исходных данных. Кроме того, обучение этих моделей может занять довольно длительной время.

В дальнейшем планируется продолжить работу по увеличению скорости сходимости реализованных алгоритмов, а также рассмотрению более сложных моделей данного класса, в том числе в новых предметных областях.

Список литературы

[1] Li L., Chu W., Langford J., Schapire R. E. A contextual-bandit approach to personalized news article recommendation // Proceedings of the 19th International Conference on World Wide Web. 2010. P. 661–670.

[2] Auer P., Cesa-Bianchi N., Fischer P. Finite-time analysis of the multiarmed bandit problem // Machine Learning. 2002. Vol. 47. No 2–3. P. 235–256.

[3] Li L., Chu W., Langford J., Moon T., Wang X. An unbiased offline evaluation of contextual bandit algorithms with generalized linear models // PMLR. Proceedings of the Workshop on On-line Trading of Exploration and Exploitation 2. 2012. Vol. 26. P. 19–26.

[4] Chu W., Park S.-T., Beaupre T., Motgi N. A case study of behavior-driven conjoint analysis on Yahoo!: front page today module // Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. 2009. P. 1097–1104

[5] MachineLearning.ru - профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных. [Электронный ресурс]: URL:http://www.machinelearning.ru (дата обращения: 17.03.19).

[6] Musical recommendations and personalization in a social network // Proceedings of the 7th ACM Conference on Recommender Systems. 2013. P. 367–370.

[7] Kiwiel, Krzysztof C. Convergence and efficiency of subgradient methods for quasiconvex minimization // Mathematical Programming. Series A. 90 (1). Berlin, Heidelberg. 2001. P. 1–25.

[8] Cichocki A., Zdunek R., Amari S. Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization // Independent Component Analysis and Signal Separation. 2007. P. 169–176.

[9] Walsh T. J., Szita I., Diuk C., Littman M. L. Exploring compact reinforcement-learning representations with linear regression. // Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence. 2009.

[10] Yahoo! Front page today module user click log dataset, version 1.0 (1.1 GB) [Электронный ресурс]: URL:https://webscope.sandbox.yahoo.com/catalog.php?datatype=r&did=49 (дата обращения: 17.03.19).

[11] Scikit-learn – free software machine learning library [Электронный ресурс]: URL:https://scikit-learn.org (дата обращения: 17.03.19).

[12] Hong L., Bekkerman R., Adler J., Davison B. D. Learning to rank social update streams // Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval. 2012. P. 651–660.

[13] Asuncion A., Welling M., Smyth P., Teh Y. W. On smoothing and inference for topic models // Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence. 2009. P. 27–34.