Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский
Академический университет Российской академии наук»
Центр высшего образования

Кафедра математических и информационных технологий

Шугаепов Ильнур Дильбарович

Эффективное обучение кликовых моделей

Магистерская диссертация

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Омельченко А. В.

Научный руководитель:
PhD Марков И. Е.

Рецензент:
к. ф.-м. н., доцент Вяткина К. В.

Санкт-Петербург
2017

SAINT-PETERSBURG ACADEMIC UNIVERSITY
Higher education centre

Department of Mathematics and Information Technology

Ilnur Shugaepov

Efficient learning of click models

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Alexander Omelchenko

Scientific supervisor:
postdoctoral researcher Ilya Markov

Reviewer:
associate professor Kira Vyatkina

Saint-Petersburg
2017

Оглавление

Введение

При разработке и исследованиях в области интернет поиска постоянно появляется необходимость в проведении экспериментов, которые вовлекают в себя пользователей. В ситуациях, когда нет возможности или желания привлекать к экспериментам реальных пользователей, используются *модели пользователей*, которые позволяют симулировать поведение реального пользователя.
Интуитивно модель пользователя можно рассматривать как некоторое множество правил, которое позволяет симулировать его поведения на SERP в виде случайного процесса.
Так как основное взаимодействие пользователя с поисковой системной осуществляется посредствам кликов, то модель пользователя называют *кликовой моделью*. Таким образом, кликовые модели позволяют симулировать клики пользователя на SERP. Также кликовые модели используются для улучшения алгоритмов ранжирования.

Кликовые модели рассматривают поведение пользователя как последовательность наблюдаемых и скрытых случайных событий. Основные случайные события, которые используются при описании кликовых моделей:

- E: пользователь изучил некоторый объект на SERP[1];
- A: пользователь привлечен описанием объекта (например, сниппетом документа);
- C: был выполнен клик по объекту;
- S: пользователь удовлетворен полученным результатом.

Кликовые модели определяют зависимости между случайными событиями и позволяют оценить вероятности соответствующих случайных величин,
например, $P(C = 1 \mid E = 1)$, $P(E = 1)$ и т. д. Как будет видно далее, некоторые вероятности рассматриваются как параметры модели.
Модель можно однозначно задать, например, одним из следующих способов:

- множество случайных величин/событий, которыми оперирует модель;
- ориентированный граф, вершинами которого являются случайные события, и ребра соответствуют зависимостям между событиями - представление в виде графической модели (см. [6]);
- соответствие между параметрами модели и объектами на SERP и пользовательским запросом.

[1]Search Engine Result Page

Перед тем как использовать модель на практике, нам нужно вычислить значения ее параметров. Традиционным подходом к обучению кликовой модели является итеративное вычисление параметров модели EM-алгоритмом по имеющемуся множеству поисковых сессий, которое часто называется *click log*. Процесс обучения модели называют *learning* или *parameters estimation*. Данный подход обладает рядом недостатков. К основным недостаткам можно отнести следующие:

- весь набор параметров модели и *click log* зачастую не помещается целиком в оперативной памяти современного компьютера;
- итеративное вычисление параметров является довольно медленным.

Цель данной работы — разработать эффективный способ обучения кликовых моделей, который свободен от недостатков традиционного подхода.
Под эффективностью (efficiency) обучения в данной работе подразумевается время, необходимое для вычисления параметров модели по имеющемуся логу поисковых сессий.

В рамках данной работы представлено несколько вариантов распределенного обучения кликовых моделей, которые свободны от вышеуказанных недостатков традиционного подхода. Кроме того, в ряде случаев алгоритм обучения был представлен в виде операций над матрицами, что позволяет сделать процесс вычисления параметров еще более эффективным.

Стоит также отметить, что на данный момент нет известных результатов для задачи распределенного обучения кликовых моделей. Таким образом, данная работа - это первый результат в этом направлении.

Кратко о последующих главах

Начнем работу с главы 1, которая есть краткое введение в кликовые модели. Глава содержит описание основных подходов к обучению и проверке качества кликовых моделей. В главе 2 изложен с формальной точки зрения довольно общий MapReduce подход к обучению и вычислению метрик качества кликовых моделей. В главе 3 описан способ представления формул EM-алгоритма для обучения Position-Based Model в векторно-матричной форме, что позволяет сделать процесс обучения в рамках MapReduce подхода еще более эффективным. Глава 4 содержит описание альтернативного MapReduce подхода к распределенному обучению Position-Based Model и User Browsing Model, который при выполнении ряда предположений позволяет значительно снизить коммуникационную сложность распределенного процесса обучения параметров модели. Также в данной главе показано, что нет возможности уменьшить

коммуникационную сложность обучения для Dynamic Bayesian Network Model и Click Chain Model.

Обозначения

Основные обозначения, которые используются в рамках данной работы:

u	Документ.
q	Поисковый запрос пользователя.
r	Ранг документа.
s	Поисковая сессия пользователя.
$\mathcal{S}$	Множество поисковых сессий.
$\mathcal{S}_c$	Множество поисковых сессий, содержащих объект c.
u_r	Документ ранга r.
r_u	Ранг документа u.
n	Максимальный ранг.
X	Событие/случайная величина.
x	Значение случайной величины X.
X_c	Событие связанное с объектом c, например C_u - клик на документе u, или C_r - клик на документе ранга r.
$x_c^{(s)}$	Значение случайной величины X_c в рамках конкретной сессии s.
$\mathcal{I}(\cdot)$	Индикатор.

1. Кликовые модели

Чтобы определить кликовую модель, мы должны описать наблюдаемые и скрытые случайные переменные, зависимости между ними и то, как они зависят от параметров модели.
Рассмотрим несколько примеров кликовых моделей.

1.1. Примеры

Наиболее простая модель, которую можно описать одним параметром - ρ, который есть вероятность клика, называется *Random Click Model*.

$$P(C_u = 1) = \rho,$$

что означает, что любой документ u может быть кликнут пользователем с вероятностью, равной ρ.
Следовательно, как только мы вычислим значение параметра ρ, мы можем генерировать клики пользователя, просто делая выборку из распределения Бернулли, с вероятностью успеха равной ρ.

1.1.1. Position-Based Model

Часто при рассмотрении кликовых моделей пользуются
examination hypothesis

$$C_u = 1 \quad \Longleftrightarrow \quad E_u = 1 \wedge A_u = 1,$$

которая в сущности означает, что пользователь кликает на документ тогда и только тогда, когда он изучил сниппет документа и был привлечен документом.
PBM является самой простой моделью, которая использует *examination hypothesis*.
В данной модели есть параметры аттрактивности (*attractiveness*) документа u при запросе q:

$$P(A_u = 1) = \alpha_{uq},$$

Кроме параметров аттрактивности, в модели есть examination параметры $(\gamma_1, \gamma_2, \ldots, \gamma_n)$, которые задают вероятность того, что пользователь изучит документ определенного ранга.
Формально PBM можно определить следующим образом, как это было сделано в [12]:

$$\begin{aligned} P(C_u = 1) &= P(E_u = 1) \cdot P(A_u = 1) \\ P(A_u = 1) &= \alpha_{uq} \\ P(E_u = 1) &= \gamma_{r_u} \end{aligned}$$

1.1.2. User Browsing Model

Модель, впервые введенная в рассмотрение в [4], является расширением PBM. Основная идея заключается в том, что в examination параметрах добавляют зависимость от ранга предыдущего документа, на котором был выполнен клик. Более формально данную зависимость можно записать следующим образом:

$$P(E_r = 1 \mid C_1 = c_1, \ldots, C_{r-1} = c_{r-1}) = \gamma_{rr'}, \tag{1.1}$$

где $r' = \max\{k \in \{0, \ldots, r-1\} \colon c_k = 1\}$, и для удобства считаем, что $c_0 = 1$.
Можно записать (1.1) в несколько иной форме:

$$P(E_r = 1 \mid \boldsymbol{C}_{<r}) = P(E_r = 1 \mid C_{r'} = 1, C_{r'+1} = 0, \ldots, C_{r-1} = 0) = \gamma_{rr'}.$$

1.1.3. Click Chain Model

CCM (см. [3]) также как и PBM использует *examination hypothesis*. Данная модель позволяет описать поведение пользователя в случаях, когда он бросает сессию, не сделав клик. Кроме того, вероятность перейти к следующему сниппету при условии, что был сделан клик, зависит не только от ранга, но и от аттрактивности.
Строго можно записать модель следующим образом:

$$\begin{aligned}
&P(A_r = 1) = \alpha_{u_r q} \\
&P(E_1 = 1) = 1 \\
&P(E_r = 1 \mid E_{r-1} = 0) = 0 \\
&P(E_r = 1 \mid E_{r-1} = 1, C_{r-1} = 0) = \tau_1 \\
&P(E_r = 1 \mid C_{r-1} = 1) = \tau_2(1 - \alpha_{u_{r-1} q}) + \tau_3 \alpha_{u_{r-1} q},
\end{aligned}$$

где τ_1, τ_2, τ_3 - константы (*continuation parameters*).
Данная модель вводит в рассмотрение новую случайную величину S_r (*satisfaction*) - был ли пользователь удовлетворен результатом, который он получил, кликнув на сниппет документа.

$$\begin{aligned}
&P(S_r = 1 \mid C_r = 0) = 0 \\
&P(S_r = 1 \mid C_r = 1) = \alpha_{u_r q} \\
&P(E_r = 1 \mid C_{r-1} = 1, S_{r-1} = 0) = \tau_2 \\
&P(E_r = 1 \mid C_{r-1} = 1, S_{r-1} = 1) = \tau_3
\end{aligned}$$

1.1.4. Dynamic Bayesian Network Model

В отличии от PBM, в DBN (см. [1]) есть дополнительный набор запросо-зависимых параметров - σ_{uq}, которые называются *actual relevance* и позволяют оценить вероят-

ность того, что пользователь будет доволен документом, который он получил в результате клика на сниппете документа.
Данная модель, так же как и PBM, использует *examination hypothesis*.
Более формально:

$$
\begin{aligned}
&P(A_r = 1) = \alpha_{u_r q} \\
&P(E_1 = 1) = 1 \\
&P(E_r = 1 \mid E_{r-1} = 0) = 0 \\
&P(S_r = 1 \mid C_r = 1) = \sigma_{u_r q} \\
&P(E_r = 1 \mid S_{r-1} = 1) = 0 \\
&P(E_r = 1 \mid E_{r-1} = 1, S_{r-1} = 0) = \gamma,
\end{aligned}
$$

где γ - вероятность того, что пользователь перейдет к изучению следующего сниппета при условии, что он не был доволен предыдущим.

Ознакомиться с большим числом кликовых моделей можно в [2].

1.2. Вычисление параметров

Прежде чем проверять качество моделей, необходимо их *обучить*, то есть вычислить значения параметров модели. Обучение моделей выполняется на основе множества поисковых сессий.
Кликовые модели, рассмотренные в [2], могут быть разделены на два класса на основании того, какие именно случайные переменные используются в модели. В зависимости от класса, в который попадает модель, определяется метод обучения параметров модели.
В *первый* класс попадают модели, случайные переменные которых являются наблюдаемыми. Обучение моделей первого класса выполняется методом максимального правдоподобия (MLE). Кликовые модели *второго* класса имеют одну или более скрытую случайную переменную. В данном случае для обучения используется EM-алгоритм.

Стоит также отметить, что MLE и EM это не единственные методы обучения кликовых моделей. Например, в работе [8] представлен Байесовский подход к обучению моделей, который делает некоторые предположения касательно распределения параметров модели.
Подробное описание методов обучения кликовых моделей можно найти в [2].

Далее будут представлены формулы EM-алгоритма для PBM и UBM.

1.2.1. Position-Based Model

PBM имеет два набора параметров: α_{uq} для каждой пары запрос-документ, и γ_r, который зависит от ранга. В PBM случайные переменные A и E являются скрытыми, поэтому для обучения параметров модели используется EM-алгоритм.

