Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математических методов прогнозирования

Коваленко Павел Антонович

Использование случайных блужданий по графу для повышения качества рекомендаций

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Научный руководитель:

д.ф-м.н., профессор

Дьяконов Александр Геннадьевич

Москва, 2019

Содержание

1 Введение

Рекомендательные системы — это семейство алгоритмов, предназначенных для выделения из широкого множества объектов тех, которые должны быть интересны пользователю. Рекомендации могут быть основаны на истории действий пользователя, например, его предыдущих покупках или просмотрах, а также на внешней информации о пользователе, такой как пол, возраст, сфера деятельности, и об объекте, например, жанр, год выпуска и длительность для фильма.

Рекомендательные системы стали неотъемлемой частью онлайн-сервисов, где объем контента слишком велик, чтобы пользователь мог просмотреть его весь. В частности, рекомендации используются в онлайн-магазинах и интернет-торговле [6, 8], сервисах онлайн-прослушивания музыки [15], просмотра видео [19], фильмов и сериалов [2, 5], социальных сетях [10].

В ряде задач машинного обучения получили распространение методы аугментации обучающей выборки, то есть пополнения ее искусственными примерами. Аугментация хорошо зарекомендовала себя в работе с изображениями [13] и временными рядами [4]. Однако во многих задачах ее сложно применить, поскольку это требует глубокого погружения в предметную область и физический смысл признаков.

В настоящей работе предлагается алгоритм аугментации для рекомендательных систем, который может быть применен к широкому классу задач без погружения в предметную область. Идея метода основана на случайных блужданиях по графу специального вида. Детальное описание метода дано в разделе 4, в разделе 6 приведены эксперименты с использованием предложенного метода аугментации на ряде общедоступных наборов данных.

2 Обзор предметной области

Рекомендательные системы разделяются в зависимости от используемых ими данных и предположений на контентные (content-based filtering) и коллаборативные (collaborative filtering).

Контентные модели используют признаковое описание пользователя и объекта для предсказания релевантности объекта к пользователю. Таким образом, задача рекомендаций сводится к задаче регрессии меры релевантности или классификации на релевантные и не релевантные объекты.

Коллаборативные модели действуют в предположении, что похожим пользователям нравятся похожие объекты. Они часто используются в случаях, когда у пользователей и объектов нет признакового описания, только история взаимодействия. В частности, к коллаборативным моделям относятся SVD и случайные блуждания, которые будут рассмотрены ниже.

Данная классификация не покрывает все возможные случаи. Существуют модели, которые используют похожесть между пользователями совместно с признаками пользователей и объектов, например, факторизационные машины [16]. Также часто предсказания коллаборативных моделей используют как признаки для обучения контентных моделей [18].

Не всегда целью рекомендаций является пользователь. На практике часто возникает задача рекомендаций объекта к объекту. Например, рекомендации треков для добавления их в плейлист [15] или рекомендации объектов, похожих на некоторый заданный [12]. В общем случае объект или пользователь, для которого строятся рекомендации, ниже будет называться запросом.

2.1 Математическая постановка задачи рекомендаций

Имеется множество пользователей (запросов) $U = \{u\}$ и множество объектов (документов) $I = \{i\}$, $|U| = N, |I| = M$. Для некоторых пар (u, i) известна оценка r_{ui}, которую пользователь u поставил объекту i. Требуется заполнить пропуски, то есть для всех пар (u, i) предсказать оценку (релевантность) $\hat{r}_{ui}$. Другая возможная поста-

новка — для каждого пользователя u найти K максимально релевантных объектов $i_1, ..., i_K$, то есть объектов с наибольшей предсказанной оценкой.

Часто рассматривают задачи с неявным откликом — все $r_{ui} = 1$. Например, если человек слушает музыку, то мы знаем, какие треки он слушал, и считаем, что они ему нравятся, но ничего не знаем про все остальные треки. В таком случае обычно считают, что все пары (u, i), которых нет в выборке (то есть все треки, которые пользователь не слушал), являются отрицательными примерами, для них считаем $r_{ui} = 0$. **Далее в работе будет рассматриваться только задача рекомендаций с неявным откликом.**

Для удобства часто вводят матрицу $R = \{r_{ui}\} \in \mathbb{R}^{N \times M}$, считая, что не все ее ячейки могут быть заполнены, или заполняя пропуски нулями.

2.2 Метрики качества рекомендаций

Для оффлайн-замера качества алгоритма обычно разбивают множество всех оценок на две части — обучающую и тестовую выборку. Разбиение может быть случайным или по времени оценки, если оно известно — оценки, поставленные раньше, попадают в тренировочную выборку, а поставленные позже — в тестовую, что делает замер более приближенным к реальным условиям.

