Санкт–Петербургский государственный университет
Кафедра технологии программирования

Бельков Роман Андреевич

Выпускная квалификационная работа

Калибровка рекомендательных систем. Поиск гетерогенного эффекта и нетипичных пользователей для задач рекомендаций контента

Направление 01.03.02
«Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2015 «Прикладная математика, фундаментальная информатика и программирование»

Научный руководитель:
кандидат техн. наук,
доцент
Блеканов И. С.

Рецензент:
старший преподаватель
Давыденко А. А.

Санкт-Петербург
2020 г.

Содержание

Введение

В настоящее время, количество информации растет очень быстрыми темпами и чтобы предоставлять наиболее релевантную и полезную для пользователей информацию, в веб сервисах начали использовать рекомендательные системы. Рекомендательные системы обеспечивают персонализированный пользовательский опыт во многих различных областях применения, включая интернет-магазины, социальные сети и потоковое воспроизведение музыки/видео. Рекомендательная система – это алгоритм, который предсказывает наиболее интересные объекты для конкретного пользователя на основе некоторой информации о нем.

Одна из проблем, которые имеют рекомендательные системы – это удовлетворение не всех интересов пользователя, рассмотрим пример рекомендательной системы фильмов. Если пользователь посмотрел, 80 артхаусных фильмов и 20 комедий, то вполне разумно ожидать, что персонализированный список рекомендуемых фильмов будет состоять примерно из 80% артхаусных и 20% комедий. Это важное свойство, известное как калибровка, недавно получило новое внимание в контексте справедливости машинного обучения. В рекомендуемом списке элементов калибровка гарантирует, что различные области интересов пользователя будут отражены в соответствующих пропорциях.

Калибровка особенно важна в свете того факта, что рекомендательные системы, оптимизированные в сторону точности в обычном автономном режиме, могут легко привести к рекомендациям, где меньшие интересы пользователя вытесняются основными интересами пользователя. В этой статье мы покажем, что рекомендательные системы, обученные точности, могут легко генерировать списки рекомендуемых элементов, которые фокусируются на основных областях интересов пользователя, в то время как меньшие области интересов пользователя, как правило, недопредставлены или даже отсутствуют. Со временем такие несбалансированные рекомендации несут в себе риск постепенного сужения областей интересов пользователя – что аналогично эффекту пузыря фильтров. Эта проблема также применима в случае нескольких пользователей, совместно использующих одну учетную запись, когда интере-

сы менее активных пользователей в рамках одной учетной записи могут быть вытеснены в рекомендациях.

Калибровка – это общая концепция машинного обучения, и в последнее время она переживает возрождение в контексте справедливости алгоритмов машинного обучения. Алгоритм классификации называется калиброванным, если прогнозируемые пропорции различных классов согласуются с фактическими пропорциями точек данных в имеющихся данных.

Постановка задачи

Основная цель работы заключается в реализации алгоритма калибровки рекомендательных систем и сравнении с существующими решениями. Реализуемый алгоритм должен не сильно ухудшать точность работы рекомендательной системы, но в то же время учитывать все интересы пользователя.

Для достижения цели были поставлены следующие задачи:

1. Обзор существующих методов калибровки рекомендательных систем.
2. Реализация алгоритма на языке Python.
3. Поиск или сбор данных и проверка реализованного метода на данных.
4. Сравнение результатов с существующими методами.

Обзор литературы

Калибровка рекомендация на данный момент имеет несколько направлений, первое заключается в получении более справедливых рекомендаций, а второе в учете всех интересов пользователя при генерации рекомендаций.

Рассмотрим первое направление. Как правило, в рекомендательных системах речь идет о рекомендациях товаров пользователям. Однако в контексте справедливости имеет смысл абстрагироваться от объектов и субъектов. Например, при рекомендациях предложений о работе испытуемыми являются люди, которые могут подвергаться дискриминации по признаку расы или пола; при поиске поездки рекомендуемыми объектами являются таксисты; в социальных сетях предложения друзей включают отдельных людей в качестве субъектов и объектов. Таким образом, справедливость в рекомендателями могут иметь несколько точек зрения, как описано в [1]: справедливость для субъектов, называются потребительской справедливостью или C-справедливостью, а справедливость для объектов, называется справедливостью производителя или P-справедливостью.

Дискриминация в рекомедательных системах является отдельной задачей, которая достаточно хорошо описана в статье об исследовании честности иснтрументов для прогнозирования рецидива преступлений [2].

Второе направление более сосредоточено на диверсификации, полноте и разнообразии рекомендации. Основным интсрументом калибровки является переранжирование списка уже сгенерированных рекомендаций. В работе [3] используется выделение подпрофилей, на основе положительных оценок, для каждого пользователя, которые отражают определенные интересы. Другой метод предлагают исследователи из Netflix [4], он основан на расстоянии Кульбака-Лейблера, которое применяется для опеределения схожести пользовательского распределения и сгенерированнного. В исследовании [5] используется метод Сент-Лагю для выбора N наиболее релевантных и разнообразных рекомендаций. Обычно метод Сент-Лагю применяется для распределения мест в правительстве, но в рассмотренной работе, его модифицировали для рекомендаций фильмов.

