Санкт-Петербургский государственный университет
Кафедра математической теории игр и статистических решений

Христенко Евгений Александрович

Магистерская диссертация

Оптимизация инструментария для работы с большими данными в задаче построения рекомендательных систем

Направление 01.04.02
Прикладная математика и информатика

Научный руководитель,
доктор физ.–мат. наук,
профессор
Богданов А.В.

Санкт-Петербург
2020

Содержание

Введение

Развитие электронной коммерции и социальных сетей обусловило рост необходимости в качественных системах рекомендации. Рядовой пользователей затруднен в выборе релевантных объектов среди всего обилия предлагаемых товаров и цифрового контента. Кроме того, алогритмы машинного обучения и анализа данных увеличавают не только удобство пользователя, но также напрямую влияют на выручку компаний. Так, например, 35% покупок крупнейшей компании на рынке электронной коммерции Amazon, а также 75% просмотров видео сервиса Netflix, приходится на продукты, предложенные рекомендательной системой [3].

Данная работа посвящена построению рекомендательной системы для одного из крупнейших игроков Азии на рынке розничной продажи товаров из области "Здоровье и Красота". Готовые решения обычно не имеют открытого кода, медленно расширяют инструментарий доступных методов и недостаточно гибко адаптируются под особенности данных компании.

Рекомендательные системы в большинстве своем обучаются на истории активности пользователей, что требует привлечения инструментов из области Big Data, т.е. обработки больших объемов данных. В частности, в данной задаче имелись данные

о примерно 12 миллионах транзакций пользователей.

В работе представлено решение основное на открытом программном обеспечении, таком как Apache Hadoop, Apache Spark и TensorFlow. Таким образом описанные алгоритмы могут быть использованы при построении рекомендательной системы для решения других задач.

Глава 1 содержит предварительные сведения необходимые для корректного обучения и оценки моделей, а также базовые модели рекомендаций.

Главы 2-3 содержат описание основных моделей: модификация распространенного алгоритма SVD, а также решение, основанное на глубоком обучении.

Глава 4 содержит описание эффективного способа ансамблирования моделей, позволяющего совместить преимущества всех использованных моделей.

Постановка задачи

Формализация задачи

$$U = \{u_1, u_2, ..., u_N\}$$

множество пользователей

$$I = \{i_1, i_2, ..., i_M\}$$

множество товаров

$$R = (r_{i,j})$$

матрица рейтингов размера $N \times M$, где $i \in 1 \dots N$, $j \in 1 \dots M$. В классической постановке задачи рейтинги могут принимать вещественные значения, однако в данной задаче отсутствуют явные предпочтения пользователя - имеется лишь факт покупки товара. Такие данные называются данными с неявной информацией [7]. В случае покупки пользователем J товара I значением $R_{i,j}$ будет являться 1, в прочих случаях - пропуск. Матрицу рейтингов можно представить в виде таблицы 1.

Задачей алгоритма машинного обучения является восстановление пропущенных значений. В дальнейшем на основе

	Товар 1	Товар 2	Товар 3
Пользователь 1	?	1	?
Пользователь 2	1	1	?

Таблица 1: Матрица рейтингов

упорядочивания восстановленных рейтингов для пользователя формируется N-мерный вектор, состоящий из релевантных товаров, где N - количество рекомендаций. Набор товаров предоставляется пользователю в качестве рекомендаций.

Выделяют 3 типа рекомендательных систем [5]:

- Коллаборативная фильтрация — рекомендации основаные на истории оценок пользователей.
- Основанные на контенте — рекомендации основаные на схожести признаков товаров.
- Гибридные — рекомендации комбинирующие коллаборативные и контентные подходы.

Рассматриваемые данные не имели достаточно качественного описания товаров, поэтом в данной работе развиваются методы, основанные на коллаборативной фильтрации.