Большая часть метрик качества для задачи рекомендаций вводится как метрика для одного пользователя, а далее усредняется по пользователям.

Обозначим множество всех оценок пользователя из обучающей выборки как I_{train}, из тестовой выборки — как I_{test}, $I_{train} \cap I_{test} = \varnothing$. Результатом работы алгоритма ранжирования можно считать последовательность всех доступных для ранжирования объектов $I \setminus I_{train}$, упорядоченную по предсказанной алгоритмом релевантности $\hat{r}_{ui}$. Обозначим эту последовательность за I_{pred}. Также обозначим за $I_{pred}[:k]$ множество из k первых (наиболее релевантных) объектов из I_{pred}.

Precision at k — точность по первым k объектам. $P@k = \dfrac{|I_{pred}[:k] \cap I_{test}|}{k}$.

Recall at k — полнота по первым k объектам. $R@k = \dfrac{|I_{pred}[:k] \cap I_{test}|}{|I_{test}|}$.

Average precision at k — средняя точность для тех позиций среди первых k, на которые алгоритм поставил релевантные объекты. $AP@k = \dfrac{1}{k} \sum_{j=1}^{k} \mathbb{I}[I_{pred}[j] \in I_{test}] P@j$.

Discounted cumulative gain at k — похожа на precision, но объекты, идущие раньше в ранжировании, имеют больший вес. Для случая неявного отклика формула имеет следующий вид: $DCG@k = \sum_{j=1}^{k} \frac{\mathbb{I}[I_{pred}[j] \in I_{test}]}{\log_2(j+1)}$. В отличие от предыдущих метрик, для DCG максимальным возможным значением является не 1. Поэтому чаще используют следующую метрику.

Normalized discounted cumulatine gain at k — DCG, нормированная на максимальное возможное значение метрики, поэтому изменяется на отрезке $[0, 1]$. $NDCG@k = \frac{DCG@k}{DCG^*@k}$, $DCG^*@k = \sum_{j=1}^{\min(|I_{test}|,k)} \frac{1}{\log_2(j+1)}$.

Также часто рассматриваются варианты DCG и NDCG, учитывающие все объекты в ранжировании: $DCG = DCG@M$, $NDCG = NDCG@M$.

3 Рассматриваемые алгоритмы

3.1 Singular Value Decomposition

Данная модель предполагает, что есть некоторое d-мерное вещественное пространство, и всем пользователям u и объектам i соответствуют вектора в этом пространстве p_u и q_i, называемые латентными векторами. Близость (релевантность) между пользователем и объектом определяется как скалярное произведение между их латентными векторами: $\hat{r}_{ui} = \langle p_u, q_i \rangle$. Задача состоит в поиске латентных векторов пользователей и объектов, для которых среднеквадратичное отклонение на объектах обучающей выборки является минимальным:

$$L(p, q) = \sum_{u,i} (r_{ui} - \hat{r}_{ui})^2 = \sum_{u,i} (r_{ui} - \langle p_u, q_i \rangle)^2 \to \min_{p,q}$$

Данная модель часто называется Singular Value Decomposition (SVD), поскольку в случае полностью заполненной матрицы R оптимальными значениями для p и q являются соответствующие столбцы из сингулярного разложения матрицы R: $R = P\Sigma Q^T$. Также для поиска латентных векторов можно использовать алгоритм Alternating Least Squares, основанный на попеременной оптимизации пользовательских и объектных векторов. Поэтому описанная выше модель также встречается под названием Alternating Least Squares (ALS) [14].

3.2 Bayesian Personalized Ranking

Данный метод является развитием ALS. Авторы статьи [3] предположили, что для задачи ранжирования больше подходит не поточечная функция ошибки, а попарная, которая штрафует не за отклонение истинной оценки от предсказанной, а за инверсии между истинным ранжированием и предсказанным. В статье предлагается следующая функция ошибки, называемая логистической:

$$L(u, i_p, i_n) = \log\left(\sigma(\langle p_u, q_{i_p}\rangle - \langle p_u, q_{i_n}\rangle)\right)$$

где u — пользователь, i_p, i_n — два объекта, таких что $r_{u,i_p} > r_{u,i_n}$, p_u — латентный вектор пользователя, q_{i_p}, q_{i_n} — латентные вектора объектов, $\sigma(x) = \frac{1}{1+e^{-x}}$ — сигмоидальная функция. Итоговый функционал можно записать в следующем виде:

$$L(p, q) = \sum_{u,i,i'} \mathbb{I}[r_{u,i} > r_{u,i'}] \log\left(\sigma(\langle p_u, q_i\rangle - \langle p_u, q_{i'}\rangle)\right) \to \min_{p,q}$$