Ниже представлены формулы для итеративного пересчета параметров модели:

$$\alpha_{uq}^{(t+1)} = \frac{1}{|\mathcal{S}_{uq}|} \sum_{s \in \mathcal{S}_{uq}} \left(c_u^{(s)} + (1 - c_u^{(s)}) \frac{(1 - \gamma_r^{(t)}) \alpha_{uq}^{(t)}}{1 - \gamma_r^{(t)} \alpha_{uq}^{(t)}} \right), \tag{1.2}$$

где $\mathcal{S}_{uq} = \{s_q \mid u \in s_q\}$.

$$\gamma_r^{(t+1)} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \left(c_u^{(s)} + (1 - c_u^{(s)}) \frac{(1 - \alpha_{uq}^{(t)}) \gamma_r^{(t)}}{1 - \gamma_r^{(t)} \alpha_{uq}^{(t)}} \right). \tag{1.3}$$

Напомним, что $c_u^{(s)}$ - есть значение случайной величины C_u в рамках сессии s, проще говоря $c_u^{(s)} = \mathcal{I}$(был клик на документе u во время сессии s).

1.2.2. User Browsing Model

Так как UBM по сути есть расширение PBM, то несложно понять, что для обучения UBM также используется EM-алгоритм.
Итеративный пересчет параметров модели:

$$\alpha_{uq}^{(t+1)} = \frac{1}{|\mathcal{S}_{uq}|} \sum_{s \in \mathcal{S}_{uq}} \left(c_u^{(s)} + (1 - c_u^{(s)}) \frac{(1 - \gamma_{rr'}^{(t)}) \alpha_{uq}^{(t)}}{1 - \gamma_{rr'}^{(t)} \alpha_{uq}^{(t)}} \right), \tag{1.4}$$

где $r := r_u^{(s)}$, $r' = \max\{k \in \{0, 1, \ldots, r-1\} \mid c_k^{(s)} = 1\}$.

$$\gamma_{rr'}^{(t+1)} = \frac{1}{|\mathcal{S}_{rr'}|} \sum_{s \in \mathcal{S}_{rr'}} \left(c_u^{(s)} + (1 - c_u^{(s)}) \frac{(1 - \alpha_{uq}^{(t)}) \gamma_{rr'}^{(t)}}{1 - \gamma_{rr'}^{(t)} \alpha_{uq}^{(t)}} \right), \tag{1.5}$$

где $\mathcal{S}_{rr'} = \{s \mid c_{r'}^{(s)} = 1, c_{r'+1}^{(s)} = 0, \ldots, c_{r-1}^{(s)} = 0\}$.

Исчерпывающее описание того, как именно были получены данные формулы и формулы для вычисления параметров CCM и DBN, можно найти в [2].

1.3. Оценка качества

В данном разделе мы рассмотрим два основных подхода к проверке качества обученных кликовых моделей - логарифм правдоподобия (Log-Likelihood) и перплексия (Perplexity).

Стоит отметить, что в зависимости от задачи используются более специфические метрики. Например, в работе [1] описаны метрики *Click-through rate* и *Relevance Prediction* (подходит для CCM, DBN).
Ознакомиться с большим количеством метрик качества моделей и более детальным описанием можно в [2].

1.3.1. Правдоподобие

Рассмотрим кликовую модель M, тогда правдоподобие (см. [5]) модели M для заданной поисковой сессии s есть

$$\mathcal{L}(s \mid M) = P_M\left(C_1 = c_1^{(s)}, \ldots, C_n = c_n^{(s)}\right).$$

Следовательно, правдоподобие M для $\mathcal{S}$ есть $\mathcal{L}(\mathcal{S} \mid M) = \prod\limits_{s \in \mathcal{S}} \mathcal{L}(s \mid M)$. Если мы предположим, что сессии в $\mathcal{S}$ независимы, то мы можем вычислить логарифм правдоподобия:

$$\begin{aligned}\mathcal{LL}(\mathcal{S} \mid M) &= \log \mathcal{L}(\mathcal{S} \mid M) \\ &= \sum_{s \in \mathcal{S}} \log P_M\left(C_1 = c_1^{(s)}, \ldots, C_n = c_n^{(s)}\right) \\ &= \sum_{s \in \mathcal{S}} \sum_{r=1}^{n} \log P_M\left(C_r = c_r^{(s)} \mid \boldsymbol{C}_{<r} = \boldsymbol{c}_{<r}^{(s)}\right). \end{aligned} \tag{1.6}$$

Мы воспользовались тем фактом (см. [13]), что для множества событий $A_1, A_2, \ldots, A_n$ вероятность $P(A_1 A_2 \ldots A_n)$ есть

$$P(A_1 A_2 \ldots A_n) = P(A_1) P(A_2 \mid A_1) \cdot \ldots \cdot P(A_n \mid A_1 \ldots A_{n-1}).$$

Ясно, что $\mathcal{LL}$ всегда не положительное число, и логарифм правдоподобия идеальной модели есть 0. Таким образом, чем ближе к 0 значение $\mathcal{LL}$, тем лучше модель.

1.3.2. Перплексия

В работе [4] впервые было предложено использовать перплексию в качестве метрики качества кликовых моделей.
Рассмотрим кликовую модель M, тогда перплексия ранга r есть:

$$p_r(M) = 2^{-\frac{1}{|\mathcal{S}|} \sum\limits_{s \in \mathcal{S}} \left(c_r^{(s)} \log_2 q_r^{(s)} + (1 - c_r^{(s)}) \log_2 (1 - q_r^{(s)})\right)}, \tag{1.7}$$

где $q_r^{(s)} = P_M(C_r = 1 \mid q, \boldsymbol{u})$, $\boldsymbol{u}$ - список документов на SERP .
Наряду с перплексией ранга рассматривается средняя по рангам перплексия модели:

$$p(M) = \frac{1}{n} \sum_{r=1}^{n} p_r(M). \tag{1.8}$$

Заметим, что перплексия идеальной модели M есть 1. Действительно, идеальная модель будет предсказывать вероятность клика q_r таким образом, что вероятность будет равна значению c_r. Для RCM

$$\forall s \in \mathcal{S}\ \forall r = \overline{1,n}\ (q_r^{(s)} = 0.5) \quad \Longrightarrow \quad (p_r = 2 \wedge p = 2)\,.$$

В рамках данной работы мы не освещаем вопросы, связанные с вычислением вероятностей, участвующих в выражениях (1.6) и (1.7). То, как можно вычислить данные вероятности для конкретных кликовых моделей, можно найти в [2, 12, 4, 3, 1].

2. MapReduce обучение

Так как цель данной работы есть разработать более эффективный подход к обучению кликовых моделей, то вполне естественной выглядит идея сделать процесс обучения распределенным.

При реализации распределенных алгоритмов перед разработчиком встают, в частности, следующие вопросы:

- Каким образом выполнять разбиение исходных данных на части меньшего размера? Или каким образом выполнить разбиение данных, чтобы была возможность независимо решить задачу для каждой части?
- Как выполнить распределение задач между узлами кластера с учетом характеристик узлов?
- Как обеспечить то, что каждый узел получит необходимую ему часть исходных данных?
- Каким образом обеспечить синхронизацию узлов кластера?
- Каким образом один узел может знать о результатах, полученных на другом узле, в случае необходимости?

При традиционных подходах (OpenMP[2], MPI[3]) к организации параллельных/распределенных вычислений, разработчику приходится в явном виде решать данные вопросы.
Одним из наиболее значимых преимуществ MapReduce является то, что MapReduce предоставляет дополнительный уровень абстракции, который скрывает от разработчика многие детали и позволяет сконцентрироваться над решением задачи, а не над попытками ответить на перечисленные выше вопросы. При использовании MapReduce, нам явным образом необходимо ответить только на вопросы из первого пункта. Цель данного раздела - дать ответ на эти вопросы.
Подробное описание того, что из себя представляет MapReduce, можно найти в [7, 10].

Так как кликовые модели, содержащие скрытые случайные переменные, позволяют с большей точностью описать поведение реального пользователя на SERP, то в рамках данной работы мы рассматриваем EM подход к обучению моделей. EM подход к обучению является гораздо более общим и используется для обучения большего числа моделей по сравнению с MLE.

[2]http://www.openmp.org/
[3]http://www.mcs.anl.gov/mpi/

В [7] изложен довольно общий подход к представлению EM-алгоритма в рамках MapReduce парадигмы вычислений.

2.1. Вычисление параметров

Заметим, что для любой из рассмотренных в разделе 1 кликовых моделей верно утверждение о том, что весь набор параметров модели можно поделить на два непересекающихся поднабора на основании того, зависит параметр от запроса или нет. Формально введем общее понятие кликовой модели:

Определение 2.1. *Кликовая модель M (обученная на множестве сессий $\mathcal{S}$ алгоритмом обучения $\mathcal{A}$) - есть в общем случае пара $\langle Q, I\rangle$, где Q - набор запросо-зависимых параметров, I - набор запросо-независимых параметров.*

Так как в данной работе под алгоритмом $\mathcal{A}$ подразумевается EM-алгоритм, который является итерационным, то введем в рассмотрение следующую нотацию:

Определение 2.2. *Через $M^{(t)} := \langle Q^{(t)}, I^{(t)}\rangle$ будем обозначать кликовую модель, полученную после t итераций алгоритма $\mathcal{A}$ на данных $\mathcal{S}$.*

Зададимся теперь вопросом о том, каким образом можно сделать кликовую модель M распределенной. Для начала предположим, что под распределенной кликовой моделью мы будем понимать некоторое разбиение набора параметров, например, между узлами кластера, которое определяется тем, каким образом было выполнено распределение данных из $\mathcal{S}$ между узлами кластера. То есть можно Q представить в виде $Q_1 \cup Q_2 \cup \ldots \cup Q_K$, и $I = I_1 \cup I_2 \cup \ldots \cup I_K$, где K - количество узлов в кластере.
В общем случае можно говорить о том, что каким бы ни было разбиение Q параметров между узлами кластера, перед началом очередной итерации EM-алгоритма, параметры I должны быть на каждом узле кластера для обеспечения корректного обновления Q параметров. Кроме того, разбиение набора параметров должно быть таким, чтобы после того, как из $I_i^{(t)}$ будет получено I_i', мы должны быть способны из набора $(I_1', I_2', \ldots, I_K')$ получить $I^{(t+1)}$ способом, не противоречащим EM-алгоритму.
Более формально:

Определение 2.3. *Пусть $M = \langle Q, I\rangle$ - кликовая модель, и разбиение*

$$Q = Q_1 \cup Q_2 \cup \ldots \cup Q_K$$

порождается разбиением $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \ldots \cup \mathcal{S}_K$ таким, что построение $Q_i^{(t+1)}$ алгоритмом обучения $\mathcal{A}$ зависит только от $(\mathcal{S}_i, Q_i^{(t)}, I^{(t)})$, тогда разбиение $\mathcal{S}$ будем называть допустимым.

Таким образом, при допустимом разбиении $\mathcal{S}$, обновление запросо-зависимых параметров модели при переходе к следующей итерации EM-алгоритма, требует загрузки только запросо-независимыми параметров модели.

Зададимся теперь вопросом о поиске допустимого разбиения $\mathcal{S}$.

Теорема 2.1. *Пусть $\mathcal{Q} = \{q_1, q_2, \ldots, q_l\}$ - есть множество уникальных поисковых запросов из множества $\mathcal{S}$. Тогда разбиение $\mathcal{S} = \mathcal{S}_{q_1} \cup \mathcal{S}_{q_2} \cup \ldots \cup \mathcal{S}_{q_l}$ является допустимым.*

Доказательство. Если считать, что любой параметр из Q зависит только от одного запроса из множества $\mathcal{Q}$, то мы можем рассмотреть разбиение

$$Q = Q_{q_1} \cup Q_{q_2} \cup \ldots \cup Q_{q_l},$$

где Q_q - множество параметров, зависящих от q. Так как алгоритм $\mathcal{A}$ является итеративным, то обновление Q_{q_i} зависит только от Q_{q_i}, S_{q_i} и I. Следовательно, рассмотренное разбиение является допустимым. □

Представление множества $\mathcal{S}$ в виде разбиения $\mathcal{S}_{q_1}, \mathcal{S}_{q_2}, \ldots, \mathcal{S}_{q_l}$ мы будем называть *агрегацией по запросу.*

Введем в рассмотрение понятие распределенной кликовой модели, порожденной допустимым разбиением множества $\mathcal{S}$:

Определение 2.4. *Распределенной кликовой моделью будем называть вектор $\mathcal{M} = (M_{q_1}, M_{q_2}, \ldots, M_{q_l})$, где $M_{q_i} = \langle Q_{q_i}, I_{q_i} \rangle$ - есть кликовая модель, обученная на $\mathcal{S}_{q_i}$ алгоритмом $\mathcal{A}$. Кроме того, $\forall i, j \ (i \neq j \Longrightarrow Q_{q_i} \cap Q_{q_j} = \emptyset)$.*

Очевидно, что если кликовые модели M и $\mathcal{M}$ были обучены на одном и том же множестве $\mathcal{S}$, то мы требуем эквивалентности данных моделей, где под эквивалентностью понимается следующее:

Определение 2.5. *Модели M, M' эквивалентны на $\mathcal{S}$:*

$$M \equiv_{\mathcal{S}} M' \quad \Longleftrightarrow \quad \mathcal{LL}(\mathcal{S} \mid M) = \mathcal{LL}(\mathcal{S} \mid M') \wedge p(M) = p(M').$$

Далее будем писать просто $\equiv$, если понятно, о каком множестве идет речь.

