ГОУ ВПО Российско-Армянский университет

ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАТИКИ

СПЕЦИАЛЬНОСТЬ: ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

КАФЕДРА МАТЕМАТИКИ И МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

М А Г И С Т Е Р С К А Я Д И С С Е Р Т А Ц И Я

Т е м а: «Аугментация данных для задач рекомендательных систем с помощью генеративно-состязательных сетей (GAN)»

Руководитель

к.ф.-м.н. - Дарбинян А.А.

Студент

Цатурян Д.К.

Е Р Е В А Н 2020

Содержание

Введение

Рекомендательные системы стали неотъемлемой частью интернет-магазинов и других систем для рекомендаций товаров клиентам. Несмотря на отточенные годами изобилия существующих методов коллаборативных фильтрации, большинство из них сталкиваются с проблемой разреженности матрицы кросс-табуляций (матрица user-item R), которую надо каким-нибудь способом побороть. С другой стороны генеративно-состязательные сети (GAN), появившиеся в последние годы, хорошо зарекомендовали себя в задачах восстановления скрытого распределения выборки, особенно, если объекты изображения. С помощью GAN нередко делается аугментация тренировочных данных - пополнение данных искусственными объектами. Проблему снижения разреженности матрицы R можно смягчить с помощью аугментации матрицы R. Однако прямое использование существующих GAN для аугментации данных рекомендательных систем практически невозможно, по крайней мере по нижеперечисленным 3 причинам:

1. Классический GAN (Goodfellow и др.) и его улучшенный вариант Вассерштайн GAN (Arjovsky и др.) предполагают непрерывные данные для обучения.
2. Малое количество существующих качественных GAN для дискретных данных.
3. Разреженность данных матрицы R.

В данной работе предлагается метод для аугментаций данных матрицы R, основанный на переработке дискретного Вассерштайн GAN. Как бейслайн берутся классические методы коллаборативной фильтрации: низкоранговое приближение матрицы R с помощью разреженного сингулярного разложения (SVD) и метода чередующихся квадратов (ALS). Результаты экспериментов, сделанные на датасетах MovieLens-100k и MovieLens-1M, улучшая оценки MSE И P@K на тестовой выборке, показывают работоспособность предлагаемого метода.

1. Введение в рекомендательные системы

1.1 Постановка задачи

Рассматривается следующая задача:

U – Множество пользователей (users).

T – Множество объектов (items).

Y – Пространство описаний транзакций.

$R = (r_{ui})$ - Матрица user-item размера $|U| \times |I|$, где $r_{ui} \in Y$ - некоторое число, которое говорит о том, насколько товар i понравился пользователю u, если $r_{ui} > 0$, а если $r_{ui} = 0$ - это означает пользователь u не оценивал товар i. Последнее может иметь две причины: либо пользователь не знает о его существовании, либо у него нет интереса к этому товару.

Матрица R бывает в основном сильно разреженной, т.е. много ячеек матрицы пропущены. Требуется сделать предсказания на незаполненных местах, т.е. заполнить пропущенные значения матрицы R, после чего товары, соответствующие предсказанным значениям с высокой оценкой можно рекомендовать пользователям. Когда $Y = \{0, 1\}$ - задача называется задачей неявной оценки (implicit feedback), а если вместо единицы оценки, например $Y = \{0, 1, 2, 3, 4, 5\}$ - задача явной оценки (explicit feedback). Рекомендательные системы используются практически во всех крупных интернет-магазинах, таких как amazon.com, ozon.ru, и т.д., и не только в интернет-магазинах. С 2 октября 2006 по 21 сентября 2009 года проводился конкурс Netflix Prize, где были даны $Y = \{0, 1, 2, 3, 4, 5\}$, $|U| = 0.48 \times 10^6$, $|I| = 17 \times 10^3$ и 10^8 рейтингов. Точность оценивалась среднеквадратичным отклонением по тестовой выборке D^*:

$$L = \frac{1}{|D^*|} \sum_{(u,i) \in D^*} (r_{ui} - \hat{r}_{ui})^2$$

Требовалось уменьшить L с 0.9514 до 0.8563 (на 10%). Приз составлял 10^6\$. В следующем параграфе будем рассматривать решение, занявшее 2-е место на этом конкурсе.