3.3 SVD++

Идея метода очень близка к классическому SVD, однако в этом методе более явно учитывается история оценок пользователя [11]. Для всех объектов помимо веторов q_i также вводятся вектора y_i. Значение релевантности объекта пользователю вычисляется как

$$\hat{r}_{ui} = \left\langle p_u + \frac{1}{\sqrt{|I(u)|}} \sum_{j \in I(u)} y_j, q_i \right\rangle$$

Как и SVD, данный алгоритм минимизирует среднеквадратичное отклонение по обучающей выборке:

$$L(p, q, y) = \sum_{u,i} (r_{ui} - \hat{r}_{ui})^2 = \sum_{u,i} \left(r_{ui} - \left\langle p_u + \frac{1}{\sqrt{|I(u)|}} \sum_{j \in I(u)} y_j, q_i \right\rangle \right)^2 \to \min_{p,q,y}$$

3.4 Random walks

Пусть задан ориентированный граф. Неформально определить случайное блуждание можно следующим образом. Есть агент, который перемещается по графу, стартуя в случайной вершине v_0. Далее в каждый момент времени t он перемещается из

вершины v_t в случайную вершину v_{t+1}, в которую ведет ребро из вершины v_t. Вероятности переходов между вершинами являются частью алгоритма, обычно вероятности равные или зависят от степени конечной вершины. Таким образом получаем бесконечную последовательность вершин $\{v_k\}$, в которой соседние вершины связаны ребром.

Чтобы использовать случайные блуждания для задачи рекомендаций, построим двудольный граф. Вершинами первой доли будут пользователи, второй — объекты. Между пользователем и объектом есть ребро, если пользователь положительно оценил объект (например, прослушал трек).

Чтобы построить рекомендации для пользователя u, запустим из соответствующей пользователю вершины T случайных блужданий, каждое из которых остановим после нечетного числа шагов L. В этом случае в силу двудольности графа все блуждания завершатся в вершинах, соответствующих некоторым объектам. После этого можно ввести меру близости между пользователем u и объектом i как число случайных блужданий, завершившихся в вершине i, и выбрать объекты, в которых заканчивается наибольшее число путей.

В случае $L = 1$ получатся абсолютно точные и бесполезные рекомендации — случайное блуждание всегда будет останавливаться в объектах, уже оцененных пользователем.

В данной работе метод случайного блуждания будет использован не как самостоятельный алгоритм ранжирования, а только как часть более сложной модели.

3.5 High-Order Proximity for Implicit Recommendation

Одной из проблем при обучении латентных векторов является низкая плотность матрицы R, из-за чего алгоритму требуется много шагов оптимизации для нахождения оптимальных латентных векторов [1]. В статье [7] предложен гибридный подход, совмещающий преимущества случайных блужданий и Bayesian Personalized Ranking.

Используется логистический функционал, описанный в 3.2. Латентные вектора обучаются стохастическим градиентным спуском на батчах, сформированных следующим образом. Для каждого примера из батча случайно выбирается пользова-

тель, для него семплируется случайный объект в качестве отрицательного примера, а в качестве положительного примера используется конечная вершина случайного блуждания с началом в вершине пользователя. Таким образом, обучающая выборка помимо объектов, которые пользователь явно оценил, пополняется похожими на них объектами. За счет этого понижается разреженность матрицы R.

В настоящей работе предлагается развитие и более полное исследование идей, предложенных в данной статье.

4 Предложенный метод

В работе предлагается метод аугментации[1] датасета для задачи рекомендаций с неявным откликом.

Исходная обучающая выборка представляется в виде двудольного неориентированного графа, как это описано в разделе 3.4. Далее по этому графу создается с нуля новая обучающая выборка. Новая выборка состоит только из положительных примеров, в каждом из которых пользователь и объект являются начальной и конечной вершинами случайного блуждания.

У случайных блужданий есть множество параметров, в данной работе рассмотрены следующие:

- Число случайных блужданий. Далее обозначается за N.
- Стартовая доля в графе — случайные блуждания можно начинать с пользователя или с объекта. Далее обозначается за $StartFromUser$.
- Зависимость вероятности перехода в вершину от числа исходящих из нее ребер. В работе был рассмотрен только вариант степенной зависимости: $p(v) \sim \deg(v)^K$. Вероятность вершины быть выбранной в качестве начала случайного блуждания также зависит от количества исходящих из нее ребер.
- Критерий остановки. В работе рассмотрены два критерия:

[1]Аугментация — увеличение объема обучающей выборки за счет добавления синтетических примеров

– По длине пути. Все случайные блуждания должны быть фиксированной длины L. Чтобы начальная и конечная вершина лежали в разных долях, число ребер L должно быть нечетным.

– С заданной вероятностью остановки после каждого шага. После каждого перехода в долю, не совпадающую с начальной, путь прерывается с заданной заранее вероятностью P.