Глава 1. Обзор существующих решений в области калибровки рекомендательных систем

1.1 Обзор существующих сервисов и инструментов

Рекомендательные системы все чаще используются в различных сервисах. Очень трудно найти сервис стримингового просмотра видео контента или прослушивания музыки, в котором бы не применялись рекомендательные механизмы. Рассмотрим крупнейшие сервисы, внедрившие рекомендательные системы.

Наиболее часто рекомендации применяются для рекомендаций фильмов. **Netflix** – одна из передовых компаний в области рекомендаций контента, в которой на данный момент применятся огромное количество подходов и методов для рекомендаций не только фильмов и сериалов, но и постеров, а иногда для порядка серий. Но не одни Netflix занимаются видео контентом. **MovieLens** – это веб-сайт, разработанный исследовательской группой из университета Миннесоты GroupLens, который помогает людям найти фильмы для просмотра. У него есть сотни тысяч зарегистрированных пользователей. В MovieLens проводят онлайн-полевые эксперименты в области автоматизированной рекомендации контента, интерфейсов рекомендаций, основанных на тегировании рекомендателей и интерфейсов, баз данных, и интеллектуального дизайна пользовательского интерфейса.

Помимо фильмов довольно часто рекомендательные системы применяются в музыкальных сервисах, например **Last.FM** – это веб-сайт, который собирает данные о прослушивании пользователями композиций и с помощью алгоритма коллаборативной фильтрации, формирует персонализированный и общий хит-парад. **Spotify** – интернет сервис для потокового прослушивания аудио. В этом сервисе применяется сразу несколько подходов в рекомендации композиций: первый, такой же как в Last.FM, основанный на поиске похожих пользователей, второй использует обработку естественного языка для анализа тектов песен, третий построен на нейронной сети, которая обрабатывает аудио состовляющую композиций.

Также, в последнее время, многи социальные сети начинают внедрять

рекомендательные алгоритмы в свои сервисы. Так, например **Instagram** применяют собственный фреймворк ig2vec [8], основаный на word2vec, для представления изображений в векторном виде, а дальше, сравнивая вектора пользователей, в онлайн режиме формируют бесконечную персонализированную ленту изображений.

Из открытых инструментов для построения рекомендательных систем, мне удалось найти фреймворк **SurPRISE** [9] для языка программирования Python, реализованный с помощью SciKits [10] (сокращенно от scipy Toolkits) – это дополнительные пакеты для SciPy [11], размещенные и разработанные отдельно и независимо от основного дистрибутива SciPy. SciPy – это основанная на Python экосистема программного обеспечения с открытым исходным кодом для математики, естественных наук и инженерии.

1.2 Обзор методов калибровки рекомедательных систем

1.2.1 Расстояние Кульбака-Лейблера

В ходе работы был проведен обзор литературы и найдено два метода потенциально подходящих для решения поставленной проблемы. Первый метод заключается в пересчете целевой метрики с помощью расстояния Кульбака-Лейблера (1) [4].

$$C_{KL}(p, q) = KL(p||\tilde{q}) = \sum_{g} p(g|u) \log \frac{p(g|u)}{\tilde{q}(q|u)}, \tag{1}$$

где p это целевое распределение жанра g для пользователя u, q – полученное распределение жанров для пользователя. Во избежание случая $q(g|u) = 0$, будем использовать

$$\tilde{q}(q|u) = (1 - \alpha) \cdot q(g|u) + \alpha \cdot p(g|u)$$

с очень маленьким $\alpha > 0$, такое что $q \approx \tilde{q}$.

Сама же калибровка выполняется по формуле:

$$I^* = \arg \max_{I,|I|=N} (1 - \lambda) \cdot s(I) - \lambda \cdot C_{KL}(p, q(I)), \tag{2}$$

где $\lambda \in [0, 1]$, которая определяет компромисс между расстоянием Кульбака-Лейблера и значением метрики полученным рекомендательной системой. $s(I) = \sum_{i \in I} s(i)$, где $s(i)$ – степень уверенности, что фильм i подойдет пользователю, предсказанная рекомендательной системой.

1.2.2 Метод Сент-Лагю

Второй алгоритм является адаптацией метода Сент-Лагю. [5] Метод Сент-Лагю, был изобретен французским математиком Андре Сент-Лагю для пропорционального распределения мандатов в правительстве. Суть метода заключается в поочередном присуждении мандатов партии с наибольшей квотой, которая на кажом шаге считается по формуле $\frac{V}{2s+1}$, где V – количество голосов, полученных партией, s – количество мандатов, выделенных партии на данном шаге.

Можно модифицировать данный метод под наш случай. Имея список наиболее релевантных фильмов, мы будем формировать новый список, выбирая фильмы по одному методом Сент-Лагю, только вместо партий у нас будут жанры, а голоса, полученные партией, заменятся на количество понравившихся пользователю фильмов конкретного жанра.

1.3 Обзор метрик качества методов калибровки рекомендаций

В качестве метрики качества рекомендательной системы будет использоваться точность (3) (precission) – это доля релевантных экземпляров среди извлеченных экземпляров [6], также называемая положительная прогностическая ценность и расчитывается по формуле

$$Precision = \frac{tp}{tp + tn}, \tag{3}$$

где tp – true positive, количество элементов корректно принятых алгоритмом, tn – true negative, количество элементов корректно отклоненных алгоримтом. В нашем случае tp будет количеством элементов, которые рекомендательная система порекомендует верно, а tn – количество элементов верно не рекомендованных алгоритмом.

