Санкт–Петербургский государственный университет

Эссе на тему:

Калибровка рекомендательных систем.

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. Калибровка – это общая концепция машинного обучения, и в последнее время она переживает возрождение в контексте справедливости алгоритмов машинного обучения.

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. Со временем такие несбалансированные рекомендации несут в себе риск постепенного сужения областей интересов пользователя – что аналогично эффекту пузыря фильтров.

Постановка задачи

Цель работы заключается в реализации алгоритма калибровки рекомендательных систем, учитывающего все интересы пользователя, и сравнении с существующими решениями.

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. В данной работе я решил рассмотреть именно второе.

Второе направление более сосредоточено на диверсификации, полноте и разнообразии рекомендации. Основным инструментом калибровки является переранжирование списка уже сгенерированных рекомендаций. В работе [1] используется выделение подпрофилей, на основе положительных оценок, для каждого пользователя, которые отражают определенные интересы. Другой метод предлагают исследователи из Netflix [2], он основан на расстоянии Кульбака-Лейблера, которое применяется для определения схожести пользовательского распределения и сгенерированнного. В исследовании [3] 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. 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.

Также, в последнее время, многи социальные сети начинают внедрять рекомендательные алгоритмы в свои сервисы. Так, например **Instagram** применяют собственный фреймворк ig2vec [5], основанный 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** [6] для языка программирования Python, реализованный с помощью SciKits [7] (сокращенно от scipy Toolkits) – это дополнительные пакеты для SciPy [8], 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 рекомендательных 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) [2].

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

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. [3] 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 каждом 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 [4], также называемая положительная прогностическая ценность и рассчитывается 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.

Метод оценки схожести распределений предложен в статье от Netflix [2], 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 инструментов 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.
- SurPrise [6] – фреймворк для языка программирования Python с готовой реализацией нескольких методов рекомендательных систем.
- SciPy [8] – это бесплатная библиотека Python с открытым исходным кодом, используемая для научных вычислений и технических вычислений.
- Implicit [9] – библиотека с реализованной коллаборативной фильтрацией для неизвестных данных.

2.2 Алгоритм рекомендательной системы

В качестве алгоритма рекомендаций было выбрано Байесовское персонализированное ранжирование [10]. 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, чтобы предсказать персонализированную

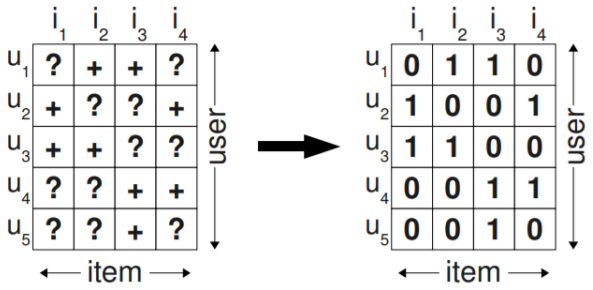

Рис. 1

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, будут рассматриваться пары элементов как обучающие данные. Набор данных, который будет рассматриваться, сформулирован следующим образом

$$(u, i, j) \in D_S$$

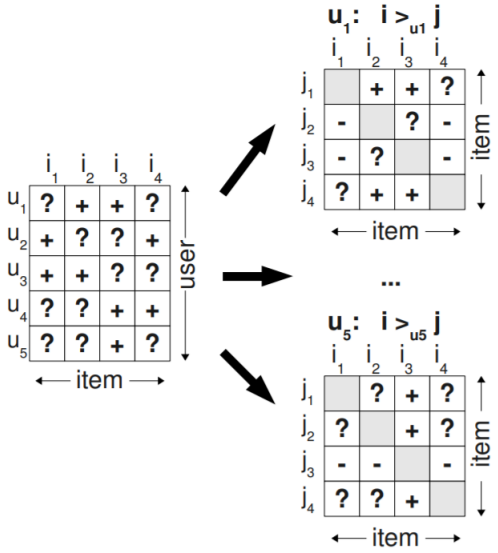

Рис. 2

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

$$p(\Theta| >_u) \propto p(>_u |\Theta)p(\Theta) \tag{4}$$

Функция правдоподобия (4), 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:

$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta)) \tag{5}$$

где σ логистическая сигмоида:

$$\sigma(x) := \frac{1}{1 + e^{-x}} \tag{6}$$

$\hat{x}_{uij}(\Theta)$ – это в приведенном выше уравнении (5) является вещественной значимой функцией, которая представляет собой отношение между пользователем u, элементом i и элементом j и обычно вычисляется с использованием модели матричного факторизации. Сигмовидная функция (6) 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 (7) является окончательным критерием Байесовского персонализированного ранжирования, который должен быть оптимизирован

$$\sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2 \tag{7}$$

где λ_Θ – специфические параметры регуляризации модели.

2.3 Реализация на языке программирования Python

Выберем порог оценки элемента, выше которого оценка считается положительной. Для дальнейшей работы представляем данные в виде разряженной таблицы, для более быстрой работы с большими таблицами. Разделяем исходные данные на тренировочную и тестовую выборки, 80% данных на тренировочную и 20% на тестовую. На тренировочной выборке строим алгоритм рекомендаций байесовского персонализированного ранжирования.

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 распределения 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 [3].

Для сравнения с готовым решением будем использовать фреймворк SurPRISE [6] на языке программирования Python. Из набора реализованных во фреймворке методов используем основанный на предположении о нормальном распределении NormalPredictor.

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 [2].
- Построение рекомендаций Байесовским персонализированным ранжированием и калибровка методом Сент-Лагю [3].
- Построение рекомендаций с помощью готового фреймворка SurPRISE [6].