В алгоритме 1 записан в виде псевдокода предлагаемый алгоритм построения датасета. Критерии остановки представлены в алгоритмах 2 и 3.

Алгоритм 1 Построение аугментированного датасета

$Dataset \leftarrow \emptyset$
for $i \in \overline{1, N}$ **do**
    if $StartFromUser$ **then**
        $v_0 \sim Cat(v : \deg(v)^K, v \in U)$
    else
        $v_0 \sim Cat(v : \deg(v)^K, v \in I)$
    end if
    $k \leftarrow 0$
    while stopping criterion not satisfied **do**
        $v_{k+1} \sim Cat(v : \deg(v)^K, v \in Neighbours(v_k))$
        $k \leftarrow k + 1$
    end while
    if $StartFromUser$ **then**
        $Dataset \leftarrow Dataset \cup \{(v_0, v_k)\}$
    else
        $Dataset \leftarrow Dataset \cup \{(v_k, v_0)\}$
    end if
end for

Алгоритм 2 Критерий остановки по длине пути

```
function StoppingCriterionSatisfied
    return (k = L)
end function
```

Алгоритм 3 Критерий остановки с вероятностью остановки

```
function StoppingCriterionSatisfied
    if k % 2 = 0 then
        return False                ▷ Путь не должен закончиться в стартовой доле
    else
        return x ~ Cat(True : P, False : 1 − P)
    end if
end function
```

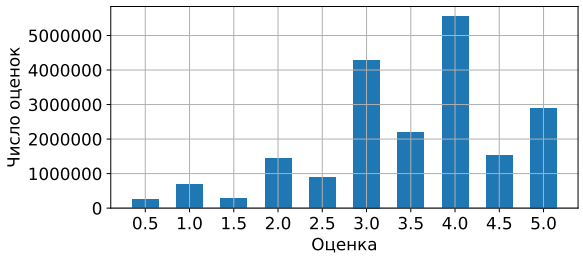
5 Описание датасетов

Для проверки работоспособности предложенного метода был проведен ряд экспериментов, которые будут подробно описаны в разделе [6]. Данный раздел посвящен описанию рассмотренных датасетов.

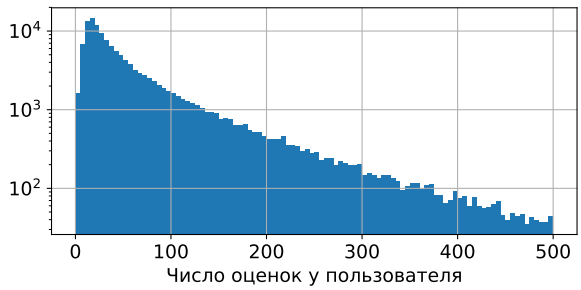
5.1 MovieLens

Данный датасет является выгрузкой оценок с сайта `movielens.org`, где пользователи могут ставить оценки фильмам, которые они просмотрели, и получать на основе этих оценок рекомендации для просмотра [5]. Датасет предоставляется в четырех вариантах: MovieLens 100K, MovieLens 1M, MovieLens 10M и MovieLens 20M, которые содержат $10^5, 10^6, 10^7$ и 2×10^7 оценок соответственно. Оценки — целые или полуцелые числа от 0.5 до 5.0. Распределение оценок для датасета MovieLens 20M, а также распределение числа оценок для пользователей и фильмов, представлены на рис. 1. Во всех четырех датасетах есть только пользователи, оценившие не менее 20 фильмов.

Для экспериментов все четыре датасета были приведены к бинарному виду. Из них были удалены все примеры, где пользователь поставил фильму оценку мень-



(a) Распределение оценок



(b) Гистограмма числа положительных оценок пользователя

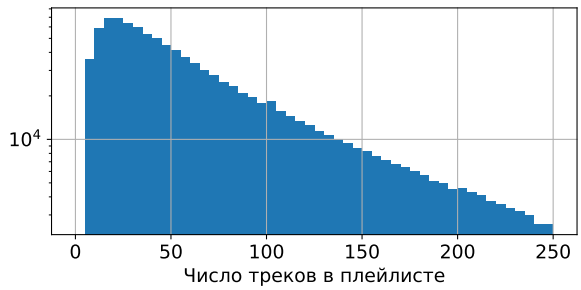
(c) Гистограмма числа положительных оценок фильма

Рис. 1: Характеристики датасета MovieLens 20M

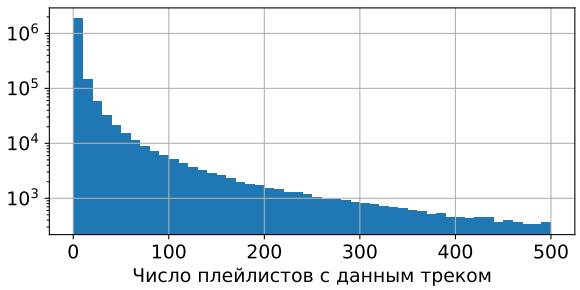
ше 4. Все оставшиеся примеры считаются положительными отзывами независимо от конкретной оценки.