Если же рекомендательная система работает не как бинарный классификатор, рекомендовать определенный элемент пользователю или нет, но и предсказывает, например оценку этого элемента конкретным пользователем, то можно использовать две похожие между собой метрики MSE (4) (mean squared error) – среднеквадратичная ошибка и RSME (5) (root mean squared error) – корень среднеквадратичной ошибки.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \tag{4}$$

где n – это количество предсказаний, Y_i реальные значения предсказываемой величины, $\hat{Y}_i$ предсказанные значения.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}. \tag{5}$$

Для вычисления схожести распределения реальных интересов пользователя и распределеня, полученного рекомендательной системой, имеется несколько метрик. Первый способ, это использование критерия согласия Пирсона [7], это непараметрический критерий, который применяют для проверки гипотезы о соответствии эмпирического распределения предполагаемому. Статистика критерия Пирсона вычисляется по формуле (6)

$$\chi^2 = \sum_{i=1}^{n}\frac{(k_i - p_i)^2}{p_i}, \tag{6}$$

где n – количество классов в распределении величины, k_i – фактические частоты встречаемости величины в i-ом классе, p_i – ожидаемые частоты встречаемости величины в i-ом классе. Чем меньше эта статистика, тем более схожи между собой распределения.

Еще один метод оценки схожести распределений предложен в статье от Netflix [4], там используется дивергенция Кульбака-Лейблера (1) из класса f-дивергенций $D_f(P \parallel Q)$, определяющих в общем случае несимметричную меру расхождения между двумя распределениями вероятностей P и Q.

Глава 2. Реализация программного комплекса калибровки результатов рекомендательных системы

2.1 Выбор технологий и интрументов для реализации

Для программной реализации методов калибровки рекомендательных систем были выбраны следующие инструменты:

- Python3 – скриптовый динамический язык программирования, выбран для реализации методов и использования готовых библиотек.
- Jupyter Notebook – веб-приложение для написания кода на языке Python с возможностью написания и запуска кода блоками, а не целиком.
- SurPrise [9] – фреймворк для языка программирования Python с готовой реализацией нескольких методов рекомендательных систем.
- Pandas [12] – это инструмент для анализа и манипулирования данными с открытым исходным кодом, построен на основе языка программирования Python.
- NumPy [13] – это библиотека для языка программирования Python, добавляющая поддержку больших многомерных массивов и матриц, а также большую коллекцию высокоуровневых математических функций для работы с этими массивами.
- Matplotlib [14] – это комплексная библиотека для создания статических, анимированных и интерактивных визуализаций в Python.
- SciPy [11] – это бесплатная библиотека Python с открытым исходным кодом, используемая для научных вычислений и технических вычислений.
- Implicit [15] – библиотека с реализованной коллаборативной фильтрацией для неизвестных данных.

2.2 Алгоритм рекомендательной системы

В качестве алгоритма рекомендаций было выбрано Байесовское персонализированное ранжирование [16]. Основная задача персонализированного ранжирования состоит в том, чтобы предоставить пользователю ранжированный список элементов.

Пусть U-множество всех пользователей,а I-множество всех элементов. Ниже на рисунке 1 показано, как обрабатываются неявные данные в случае общих рекомендателей элементов. Обычный подход заключается в том, что-