Глава 1. Описание процедуры обучения и базовая модель

1.1. Оценка качества предложенных моделей

Для оценки качества рассматриваемых моделей предварительно небходимо определить критерии сравнения. Оценки качества могут быть разделены на две группы:

- Оцениваемые на отложенной выборке из исторических данных пользователей
- Оцениваемые на пользователях в реальном времени

Необходимость обязательного наличия двух подоходов к оценивания обусловлена возможностью значительного расхождения оценок двух групп [2]. Оценки первой группы представляют критерии сравнения с точки зрения формальной математической модели, в то время как оценки второй группы предоставляют возможность оценить эффект модели с точки зрения целесообразности для бизнеса, например, количество пользователей, купивших рекомендованный товар.

В первой группе были выбраны стандартные для задачи построения рекоммендательных систем оценки - MAPk и NDCGk.

Данные оценки, взятые из задачи ранжирования списка объектов, обладают тем преимуществом, что позволяют оценивать не только точность рекомендации, но и насколько близко к началу выдачи был объект. Данное свойство важно в данной задаче, поскольку пользователь редко просматривает выдачу полностью и акцентирует внимание только на начале списка. Исходя из технической реализации рассылки оптимизировался список длиной 20.

MAPk (Mean average precision at K)

$$pr_K = \frac{\sum_{i=1}^{K} r_{true}(e_i)}{K}$$

Где e_i– i-й элемент в выдаче пользователя, $r_true(e)$ – функция равная 1 если e релевантен, 0 иначе, K - размер выдачи

$$apr_K = \frac{1}{K} \sum_{i=1}^{K} r_{true}(e_i) * pr_i$$

$$MAP_K = \frac{1}{N} \sum_{i=1}^{N} apr_K^i$$

Где N - количество пользователей

NDCGk (Normalized Discounted Cumulative Gain at K)

$$G(i) = 2^{r_{true}(e_i)} - 1$$

$$D(i) = \frac{1}{log_2(i+1)}$$

$$DCG_K = \sum_{i=1}^{N} G * D$$

$$NDCG_K = \frac{DCG_K}{maxDCG_K}$$

Так как основным каналом доставки рекомендации являлись рассылки электронной почты основной оценкой второй группы было выбрано отношение количества переходов по ссылке рекомендуемых товаров к количеству разосланных писем — Rel_{link}.

1.2. Разбиение исторических данных

Для оценивания обобщающей способности моделей применим процедуру скользящего контроля. Стандартный подход заключается в разбиении данных различными K способами на пары непересекающихся множеств - обучающего и тестового [1]. Для каждого разбиения производится обучение на обучающем множестве и вычисляется значение оценок качества на тестовом множестве.

Оригинальный метод предполагает случайное разбиение элементов. В случае транзакционных данных пользователей необходимо учитывать время совершения тразакции - модель должна обучаться на транзакциях совершенных раньше, нежели те, что представлены в тестовом множестве. Пример подобного разбиения изображен на Рис. 1.

