



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

<b>1</b>	<b>Введение</b>	<b>2</b>
<b>2</b>	<b>Обзор предметной области</b>	<b>2</b>
2.1	Математическая постановка задачи рекомендаций . . . . .	3
2.2	Метрики качества рекомендаций . . . . .	3
<b>3</b>	<b>Рассматриваемые алгоритмы</b>	<b>4</b>
3.1	Singular Vector Decomposition . . . . .	4
3.2	Bayesian Personalized Ranking . . . . .	5
3.3	Random walks . . . . .	5
3.4	High-Order Proximity for Implicit Recommendation . . . . .	6
<b>4</b>	<b>Предлагаемый метод</b>	<b>7</b>
<b>5</b>	<b>Описание датасетов</b>	<b>7</b>
5.1	MovieLens . . . . .	7
5.2	Million Playlist Dataset . . . . .	7
5.3	Amazon Review . . . . .	8
5.4	Сравнение датасетов . . . . .	8
<b>6</b>	<b>Эксперименты</b>	<b>8</b>

# 1 Введение

Рекомендательные системы — это семейство алгоритмов, предназначенных для выделения из широкого множества объектов подмножества, релевантного поданному на вход запросу.

Вода: что такое рекомендательные системы, примеры задач.

Частым случаем использования рекомендательных систем является построения персонализированных рекомендаций для пользователя некоторого сервиса. Например, рекомендации фильмов, музыки, товаров в интернет-магазине и т.д. В этом случае запросом является пользователь, а множеством объектов для рекомендации — все доступные книги, музыка и товары соответственно. Далее в работе будет рассматриваться задача рекомендаций объектов (или документов) пользователю, однако все идеи легко переносятся на более общий случай.

## 2 Обзор предметной области

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

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 далее.

Данная классификация не покрывает все возможные случаи. Существуют модели, которые используют похожесть между пользователями совместно с признаками пользователей и объектов, например, факторизационные машины [ссылка или опи-

сание]. Также часто коллаборативные модели используются как признаки для обучения контентных моделей [ссылки на победителей рексисов].

## 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 постановка — для каждого пользователя  $u$  найти  $K$  максимально релевантных объектов  $i_1, \dots, i_K$ , 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-оценки качества алгоритма обычно разбивают множество всех оценок на две части — обучающую и тестовую выборку. Разбиение может быть случайным или по времени, когда поставлена оценка, если оно известно.

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

Обозначим множество всех оценок пользователя из обучающей выборки как  $I_{train}$ , из тестовой выборки — как  $I_{test}$ ,  $I_{train} \cap I_{test} = \emptyset$ . Результатом работы алгоритма ранжирования можно считать последовательность всех доступных для ранжирования объектов  $I \setminus I_{train}$ , упорядоченную по предсказанной алгоритмом релевантности  $r_{ui}$ .

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.  $P@k = \frac{|I_{pred}[:k] \cap I_{test}|}{k}$ .

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

**Average precision at  $k$**  — средняя точность по первым  $1, 2, \dots, k$  объектам.  $AP@k = \frac{1}{k} P@k$ .

**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:  $DCG@k = \sum_{i=1}^k \frac{\mathbb{I}[I_{pred}[k] \in I_{test}]}{\log_2(k+1)}$ . В отличие от предыдущих метрик, для DCG максимальным возможным значением является не 1. Поэтому чаще используют следующую метрику.

**Normalized Discounted Cumulative Gain at  $k$**  — DCG, нормированный на максимальное возможное значение метрики, поэтому изменяется на отрезке  $[0, 1]$ .

$$nDCG@k = \frac{DCG@k}{DCG^*@k}, \quad DCG^*@k = \sum_{i=1}^{\min(|I_{test}|, k)} \frac{1}{\log_2(k+1)}.$$

## 3 Рассматриваемые алгоритмы

### 3.1 Singular Vector 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 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:

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

Данная модель часто называется Singular Vector Decomposition (SVD), поскольку в случае полностью заполненной матрицы  $R$  оптимальными значениями для  $p$  и  $q$  являются соответствующие столбцы из сингулярного разложения матрицы  $R$ :

$R = P\Sigma Q^T$  [ссылка]. 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 ALS. [ссылка на статью Миши]

### 3.2 Bayesian Personalized Ranking

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

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

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. Итоговый функционал можно записать в следующем виде:

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

### 3.3 Random walks

Пусть задан ориентированный граф. Неформально определить случайное блуждание можно следующим образом. Есть некий агент, который перемещается по графу, стартуя в некоторой вершине  $v_0$ . Далее в каждый момент времени  $t$  он перемещается из вершины  $v_t$  в случайную вершину  $v_{t+1}$ , соседнюю с вершиной  $v_t$ . Вероятности переходов между вершинами являются частью алгоритма, обычно вероятности равные или зависят от степени 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  $K$  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.