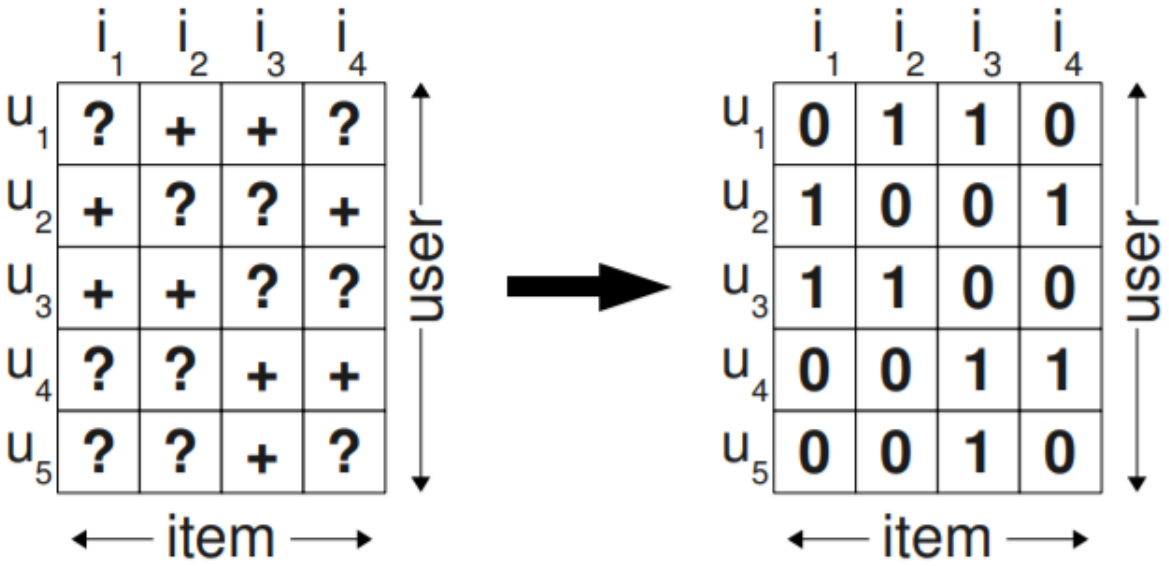

Рис. 1

бы предсказать персонализированную оценку для элемента, которая отражает предпочтения пользователя для этого элемента. После этого предметы будут ранжированы на основе этого балла. Здесь, как вы можете видеть на рисунке 1, все существующие взаимодействия между Пользователем и элементом помечаются как положительный класс(1), а остальные взаимодействия помечаются как отрицательный класс(0).

В подходе Байесовского персонализированного ранжирования, вместо взятия одного элемента, пары элементов будут рассматриваться как обучающие данные. Оптимизация будет выполняться на основе ранжирования этих

пар пользователь-элемент, а не только на основе взаимодействия пользователя и элемента. Набор данных, который будет рассматриваться, сформулирован следующим образом

$$(u, i, j) \in D_S \tag{7}$$

Семантика (7) заключается в том, что пользователь u, как предполагается, более предпочитает i, чем j.

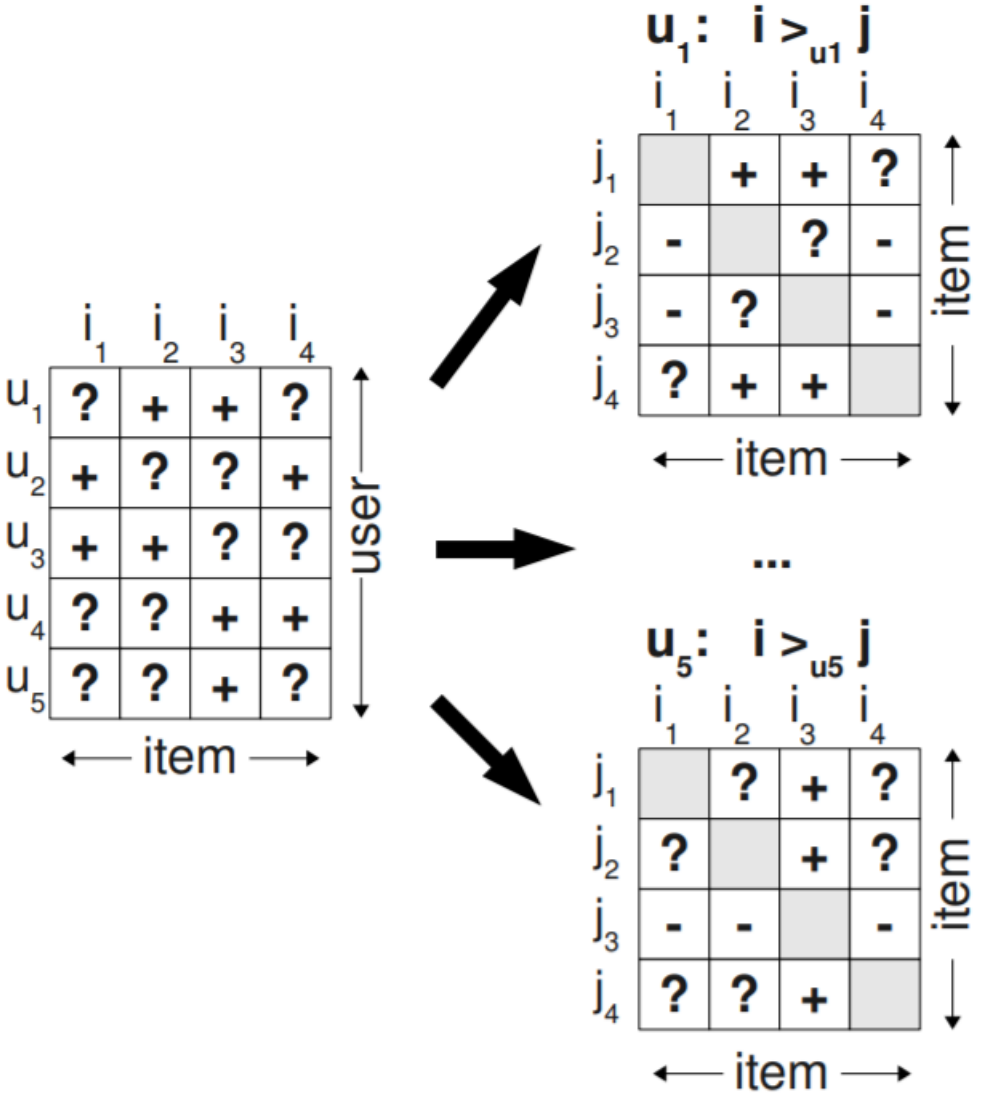

Рис. 2

Здесь триплеты, сгенерированные для обучающих данных, представляют собой специфические для пользователя попарные предпочтения между парой элементов.