Следствие 2.1. $M \equiv_{\mathcal{S}} \mathcal{M} \quad \Longrightarrow \quad \forall q \in \mathcal{Q} \ (M_q \equiv_{\mathcal{S}_q} M)$.

После того, как мы определили, что из себя представляет распределенная кликовая модель, мы можем перейти к вопросу о том, каким образом можно представить EM-алгоритм $\mathcal{A}$ для обучения параметров в виде последовательности MapReduce фаз.

Определение 2.6. *Через $\mathcal{A}_{MR}$ будем обозначать распределенную версию алгоритма $\mathcal{A}$ такую, что*

$$\forall t \ (\mathcal{M}^{(t)} \equiv M^{(t)}), \tag{2.1}$$

где $M^{(t)}$ - есть модель, полученная после t-ой итерации алгоритма $\mathcal{A}$, и $\mathcal{M}^{(t)}$ получена после t-ой итерации алгоритма $\mathcal{A}_{MR}$.

Представленные выше рассуждения позволяют нам сформулировать общий вид итерации распределенного EM-алгоритма обучения параметров кликовой модели:

Алгоритм 2.1 Общая схема $(t+1)$-ой итерации алгоритма $\mathcal{A}_{MR}$

1: function $\mathcal{A}_{MR}$ ITERATION$(\mathcal{M}^{(t)}, \mathcal{S}_{q_1}, \mathcal{S}_{q_2}, \ldots, \mathcal{S}_{q_l})$

2: Fit(Map)

$$(M_q^{(t)}, \mathcal{S}_q) \quad \underset{\mathcal{A}}{\rightarrow} \quad M_q'^{(t+1)}$$

Выполняется обучение кликовой модели $M_q^{(t)}$, полученной на предыдущей итерации алгоритма $\mathcal{A}_{MR}$, с помощью алгоритма $\mathcal{A}$ на данных $\mathcal{S}_q$.

3: Extract(Map)

$$M_q'^{(t+1)} \quad \rightarrow \quad I_q'^{(t+1)}$$

Из модели, обученной во время фазы Fit, происходит извлечение запросо-независимых параметров

4: Merge(Reduce)

$$(I_{q_1}'^{(t+1)}, I_{q_2}'^{(t+1)}, \ldots, I_{q_l}'^{(t+1)}) \quad \underset{C}{\rightarrow} \quad I^{(t+1)}$$

Из запросо-независимых параметров, извлеченных во время фазы Extract, под действием оператора композиции C, строится результирующий набор запросо-независимых параметров.

5: Update(Map)

$$(M_q'^{(t+1)}, I^{(t+1)}) \quad \rightarrow \quad \langle Q_q^{(t+1)}, I^{(t+1)} \rangle =: M_q^{(t+1)}$$

Выполняется замена запросо-независимых параметров в кликовой модели.

Теорема 2.2. *Алгоритм 2.1 обеспечивает выполнение условия (2.1).*

Доказательство. Докажем утверждение по индукции.
База: $t = 0$ — утверждение очевидно.
Переход: $t \rightarrow t+1$ — не трудно понять, что после выполненных преобразований $\mathcal{M}^{(t+1)} \equiv_{\mathcal{S}} M^{(t+1)}$, так как разбиение $\mathcal{S}$ является допустимым. □

Давайте рассмотрим некоторые детали алгоритма, на примере Position-Based Model.

Ясно, что для Position-Based модели M, набор запросо-зависимых параметров $Q = (\alpha_{uq})_{q \in \mathcal{Q}, u \in \mathcal{S}_q}$ и набор запросо-независимых параметров $I = (\gamma_1, \gamma_2, \ldots, \gamma_n)$.
Рассмотрим $\mathcal{A}_{MR}$ алгоритм для PBM (алгоритм $\mathcal{A}$ для рассматриваемой модели описан в разделе 1.2.1).

Единственным моментом, требующим уточнения, является вид оператора композиции C, который используется во время фазы Merge алгоритма 2.1.
Во время Extract фазы, на узле кластера происходит построение

$$I'^{(t+1)}_q = \begin{pmatrix} \gamma^{(t+1)}_{1,q} & \gamma^{(t+1)}_{2,q} & \cdots & \gamma^{(t+1)}_{n,q} \end{pmatrix}.$$

Таким образом, после фазы Extract мы имеем $\begin{pmatrix} I'^{(t+1)}_{q_1} & I'^{(t+1)}_{q_2} & \cdots & I'^{(t+1)}_{q_l} \end{pmatrix}$. Заметим, что, согласно формуле (1.3), $\gamma^{(t+1)}_r \neq \sum\limits_{i=1}^{l} \gamma^{(t+1)}_{r,q_i}$. Это означает, что оператор C - есть нечто более сложное, чем просто $I'^{(t+1)}_{q_1} + I'^{(t+1)}_{q_2} + \ldots + I'^{(t+1)}_{q_l}$.
Для решения данной проблемы, представим $\gamma^{(t+1)}_r$ в виде пары $(x^{(t+1)}_r, y^{(t+1)}_r)$ такой, что $\gamma^{(t+1)}_r = x^{(t+1)}_r / y^{(t+1)}_r$. Например:

$$x^{(t+1)}_r = \sum_{s \in \mathcal{S}} \left(c^{(s)}_u + (1 - c^{(s)}_u) \frac{(1 - \alpha^{(t+1)}_{uq}) \gamma^{(t+1)}_r}{1 - \alpha^{(t+1)}_{uq} \gamma^{(t+1)}_r} \right),$$
$$y^{(t+1)}_r = |\mathcal{S}|.$$

Тогда не трудно понять, что

$$\gamma^{(t+1)}_r = \frac{\sum\limits_{i=1}^{l} x^{(t+1)}_{r,q_i}}{\sum\limits_{i=1}^{l} y^{(t+1)}_{r,q_i}}.$$

В заключении данного раздела хочется отметить, что описанный выше MapReduce подход к обучению кликовых моделей подходит для обучения PBM, UBM, DBN и CCM. Действительно, если посмотреть на формулы EM-алгоритма данных моделей (см. [2]), то становится ясно, что оператор композиции C имеет для всех рассмотренных моделей один и тот же вид. То есть запросо-независимые параметры следует хранить в виде пары (числитель, знаменатель).

2.2. Оценка качества

Используя описанный выше подход, мы можем распределенно обучать параметры кликовых моделей. Но нам также интересно выполнять сравнение качества обученных моделей. В данном разделе мы покажем, каким образом метрики из раздела 1.3 могут быть вычислены в рамках MapReduce парадигмы вычислений.

2.2.1. Правдоподобие

Заметим, что для того, чтобы вычислить (1.6) в рамках MR модели, достаточно выполнить Map и Reduce фазы при условии, что модели для запросов уже обучены.

Таким образом, (1.6) можно переписать следующим образом:

$$\begin{aligned}\mathcal{LL}(\mathcal{S} \mid \mathcal{M}) &= \sum_{q \in \mathcal{Q}} \sum_{s \in \mathcal{S}_q} \log P_{M_q}\left(C_1 = c_1^{(s)}, \ldots, C_n = c_n^{(s)}\right) \\ &= \sum_{q \in \mathcal{Q}} \mathcal{LL}(\mathcal{S}_q \mid M_q). \end{aligned} \tag{2.2}$$

Данная группировка слагаемых позволяет выполнить вычисление следующим образом:

Алгоритм 2.2 MapReduce вычисление логарифма правдоподобия

1: function $\mathcal{LL}_{MR}(\mathcal{M}^{(t)}, \mathcal{S}_{q_1}, \mathcal{S}_{q_2}, \ldots, \mathcal{S}_{q_l})$

2: Map

$$(M_q, \mathcal{S}_q) \quad \to \quad \mathcal{LL}(\mathcal{S}_q \mid M_q)$$

3: Reduce

$$(\mathcal{LL}(\mathcal{S}_{q_1} \mid M_{q_1}), \ldots, \mathcal{LL}(\mathcal{S}_{q_l} \mid M_{q_l})) \quad \xrightarrow[(2.2)]{} \quad \mathcal{LL}(\mathcal{S} \mid \mathcal{M})$$

2.2.2. Перплексия

Вычисление значения перплексии требует от нас несколько больших усилий, чем вычисление логарифма правдоподобия.

Нас интересует значение $p(\mathcal{M})$, которое ввиду того, что $\mathcal{M} \equiv_{\mathcal{S}} M$, равно $p(M)$, однако очевидно, что $\sum\limits_{q \in \mathcal{Q}} p(M_q) \neq p(M)$. Тем не менее вычисление $p(\mathcal{M})$ легко представимо в виде MapReduce алгоритма:

Алгоритм 2.3 MapReduce вычисление перплексии

1: function $p_{MR}(\mathcal{M}^{(t)}, \mathcal{S}_{q_1}, \mathcal{S}_{q_2}, \ldots, \mathcal{S}_{q_l})$

2: Map

$$(M_q, \mathcal{S}_q) \quad \to \quad (|\mathcal{S}_q|, \boldsymbol{p}_q),$$

где $\boldsymbol{p}_q := (\log p_1(M_q) \cdot |\mathcal{S}_q|, \ldots, \log p_n(M_q) \cdot |\mathcal{S}_q|)$.

3: Reduce

$$((|\mathcal{S}_{q_1}|, \boldsymbol{p}_{q_1}), (|\mathcal{S}_{q_2}|, \boldsymbol{p}_{q_2}), \ldots, (|\mathcal{S}_{q_l}|, \boldsymbol{p}_{q_l})) \quad \to \quad p(\mathcal{M})$$

Нетрудно понять, что

$$p_r(\mathcal{M}) = 2^{\sum_{i=1}^{l} (\boldsymbol{p}_{q_i})_r / \sum_{i=1}^{l} |\mathcal{S}_{q_i}|}.$$

Далее достаточно воспользоваться формулой (1.8), чтобы получить $p(\mathcal{M})$.

2.3. Реализация

Идеи, представленные в данной главе, были реализованы с помощью Hadoop (см. [10]) в рамках библиотеки PyClick[4]. С исходным кодом реализации можно ознакомиться по ссылке `https://bitbucket.org/Sh_Nur/pyclick2/branch/hadoop`.

Стоит отметить, что при реализации на Hadoop, фаза Fit алгоритма 2.1 не является просто Map фазой. Проблема в том, что мы хотим выполнять некоторые действия над парой объектов $(M_q, \mathcal{S}_q)$, а не над одним объектом. Для того, чтобы иметь возможность обрабатывать пару объектов, необходимо выполнить сначала Map фазу, во время которой выполняется отображение $M_q \mapsto (q, M_q)$, $\mathcal{S}_q \mapsto (q, \mathcal{S}_q)$, а затем Reduce фазу, в рамках которой будет обработан набор $(q, (M_q, \mathcal{S}_q))$. Таким образом, фаза Fit представляет из себя две MapReduce фазы.
Далее фазы, подобные Fit, мы будем обозначать как Map & Reduce. В качестве примеров подобных фаз можно рассмотреть первую фазу вычисления перплексии (алгоритм 2.3) и логарифма правдоподобия (алгоритм 2.2).

Для моделей PBM,UBM,CCM,DBN, корректность алгоритма 2.1 была проверена на первом миллионе строк из лога *Yandex Relevance Prediction Challeng*(см. [11]): было выполнено сравнение значений метрик качества, посчитанных MapReduce и итеративным алгоритмами.