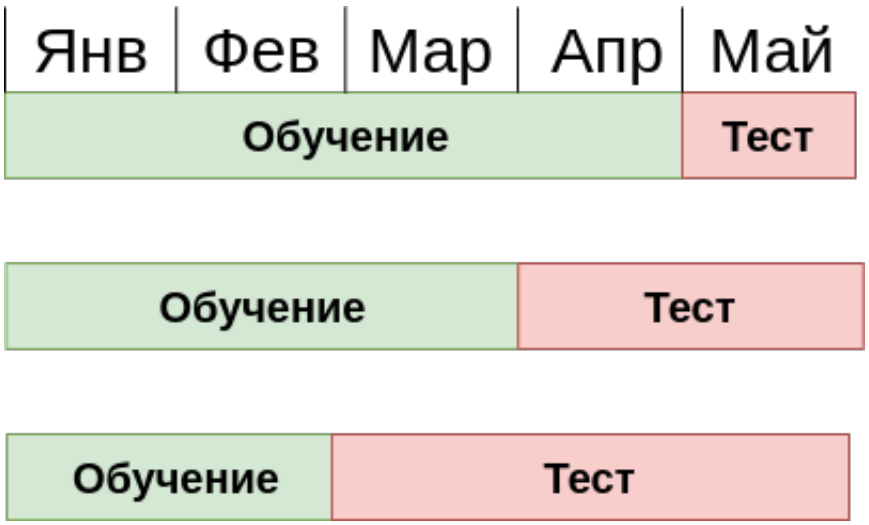

Рис. 1: Разделение обучающей выборки

1.3. Вычислительные ресурсы

Apache Hadoop

Основным фреймворком для храненнния данных и организации вычислений являлся кластер Apache Hadoop [9]. Apache Hadoop позволяет осуществлять распределенную обработку больших объемов данных и является стандартом отрасли интелектуального анализа данных.

Преимуществом решения построенного на основе Hadoop является возможность технологии адаптироваться под ЭВМ общего назначения и масштабироваться под любой объем данных. В данном случае использовался кластер из 21 узла конфигурации указанной в Таб. 2.

Количество ядер процессора	8
Оперативная память, Гб	48
Объем постоянной памяти, Тб	2

Таблица 2: Конфигурация узла кластера

GPU

Для обучения нейронной сети использовалась GPU Tesla P100.

1.4. Базовая модель

Простейшей моделью в рекомендательных системах является частотная модель. Согласно данному подходу выдача формируется из N наиболее популярных товаров. Модель не является персонализированной моделью, то есть список рекомендации является единообразным для всех пользователей.

Построение подобной модели необходимо для определения нижней границы оценок качества. Модели, имеющие качество не превосходящее качество базовой модели следует исключить из рассмотрения.

Также предложена альтернативная базовая модель - модель основанная на демографических данных пользователей[5]. Данная модель является модификацией частотной модели:

1. Группируем пользователей на основе пола и возраста

2. Составляем списки наиболее популярных товаров в пределах одного сегмента

3. Используем полученные списки для формирования выдачи нового пользователя, попадающего в определенный сегмент

1.5. Результаты

	MAPk	NDCGk	Rel_{link}
Popularity	0.0521	0.2283	0.1151
Demographic	**0.0947**	**0.2631**	**0.1627**

Таблица 3: Оценки базовых моделей

Модель, использующая демографию пользователя превосходит частотную модель во всех показателях. Поскольку в качестве базовой необходима только одна модель, в дальнейшем будет рассмтариваться только модель с демографией.

Глава 2. Алгоритм SVD

2.1. Описание модели

SVD(сингулярное разложени матрицы) — один из основных алгоритмов в области рекомендательных систем. Данный подход лежал в основе решения победившего на конкурсе Netflix Prize [8], конкурсе послужившем огромным толчком в развитии области. Основой метода служит идея представления матрицы в виде произведения двух матриц меньшей размерности.

В таком случае оценка конкретного товара i пользователем u представляется в виде скалярного произведения:

$$r_{ui}(\theta) = p_u^T q_i$$

$$\theta = p_u, q_i | u \in U, i \in I$$

Необходимо найти оптимальные параметры θ, при которых модель предсказывала оценки как можно лучше:

$$E_{(u,i)}(r_{ui}(\theta) - r_{ui})^2 \to \min$$

Как следствие нам необходимо оптимизировать следующий функционал:

$$J(\theta) = \sum_{(u,i)\in D} (p_u^T q_i - r_u i)^2 + \lambda(\sum_u ||p_u||^2 + \sum_i ||q_i||^2)$$

Оптимизация производится методом градиентного спуска.

Полученные вектора пользователей и тоаров обладают дополнительным свойством - они задают векторное представление в некотором новом пространстве латентных факторов. Это позволяет использовать полученные вектора отдельно от задачи построения рекомендаций, например, в задаче нахождения похожих товаров. Данное свойство будет использовано при построении модели следующей главы

2.2. Программная реализация

Реализация матричного разложения создана на основе библиотеки Apache Spark. Особенностью данного фреймворка является обработка всех распределенных вычислений в оперативной памяти без необходимости записи промежуточных результатов на жесткий диск. Это позволяет значительно ускорить вычисления на больших данных. Кроме того, Spark способен работать поверх вычислительного кластера Hadoop.