На приведенном выше рисунке пользователь u_1 просматривал элемент i_2, но не элемент i_1, поэтому алгоритм предполагает, что этот пользователь предпочитает элемент i_2 i_1 $(i_2 > i_1)$ и дает положительный знак. Никакой вывод не может быть сделан о предпочтении для элементов, которые были замечены пользователем и показаны как отметка ?. То же самое верно и для двух элементов, которые пользователь еще не видел (например, элемент i_1 и i_4 для пользователя u_1). Напротив, вы можете наблюдать отрицательный знак для (i_1, j_2), поскольку пользователь предпочитает item2 item1.

Как и в любом байесовском подходе, в этом подходе мы имеем функцию правдоподобия, априорную вероятность и апостериорную вероятность. Байесовская формулировка нахождения правильного персонализированного ранжирования для всех элементов $i \in I$ заключается в максимизации следующей апостериорной вероятности, где Θ представляет вектор параметров произвольного класса моделей (например, матричная факторизация).

$$p(\Theta| >_u) \propto p(>_u |\Theta)p(\Theta) \tag{8}$$

Функция правдоподобия (8), где $>_u$ – это необходимые, но латентные предпочтения структуры для пользователя u.

Предполагая, что пользователи будут действовать независимо и порядок каждой пары элементов (i, j) для конкретного пользователя не зависит от порядка каждой другой пары, мы можем сформулировать индивидуальную вероятность того, что пользователь предпочитает элемент i элементу j следующим образом:

$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta)) \tag{9}$$

где σ логистическая сигмоида:

$$\sigma(x) := \frac{1}{1 + e^{-x}} \tag{10}$$

$\hat{x}_{uij}(\Theta)$ – это в приведенном выше уравнении (9) является вещественной значимой функцией, которая представляет собой отношение между пользователем u, элементом i и элементом j и обычно вычисляется с использовани-

ем модели матричного факторизации. Другими словами, баллы, отражающие отношение между пользователем u, элементом i и элементом j, будут рассчитываться с использованием данных триплетного обучения и матричного факторизации и завернуты в сигмоидную функцию (10). Эта сигмовидная функция дает индивидуальную вероятность, которая будет оптимизирована в ходе процесса.

$p(\Theta)$ - это априорная вероятность, представляющая собой нормальное распределение с нулевым средним и дисперсионно-ковариационной матрицей.

$$p(\Theta) \sim N(0, \Sigma_\Theta)$$

Следующее уравнение (11) является окончательным критерием Байесовского персонализированного ранжирования, который должен быть оптимизирован

$$\sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2 \tag{11}$$

где λ_Θ – специфические параметры регуляризации модели.

2.3 Реализация на языке программирования Python

Так как предполагается, что данные будут подаваться в файлах фората csv, то была выбрана библиотека pandas [12] для работы с данными. Данные представлены в формате id пользователя, id элемента, оценка элемента пользователем, класс, к которому принадлежит элемент. Выбирается порог оценки элемента, выше которого оценка считается положительной. Для дальнейшей работы представляем данные в виде разряженной таблицы, для более быстрой работы с большими таблицами. Из библиотеки Implicit [15] и метода BayesianPersonalizedRanking берем реализацию Байесовского персонализированного ранжирования. Разделяем исходные данные на тренировочную и тестовую выборки, 80% даннхы на тренировочную и 20% на тестовую. На тренировочной выборке строим алгоритм рекомендаций.

Далее реализуем метрики для проверки качества полученных рекомендаций. В качестве метрики для проверки точности рекомендаций используем precision (3) (точность), а в качестве метрики сравнения исходного распреде-

ления и полученного, будем использовать дивергенцию Кульбака-Лейблера (1). Для проверки будем брать 30 лучших рекомндаций для пользователя.

Калибровку рекомедаций будем производить двумя методами: первый (2) основан на дивергенции Кульбака-Лейблера, а второй является адаптацией метода Сент-Лагю [5].

Для визуализации результатов используется библиотека Matplotlib [14], с помощью нее строится график распределения классов рекомендованных элементов, для сравнения реального, сгенерированного рекомендательной системой и откалиброванного распределений.

Для сравнения с готовым решением будем использовать фреймворк SurPRISE [9] на языке программирования Python. Из набора реализованных во фреймворке методов используем основанный на предположении о нормальном распределении NormalPredictor.

Алгоритм прогнозирования случайной оценки основан на распределении обучающего множества, которое принято считать нормальным. Предсказание $\hat{r}_{ui}$ формируется из нормального распределения $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$, где $\hat{\mu}$ и $\hat{\sigma}$ оцениваются из обучающих данных, используя оценку максимального правдоподобия:

$$\hat{\mu} = \frac{1}{|R_{train}|} \sum_{r_{ui} \in R_{train}} r_{ui}$$

$$\hat{\sigma} = \sqrt{\sum_{r_{ui} \in R_{train}} \frac{(r_{ui} - \hat{\mu})^2}{|R_{train}|}}$$

Глава 3. Проведение эксперимента

3.1 Постановка эксперимента

Для проверки качества и сравнения алгоритмов было поставлено 3 эксперимента:

- Построение рекомендаций Байесовским персонализированным ранжированием и калибровка методом Кульбака-Лейблера [4].
- Построение рекомендаций Байесовским персонализированным ранжированием и калибровка методом Сент-Лагю [5].
- Построение рекомендаций с помощью готового фреймворка SurPRISE [9].