Формат данных в логе отличается от того представления, которое нужно для работы алгоритма 2.1. Поэтому на этапе предобработки данных мы из множества $\mathcal{S}$ строим разбиение $\mathcal{S}_{q_1}, \mathcal{S}_{q_2}, \ldots, \mathcal{S}_{q_l}$, то есть выполняем *агрегацию по запросу.*

[4]https://bitbucket.org/markil/pyclick2

3. Векторно-матричное представление Position-Based Model

В предыдущем разделе мы показали, что обучение кликовых моделей может быть представлено в рамках MapReduce парадигмы вычислений. Несмотря на то, что процесс вычисления параметров кликовой модели можно сделать распределенным, у нас есть возможность сделать процесс обучения более эффективным, если проводить вычисления в векторно-матричном виде в рамках фазы Fit алгоритма 2.1 на каждом узле кластера.
Большая эффективность процесса обучения достигается за счет того, что вычисления в векторно-матричной форме, при использовании GPU, выполняются значительно быстрее, чем при стандартном подходе и использовании CPU.

Таким образом, наша задача состоит в том, чтобы представить формулы EM-алгоритма для обновления параметров в виде операций над матрицами.

3.1. Частный случай

Заметим, что в формуле (1.2) ранг рассматриваемого документа u есть $r := r_u^{(s)}$, то есть в общем случае $\exists s, s' \in \mathcal{S}_{uq}(s \neq s' \wedge r_u^{(s)} \neq r_u^{(s')})$.
Предположим, что

1. Алгоритм ранжирования документов является детерминированным;
2. Между поисковыми сессиями в базе данных не появляется новых документов.

Более формально результат выдвинутого предположения можно сформулировать следующим образом:

Предположение 1**.**

$$\forall q \in \mathcal{Q}\, \forall r = \overline{1,n}\, \forall s, s' \in \mathcal{S}_q \left(u_r^{(s)} = u_r^{(s')}\right).$$

То есть пользователь по запросу q всегда будет получать в качестве результата один и тот же список документов в определенном порядке.

Следствием предположения 1 является то, что $\forall s, s' \in \mathcal{S}_{uq}\ (r_u^{(s)} = r_u^{(s')})$.

Замечание 3.1**.** *Выдвинутое предположение не выполняется лишь для 15% уникальных запросов в Yandex Relevance Prediction Challenge (см. [11]).*

Пусть $u_1, u_2, \ldots, u_m$ - множество документов, которые участвуют в сессиях из $\mathcal{S}_q$, то есть $\mathcal{S}_q = \mathcal{S}_{u_1 q} \cup \mathcal{S}_{u_2 q} \cup \ldots \cup \mathcal{S}_{u_m q}$.

Attractiveness Сделанное допущение позволяет нам записать формулу (1.2) в следующем виде:

$$\alpha_{uq}^{(t+1)} = \frac{1}{|\mathcal{S}_{uq}|}\left(c_u^q + (|\mathcal{S}_{uq}| - c_u^q)\frac{(1-\gamma_r^{(t)})\alpha_{uq}^{(t)}}{1-\gamma_r^{(t)}\alpha_{uq}^{(t)}}\right), \tag{3.1}$$

где $c_u^q = \sum\limits_{s\in\mathcal{S}_{uq}} c_u^{(s)}$ - общее количество кликов на документе u, в рамках поисковых сессий инициированных запросом q.
Действительно, ввиду того, что теперь вне зависимости от сессии $s \in \mathcal{S}_q$ документ u имеет всегда один и тот же ранг r, слагаемое $(1-\gamma_{r_u^{(s)}})\alpha_{uq}/(1-\gamma_{r_u^{(s)}}\alpha_{uq})$ более не зависит от s, а зависит лишь от пары (u,q). Данное наблюдение позволяет нам выполнить группировку слагаемых в сумме (1.2).

Далее мы будем опускать индекс (t) для упрощения записей, если в нем нет необходимости.

Определение 3.1. *Рассмотрим тройку $(\mathbb{R}^m, \circ, +)$, в котором операция $\circ$ - есть поэлементное умножение векторов, а $+$ - поэлементное сложение. Таким образом, единица данной тройки есть $\mathbf{1} = \begin{pmatrix}1 & 1 & \cdots & 1\end{pmatrix}$.*
Пусть $\boldsymbol{a} \in \mathbb{R}^m$ - есть вектор без нулевых компонент, то через $\boldsymbol{a}^{-1}$ будем обозначать обратный элемент, в котором каждый элемент вектора $\boldsymbol{a}$ заменен на обратный к нему.
Для векторов из $\mathbb{R}^m$ определим операцию умножения на скаляр из $\mathbb{R}$ - элементы исходного вектора домножаются на скаляр.

Покажем теперь, как можно перейти от обновления α_{uq} к обновлению вектора $(\alpha_{u_1q}, \alpha_{u_2q}, \ldots, \alpha_{u_mq})$.

Лемма 3.1. *Рассмотрим вектора:* $\boldsymbol{\alpha}_q = \begin{pmatrix}\alpha_{u_1q} & \alpha_{u_2q} & \cdots & \alpha_{u_mq}\end{pmatrix}$,
$\boldsymbol{c}^q = \begin{pmatrix}c_{u_1}^q & c_{u_2}^q & \cdots & c_{u_m}^q\end{pmatrix}$, $\boldsymbol{\gamma}_q = \begin{pmatrix}\gamma_{r_{u_1}} & \gamma_{r_{u_2}} & \cdots & \gamma_{r_{u_m}}\end{pmatrix}$, $\boldsymbol{S}_q = \begin{pmatrix}|\mathcal{S}_{u_1q}| & |\mathcal{S}_{u_2q}| & \cdots & |\mathcal{S}_{u_mq}|\end{pmatrix}$.

$$\boldsymbol{\alpha}_q \leftarrow \left((\mathbf{1}-\boldsymbol{\gamma}_q)\circ\boldsymbol{\alpha}_q\circ(\mathbf{1}-\boldsymbol{\gamma}_q\circ\boldsymbol{\alpha}_q)^{-1}\circ(\boldsymbol{S}_q-\boldsymbol{c}^q)+\boldsymbol{c}^q\right)\circ\boldsymbol{S}_q^{-1}. \tag{3.2}$$

Доказательство. Для того, чтобы убедиться в том, что (3.2) есть то же, что и (3.1), но сразу для всех документов $u_1, u_2, \ldots, u_m$, достаточно расписать формулу. □

Замечание 3.2. *Заметим, что вектора $\boldsymbol{c}^q$, $\boldsymbol{S}_q$ не зависят от t, то есть не изменяются при переходе к следующей итерации EM-алгоритма. Следовательно, их можно посчитать один раз и использовать на последующих итерациях.*

Вектор $\boldsymbol{\gamma}_q := \boldsymbol{\gamma}_q^{(t)}$ можно получить следующим образом:

$$\begin{pmatrix} \mathbf{1}_{r_{u_1}} \\ \mathbf{1}_{r_{u_2}} \\ \vdots \\ \mathbf{1}_{r_{u_m}} \end{pmatrix} \cdot \begin{pmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_n \end{pmatrix}^T = \boldsymbol{\gamma}_q,$$

где $\mathbf{1}_i$ - есть единичный вектор-строка с единицей на позиции i. Заметим, что первый множитель не зависит от t, следовательно, может быть построен лишь один раз и использован на последующих итерациях алгоритма.

Examination Еще раз обратим внимание на то, что вычисления выполняются в рамках узла кластера, то есть в соответствии с алгоритмом 2.1, узел располагает информацией о множестве $\mathcal{S}_q$, то есть в формуле (1.3) вместо $\mathcal{S}$ будет $\mathcal{S}_q$.
Полезным следствием предположения 1 является то,
что $\forall r = \overline{1,n}\ \forall s, s' \in \mathcal{S}_q\ (u_r^{(s)} = u_r^{(s')})$ и $m \equiv n$. Данное следствие, так же как и в случае для параметров аттрактивности, позволяет нам записать (1.3) следующим образом:

$$\gamma_r^{(t+1)} = \frac{1}{|\mathcal{S}_q|}\left(c_u^q + (|\mathcal{S}_q| - c_u^q)\frac{(1-\alpha_{uq}^{(t)})\gamma_r^{(t)}}{1-\alpha_{uq}^{(t)}\gamma_r^{(t)}}\right), \tag{3.3}$$

где $u := u_r$ и $c_u^q = \sum\limits_{s\in\mathcal{S}_q} c_u^{(s)}$.
Мы можем сформулировать следующую лемму:

Лемма 3.2. *Так как* $\forall s \in \mathcal{S}_q(SERP_q = u_1\ u_2\ \ldots\ u_n)$ *и* $u_1, u_2, \ldots, u_n = u_1, u_2, \ldots, u_m$, *то есть*

$$\mathcal{S}_{u_1,q} = \mathcal{S}_{u_2,q} = \ldots = \mathcal{S}_{u_n,q} \equiv \mathcal{S}_q. \tag{3.4}$$

То мы можем воспользоваться векторами, определенными в лемме 3.1, и получим

$$\boldsymbol{\gamma} \leftarrow \left(\boldsymbol{c}^q + (\boldsymbol{S}_q - \boldsymbol{c}^q) \circ (\mathbf{1} - \boldsymbol{\alpha}_q) \circ \boldsymbol{\gamma} \circ (\mathbf{1} - \boldsymbol{\alpha}_q \circ \boldsymbol{\gamma})^{-1}\right) \circ \boldsymbol{S}_q^{-1}. \tag{3.5}$$

Доказательство. Очевидно. □

Стоит также отметить, что ввиду (3.4), вектор $\boldsymbol{\gamma}_q$ из леммы 3.1 есть не что иное, как $\boldsymbol{\gamma}$.

3.2. Общий случай

Рассмотрим векторно-матричное представление формул (1.2), (1.3) в случае, когда предположение 1 не выполняется.

Attractiveness В общем случае, когда предположение 1 не верно,
слагаемое $(1-\gamma_r)\alpha_{uq}/(1-\gamma_r\alpha_{uq})$ суммы (1.2) в явном виде зависит от сессии, так как $r := r_u^{(s)}$, что не позволяет выполнить такую же компактную группировку слагаемых, как в (3.1).
Но мы знаем, что $r_u^{(s)}$ может принимать одно из n значений.
Рассмотрим разбиение

$$\mathcal{S}_{uq} = \mathcal{S}_{uq}^{(1)} \cup \mathcal{S}_{uq}^{(2)} \cup \ldots \cup \mathcal{S}_{uq}^{(n)}, \text{ где } \mathcal{S}_{uq}^{(i)} \colon \forall s \in \mathcal{S}_{uq}^{(i)}\ (r_u^{(s)} = i).$$

Тогда в (1.2) можно сгруппировать слагаемые и получить:

$$\begin{aligned}\alpha_{uq} &\leftarrow \frac{1}{|\mathcal{S}_{uq}|}\sum_{r=1}^{n}\sum_{s\in\mathcal{S}_{uq}^{(r)}}\left(c_u^{(s)} + (1-c_u^{(s)})\frac{(1-\gamma_r)\alpha_{uq}}{1-\gamma_r\alpha_{uq}}\right)\\ &= \frac{1}{|\mathcal{S}_{uq}|}\sum_{r=1}^{n}\left(c_{uq}^{r} + (|\mathcal{S}_{uq}^{(r)}| - c_{uq}^{r})\frac{(1-\gamma_r)\alpha_{uq}}{1-\gamma_r\alpha_{uq}}\right),\end{aligned} \tag{3.6}$$

где $c_{uq}^r = \sum\limits_{s\in\mathcal{S}_{uq}^{(r)}} c_u^{(s)}$.
Заметим, что если $\exists! r \colon \forall s \in \mathcal{S}_{uq}\ (r_u^{(s)} = r)$, то формула (3.6) эквивалента формуле (3.1), полученной для частного случая.

Воспользуемся полученной формулой применительно к вычислению параметров PBM в MapReduce модели вычислений.
Так же как и при рассмотрении частного случая, нас интересует возможность выделить в рассматриваемых выражениях части, которые не изменяются от итерации к итерации.
Рассмотрим матрицы $\boldsymbol{C} = \left(c^i_{u_jq}\right)^{n,m}_{i,j=1,1}$ и $\boldsymbol{S} = \left(|\mathcal{S}^{(i)}_{u_jq}|\right)^{n,m}_{i,j=1,1}$. Так же как и в частном случае, нам хочется перейти от обновления элемента α_{uq} к обновлению вектора $\boldsymbol{\alpha}_q$. Это можно сделать следующим образом:

$$\boldsymbol{\alpha}_q \leftarrow \boldsymbol{S}_q^{-1} \circ \left(\sum_{r=1}^{n}\left(\boldsymbol{C}_r + (\boldsymbol{S}_r - \boldsymbol{C}_r) \circ ((1-\gamma_r)\boldsymbol{\alpha}_q) \circ (\boldsymbol{1} - \gamma_r\boldsymbol{\alpha}_q)^{-1}\right)\right), \tag{3.7}$$

где $\boldsymbol{C}_r, \boldsymbol{S}_r$ - есть r-ая строка матрицы $\boldsymbol{C}$ и $\boldsymbol{S}$ соответственно.
Не составляет труда убедиться в справедливости полученной формулы.
Ввиду того, что матрицы $\boldsymbol{C}$ и $\boldsymbol{S}$ не изменяются во время работы EM-алгоритма, их достаточно построить лишь один раз.