2.3. Результаты

	MAPk	NDCGk	Rel_{link}
Demographic	0.0947	0.2631	0.1627
SVD	**0.2117**	**0.4034**	**0.2651**

Таблица 4: Оценки матричного разложения

Результаты SVD ожидаемо значительно превосходит базовую модель. Это обусловлено тем, что разложение является первой по-настоящему персонализированной моделью: в отличие от демографической рекомендательной модели, где оценка получается путем усреднения оценок достаточно большого числа пользователей.

Глава 3. Полносвязная нейронная сеть и исследование применимости word2vec в задаче рекомендации

3.1. Описание модели

Основной идеей предлагаемой архитектуры нейронной сети является попытка предсказать очередной товар пользователя, основываясь на предыдущей покупке. Данный подход хорошо подходит для детектирования товаров покупаемых вместе, что является распространненым случаем в розничной торговле.

Для представления пользователя и товара нам необходимы их векторные представления. В частности мы имеем подобные вектора из алгоритма разложения, описанного в прошлой главе. Также предлагается добавить дополнительное представление товара используя алгоритм word2vec.

word2vec [6] — нейросетевой метод получения векторного представления слова, разработанный компанией Google. Наиболее часто применяется в обработке естественного языка, однако может быть адаптирован к рассматриваемой задаче. Для этого необходимо рассмотреть последовательный набор покупок пользователя в качестве корпуса текста, таким образом уни-

кальные идентификаторы товара будут представлять слово в некотором словаре.

OHE(one hot encoding) — метод, преобразующий слово в вектор длиной равной мощности словарая, с единственным отличным от нуля значением на позиции, соответствующей номеру данного слова в словаре.

Используем архитуектуру CBOW(Continuous Bag of Words). На вход имеем 2k+1 наборов OHE представлений слов из текста, где центральное слово прогнозируется из контекста фиксированной длины:

$$p(v|w) = \frac{exp(Y_v^T X_w)}{\sum_s exp(Y_s^T X_w)}$$

где Y_v - вектор предсказываемых терминов, а X_w - вектора слова на скрытом слое.

В результате экспериментов с различными архитектурами нейронной сети было предложено следующее решение (Рис. 2):

- Полученное в результате обучения word2vec векторное предствление товара, а также вектора, полученные из матричной факторизации поступают на входной слой нейронной сети

- Добавляем скрытый слой меньшей размерности

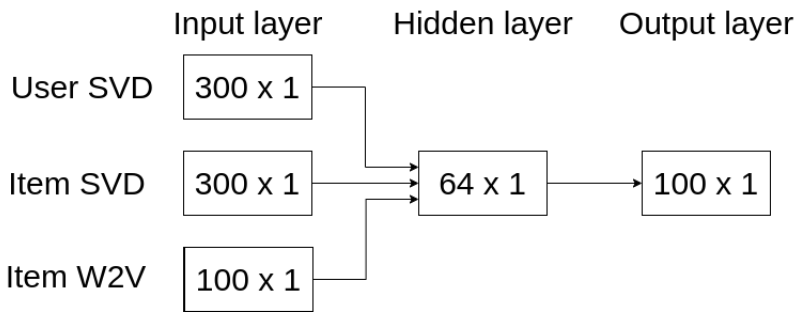

Рис. 2: Архитектура нййронной сети

- Добавляем выходной слой размерности, соответствующей размерности вектора товара - вектор следующей покупки пользователя

3.2. Программная реализация

Архитектура нейронной сети была реализована с помощью библиотеки Keras, представляющую собой верхнеуровневую абстракцию над библиотекой Tensorflow. Решения, построенные на основе Tensorflow являются стандартом отрасли, что обеспечивает простое перенесение обученных моделей из одной системы в другую.