3.2 Набор данных

Для проведения эксперимента был выбран датасет MovieLens 25M [17], включающий в себя 25 миллионов оценок к 62,000 фильмов от 162,000 пользователей. Данные представленны в виде двух csv файлах. Первый rating.csv состоит из четырех колонок: userId – id пользователя, moviesId – id фильма, rating – оценка, поставленная по пятибальной шкале, пользователем для данного фильма, timestamp – временная отметка оценки.

userId	movieId	rating	timestamp
1	2	3.5	1112486027
1	29	3.5	1112486676
1	32	3.5	1112486819
1	47	3.5	1112486727
1	50	3.5	1112486580
1	112	3.5	1112486740
1	151	4	1112486734
1	223	4	1112486573
1	253	4	1112486940
1	260	4	1112486826

Второй файл movies.csv состоит из трех столбцов: moviesId – id фильма, title – название фильма, genres – жанры, к которым относится фильм.

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller

Во фреймворке SurPRISE [9] встроены несколько наборов данных, например MovieLens 1M, но для более честного сравнения, будем вручную загружать набор MovieLens 25M.

3.3 Результаты

В результате эксперимента 1 была произведена калибровка рекомендаций и подбор коэффициента λ для калибровки Кульбака-Лейблера (2) и были получены следующие результаты:

λ	Precision	C_{KL}
$\lambda = 0$ (нет калибровки)	0.43	0.93
$\lambda = 0.2$	0.49	0.47
$\lambda = 0.3$	0.45	0.39
$\lambda = 0.5$	0.33	0.31
$\lambda = 0.7$	0.21	0.25
$\lambda = 0.8$	0.18	0.11
$\lambda = 0.99$	0.14	0.019

Заметно, что с увеличением λ идет уменьшение дивергенции Кульбака-Лейблера между распределениями, следовательно полученные рекомендации больше

поможи на интересы пользователя. Но также с увеличением λ идет сначала небольшой рост точности, а после ее снижение. Для наилучшего результата, я считаю оптимальным взять $\lambda = 0.3$.

В эксперименте 2 после применения калибровки адаптированным методом Сент-Лагю [5], получились следующие результаты:

Precision	C_{KL}
0.12	0.3

Точность получилась заметно хуже исходной, а сходство полученного распределения интересов пользователя с реальным заметно меньше, чем при калибровке Кульбака-Лейблера с таким же показателем точности.

Эксперимент 3, при использовании готового фреймворка, показал достаточно хорошие результаты:

Precision	C_{KL}
0.53	0.61

этот метод показывает наилучшую точность, но не впечатляющие значения дивергенции Кульбака-Лейблера.

Для более наглядной демонстрации результатов привожу несколько графиков для сравнения распределений интересов пользователя. На графике отобразим долю каждого жанра среди интересов для пользователей. Синим цветом обонзачены исходные данные, на которых обучались алгоримы, оранжевым цветом обозначены данные, полученные рекомендательной системой, а зеленым цветом обонзачены откалиброванные данные, сгенерированные рекомендательной системой.