Examination Заметим, что в формуле (1.3) $u := u_r^{(s)}$, то есть в зависимости от рассматриваемой сессии на позиции r могут быть разные документы.
Пусть $\mathcal{S}_q = \{s_i\}_{i=1}^K$, где $s_i = \begin{pmatrix}(u_1,c_1) & (u_2,c_2) & \cdots & (u_n,c_n)\end{pmatrix}^T$. Рассмотрим матрицу $\boldsymbol{SC} = \begin{pmatrix}s_1 & s_2 & \cdots & s_K\end{pmatrix}$ - данная матрица хранит в себе информацию обо всех сесси-

ях.

Замечание 3.3. *На практике в каждой строке матрицы* $\boldsymbol{SC}$ *есть лишь небольшое количество различных документов. Так, например, при условии, что выполнено предположение 1, в* r*-ой строке матрицы хранится информация только о документе -* u_r.

Определение 3.2. *Пусть* $u(r) = \{v_1^r, v_2^r, \ldots, v_{k_r}^r\}$ *- множество уникальных документов в строке* $\boldsymbol{SC}_r$, *и* $k_r := |u(r)| = \mathcal{O}(1)$.

Пользуясь определением, можно $\mathcal{S}_q$, при фиксированном r, представить в следующем виде:

$$\mathcal{S}_q = \mathcal{S}_q^{v_1^r} \cup \mathcal{S}_q^{v_2^r} \cup \ldots \cup \mathcal{S}_q^{v_{k_r}^r},$$

где $\mathcal{S}_q^v = \{s \in \mathcal{S}_q \mid u_r^{(s)} = v\}$.
Полученное разбиение позволяет сгруппировать слагаемые в (1.3):

$$\begin{aligned}\gamma_r &\leftarrow \frac{1}{|\mathcal{S}_q|} \sum_{v \in u(r)} \sum_{s \in \mathcal{S}_q^v} \left(c_v^{(s)} + (1 - c_v^{(s)}) \frac{(1 - \alpha_{vq})\gamma_r}{1 - \alpha_{vq}\gamma_r} \right) \\ &= \frac{1}{|\mathcal{S}_q|} \sum_{v \in u(r)} \left(c_v + (|\mathcal{S}_q^v| - c_v) \frac{(1 - \alpha_{vq})\gamma_r}{1 - \alpha_{vq}\gamma_r} \right),\end{aligned}$$

где $c_v = \sum\limits_{s \in \mathcal{S}_q^v} c_v^{(s)}$.
Так как вектор $\begin{pmatrix} c_{v_1^r} & c_{v_2^r} & \cdots & c_{v_{k_r}^r} \end{pmatrix}$ не зависит от t, то его можно посчитать лишь однажды.
Таким образом, мы можем *итеративно* обновить вектор $\boldsymbol{\gamma}$ за время $\mathcal{O}(n)$.

Для упрощения дальнейших записей положим, что $v_1^r, v_2^r, \ldots, v_{k_r}^r = v_1, v_2, \ldots, v_k$.
Для представления предыдущего результата в векторно-матричной форме нам потребуются следующие вектора: $\boldsymbol{C}_r = \begin{pmatrix} c_{v_1} & c_{v_2} & \cdots & c_{v_k} \end{pmatrix}$, $\boldsymbol{A}_r = \begin{pmatrix} \alpha_{v_1 q} & \alpha_{v_2 q} & \cdots & \alpha_{v_k q} \end{pmatrix}$, $\hat{\boldsymbol{C}}_r = \begin{pmatrix} |\mathcal{S}_q^{v_1}| - c_{v_1} & |\mathcal{S}_q^{v_2}| - c_{v_2} & \cdots & |\mathcal{S}_q^{v_k}| - c_{v_k} \end{pmatrix}$.
Тогда получаем:

$$\gamma_r \leftarrow \frac{1}{|\mathcal{S}_q|} \left(\langle \boldsymbol{C}_r, \mathbf{1} \rangle + \gamma_r \langle \hat{\boldsymbol{C}}_r, (\mathbf{1} - \boldsymbol{A}_r) \circ (\mathbf{1} - \boldsymbol{A}_r \gamma_r)^{-1} \rangle \right),$$

где $\langle \cdot, \cdot \rangle$ - есть скалярное произведение векторов.
Как было отмечено ранее, вектор $\boldsymbol{C}_r$ не зависит от t, следовательно, и вектор $\hat{\boldsymbol{C}}_r$ не зависит от t. Данные вектора могут быть построены по вектору $\boldsymbol{SC}_r$. Вектор $\boldsymbol{A}_r$, в свою очередь, явно зависит от t, и есть не что иное, как $\boldsymbol{\alpha}_q$ из (3.7).

Можно также отметить, что так как обновления γ_r и $\gamma_{r'}$ независимы для $r \neq r'$, следовательно, их можно выполнять *параллельно*.

В заключении хочется отметить, что векторизация EM-алгоритма в общем случае имеет большую пространственную сложность по сравнению с частным случаем.

4. Агрегация по рангу

В рамках глав 2 и 3 мы показали, что процесс обучения кликовых моделей можно эффективно сделать распределенным при использовании *агрегации по запросу*. В данном разделе мы рассмотрим альтернативный способ представления информации о сессиях, который мы будем называть *агрегацией по рангу*, и покажем какие преимущества нам дает данный способ.

На примере PBM (см. раздел 1.2.1) видно, что если на узле кластера оказывается информация о всей сессии $s \in \mathcal{S}$ целиком, то обновление запросо-зависимых параметров, связанных с документами из данной сессии, требует наличия на узле кластера вектора $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ - это можно видеть, если посмотреть на формулу (1.2). Следовательно, фазы Extract, Merge и Update итерации EM-алгоритма 2.1 являются *необходимыми*.
Можно поставить следующий вопрос:
Можно ли информацию о сессиях представить в таком виде и распределить ее между узлами кластера таким образом, чтобы процесс обучения параметров модели не требовал обмена данными между узлами кластера?
В данном разделе мы дадим ответ на этот вопрос для PBM и UBM.

4.1. Position-Based Model

Рассмотрим альтернативный способ хранения информации о сессиях. Положим, что $\mathcal{S} := \{s_1, s_2, \ldots, s_T\}$, тогда построим множества

$$\mathcal{R}_r := \left\{ \left(q_1, u_r^{(s_1)}, c_r^{(s_1)}\right), \left(q_2, u_r^{(s_2)}, c_r^{(s_2)}\right), \ldots, \left(q_T, u_r^{(s_T)}, c_r^{(s_T)}\right) \right\}, \quad r = 1, 2, \ldots, n$$

Таким образом, множество $\mathcal{R}_r$ содержит информацию о том, что происходило с документами ранга r в сессиях из $\mathcal{S}$, то есть *агрегирует информацию по рангу*.

Предположение **2.** *Узел кластера способен хранить и обрабатывать параметры, связанные с множеством* $\mathcal{R}_r$.

Следствием выдвинутого предположения является то, что в кластере обработкой данных должно заниматься не более n узлов.

Основная идея заключается в том, что узел, который обрабатывает $\mathcal{R}_r$, хранит информацию только об одном запросо-независимом параметре — γ_r.
С множеством $\mathcal{R}_r$ связана модель $M_r := \langle Q_r, I_r \rangle$, где $I_r := \{\gamma_r\}$ и

$$Q_r := \left\{ \alpha_{u_r^{(s_1)}, q_1}, \alpha_{u_r^{(s_2)}, q_2}, \ldots, \alpha_{u_r^{(s_T)}, q_T} \right\}.$$

Так как мы хотим обновлять параметры модели, связанные с $\mathcal{R}_r$, независимо от других узлов кластера, нужно показать, что

Теорема 4.1. *1. Модель $\mathcal{M} = (M_1, M_2, \ldots, M_n)$ - есть допустимая распределенная модель;*

2. Построение параметров модели $M_r^{(t+1)}$ зависит только от $(M_r^{(t)}, \mathcal{R}_r)$.

Доказательство. 1. Нужно показать, что

$$\forall r, r' \ (r \neq r' \quad \Longrightarrow \quad Q_r \cap Q_{r'} = \emptyset \wedge I_r \cap I_{r'} = \emptyset) \quad \wedge \quad \cup_r Q_r = Q \quad \wedge \quad \cup_r I_r = I.$$

Очевидно, что так как I_r по определению есть $\{\gamma_r\}$, следовательно $\cup_r I_r$ есть $(\gamma_1, \gamma_2, \ldots, \gamma_n) =: I$.
Рассмотрим пару (u, q): $\exists s \in \mathcal{S}_q(u \in s)$. В общем случае возможна ситуация такая, что $\exists s, s' \in \mathcal{S}_q(r_u^{(s)} \neq r_u^{(s')})$, следовательно $Q_{r_u^{(s)}} \cap Q_{r_u^{(s')}} \neq \emptyset$,
так как $\alpha_{uq} \in Q_{r_u^{(s)}}$ и $\alpha_{uq} \in Q_{r_u^{(s')}}$. Поэтому положим, что предположение 1 верно.
Тогда ввиду того, что $\forall s, s' \in \mathcal{S}_{uq}$ $(r_u^{(s)} = r_u^{(s')})$, мы получаем, что для пары $(u, q) \exists! r$: $\alpha_{uq} \in Q_r$. Следовательно, $\forall r, r' \ (r \neq r' \Longrightarrow Q_r \cap Q_{r'} = \emptyset)$.
Очевидно, что $\cup_r Q_r = Q$, так как при переходе $\mathcal{S} \quad \rightarrow \quad (\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n)$ мы не потеряли никакую информацию о сессиях.

2. Покажем, что мы можем корректно выполнить обновление параметра $\alpha_{u_r^{(s_1)}, q_1}$. Согласно формуле (1.2), для обновления параметра $\alpha_{u_r^{(s_1)}, q_1}$ нужно иметь в наличии множество $\mathcal{S}_{u_r^{(s_1)}, q_1}$. Так как мы считаем, что предположение 1 верно, то значит во всех сессиях, инициированных запросом q_1, документ $u_r^{(s_1)}$ имеет ранг r, следовательно, информация обо всех таких сессиях представлена в множестве $\mathcal{R}_r$. Более формально, это значит, что $\mathcal{R}_r$ можно представить в следующем виде:

$$\mathcal{R}_r = \mathcal{R}_{r,q_1} \cup \mathcal{R}_{r,q_2} \cup \ldots \cup \mathcal{R}_{r,q_l}, \tag{4.1}$$
$$\text{где } \mathcal{R}_{r,q} := \bigcup_{s \in \mathcal{S}_q} \left\{ \left(q, u_r^{(s)}, c_r^{(s)} \right) \right\}$$

и $q_1, q_2, \ldots, q_l$ - уникальные запросы из $\mathcal{S}$.
Из формулы (1.3) видно, что обновление γ_r зависит только от $(Q_r, \mathcal{R}_r)$.
Это завершает доказательство того, что $M_r^{(t+1)}$ зависит только от $(M_r^{(t)}, \mathcal{R}_r)$.

□

Таким образом, заключаем, что коммуникационная сложность итерации обучения параметров модели, при выполнении предположений 2 и 1, стремится к нулю.

Опишем общую схему обучения модели, которая может быть использована даже при условии, что предположение 2 не выполняется.