5.2 Million Playlist Dataset

Данный датасет был собран сервисом онлайн-стриминга музыки `spotify.com` для соревнования RecSys Challenge 2018 [15]. Объектами в данном датасете являются



(a) Гистограмма размера плейлистов



(b) Гистограмма числа плейлистов на трек

Рис. 2: Характеристики датасета Million Playlist

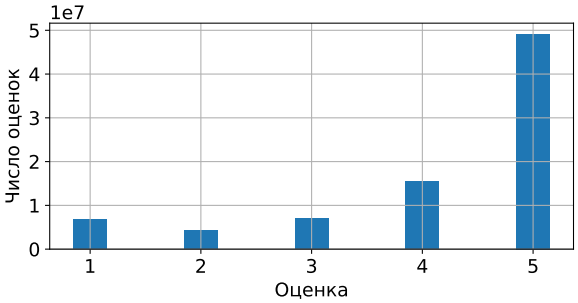
аудио-треки, а запросами — плейлисты, составленные пользователями сервиса. В датасете представлен 1 миллион плейлистов, каждый из которых содержит не менее 5 и не более 250 треков. Распределение популярности плейлистов и треков представлено на рис. 2.

Этот датасет является примером датасета с неявным откликом — для каждого плейлиста известно только, какие треки ему точно релевантны (то есть содержатся в нем), а про все остальные треки не известно ничего. Поэтому все треки, не включенные в плейлист, считаются не релевантными этому плейлисту.

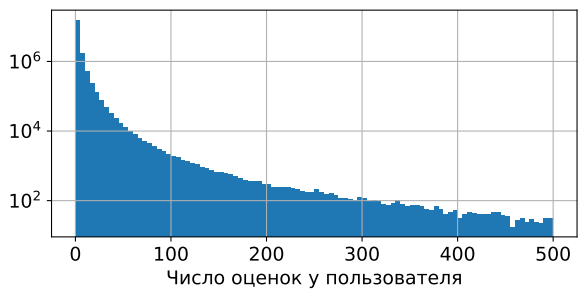
Для экспериментов из датасета были удалены все треки, которые встречаются только в одном плейлисте.

5.3 Amazon Review

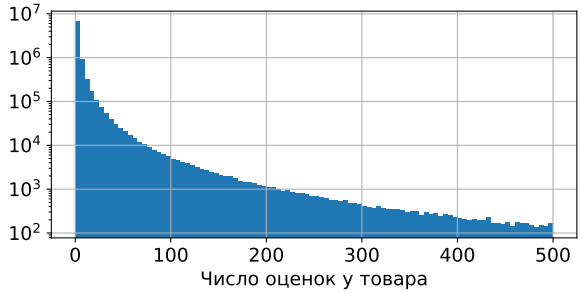
Данный датасет состоит из оценок, оставленных пользователями интернет-магазина `amazon.com` [6, 8]. Оценки — целые числа от 1 до 5. Как и в случае



(a) Распределение оценок



(b) Гистограмма числа положительных оценок пользователя



(c) Гистограмма числа положительных оценок товара

Рис. 3: Характеристики датасета Amazon Review

MovieLens, из датасета были удалены все оценки меньше 4, а для оставшихся конкретное значение оценок не учитывалось.

5.4 Сравнение датасетов

Датасет	**Число запросов**	**Число документов**	**Число оценок**	**Плотность матрицы R**
MovieLens 100K	942	1 447	5.5×10^4	0.041
MovieLens 1M	6 038	3 533	5.75×10^5	0.027
MovieLens 10M	6.98×10^4	1.03×10^4	5×10^6	0.007
MovieLens 20M	1.38×10^5	2.07×10^4	1×10^7	0.0035
Million Playlist	6.63×10^7	1×10^6	1.19×10^6	5.49×10^{-5}
Amazon Review	1.8×10^7	8.64×10^6	6.46×10^7	4.15×10^{-7}

Таблица 1: Сравнение рассмотренных датасетов

6 Эксперименты

Для подтверждения работоспособности метода были проведены эксперименты на датасетах, описанных в разделе 5 — MovieLens 100K, MovieLens 1M, MovieLens 10M, MovieLens 20M, Million Playlist Dataset и Amazon Review. В качестве алгоритмов были испытаны ALS, BPR и SVD++, описанные в разделе 3. Для всех трех алгоритмов использовалась размерность скрытого пространства 64 и значения остальных параметров по умолчанию — они в меньшей степени влияют на конечный результат. Для ALS и BPR была использована реализация из python-библиотеки `implicit` [9], для SVD++ — из python-библиотеки `surprise` [17].