3.2 Набор данных

Для проведения эксперимента был выбран датасет MovieLens 25M [11], 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 пятибалльной шкале, пользователем для данного фильма, timestamp – временная отметка оценки.

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

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 похожи 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 [3], 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 обозначены исходные данные, на которых обучались алгоримы, оранжевым цветом обозначены данные, полученные рекомендательной системой, а зеленым цветом обозначены откалиброванные данные, сгенерированные рекомендательной системой.

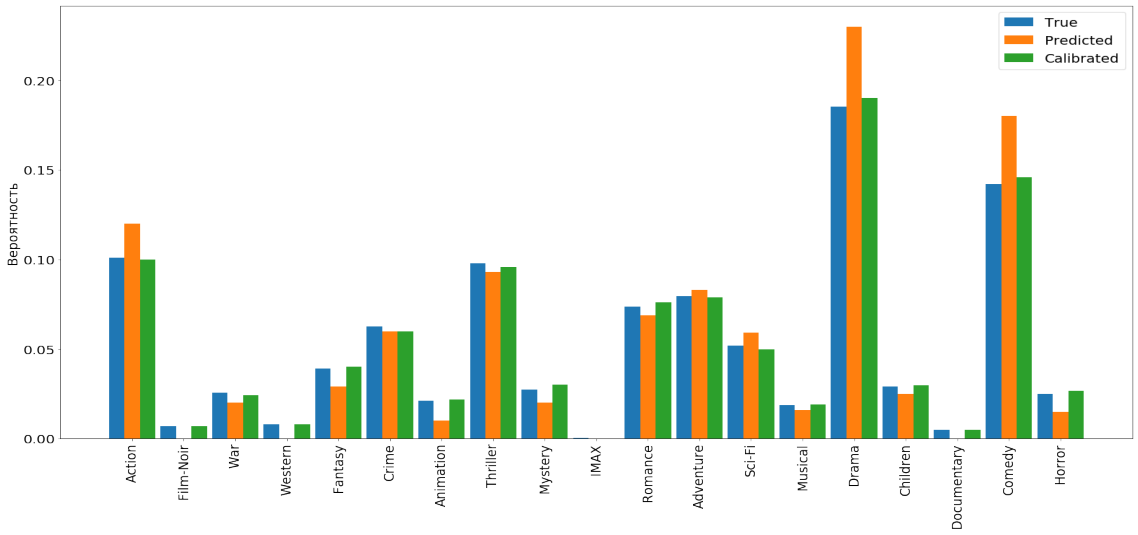

Рис. 3: Распределение жанров при Байесовском персонализированном ранжировании.

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.

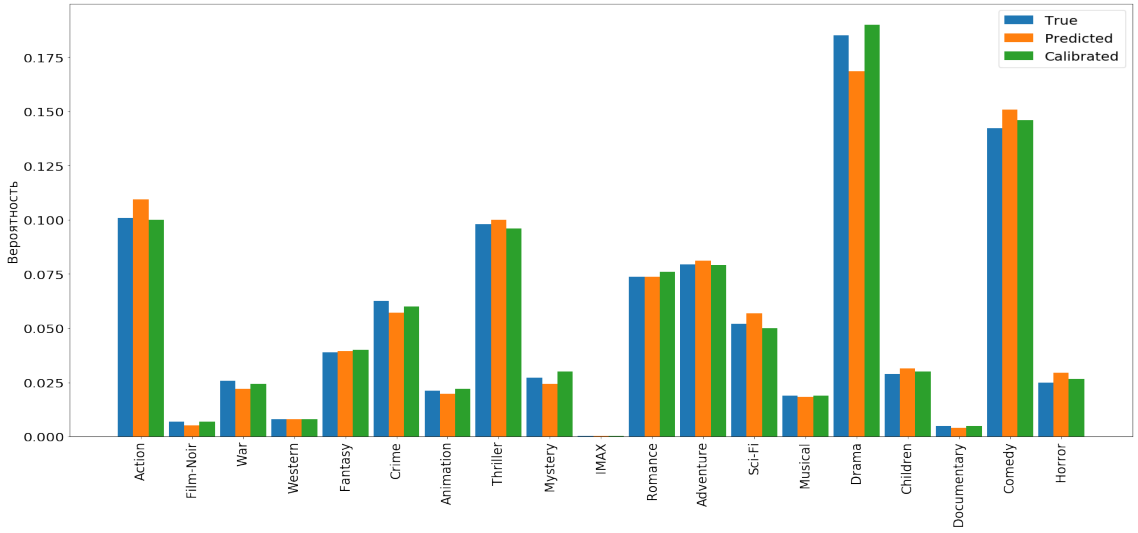

Рис. 4: Распределение жанров при использовании фреймворка SurPRISE.

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 пользователей 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;
- Реализован алгоритм на языке Python и проведены эксперименты на данных;
- Проведено сравнение полученных результатов калибровки между собой и с готовой реализацией из фреймворка;

Цель работы была достигнута, реализованный алгоритм калибровки рекомендательных систем заметно улучшает учет пользовательских интересов в рекомендациях 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 алгоритм можно применять практически в любой рекомендательной системе, где фигурируют какие-то классы объектов рекомендаций, а не только для фильмов, как представлено в эксперименте. Например, в музыкальных сервисах, социальных сетях, новостных агрегаторах, интернет-магазинах.

Список литературы

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

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

[8] Pauli Virtanen, Ralf Gommers «SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python». Nature Methods 2020, doi.org/10.1038/s41592-019-0686-2.

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

[10] 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.

[11] 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.