То, что $M_r^{(t+1)}$ зависит только от $(M_r^{(t)}, \mathcal{R}_r)$ в частности означает, что

$$(Q_r^{(t)}, \gamma_r^{(t)}, \mathcal{R}_r) \quad \underset{\mathcal{A}}{\rightarrow} \quad Q_r^{(t+1)}.$$

Используя выражение (4.1), можно выполнить декомпозицию предыдущей зависимости, а именно:

$$\begin{aligned} Q_r &= Q_{r,q_1} \cup Q_{r,q_2} \cup \ldots \cup Q_{r,q_l} \\ (Q_{r,q}^{(t)}, \gamma_r^{(t)}, \mathcal{R}_{r,q}) &\quad \underset{\mathcal{A}}{\rightarrow} \quad Q_{r,q}^{(t+1)}. \end{aligned} \tag{4.2}$$

Данная декомпозиция позволяет нам избавиться от предположения 2 при условии, что узел кластера способен обработать $\mathcal{R}_{r,q}$ для любых r и q.

Определение 4.1. *Множество узлов кластера, на которых происходит обработка данных из множества $\mathcal{R}_r$, будем называть группой и обозначать через G_r.*

Так как каждый из узлов группы G_r использует один и тот же запросо-независимый параметр - γ_r, то по аналогии с алгоритмом 2.1 мы можем утверждать о том, что по окончании $t+1$-ой итерации EM-алгоритма, значение параметра $\gamma_r^{(t+1)}$ должно быть загружено в каждый из узлов группы (фаза Update алгоритма 2.1).
Определение группы позволяет нам формализовать тот факт, что обмен информацией осуществляется в рамках групп, а не всего кластера целиком. Более формально данное утверждение можно записать следующим образом:

$$(Q_{r,q}^{(t)}, \gamma_r^{(t)}, \mathcal{R}_{r,q}) \quad \underset{\mathcal{A}}{\rightarrow} \quad \gamma_{r,q}^{(t+1)}.$$

Итоговое значение $\gamma_r^{(t+1)}$ можно вычислить следующим образом:

$$\gamma_r^{(t+1)} = \frac{\sum\limits_{q \in \mathcal{Q}} |\mathcal{R}_{r,q}| \cdot \gamma_{r,q}^{(t+1)}}{\sum\limits_{q \in \mathcal{Q}} |\mathcal{R}_{r,q}|}. \tag{4.3}$$

Замечание 4.1. *Заметим, что в общем случае $G_1, G_2, \ldots, G_n$ - не обязательно разбиение множества узлов кластера.*

Зададимся теперь вопросом о том, какова же будет коммуникационная сложность (см. [9]) одной итерации алгоритма в данном случае. Но для того, чтобы нам было с чем проводить сравнение, выясним, какова коммуникационная сложность итерации алгоритма 2.1.
Предположим, что данные $\mathcal{S}_q$, однажды оказавшись на узле кластера, более его не покидают, тогда справедливо следующее утверждение:

Лемма 4.2. *Коммуникационная сложность $\mathcal{A}_{MR}iteration$ есть*

$$CC_{\mathcal{A}_{MR}iteration} = \mathcal{O}(n \cdot K),$$

где K - количество узлов кластера, которые занимаются обработкой данных, и n - максимальный ранг.

Доказательство. Заметим, что обмен данными между узлами кластера осуществляется только на фазах Extract и Update алгоритма 2.1.
Оценим объем передаваемых данных для каждой из этих фаз в отдельности:

Extract Во время данной фазы каждый узел кластера отправляет узлу-Reducer'у свой вектор параметров - $I_q'^{(t+1)}$, размер которого есть $\mathcal{O}(n)$, следовательно, суммарный объем данных, которые получит узел-Reducer, есть $\mathcal{O}(n \cdot K)$.

Update Узел-Reducer отправляет вектор $I^{(t+1)}$ всем узлам, которые занимаются обработкой данных, следовательно, суммарный объем отправленных данных есть $\mathcal{O}(n \cdot K)$.

Таким образом мы показали, что, действительно, $CC_{\mathcal{A}_{MR}iteration} = \mathcal{O}(n \cdot K)$. □

Вернемся теперь к разговору о коммуникационной сложности рассматриваемого алгоритма. Для этого чуть в больших деталях опишем то, как именно организован процесс обучения PBM в рамках группы G_r:

1. В рамках группы G_r выбирается узел, который отвечает за сбор, обработку и распределение запросо-независимых параметров группы. Будем называть этот узел *представителем группы* и обозначать через $p(G_r)$. Таким образом, представитель G_r знает то, какие именно узлы входят в G_r, а все узлы G_r знают, кто является представителем.

2. В конце $t+1$-ой итерации EM-алгоритма, узел группы G_r, который обрабатывал $\mathcal{R}_{r,q}$, отправляет $p(G_r)$ пару $(|\mathcal{R}_{r,q}|, \gamma_{r,q}^{(t+1)})$.

3. На узле $p(G_r)$ выполняется преобразование:

$$\left((|\mathcal{R}_{r,q_1}|, \gamma_{r,q_1}^{(t+1)}), (|\mathcal{R}_{r,q_2}|, \gamma_{r,q_2}^{(t+1)}), \ldots, (|\mathcal{R}_{r,q_l}|, \gamma_{r,q_l}^{(t+1)})\right) \quad \xrightarrow[(4.3)]{} \quad \gamma_r^{(t+1)}.$$

4. Узел $p(G_r)$ сообщает остальным узлам группы G_r значение $\gamma_r^{(t+1)}$.

Прежде чем перейти к анализу коммуникационной сложности предложенного подхода, нам потребуется еще немного терминологии:

Определение 4.2. *Рассмотрим узел x кластера. Через $g(x)$ будем обозначать следующее:*

$$g(x) := |\{r \colon \mathcal{R}_{r,q} \text{ обрабатывается на } x\}|.$$

Теперь мы можем выразить коммуникационную сложность в терминах $g(x)$:

$$CC_{iteration} = \mathcal{O}\left(\sum_{x} g(x) - \sum_{r=1}^{n} g(p(G_r))\right). \tag{4.4}$$

Действительно, ведь узел $p(G_r)$ не должен отправлять сам себе данные.
Теперь мы в состоянии сформулировать следующую теорему:

Теорема 4.3***.*** *Пусть K - количество узлов кластера, на которых происходит обработка данных, тогда:*

$$0 \leqslant CC_{iteration} \leqslant \mathcal{O}(n \cdot K).$$

Доказательство. Рассмотрим ряд случаев, которые позволят нам убедиться в справедливости неравенства:

1. В ситуации, когда являются верными предположения 1 и 2, мы получаем, что $\forall r = \overline{1,n}\ (|G_r| = 1)$, то есть $\forall x\ (g(x) = 1)$, как следствие $CC_{iteration} = 0$.

2. $G_1, G_2, \ldots, G_n$ - разбиение множества узлов кластера, и $\forall r = \overline{1,n}\ (G_r \neq \emptyset)$, и $K > n$. В данном случае $\forall x\ (g(x) = 1)$, тогда согласно (4.4) получаем, что $CC_{iteration} = \mathcal{O}(K - n)$.

3. Очевидно, что $CC_{iteration} = \mathcal{O}(n \cdot K)$ в случае, когда $\forall x\ (g(x) = n)$.
   Рассмотрим распределение данных, которое позволяет достичь оценку в худшем случае. Пусть $K = l$, где l - количество уникальных запросов в $\mathcal{S}$, тогда пусть на узле x_i будут данные $\mathcal{R}_{1,q_i}, \mathcal{R}_{2,q_i}, \ldots, \mathcal{R}_{n,q_i}$, объединение которых в сущности есть $\mathcal{S}_{q_i}$, то есть мы перешли к случаю из леммы 4.2.

□

Таким образом мы показали, что коммуникационная сложность итерации рассматриваемого алгоритма существенно зависит от способа распределения данных между узлами кластера.

Можно сформулировать следующую задачу: При фиксированном K, найти способ распределения данных, минимизирующий коммуникационную сложность итерации алгоритма обучения.
Однако решение данной задачи выходит за рамки данной работы, поэтому мы не будем его рассматривать.

4.1.1. Векторно-матричное представление

По аналогии с разделом 3.1, с целью сделать обучение более эффективным, мы хотим получить векторно-матричное представление формул для EM-алгоритма в рас-

сматриваемом случае.

Attractiveness Заметим, что ввиду того, что мы считаем предположение 1 верным, множество Q_{r,q_i} в разбиении (4.2), состоит только из одного элемента — α_{u_i,q_i},
где $u_i := u_r^{(s)}, s \in \mathcal{S}_{q_i}$. Следовательно, выражение (1.2) можно записать в следующей форме:

$$\alpha_{u_i,q_i} \leftarrow \frac{1}{|\mathcal{R}_{r,q_i}|}\left(c_{u_i}^{(q_i)} + \left(|\mathcal{R}_{r,q_i}| - c_{u_i}^{(q_i)}\right)\frac{(1-\gamma_r)\alpha_{u_i,q_i}}{1-\gamma_r\alpha_{u_i,q_i}}\right),$$

где $c_{u_i}^{(q_i)} = \sum_{\left(q_i,u_i,c_r^{(s)}\right)\in\mathcal{R}_{r,q_i}} c_r^{(s)} = \sum_{s\in\mathcal{S}_{u_i,q_i}} c_{u_i}^{(s)}$.
Тут мы, так же как и в разделе 3.1, выполнили группировку слагаемых.
Так как в рассматриваемом случае запрос q_i однозначно определяет документ u_i, то, с целью упрощения записей, индекс u_i в ряде случаев будет опущен.
Для того, чтобы перейти к векторно-матричной форме записи, введем в рассмотрение следующие вектора:

$$\begin{aligned}\boldsymbol{\alpha}_r &:= \begin{pmatrix}\alpha_{q_1} & \alpha_{q_2} & \cdots & \alpha_{q_l}\end{pmatrix},\\ \boldsymbol{c}_r &:= \begin{pmatrix}c^{(q_1)} & c^{(q_2)} & \cdots & c^{(q_l)}\end{pmatrix},\\ \boldsymbol{\mathcal{R}}_r &:= \begin{pmatrix}|\mathcal{R}_{r,q_1}| & |\mathcal{R}_{r,q_2}| & \cdots & |\mathcal{R}_{r,q_l}|\end{pmatrix}.\end{aligned}$$

Используя данные вектора, мы можем записать выражение для обновления $\boldsymbol{\alpha}_r$ в векторно-матричной форме:

$$\boldsymbol{\alpha}_r \leftarrow \boldsymbol{\mathcal{R}}_r^{-1} \circ \left(\boldsymbol{c}_r + (\boldsymbol{\mathcal{R}}_r - \boldsymbol{c}_r) \circ ((1-\gamma_r)\cdot\boldsymbol{\alpha}_r) \circ (\mathbf{1} - \gamma_r\cdot\boldsymbol{\alpha}_r)^{-1}\right).$$

Examination Так как вместо $\mathcal{S}$ в выражении (1.3) теперь используется $\mathcal{R}_r$, то мы можем выполнить группировку слагаемых и получить следующее выражение:

$$\begin{aligned}\gamma_r &\leftarrow \frac{1}{|\mathcal{R}_r|}\sum_{q\in\mathcal{Q}}\sum_{\left(q,u_q,c_r^{(s)}\right)\in\mathcal{R}_r}\left(c_r^{(s)} + \left(1-c_r^{(s)}\right)\frac{(1-\alpha_q)\gamma_r}{1-\gamma_r\alpha_q}\right)\\ &= \frac{1}{|\mathcal{R}_r|}\sum_{q\in\mathcal{Q}}\left(c^q + (|\mathcal{R}_{r,q}| - c^q)\frac{(1-\alpha_q)\gamma_r}{1-\gamma_r\alpha_q}\right).\end{aligned}$$

Последнее выражение не составляет никакого труда записать в векторно-матричной форме, используя ранее определенные вектора:

$$\gamma_r \leftarrow |\mathcal{R}_r|^{-1}\cdot\left(\langle\boldsymbol{c}_r,\mathbf{1}\rangle + \langle\boldsymbol{\mathcal{R}}_r - \boldsymbol{c}_r, (\mathbf{1}-\boldsymbol{\alpha}_r)\circ(\mathbf{1}-\gamma_r\cdot\boldsymbol{\alpha}_r)^{-1}\rangle\cdot\gamma_r\right).$$

Ясно, что среди введенных в рассмотрение векторов только $\boldsymbol{\alpha}_r$ зависит от итерации EM-алгоритма, а значит остальные вектора можно посчитать один раз и использовать далее на всех итерациях.