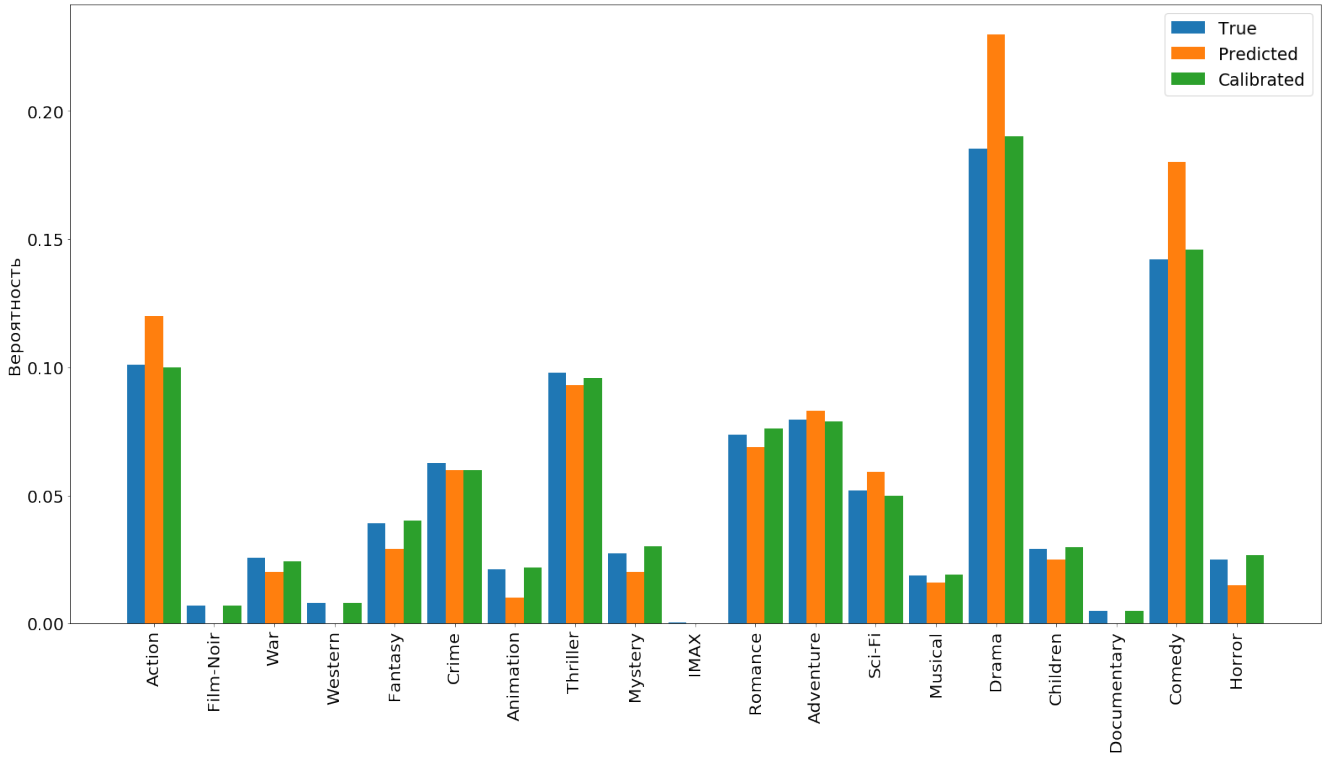

Рис. 3: Распределение жанров при Байесовском персонализированном ранжировании.

На рисунке 3 демнострируется эффект калибровки Кульбака-Лейблера с $\lambda = 0.3$ для Байесовского персонализированного ранжирования. Заметно, что жанры, которые больше остальных представлены в обучающих данных, выдаются рекомендательной системой еще чаще, в то время как калибровка приближает данные к реальным. С малочисленными жанрами ситуация обстоит немного иначе, Байесовское персонализированное ранжирование не представило некоторые жанры, например вестерн и документальное кино, калибровка же исправила это упущение. Таким образом, можно сделать вывод, что калиброванные рекомендации намного лучше удовлетворяют интересы пользователя.

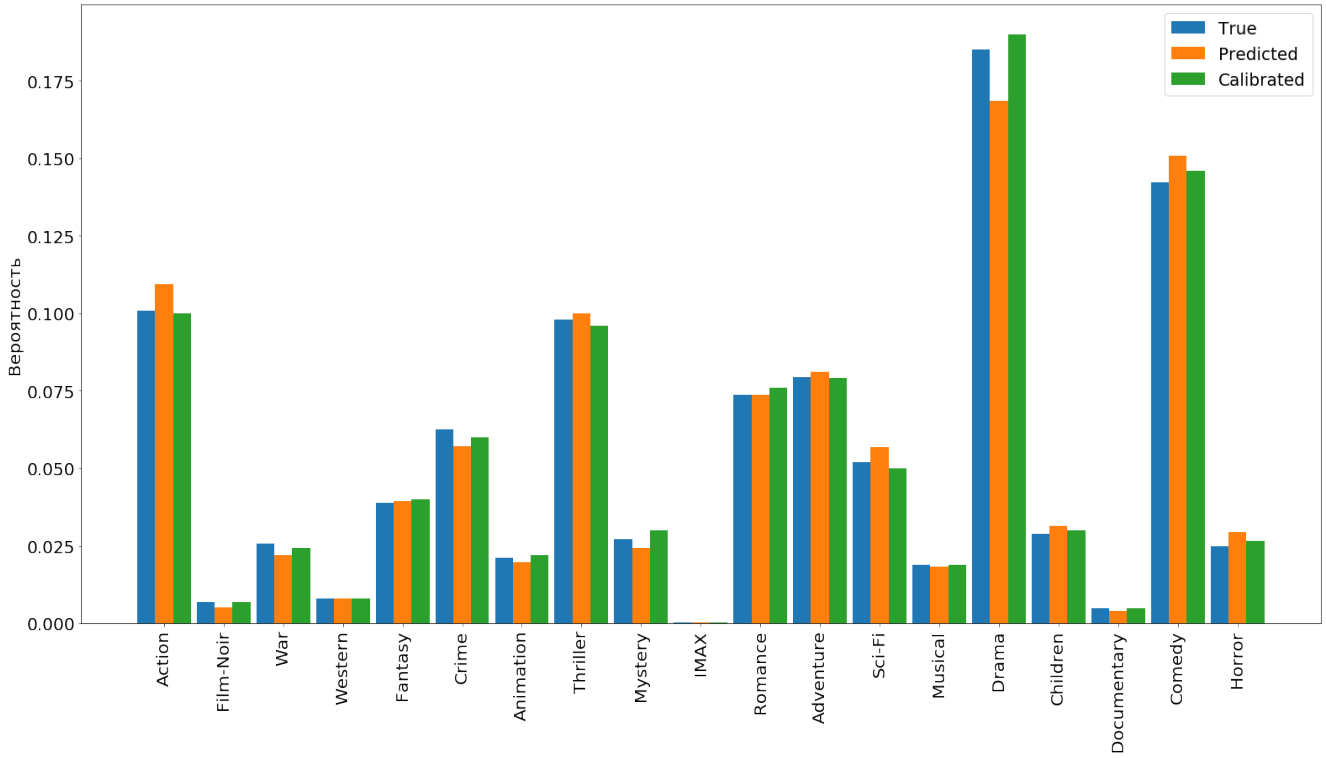

Рис. 4: Распределение жанров при использовании фреймворка SurPRISE.