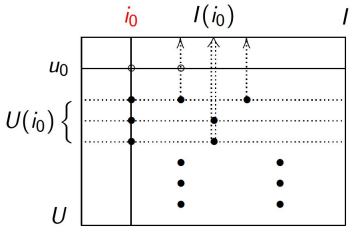
Существуют два основных подхода к решению задач рекомендательных систем:

1. Корреляционные модели (memory based collaborative filtering).
   При таком подходе хранится вся исходная матрица R и методы, основанные на таком подходе, ищут корреляции между пользователями (между строками) и корреляции между товарами (между столбцами)
2. Латентные модели (latent models for collaborative filtering).
   Основная идея основана на низкоранговом приближении матрицы R. Вместо большой

исходной матрицы хранятся две маленькие матрицы латентных характеристик (профилей) пользователей и товаров. Сходство клиентов и объектов оценивается сходством соответствующих профилей.

Первый подход уже практически не используют, так как он обладает существенными недостатками, а именно: плохое прогнозирование, хранение большой исходной матрицы и т.д. Фактически это два основополагающих решения задач коллаборативных фильтраций. Существуют и другие решения, основанные на комбинированном подходе 1 и 2, на ограниченных машинах Больцмана [23], на автокодировщике [8] и т.д.. Заметим, что в данной работе нас будет интересовать только второй подход.

Ради полноты изложения рассмотрим простейший метод коллаборативной фильтрации, принадлежащей категории методов 1.



Реализация идеи <<клиенты, купившие i_0, также покупали $I(i_0)$ >>.

$$U(i_0) = \{u \in U; r_{ui} > 0, u \neq u_0\}$$

$$I(i_0) = \left\{ i \in I; \operatorname{sim}(i, i_0) = \frac{|U(i) \cap U(i_0)|}{|U(i) \cup U(i_0)|} > \delta \right\}$$

Отсортируется $I(i_0)$ и рекомендуются топ N объектов.

1.2 Латентные модели - низкоранговое приближение матрицы R

Для разложения матрицы R используется предположение о ее низком ранге и строится приближение:

$$R \approx \hat{R} = UV^T \qquad (1.2.1)$$

$$R \in \boldsymbol{R}^{m \times n}, U \in \boldsymbol{R}^{m \times k}, V \in \boldsymbol{R}^{n \times k}, k \ll n$$

Латентные характеристики объектов и клиентов невозможно точно интерпретировать, но интуитивно эти характеристики можно воспринимать как особенности этих объектов (например, жанры для фильмов), а для пользователей - как предпочтения пользователя.

После разложения R, предсказанная оценка объекта i для пользователя u - есть значение $\hat{R}_{ui} = (U_u, V_i)$, а так как скалярное произведение прямо пропорционально косинусной мере

$(U_u, V_i) = \|U_u\| \, \|V_i\| \cos\theta$, то его можно интерпретировать как корреляцию между характеристиками объекта и предпочтениями клиента. Приближение (1.2.1) делается в основном одним из двух широко известных алгоритмов - стохастическим градиентным спуском(stochastic gradient descent) и методом чередующихся наименьших квадратов (alternating least squares). Рассмотрим поподробнее эти методы, так как именно эти методы выбраны для экспериментов настоящей работы.

1.2.1 Разреженный SVD

Заметим, что здесь невозможно применять хорошо знакомые нам разложения, такие как SVD, LU, спектральное разложение, и т.д.. Для этого много причин, но самая главная причина в том, что все эти методы предполагают, что матрица известна целиком. Однако в нашем случае это не так. 2-е место в конкурсе Netflix, описанный в первой главе, занял следующий метод (с незначительными изменениями), названный как разреженный SVD [4]:

Разреженный SVD:

$$\sum_{(u,i)\in S^*} \left(R_{ui} - \bar{r}_u - \bar{r}_i - \sum_{t=1}^{k} U_{tu} V_{ti} \right)^2 \to \min_{U,V} \qquad (1.2.1.1)$$

где $S^* = \{(u,i) \in S;\ (u,i) > 0\}$, $S \in R$-тренировочная выборка $\bar{r}_u$ - bias объекта, $\bar{r}_i$ - bias товара. Иногда добавляются регуляризации $\lambda \|U\|$, $\mu \|V\|$ и другие члены, этот функционал можно варьировать по-разному, но мы рассмотрим простейший его вид (1.2.1.1). Замечаем, что главное отличие от обычного SVD это то, что сумма берется не по всем значениям R, а по S^*.