4.1.2. MapReduce

Заметим, что обучение с агрегацией по рангу также может быть представлено в рамках MapReduce парадигмы. Рассмотрим ситуацию, когда предположение 2 верно:

Алгоритм 4.4

1: function $EM(\mathcal{S})$

2: Map В рамках данной фазы происходит обработка множества $\mathcal{S}$. Более формально, происходит следующее преобразование:

$$s_q \quad \mapsto \quad \left\{(r, (q, u_r^{(s_q)}, c_r^{(s_q)}))\right\}_{r=1}^{n}.$$

3: Reduce Мы знаем, что все записи (пары ключ-значение), которые были получены во время Map фазы, и имеют один и тот же ключ, будут обработаны одним Reducer'ом, то есть для фиксированного r, на Reducer'е будет набор

$$\left(r, \left\{(q_i, u_r^{(s_i)}, c_r^{(s_i)})\right\}_{i=1}^{T}\right).$$

Ясно, что данный набор есть не что иное, как $\mathcal{R}_r$. Таким образом, процесс обучения выполняется в рамках Reduce фазы.

Покажем теперь, как можно представить процесс обучения PBM, когда предположение 2 не выполняется.
Прежде всего нам нужно выполнить преобразование данных из $\mathcal{S}$:

1. Map

$$s_q \quad \mapsto \quad \left\{\big((r, q), \quad (q, u_r^{(s_q)}, c_r^{(s_q)})\big)\right\}_{r=1}^{n}.$$

2. Reduce

$$\big((r, q), \quad \left\{(q, u_r^{(s)}, c_r^{(s)}) \mid s \in \mathcal{S}_q\right\}\big) \quad \mapsto \quad \mathcal{R}_{r,q}.$$

Теперь мы можем перейти к рассмотрению того, как выглядит итерация алгоритма обучения:

Алгоритм 4.5

1: function $EM_{iteration}(\mathcal{M}, \mathcal{R})$

2: Map & Reduce

$$(M_{r,q}^{(t)}, \mathcal{R}_{r,q}) \quad \rightarrow \quad \langle Q_{r,q}^{(t+1)}, \gamma_{r,q}^{(t+1)} \rangle =: M_{r,q}^{\prime(t+1)}$$

3: Map

$$M_{r,q}^{\prime(t+1)} \quad \rightarrow \quad (r, \gamma_{r,q}^{(t+1)})$$

4: Reduce

$$(r, (\gamma_{r,q_1}^{(t+1)}, \gamma_{r,q_2}^{(t+1)}, \ldots, \gamma_{r,q_l}^{(t+1)})) \quad \rightarrow \quad (r, \gamma_r^{(t+1)})$$

5: Map & Reduce

$$(M_{r,q}^{\prime(t+1)}, \gamma_r^{(t+1)}) \quad \rightarrow \quad M_{r,q}^{(t+1)}$$

4.1.3. Оценка качества

Несмотря на то, что мы используем агрегацию по рангу, мы все еще можем вычислить метрики качества, описанные в разделе 1.3, используя MapReduce подход (если предположение 2 верно).

Перплексия Из формулы (1.7) видно, что из всей сессии $s \in \mathcal{S}$ используется только информация об $u_r^{(s)}, c_r^{(s)}$. Мы знаем, что данная информация есть во множестве $\mathcal{R}_r$. Кроме того, мы знаем, что для PBM $P_M\left(C_r = 1 \mid q, \boldsymbol{u}\right) = \alpha_{u_r q} \cdot \gamma_r$ (см. [2]). То есть вся необходимая информация для вычисления p_r содержится во множестве $\mathcal{R}_r$ и модели M_r.
Следовательно, на Map фазе достаточно вычислить значения $p_1, p_2, \ldots, p_n$, а на фазе Reduce воспользоваться формулой (1.8) для вычисления перплексии.

Правдоподобие Вычисление значения $\mathcal{LL}$ очевидным образом представимо в виде Map и Reduce фазы, так как для PBM $P_M\left(C_r = 1 \mid \boldsymbol{C}_{<r} = \boldsymbol{c}_{<r}^{(s)}\right) = P_M\left(C_r = 1\right) = \alpha_{u_r q} \cdot \gamma_r$ (см. [2]).

4.2. User Browsing Model

Ввиду того, что UBM довольно сильно похожа на PBM, то мы в своих рассуждениях будем опираться на результаты, полученные в разделе 4.1 для PBM.

Так же как и в предыдущем разделе, мы хотим распределить запросо-независимые параметры по узлам кластера таким образом, чтобы $CC_{iteration} = 0$.

Выполним агрегацию по рангу следующим образом:

$$\mathcal{R}_r = \left\{ \left(q_1, u_r^{(s_1)}, c_r^{(s_1)}, r'^{(s_1)}\right), \left(q_2, u_r^{(s_2)}, c_r^{(s_2)}, r'^{(s_2)}\right), \ldots, \left(q_T, u_r^{(s_T)}, c_r^{(s_T)}, r'^{(s_T)}\right) \right\},$$

где $r'^{(s)} = \max\{k \in \{0, 1, \ldots, r-1\} \mid c_k^{(s)} = 1\}$.
На основании того, какое именно значение принимает $r'^{(s)}$, мы можем представить $\mathcal{R}_r$ в следующем виде:

$$\mathcal{R}_r = \mathcal{R}_{r,0} \cup \mathcal{R}_{r,1} \cup \ldots \cup \mathcal{R}_{r,r-1},$$

где $\mathcal{R}_{r,r'} = \{(q, u_r^{(s)}, c_r^{(s)}, r'^{(s)}) \in \mathcal{R}_r \mid r'^{(s)} = r'\}$.

С множеством $\mathcal{R}_r$ связана кликовая модель $M_r = \langle Q_r, I_r \rangle$, где I_r, ввиду рассмотренного выше разбиения, есть $(\gamma_{r,0}, \gamma_{r,1}, \ldots, \gamma_{r,r-1})$.
Теперь мы можем перейти к доказательству утверждения теоремы (4.1), не меняя формулировку:

Доказательство. Будем считать, что предположения 1 и 2 верны.

1. Нужно показать, что

   $$\forall r, r' \ (r \neq r' \quad \Longrightarrow \quad Q_r \cap Q_{r'} = \emptyset \wedge I_r \cap I_{r'} = \emptyset) \quad \wedge \quad \cup_r Q_r = Q \quad \wedge \quad \cup_r I_r = I.$$

   По построению очевидно, что $\forall r, r' \ (I_r \cap I_{r'} = \emptyset) \wedge \cup_r I_r = I$.
   Утверждение касательно параметров Q_r доказывается абсолютно так же, как и в доказательстве теоремы (4.1).

2. Согласно формуле (1.4), для обновления значения α_{uq} требуется множество $\mathcal{S}_{uq}$, но так как мы считаем, что предположение 1 верно, то $\forall s \in \mathcal{S}_q \ (r_u^{(s)} = r)$, то есть вся необходимая информация имеется во множестве $\mathcal{R}_{r,q}$ (см. выражение (4.1)). Это означает, что обновление $Q_r^{(t+1)}$ зависит только от $(Q_r^{(t)}, I_r^{(t)}, \mathcal{R}_r)$.
   Осталось только показать, что все необходимые параметры для пересчета $\gamma_{rr'}$ содержатся в $\mathcal{R}_r$. Действительно, согласно выражению (1.5), для обновления $\gamma_{rr'}$ необходимо множество $\mathcal{S}_{rr'}$, и мы знаем, что $\mathcal{R}_{r,r'} \subset \mathcal{R}_r$ содержит в себе информацию о том, что происходило с документами ранга r в рамках сессий из $\mathcal{S}_{rr'}$. Таким образом, действительно, $\gamma_{rr'}^{(t+1)}$ зависит только от $(Q_r^{(t)}, \gamma_{rr'}^{(t)}, \mathcal{R}_{r,r'})$.

□

Таким образом, мы показали, что коммуникационная стоимость одной итерации алгоритма обучения UBM, при некоторых предположениях, стремится к нулю.

Замечание 4.2. *По аналогии с разделом 4.1.2 (для случая, когда предположение 2 верно) нетрудно понять, что обучение UBM также представимо в виде MapReduce фаз.*

Давайте рассмотрим то, как именно можно организовать процесс обучения, когда предположение 2 не выполняется. Нетрудно понять, что для UBM имеем $CC_{\mathcal{A}_{MR}iteration} = \mathcal{O}(n^2 \cdot K)$. Так же как и в случае PBM, определим понятие группы, что, возможно, позволит нам уменьшить обмен информацией между узлами кластера.
Определим группу G_r так же, как и для PBM, пользуясь разбиением (4.1), то есть G_r - множество узлов, которые обрабатывают части множества $\mathcal{R}_r$. Но в отличии от PBM, где в рамках G_r использовался параметр γ_r, в рамках группы для UBM используется вектор параметров $(\gamma_{r,0}, \gamma_{r,1}, \ldots, \gamma_{r,r-1})$.
То есть любой узел из G_r на каждой итерации отправляет узлу $p(G_r)$, $\mathcal{O}(r)$ данных. Для того, чтобы оценить суммарный объем пересылаемых данных во время итерации, нам необходимо оценить количество узлов в группе G_r. Будем считать, что $G_1, G_2, \ldots, G_n$ образуют разбиение множества узлов кластера.
Сделаем следующее предположение:

Предположение 3. *Любой узел кластера в состоянии обработать* $\geqslant \frac{|\mathcal{R}_r|}{C}$ *записей, где* $C > 1$ *- некоторая константа.*

Следствием предположения 3 является то, что $|G_r| \leqslant C$. Тогда

$$CC_{iteration} = \mathcal{O}\left(\sum_{r=1}^{n} r \cdot C\right) = \mathcal{O}(n^2 \cdot C),$$

и кластер содержит не более $n \cdot C$ узлов.

Вопросы касательно коммуникационной сложности в случае, когда $G_1, G_2, \ldots, G_n$ не образуют разбиение, мы оставляем за рамками данной работы.

Заметим, что вместо того, чтобы обмениваться запросо-независимыми параметрами в рамках группы, мы можем определить группу таким образом, чтобы обмен происходил запросо-зависимыми параметрами. Возможно, что такой способ построения групп позволит уменьшить суммарное количество пересылаемых данных между узлами кластера.

4.2.1. Векторно-матричное представление

Покажем, каким образом можно получить векторизацию формул (1.4) и (1.5) для рассматриваемого распределенного алгоритма обучения UBM.

Examination Рассмотрим узел кластера, который обрабатывает $\mathcal{R}_r$. Данный узел использует параметры $\boldsymbol{\gamma_r} := (\gamma_{r,0}, \gamma_{r,1}, \ldots, \gamma_{r,r-1})$. Наша цель состоит в том, чтобы представить обновление параметра $\gamma_{r,r'}$ в векторно-матричной форме.