На рисунке 4 сравниваются распределения, фареймворком SurPRISE и калибровкой Кульбака-Лейблера с $\lambda = 0.3$ для Байесовского персонализированного ранжирования. Тут уже нет серьезных проблем со стороны рекомендательной системы, все жанры представлены и имеют долю достаточно близкую к реальной. Калибровка в этом случае показывает все еще лучший результат, но разница уже не так велика.

По итогам экспериментов, можно сказать, что калибровка показала отличные результаты, интересы ползователей удовлетворяются достаточно хорошо и распределение жанров получается близким к реальному. Точность при калибровке может немного снизится, но я считаю, что пользовательский опыт при этом не пострадает, так как их интересы будут более широко представлены.

Заключение

Итоги работы

В рамках работы были выполнены следующие задачи:

- Обзор литературы на тему рекомендательных систем и их калибровки;
- Обзор существующих методов решения проблемы;
- Изучены и реализованы, на языке Python, два алгоритма калибровки рекомендаций;
- Проведены эксперименты на выбранном наборе данных;
- Проведено сравнение полученных результатов калибровки между собой и с готовой реализацией из фреймворка;
- Построены графики для наглядного сравнения распределения классов интересов пользователей.

Цель работы была достигнута, реализованный алгоритм калибровки рекомендательных систем заметно улучшает учет пользовательских интересов в рекомндациях и немного повышает точность рекомендаций. В сравнении с готовым решением, калибровка все еще лучше в плане удовлетворения пользовательских интересов, но по точности немного проигрывает.

Практическое применение

Реализованный алгорит можно применять практически в любой рекомендательной системе, где фигурируют какие-то классы объктов рекомендаций, а не только для фильмов, как представлено в эксперименте. Например в музыкальных сервисах можно заменить жанры фильмов на жанры музыки и использовать тот же метод. В социальных сетях вместо жанров можно использовать теги записей, направленность группп и каналов. В новостных агрегаторах ориентироваться на тематику новости. В интернет-магазинах классами будут являться категории товаров.

Дальнейшее развитие

В дальнейшем для улучшение алгоритма, можно заменить метод применяемы рекомендательной системой на более точный и калибровать его результаты, возможно точность в этом случает будет даже выше, чем в готовом фреймворке. Также, можно попробовать реализовать идею применения нескольких методов калибровки последовательно.

Список литературы

[1] Robin Burke «Multisided Fairness for Recommendation». Presented as a poster at the 2017 Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML 2017), arxiv.org/pdf/1707.00093.pdf.

[2] Alexandra Chouldechova «Fair prediction with disparate impact: A study of bias in recidivism prediction instruments». FATML 2016 conference paper. A long version of the paper available on the author's website, arxiv.org/pdf/1610.07524.pdf.

[3] Mesut Kaya, Derek Bridge «Accurate and Diverse Recommendations Using Item-Based SubProfiles». hirty-First International Florida Artificial Intelligence Research Society Conference, 2018, pp. 462–467 www.insight-centre.org/sites/default/files/publications/kaya-bridge-2017.pdf.

[4] Harald Steck «Calibrated Recommendations». RecSys '18: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 154–162, doi.org/10.1145/3240323.3240372.

[5] Van Dang and W. Bruce Croft «Diversity by Proportionality: An Election-based Approach to Search Result Diversification». SIGIR '12: Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, 2012, pp. 65–74, doi.org/10.1145/2348283.2348296.

[6] David M W Powers «Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation». Journal of Machine Learning Technologies, 2011, pp. 37–63, csem.flinders.edu.au/research/techreps/SIE07001.pdf.

[7] Кендалл М., Стьюарт А. «Статистические выводы и связи». Наука, 1973.

[8] Ivan Medvedev, Haotian Wu, Taylor Gordon «Powered by AI: Instagram's Explore recommender system». Facebook Artificial Intelligence, 2019.

[9] Nicolas Hug «Surprise a Python library for recommender systems», 2017. surpriselib.com.

[10] Официальный сайт документации SciKits: www.scipy.org/scikits.html.

[11] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, Jarrod K. Millman, Nikolay Mayorov, Andrew Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt «SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python». Nature Methods 2020, doi.org/10.1038/s41592-019-0686-2.

[12] The pandas development team «pandas: powerful Python data analysis toolkit». pandas.pydata.org/docs/pandas.pdf.

[13] Travis E. Oliphant. «A guide to NumPy». USA: Trelgol Publishing, 2006. numpy.org/doc/stable.

[14] Hunter, J. D. «Matplotlib: A 2D graphics environment». Computing in Science & Engineering vol. 9, no. 3, pp. 90-95, 2007. matplotlib.org/citing.html.

[15] Официальный сайт документации Implicit: implicit.readthedocs.io/en/latest/quickstart.html.

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, Lars Schmidt-Thieme «BPR: Bayesian Personalized Ranking from Implicit Feedback». Appears in Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI2009), arxiv.org/abs/1205.2618.

[17] F. Maxwell Harper and Joseph A. Konstan «ACM Transactions on Interactive Intelligent Systems». ACM Trans. Interact. Intell. Syst. 5, 4, Article 19, 2015, doi.org/10.1145/2827872.