Все датасеты были разбиты на обучающую и тестовую выборку в соотношении 9:1. Для датасетов MovieLens и Amazon Review в тестовую выборку попали 10% наиболее поздних оценок. В датасете Million Playlist время добавления трека в плейлист не известно, поэтому в тестовую выборку попали случайные 10% пар (плейлист,

трек) из датасета. Случайные блуждания строились только по обучающей выборке, тестовая выборка оставалась неизменной.

У случайных блужданий есть ряд параметров, для которых в ходе экспериментов подбирались оптимальные значения для каждой задачи. Были проверены следующие параметры:

- Начинать случайные блуждания с пользователя (с плейлиста в случае MPD) или объекта. Во всех случаях качество оказалось выше, если начинать с пользователя.

- Влияние степени вершины на вероятность перехода в нее. Вероятность перехода выбиралась пропорциональной $deg(v)^K$. Для K были проверены значения $-1, -0.5, 0, 0.5, 1$.

- Количество семплированных случайных блужданий N. Были проверены значения $10^5, 10^6, 10^7, 10^8$.

- Случайные блуждания фиксированной длины либо с вероятностью остановки. Для блужданий фиксированной длины были проверены значения длины $L = 3, 5, 7$. Для вероятностных блужданий были проверены вероятности остановки $P = 0.9, 0.8, 0.7$.

В таблице 2 приведены результаты работы предложенного метода для precision at 10, в таблице 3 — для NDCG. Оптимальные параметры для каждой задачи указаны в последнем столбце таблиц. Более детальное исследование зависимости качества алгоритмов от параметров метода представлено в разделе 6.1.

Датасет	**Алгоритм**	**P@10 исх.**	**P@10 модиф.**	**Оптимальные параметры RW**
MovieLens 100K	ALS	0.0805	**0.0871**	$K: -0.5, P: 0.8, N: 10^6$
	BPR	0.1098	**0.1195**	$K: -0.5, L: 5, N: 10^7$
	SVD++	**0.0837**	0.0813	$K: 0, P: 0.9, N: 10^6$
MovieLens 1M	ALS	0.1255	**0.1365**	$K: 0, L: 3, N: 10^6$
	BPR	0.1345	**0.1942**	$K: 0, L: 3, N: 10^7$
	SVD++	**0.1391**	0.1380	$K: 0, L: 3, N: 10^6$
MovieLens 10M	ALS	0.1215	**0.1245**	$K: 0, P: 0.8, N: 10^7$
	BPR	0.1496	**0.1788**	$K: 0, P: 0.9, N: 10^8$
	SVD++	0.1308	**0.1311**	$K: 0, P: 0.9, N: 10^8$
MovieLens 20M	ALS	0.1174	**0.1308**	$K: 0, P: 0.9, N: 10^8$
	BPR	0.1514	**0.1875**	$K: 0.5, P: 0.9, N: 10^8$
	SVD++	0.1287	**0.1332**	$K: 0, L: 3, N: 10^8$
Amazon Review	ALS	0.0115	**0.0121**	$K: 0, P: 0.9, N: 10^8$
	BPR	0.0125	**0.0145**	$K: 0, P: 0.9, N: 10^8$
	SVD++	**0.0102**	0.0093	$K: 0, P: 0.9, N: 10^8$
Million Playlist	ALS	0.1556	**0.1783**	$K: 0, L: 3, N: 10^8$
	BPR	0.1842	**0.2196**	$K: 0, L: 3, N: 10^8$
	SVD++	0.1911	**0.2123**	$K: 0, L: 3, N: 10^8$

Таблица 2: Precision at 10 с применением предложенного метода и без него

Датасет	**Алгоритм**	**NDCG исх.**	**NDCG модиф.**	**Оптимальные параметры RW**
MovieLens 100K	ALS	0.0755	**0.0798**	$K: -0.5, P: 0.9, N: 10^7$
	BPR	0.1173	**0.1208**	$K: -0.5, L: 3, N: 10^6$
	SVD++	**0.0811**	0.0801	$K: 0, P: 0.9, N: 10^6$
MovieLens 1M	ALS	0.1190	**0.1374**	$K: 0, P: 0.9, N: 10^7$
	BPR	0.1348	**0.1913**	$K: 0, L: 3, N: 10^7$
	SVD++	0.1311	**0.1410**	$K: 0, L: 3, N: 10^6$
MovieLens 10M	ALS	0.1165	**0.1180**	$K: 0, P: 0.9, N: 10^7$
	BPR	0.1488	**0.1721**	$K: 0, P: 0.9, N: 10^8$
	SVD++	0.1223	**0.1289**	$K: 0, P: 0.9, N: 10^8$
MovieLens 20M	ALS	0.1112	**0.1191**	$K: 0, P: 0.9, N: 10^8$
	BPR	0.1461	**0.1704**	$K: 0, P: 0.9, N: 10^8$
	SVD++	0.1232	**0.1279**	$K: 0, L: 3, N: 10^8$
Amazon Review	ALS	0.0101	**0.0112**	$K: 0, P: 0.9, N: 10^8$
	BPR	0.0119	**0.0132**	$K: 0, P: 0.9, N: 10^8$
	SVD++	**0.0095**	0.0092	$K: 0, P: 0.9, N: 10^8$
Million Playlist	ALS	0.1432	**0.1701**	$K: 0, P: 0.9, N: 10^8$
	BPR	0.1821	**0.2121**	$K: 0, L: 3, N: 10^8$
	SVD++	0.1887	**0.2056**	$K: 0, L: 3, N: 10^8$