### 3.4 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 [хорошо бы ссылку на пруф]. В статье [ссылка!] 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 пользователь, 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 разделе настоящей работы предлагается развитие и более полное исследование идей, предложенных в данной статье.

## 4 Предлагаемый метод

// Кажется, тут чего-то не хватает

## 5 Описание датасетов

Для проверки работоспособности метода, описанного в разделе [4], 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 [ссылка]. Датасет предоставляется в четырех вариантах: `MovieLens 100K`, `MovieLens 1M`, `MovieLens 10M` и `MovieLens 20M`, которые содержат  $10^5$ ,  $10^6$ ,  $10^7$  и  $2 \times 10^7$  оценок соответственно. Оценки — целые или полуцелые числа от 1.0 до 5.0. Распределение оценок для датасета `MovieLens 20M` представлено на рис. ????. 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 меньше 4. Все оставшиеся примеры считаются положительными отзывами независимо от конкретной оценки. Оценки были разделены на обучающую и тестовую выборки по времени, в тестовую выборку попали 10% наиболее поздних оценок.

// Неплохо бы нарисовать распределение числа оценок на пользователя и на фильм.

### 5.2 Million Playlist Dataset

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

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 менее чем в трех плейлистах.

### 5.3 Amazon Review

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

### 5.4 Сравнение датасетов

- \* Количество запросов
- \* Количество объектов
- \* Количество оценок
- \* Плотность матрицы оценок

## 6 Эксперименты

Для подтверждения работоспособности метода были проведены эксперименты на датасетах, описанных в разделе 5 — MovieLens 100K, MovieLens 1M, MovieLens 10M, MovieLens 20M и Million Playlist Dataset. 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.

В таблице 1 приведены результаты работы предложенного метода для precision at 10, в таблице 2 — для NDCG.

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  $L = 1, 3, 5, 7$ . Для вероятностных блужданий были проверены вероятности остановки  $P = 1.0, 0.9, 0.8, 0.7$ .

Оптимальные параметры для каждой задачи указаны в последнем столбце таблиц. Более детальное исследование зависимости качества алгоритмов от параметров метода будет проведено в разделе ???.

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

Датасет	Алгоритм	P@10 исх.	P@10 модиф.	Оптимальные параметры RW
MovieLens 100K	ALS	0.0805	<b>0.0871</b>	$K : -0.5, P : 0.8, N : 10^6$
	BPR	0.1098	<b>0.1195</b>	$K : -0.5, L : 5, N : 10^7$
	SVD++	<b>0.0837</b>	0.0813	$K : 0, P : 0.9, N : 10^6$
MovieLens 1M	ALS	0.1255	<b>0.1365</b>	$K : 0, L : 3, N : 10^6$
	BPR	0.1345	<b>0.1942</b>	$K : 0, L : 3, N : 10^7$
	SVD++	<b>0.1391</b>	0.1380	$K : 0, L : 3, N : 10^6$
MovieLens 10M	ALS	0.1215	<b>0.1245</b>	$K : 0, P : 0.8, N : 10^7$
	BPR	0.1496	<b>0.1788</b>	$K : 0, P : 0.9, N : 10^8$
	SVD++	0.1308	<b>0.1311</b>	$K : 0, P : 0.9, N : 10^9$
MovieLens 20M	ALS	0.1174	<b>0.1308</b>	$K : 0, P : 0.9, N : 10^8$
	BPR	0.1514	<b>0.1875</b>	$K : 0.5, P : 0.9, N : 10^8$
	SVD++	0.1287	<b>0.1332</b>	$K : 0, L : 3, N : 10^8$
Amazon Review	ALS	0.0115	<b>0.0121</b>	$K : 0, P : 0.9, N : 10^8$
	BPR	0.0125	<b>0.0145</b>	$K : 0, P : 0.9, N : 10^8$
	SVD++	<b>0.0102</b>	0.0093	$K : 0, P : 0.9, N : 10^8$
Million Playlist	ALS	0.1556	<b>0.1783</b>	$K : 0, L : 3, N : 10^8$
	BPR	0.1842	<b>0.2196</b>	$K : 0, L : 3, N : 10^8$
	SVD++	0.1911	<b>0.2123</b>	$K : 0, L : 3, N : 10^8$

Таблица 1: Precision at 10 с применением предложенного метода и без него

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

Таблица 2: Normalized DCG с применением предложенного метода и без него