3.3. Результаты

	MAPk	NDCGk	Rel_{link}
Demographic	0.0947	0.2631	0.1627
SVD	0.2117	0.4034	0.2651
NN	0.2173	0.4101	
$NN_{word2vec}$	**0.2205**	**0.4306**	**0.2760**

Таблица 5: Оценки нейронной сети

Оценка качества нейросетевого подхода незначительно превосходит качество матричного разложения, несмотря на значительное усложнение модели. Похожие результаты показаны в [10]. Заметим, что модель не содержащая дополнительного word2vec представления товара уступает в качетсве расширенной модели. Это служит подтверждение применимости алгоритм word2vec в задаче построения рекомендательных систем.

Глава 4. Ансамблирование моделей

4.1. Описание модели

При анализе работы моделей, рассмотренных в прошлых главах, было замечено, что даже более сильные алгоритмы могут для отдельно взятых пользователей показывать качество ниже, чем частотная модель. Это связано с тем, что для различных пользователей представлено различное количество транзакций. Такие пользователи могут плохо описаваться сложным моделям. в силу небольшого количества данных для обучения. Для решения данной проблемы предлагается следующий алгоритм:

1. Обучить расссмотренные модели на части данных.
2. Произести оценку качества моделей для каждого отдельно взятого пользователя.
3. Зафиксировать метку наилучшего алгоритма
4. Обучить многоклассовый классификатор пользователей, используя разметку с прошлого шага
5. Для каждого представления пользователя получаем список рекомендаций наиболее вероятной модели

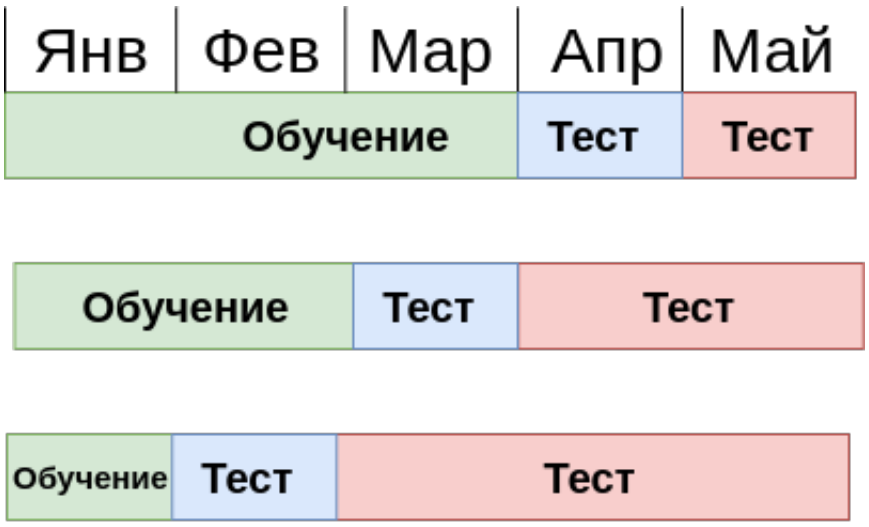

Рис. 3: Модифицированное разделение обучающей выборки

Предложенный подход требует модификации процедуры оценки, описанной в главе 1. Потребуется введение дополнительного тестового множества, по причине того, что на первом наборе будут обучаться модели нижнего уровня, а на втором тестовом наборе будет производиться оценка качетсва итогового ансамбля. Возможное модификация разбиения изображена на рис. 3

В качестве классификатор с шага 4 может быть рассмотрен любой многоклассовый алгоритм, например, логиситическая регрессия или лес решающих деревьев. В данной работе предлагается использование градиентного бустинга над решающими деревьями, как алгоритма демонстрирующего высокую точность в задачах классификации. В качестве вектора признаков пользователя использовались вектор полученные из алгоритма SVD.