Таблица 3: Normalized DCG с применением предложенного метода и без него

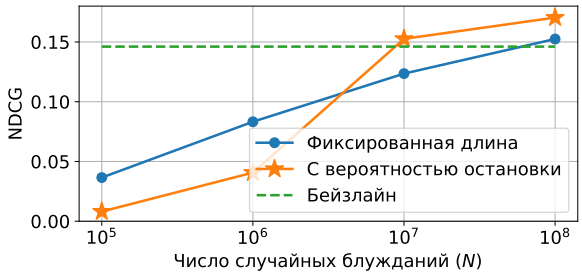
Эксперименты показывают, что для большинства рассмотренных задач и алгоритмов удается получить больший или меньший прирост качества за счет использования предложенного метода аугментации датасета. Для Bayesian Personalized Ranking прирост оказывается более значимым, предположительно, это связано с тем, что попарная функция ошибки лучше подходит для модифицированной обучающей выборки.

6.1 Исследование параметров метода

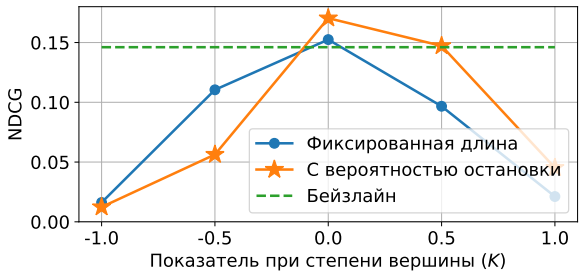
В данном разделе все эксперименты проводились на датасете MovieLens 20M с алгоритмом Bayesian Personalized Ranking. В качестве метрики качества использовалась NDCG как более чувствительная. Для остальных датасетов и алгоритмов получаются схожие результаты.

Для всех параметров кроме того, по которому исследуется зависимость, были зафиксированы параметры, дающие наибольшее значение NDCG. Для блужданий фиксированной длины $K = 0, L = 3, N = 10^8$, для блужданий с вероятностью остановки $K = 0, P = 0.9, N = 10^8$.

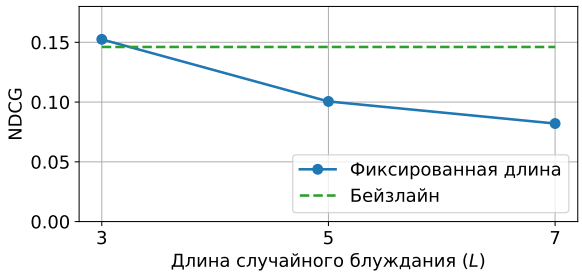
Результаты экспериментов приведены на рис. 4. На всех графиках пунктирной линией обозначена точность алгоритма, обученного на исходной, не аугментированной, выборке.



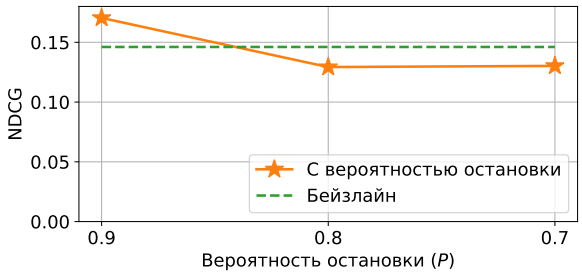
(a) От числа семплированных блужданий



(b) От показателя при степени вершины



(c) От длины случайного блуждания



(d) От вероятности остановки блуждания

Рис. 4: Зависимость NDCG алгоритма BPR на датасете MovieLens 20M от параметров случайных блужданий

Как видно из графиков, метод очень чувствителен к отклонению параметров от оптимальных значений. Параметр K, отвечающий за зависимость вероятности пере-

хода в вершину от степени конечной вершины, сильнее остальных влияет на конечную точность алгоритма. Точность ожидаемо растет с увеличением числа семплированных примеров, однако дальнейшее увеличение их числа слишком вычислительно затратно.