Обозначим:

$$\widehat{R}_{ui} = \bar{r}_u + \bar{r}_i + \sum_{t=1}^{k} U_{tu} V_{ti}$$

$$\varepsilon_{ui} = R_{ui} - \bar{r}_u - \bar{r}_i - \sum_{t=1}^{k} U_{tu} V_{ti} = R_{ui} - \widehat{R}_{ui}$$

$$\varepsilon^2 = \sum_{(u,i)\in S^*} \left(R_{ui} - \widehat{R}_{ui} \right)^2 \to \min_{U,V}$$

Случайным образом многократно перебираются все $(u,i) \in S^*$ и каждый раз делается градиентный шаг:

$$U_{tu} = U_{tu} + \eta \varepsilon_{ui} V_{ti} \qquad t = \{1, 2, \ldots k\}$$

$$V_{ti} = V_{ti} + \eta\varepsilon_{ui}U_{ti} \qquad t = \{1, 2, \ldots k\}$$
$$\bar{r}_u = \bar{r}_u - \eta\varepsilon_{ui}$$
$$\bar{r}_i = \bar{r}_i - \eta\varepsilon_{ui}$$

1.2.2 Метод чередующихся наименьших квадратов

Рассмотрим задачу наименьших квадратов:

Пусть дана система

$$Ax = b \qquad (1.2.2.1)$$

где $A \in R^{mxn}$, $x \in R^n$, $b \in R^m$. Требуется найти $\bar{x} \in R^n$, так что:

$$\|A\bar{x} - b\| = \min_{x \in R^n} \|Ax - b\| \qquad (1.2.2.2)$$

Такое решение существует, и оно удовлетворяет

$$A\bar{x} - b \perp imA$$
$$imA \perp \ker A^T$$
$$A\bar{x} - b \in \ker A^T$$
$$A^T(A\bar{x} - b) = 0$$
$$A^T Ax = A^T b \qquad (1.2.2.3)$$

где imA - это образ матрицы A: $imA = \{y \in R^m;\ \exists x \in R^n;\ Ax = y\}$, $kerA$ - ядро матрицы A: $kerA = \{x \in R^n;\ Ax = y\}$. Легко доказывается эквивалентность систем (1.2.2.1) и (1.2.2.3) и совместность системы (1.2.2.1).

$x = A^+ b$, где A^+ псевдообратная матрица матрицы A.

Если $rankA = n$ и $m \geq n$, тогда $rankA^T A = rankA = n$, откуда получаем $A^+ = (A^T A)^{-1} A$:

$$x = \left(A^T A\right)^{-1} A^T b \qquad (1.2.2.4)$$

В коллаборативной фильтрации (1.2.1) вместо обычной регрессии, как в задаче описанной выше, где x и b векторы, рассматривается задача мульти регрессии, где все компоненты - матрицы:

$$\left\|R - UV^T\right\| \to \min_{U,V}$$

Здесь, в отличие от (1.2.2.2) две переменные: U и V. Метод чередующихся квадратов, как гласит само его название, действует следующим образом:

Случайным образом инициализируются U и V и, с помощью формулы (1.2.2.4) N раз поочередно решается задача наименьших квадратов при фиксированном U или V

- Фиксируется U

  $V = (U^T U)^{-1} UR$

- Фиксируется V

$$U = (V^T V)^{-1} U R^T$$

Если добавить регуляризацию

$$\left\| R - UV^T \right\| + \lambda \left(\|U\| + \|V\| \right) \to \min_{U,V}$$

- Фиксируется U

$$V = (U^T U + \lambda I)^{-1} U R$$

- Фиксируется V

$$U = (V^T V + \lambda I)^{-1} U R^T$$

Разложить (1.2.1) можно и с помощью первого алгоритма (SGD) и с помощью второго (ALS) (для задач неявных оценок вместо ALS часто используется модифицированный вариант Weighted ALS), в [11] делается сравнение этих алгоритмов на разных датасетах. ALS работает немного быстрее, что делает его предпочтительным для больших матриц. В настоящей работе эксперименты сделаны на обоих алгоритмах.

2. Генеративно-состязательные сети (GAN)

2.1 Введение

Пусть дан некоторый набор векторов $\{x_i\}_{i=1}^m;\ x_i \in R^n$, предполагается, что все x_i взяты из некоторого распределения P. Требуется восстановить распределение P по набору $\{x_i\}_{i=1}^m$. В статистике существуют три семейства методов к решению этой задачи: параметрическое восстановление, непараметрическое восстановление и восстановление смеси распределений. При параметрическом восстановлении распределения определяется семейство параметрических плотностей $(P_\theta)_{\theta \in R^d}$ и ищется та плотность, которая максимизирует правдоподобие на нашей выборке $x-$ов.

$$\prod_{i=1}^{m} P_\theta(x_i) \to \max_\theta \qquad (2.1.1)$$

$$\sum_{i=1}^{n} \ln P_\theta(x_i) \to \max_\theta$$

Для оценивания расстояния между двумя распределениями P и Q существуют несколько метрик, таких как дивергенция Кульбака-Лейблера ($KL(P\|Q)$), расстояние Йенсена-Шеннона (JS(P||Q)), расстояние Вассерштейна (EM(P||Q)), и т.д. . Мы еще вернемся к особенностям этих метрик в следующем параграфе. Расстояние Кульбака-Лейблера - это математическое