Заметим, что даже в случае, когда предположение 1 верно, множество $\mathcal{S}_{rr'}$, которое используется в (1.5), может быть представлено в виде разбиения $\mathcal{S}_{rr'q_1} \cup \mathcal{S}_{rr'q_2} \cup \ldots \cup \mathcal{S}_{rr'q_l}$, в котором тем не менее могут присутствовать пустые множества. Так как мы считаем, что предположение 1 верно, то по запросу q_i и рангу r мы можем однозначно определить документ $u_i := u_r^{(s)} : s \in \mathcal{S}_q$. Разбиению множества $\mathcal{S}_{rr'}$ можно поставить в соответствие разбиение множества $\mathcal{R}_{r,r'}$:

$$\mathcal{R}_{r,r'} = \mathcal{R}_{r,r',q_1} \cup \mathcal{R}_{r,r',q_2} \cup \ldots \cup \mathcal{R}_{r,r',q_l},$$

где необязательно все множества непустые.
Используя данное разбиение, мы можем переписать (1.5) следующим образом:

$$\begin{aligned}\gamma_{rr'} &\leftarrow \frac{1}{|\mathcal{R}_{rr'}|} \sum_{i=1}^{l} \sum_{(q_i,u_i,c,r') \in \mathcal{R}_{r,r',q_i}} \left(c + (1-c) \frac{(1-\alpha_{u_i q_i})\gamma_{rr'}}{1-\alpha_{u_i q_i}\gamma_{rr'}} \right) \\ &= \frac{1}{|\mathcal{R}_{rr'}|} \sum_{i=1}^{l} \left(c_{u_i}^{rr'} + \left(|\mathcal{R}_{r,r',q_i}| - c_{u_i}^{rr'} \right) \frac{(1-\alpha_{u_i q_i})\gamma_{rr'}}{1-\alpha_{u_i q_i}\gamma_{rr'}} \right), \end{aligned} \tag{4.5}$$

где $c_{u_i}^{rr'} = \sum\limits_{(q_i,u_i,c,r') \in \mathcal{R}_{r,r',q_i}} c \geqslant 0$.
Для того, чтобы перейти к векторно-матричной записи предыдущего выражения, введем в рассмотрение следующие вектора:

$$\begin{aligned} \boldsymbol{c}^{rr'} &= \begin{pmatrix} c_{u_1}^{rr'} & c_{u_2}^{rr'} & \cdots & c_{u_l}^{rr'} \end{pmatrix}, \\ \boldsymbol{\alpha}_r &= \begin{pmatrix} \alpha_{u_1 q_1} & \alpha_{u_2 q_2} & \cdots & \alpha_{u_l q_l} \end{pmatrix}, \\ \boldsymbol{\mathcal{R}}_{r,r'} &= \begin{pmatrix} |\mathcal{R}_{r,r',q_1}| & |\mathcal{R}_{r,r',q_2}| & \cdots & |\mathcal{R}_{r,r',q_l}| \end{pmatrix}. \end{aligned}$$

Используя данные вектора, мы можем записать (4.5) следующим образом:

$$\gamma_{rr'} \leftarrow \frac{1}{|\mathcal{R}_{rr'}|} \left(\langle \boldsymbol{c}^{rr'}, \mathbf{1} \rangle + \gamma_{rr'} \langle \boldsymbol{\mathcal{R}}_{r,r'} - \boldsymbol{c}^{rr'}, (\mathbf{1} - \boldsymbol{\alpha}_r) \circ (\mathbf{1} - \gamma_{rr'} \cdot \boldsymbol{\alpha}_r)^{-1} \rangle \right).$$

Очевидно, что в данном выражении, наряду с $\gamma_{rr'}$, от t зависит только $\boldsymbol{\alpha}_r$. Остальные вектора могут быть посчитаны однажды, на этапе предобработки данных. Также стоит отметить, что обновление $\gamma_{rr'}$, $\gamma_{rr''}$ можно выполнять параллельно при $r' \neq r''$.

Attractiveness Мы хотим представить обновление элементов вектора $\boldsymbol{\alpha}_r$ в векторно-матричной форме.
Заметим, что значение r' в формуле (1.4) зависит от рассматриваемой сессии $s \in \mathcal{S}_{uq}$. Так как $r'^{(s)}$ принимает значение из $\{0, 1, \ldots, r-1\}$, то можно выполнить группировку

слагаемых и получить следующее выражение:

$$\alpha_{u_i q_i} \leftarrow \frac{1}{|\mathcal{R}_{r,q_i}|} \sum_{r'=0}^{r-1} \sum_{(q_i,u_i,c,r') \in \mathcal{R}_{r,r',q_i}} \left(c + (1-c) \frac{(1-\gamma_{rr'})\alpha_{u_i q_i}}{1-\gamma_{rr'}\alpha_{u_i q_i}} \right)$$
$$= \frac{1}{|\mathcal{R}_{r,q_i}|} \sum_{r'=0}^{r-1} \left(c_{u_i}^{rr'} + \left(|\mathcal{R}_{r,r',q_i}| - c_{u_i}^{rr'} \right) \frac{(1-\gamma_{rr'})\alpha_{u_i q_i}}{1-\alpha_{u_i q_i}\gamma_{rr'}} \right).$$

То же самое в векторно-матричной форме:

$$\alpha_{u_i,q_i} \leftarrow \frac{1}{|\mathcal{R}_{r,q_i}|} \left(\langle \boldsymbol{c}_{u_i}^r, \mathbf{1} \rangle + \alpha_{u_i q_i} \cdot \langle \boldsymbol{\mathcal{R}}_{r,q_i} - \boldsymbol{c}_{u_i}^r, (\mathbf{1} - \alpha_{u_i q_i} \cdot \boldsymbol{\gamma}_r)^{-1} \circ (\mathbf{1} - \boldsymbol{\gamma}_r) \rangle \right),$$

где $\boldsymbol{c}_{u_i}^r = (c_{u_i}^{r,0}, c_{u_i}^{r,1}, \ldots, c_{u_i}^{r,r-1})$, и $\boldsymbol{\mathcal{R}}_{r,q_i} = (|\mathcal{R}_{r,0,q_i}|, |\mathcal{R}_{r,1,q_i}|, \ldots, |\mathcal{R}_{r,r-1,q_i}|)$.

Ясно, что значения $\alpha_{u_i q_i}$ и $\alpha_{u_j q_j}$ для разных i, j могут быть вычислены параллельно, что позволяет увеличить эффективность вычислений.

Мы показали, что обновление параметров модели может быть векторизовано, однако мы не задаемся вопросом о том, как много нужно памяти для того, чтобы построить вектора на этапе подготовки данных, так как рассмотрение данного вопроса выходит за рамки данной работы.

4.2.2. Оценка качества

Правдоподобие В случае, когда выполняется предположение 2, вычисление $\mathcal{LL}$ может быть легко представлено в виде Map и Reduce фаз. Действительно, для начала поменяем местами суммы по s и по r в выражении (1.6), далее заметим, что в случае UBM вероятность $P_M \left(C_r = 1 \mid \boldsymbol{C}_{<r} = \boldsymbol{c}_{<r}^{(s)} \right)$, которая участвует в выражении (1.6), есть $\alpha_{u_r q} \cdot \gamma_{rr'}$ (см. [2]). Таким образом, для фиксированного r, вся информация для вычисления суммы

$$\mathcal{L}_r := \sum_{s \in \mathcal{S}} P_M \left(C_r = 1 \mid \boldsymbol{C}_{<r} = \boldsymbol{c}_{<r}^{(s)} \right)$$

содержится в $\mathcal{R}_r, M_r$.
Во время Map фазы на узле, который хранит M_r, выполняется вычисление значения $\mathcal{L}_r$. Во время фазы Reduce вычисляется сумма $\mathcal{L}_1 + \mathcal{L}_1 + \ldots + \mathcal{L}_n$, которая и есть $\mathcal{LL}$.

Перплексия Заметим также, что рассмотренное распределение данных между узлами кластера (*агрегация по рангу*) не позволяет нам легко вычислить перплексию, рассмотренную в разделе 1.3
Действительно, для вычисления (1.7) нужно вычислить $P_M(C_r = 1 \mid q, \boldsymbol{u})$. Для UBM

(см. [2, 4]).

$$P(C_r = 1) = \sum_{j=0}^{r-1} P(C_j = 1) \left(\prod_{k=j+1}^{r-1} (1 - \alpha_{u_k q} \gamma_{kj}) \right) \alpha_{uq} \gamma_{rj},$$

где $P(C_0 = 1) = 1$.

Из данной формулы видно, что при вычислении вероятности используются параметры, которых в общем случае нет на узле, обрабатывающем $\mathcal{R}_r$.

Для решения проблемы воспользуемся MapReduce подходом для того, чтобы выполнить переход:

$$(M_1, M_2, \ldots, M_n) \quad \rightarrow \quad \mathcal{M} = (M_{q_1}, M_{q_2}, \ldots, M_{q_l})$$

То есть мы хотим построить другую распределенную кликовую модель из уже вычисленных параметров.

Опишем теперь алгоритм данного перехода:

Алгоритм 4.6

1: function TRANSFORM($M_1, M_2, \ldots, M_n$)

2: Map Рассмотрим модель $M_r = \langle Q_r, I_r \rangle$. Набор запросо-зависимых параметров можно представить виде (4.2), где набор $Q_{r,q}$ содержит только те параметры из Q_r, которые зависят от запроса q. Используя данное разбиение, мы можем построить модели $M_{r,q} = \langle Q_{r,q}, I_r \rangle$. Более формально:

$$M_r \quad \mapsto \quad \left(q, \{M_{r,q}\}_{q \in \mathcal{Q}} \right).$$

3: Reduce На данной фазе Reducer обрабатывает набор

$$\left(q, \{M_{r,q}\}_{r=1}^{n} \right).$$

Заметим, что по данному набору можно восстановить кликовую модель M_q. Действительно,

$$M_q = \langle Q_{1,q} \cup Q_{2,q} \cup \ldots \cup Q_{n,q}, I_1 \cup I_2 \cup \ldots \cup I_n \rangle.$$

После того, как мы построим модель $\mathcal{M}$, мы сможем воспользоваться методами, описанными в разделе 2.2, для вычисления метрик качества из раздела 1.3.

4.3. Замечания

Так как в DBN есть один запросо-независимый параметр (γ), который необходим при вычислении каждого запросо-зависимого параметра модели (формулы можно найти в [2, 1]), то у нас нет возможности построить распределенный алгоритм

обучения, имеющий нулевую коммуникационную сложность итерации. Аналогичные соображения верны для CCM — модель обладает тремя запросо-независимыми параметрами (τ_1, τ_2, τ_3), которые используются при обновлении любого запросо-зависимого параметра.

Был проведен анализ касательно того, какие преимущества может дать агрегация по документам. В результате анализа был сделан вывод о том, что данный подход не имеет никаких преимуществ по сравнению с подходом, описанным в разделе 2.

Заключение

В рамках данной работы был реализован (в рамках библиотеки PyClick) общий подход к обучению кликовых моделей в рамках парадигмы вычислений MapReduce. Разработанный метод может быть использован для обучения моделей, которые используют EM-алгоритм для вычисления параметров модели. Данный подход свободен от недостатков традиционного EM подхода к обучению, которые были описаны во введении, и является более эффективным.

Для Position-Based Model было получено векторно-матричное представление формул EM-алгоритма, которое может сделать процесс вычисления параметров модели более эффективным. Открытым остается вопрос о том, можно ли получить в.-м. представление формул для других моделей, и если возможно, то позволит ли это сделать обучение более эффективным.

В результате того, что был рассмотрен альтернативный способ к представлению информации о поисковых сессиях, был получен новый распределенный подход к обучению PBM и UBM, который позволяет значительно уменьшить количество данных, пересылаемых между узлами кластера в процессе обучения, а при выполнении некоторых предположений, позволяет выполнить обучение без обмена информацией между узлами кластера. Также было показано, что подобный результат недостижим для CCM и DBN.

Открытыми остаются вопросы о поиске полезных способов представления информации о сессиях и распределения данных между узлами кластера.

Список литературы

[1] Chapelle Olivier, Zhang Ya. A dynamic bayesian network click model for web search ranking // Proceedings of the 18th international conference on World wide web / ACM. — 2009. — P. 1–10.

[2] Chuklin Aleksandr, Markov Ilya, Rijke Maarten de. Click models for web search // Synthesis Lectures on Information Concepts, Retrieval, and Services. — 2015. — Vol. 7, no. 3. — P. 1–115.

[3] Click chain model in web search / Fan Guo, Chao Liu, Anitha Kannan et al. // Proceedings of the 18th international conference on World wide web / ACM. — 2009. — P. 11–20.

[4] Dupret Georges E, Piwowarski Benjamin. A user browsing model to predict search engine click data from past observations. // Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval / ACM. — 2008. — P. 331–338.

[5] Fisher Ronald Aylmer. On the mathematical foundations of theoretical statistics // Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character. — 1922. — Vol. 222. — P. 309–368.

[6] Koller Daphne, Friedman Nir. Probabilistic graphical models: principles and techniques. — MIT press, 2009.

[7] Lin Jimmy, Dyer Chris. Data-intensive text processing with MapReduce // Synthesis Lectures on Human Language Technologies. — 2010. — Vol. 3, no. 1. — P. 1–177.

[8] Liu Chao, Guo Fan, Faloutsos Christos. Bbm: bayesian browsing model from petabyte-scale data // Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining / ACM. — 2009. — P. 537–546.

[9] Rao Anup, Yehudayoff Amir. Communication Complexity. — 2016.

[10] White Tom. Hadoop: The Definitive Guide 4th Edition [M]. — 2009.

[11] Yandex. Yandex Relevance Prediction Challenge. — 2011. — Access mode: `https://academy.yandex.ru/events/data_analysis/relpred2011/`.

[12] An experimental comparison of click position-bias models / Nick Craswell, Onno Zoeter, Michael Taylor, Bill Ramsey // Proceedings of the 2008 International Conference on Web Search and Data Mining / ACM. — 2008. — P. 87–94.

[13] Ширяев А.Н. Вероятность-1. — МЦНМО, 2004.