7 Заключение

В работе был предложен метод аугментации для задачи рекомендаций с неявным откликом. Метод основан на многократном применении случайных блужданий к двудольному графу пользователь-объект.

Для подтверждения работоспособности метода был проведен ряд экспериментов на общедоступных наборах данных разных природы и объема. В качестве базового алгоритма рекомендаций были использованы три алгоритма обучения скрытых представлений, хорошо зарекомендовавшие себя в практических задачах. Проведенные эксперименты показывают, что данный метод в значительном числе случаев позволяет повысить качество рекомендаций. Также было проведено исследование влияния параметров метода на итоговую точность.

По результатам работы сделан доклад на международной научной конференции студентов, аспирантов и молодых учёных «Ломоносов-2019» [20].

Список литературы

1. Augmented collaborative filtering for sparseness reduction in personalized POI recommendation / C. Cui, J. Shen, L. Nie et al. // *ACM Transactions on Intelligent Systems and Technology (TIST)*. — 2017. — Vol. 8, no. 5. — P. 71.

2. *Bennett J., Lanning S. et al.* The Netflix Prize // Proceedings of KDD cup and workshop / New York, NY, USA. — Vol. 2007. — 2007. — P. 35.

3. BPR: Bayesian personalized ranking from implicit feedback / S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme // Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence / AUAI Press. — 2009. — Pp. 452–461.

4. Data augmentation using synthetic data for time series classification with deep residual networks / H. I. Fawaz, G. Forestier, J. Weber et al. // *arXiv preprint arXiv:1808.02455.* — 2018.

5. *Harper F. M., Konstan J. A.* The MovieLens datasets: History and context // *ACM Transactions on Interactive Intelligent Systems (TIIS).* — 2016. — Vol. 5, no. 4. — P. 19.

6. *He R., McAuley J.* Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering // Proceedings of the 25th International Conference on World Wide Web / International World Wide Web Conferences Steering Committee. — 2016. — Pp. 507–517.

7. HOP-Rec: High-order proximity for implicit recommendation / J.-H. Yang, C.-M. Chen, C.-J. Wang, M.-F. Tsai // Proceedings of the 12th ACM Conference on Recommender Systems / ACM. — 2018. — Pp. 140–144.

8. Image-based recommendations on styles and substitutes / J. McAuley, C. Targett, Q. Shi, A. Van Den Hengel // Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval / ACM. — 2015. — Pp. 43–52.

9. Implicit: Fast python collaborative filtering for implicit datasets. — `https://github.com/benfred/implicit`.

10. *Jamali M., Ester M.* A matrix factorization technique with trust propagation for recommendation in social networks // Proceedings of the fourth ACM conference on Recommender systems / ACM. — 2010. — Pp. 135–142.

11. *Koren Y.* Factorization meets the neighborhood: a multifaceted collaborative filtering model // Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining / ACM. — 2008. — Pp. 426–434.

12. *Linden G., Smith B., York J.* Amazon.com recommendations: Item-to-item collaborative filtering // *IEEE Internet computing.* — 2003. — no. 1. — Pp. 76–80.

13. *Perez L., Wang J.* The effectiveness of data augmentation in image classification using deep learning // *arXiv preprint arXiv:1712.04621.* — 2017.

14. *Pilászy I., Zibriczky D., Tikk D.* Fast ALS-based matrix factorization for explicit and implicit feedback datasets // Proceedings of the fourth ACM conference on Recommender systems / ACM. — 2010. — Pp. 71–78.

15. RecSys challenge 2018: Automatic playlist continuation / M. Schedl, H. Zamani, C.-W. Chen et al. // Late-Breaking/Demos Proc. 18th Int. Soc. Music Information Retrieval Conf. — 2017.

16. *Rendle S.* Factorization machines with libFM // *ACM Transactions on Intelligent Systems and Technology (TIST).* — 2012. — Vol. 3, no. 3. — P. 57.

17. Surprise: A python scikit for recommender systems. — `http://surpriselib.com`.

18. Two-stage model for automatic playlist continuation at scale / M. Volkovs, H. Rai, Z. Cheng et al. // Proceedings of the ACM Recommender Systems Challenge 2018 / ACM. — 2018. — P. 9.

19. The YouTube video recommendation system / J. Davidson, B. Liebald, J. Liu et al. // Proceedings of the fourth ACM conference on Recommender systems / ACM. — 2010. — Pp. 293–296.

20. *Коваленко П.* Использование случайных блужданий по графу для повышения качества рекомендаций // Сборник тезисов XXVI Международной научной конференции студентов, аспирантов и молодых ученых «Ломоносов-2019». — Макс-Пресс, 2019.