Градиентный бустинг

Строим K зависимых между собой деревьев так, чтобы каждое следующее дерево старалось улучшить качество всей композиции. Классификация производится по формуле:

$$a(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{k:b_k(x)=y} a_k$$

где $b_k(x)$ – результат k-го дерева на элементе x, а a_x – вклад k-го дерева в композицию.

После обучения очередного дерева, веса неверно классифицированных элементов возрастают, тем самым каждое последующее дерево учиться давать правильный ответ на наиболее сложных, в смысле классификации, объектах.

4.2. Программная реализация

Модель разработана с помощью открытой библиотеки LightGBM. Преимущество данной реализации градиентного бустинга является возможность распределенного обчения композиции решающих деревьев. В данной работе использовалась возможность обучения на GPU.

Для реализации использовался язык Python. В качестве целевой функции был выбран softmax. Подбор гиперпараметров

осуществлялся на основе процедуры скользящего контроля.

4.3. Результаты

	MAPk	NDCGk	Rel_{link}
Demographic	0.0947	0.2631	0.1627
SVD	0.2117	0.4034	0.2651
NN	0.2205	0.4306	0.2760
Ensemble	**0.2625**	**0.5174**	**0.3205**

Таблица 6: Оценки ансамбля моделей

Качество ансамбля моделей ожидаемо превосходит качество отдельно взятых моделей. Причиной тому является то, что модель второго позволяет выбирать наиболее качественную на уровне отдельно взятого пользователя. Таким образом, возможно совместить резлультат работы каждой модели.

Заключение

Был разработан набор методов рекомендации товаров на основе факторизцации матриц и глубокого обучения. Внедрение рекомендательной системы на основе предложенных методов позволило увеличить переходы по ссылке из e-mail рассылки на 97% относительно рекомендации наиболее популярных товаров. Алгоритм ансамблирования, описанный в последнй главе, позволяет расширять систему новыми моделями рекомендации, не переживая о выборе наиболее эффективного метода.

Кроме того, в процессе разработки нейросетевой модели исследован вопрос о применимости модели анализа текстов word2vec к задаче рекомендации товаров.

Использование открытых технологий, обладающих возможность развертывания в гетерогенных системах, а также их высокая масштабирумемость позволяет адаптировать предложенную систему для решения широкого спектра задач рекомендации контента, в том числе на больших объемах данных. Использование современных языков разработки, таких как Scala и Python, облегчает поддержку и дальнейшее развитие системы.

Список литературы

1. Мерков А. Б. Распознавание образов. Введение в методы статистического обучения. URSS, 2011. 256 с.

2. Garcin F., Faltings B., Donatsch O., Alazzawi A., Bruttin C., Huber A. Offline and Online Evaluation of News Recommender Systems at swissinfo.ch // Proceedings of the 8th ACM Conference on Recommender systems. 2014. С. 169-176.

3. How retailers can keep up with consumers [Электронный ресурс]: URL:https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers (дата обращения: 21.05.20).

4. Apache Hadoop [Электронный ресурс]: URL:https://hadoop.apache.org/ (дата обращения: 21.05.20).

5. Aggarwal C. Recommender Systems: The Textbook. Springer, 2016. 498 с.

6. Mikolov T., Sutskever I., Chen K., Corrado G., Dean J. Distributed representations of words and phrases and their compositionality // Proceedings of the 26th International

Conference on Neural Information Processing Systems. 2013. C. 3111–3119.

7. Recommender Systems Handbook / Ricci, Bracha, Shapira, Kantor, Springer, 2011.

8. Bennett J., Lanning S. The Netflix Prize // In KDD Cup and Workshop in conjunction with KDD. 2007.

9. TensorFlow [Электронный ресурс]: URL:https://www.tensorflow.org/ (дата обращения: 21.05.20).

10. Ferrari M., Cremonesi P., Jannach D. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches // Proceedings of the 13th ACM Conference on Recommender Systems. 2019.