ожидание логарифмической разности между расстояниями P и Q, где математическое ожидание берется по распределению P

$$KL\left(P||Q\right) = \left[\log P\left(x\right) - \log Q\left(x\right)\right]$$

$$KL\left(P||Q\right) = \sum_{i=1}^{m} P\left(x_i\right)\left(\log P\left(x_i\right) - \log Q\left(x_i\right)\right) = \sum_{i=1}^{m} P\left(x_i\right)\log\frac{P\left(x_i\right)}{Q\left(x_i\right)}$$

Очевидно, чем ближе P и Q, тем меньше $KL(P\|Q)$. Стоит заметить, что в строго математическом смысле $KL(P\|Q)$ не является функцией расстояния, так как оно не симметрично $KL(P\|Q) \neq KL(P\|Q)$.

Покажем, что максимизируя функционал (2.1.1), минимизируется дивергенция Кульбака-Лейблера между P и P_θ – $KL(P\|P_\theta)$.

$$\min_{\theta\in R^d} KL\left(P||P_\theta\right) = \min_{\theta\in R^d}\left(\sum_{i=1}^{m} P\left[x_i\right]\log P\left(x_i\right) - P_\theta\left(x_i\right)\log P\left(x_i\right)\right) =$$

$$\min_{\theta\in R^d}\left[-\sum_{i=1}^{m} P\left(x_i\right)\log P_\theta\left(x_i\right)\right] = \max_{\theta\in R^d}\left[\sum_{i=1}^{m} P\left(x_i\right)\log P_\theta\left(x_i\right)\right] = \lim_{m\to\infty}\max_{\theta\in R^d}\frac{1}{m}\sum_{i=1}^{m}\log P_\theta\left(x_i\right).$$

Последнее равенство получается по теореме о больших числах, что и доказывает теоретическую обоснованность максимального правдоподобия (2.1.1) к восстановлению распределения. Такой подход не всегда применим: прямая оптимизация (2.1.1) далеко не всегда дает желаемые результаты, и к тому же зачастую более полезна способность легко сгенерировать новые объекты из этого распределения, чем найти численную формулу распределения, которой может и не существует. Именно последнюю идею и реализуют генеративно-состязательные сети - GAN.

2.2 Классический GAN.

GAN были изобретены Яном Гудфеллоу и другими соавторами [2]. Суть GAN состоит в следующем: определяются две нейронные сети $G_{\theta_g}(z):\ R^k \to R^n;\ k \ll n$ -генератор, который генерирует правдоподобные образцы и $D_{\theta_d}(x):\ R^n \to [0,1]$ дискриминатор, который старается отличить подлинные объекты от сгенерированных. Таким образом, имея поставленные противоположные цели, обе сети обучаются одна с помощью другой. Дискриминатор D обучается максимизировать

$$\max_{\theta_d} E_{x\sim P_r}\left[\log D_{\theta_d}\left(x\right)\right] + E_{z\sim P_z}\left[1 - \log D_{\theta_d}\left(G_{\theta_g}\left(z\right)\right)\right],$$

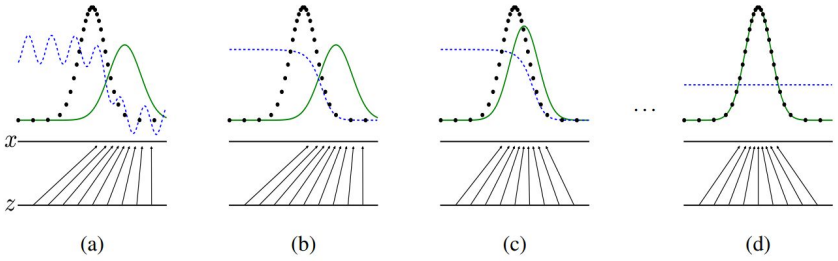
а генератор G минимизирует

$$\min_{\theta_g} E_{z\sim P_z}\left[1 - \log D_{\theta_d}\left(G_{\theta_g}\left(z\right)\right)\right].$$

Вместе они оптимизируют

$$\min_G \max_D V(D,G) = E_{x\sim P_r}[\log D(x)] + E_{z\sim P_z}[1-\log D(G(z))] \quad (2.2.1),$$

т.е. ищется седловая точка функции (2.2.1).



Изображение из [2], показывающее идеальный процесс обучения

Таким образом генератор G неявным образом приближает распределение $P_g(z) : R^k \to R^n$ к реальному распределению P_r. Легко доказывается, что функционал (2.2.1) имеет глобальный оптимум, когда $P_r = P_g$. Рассматривается оптимальный дискриминатор D при фиксированном генераторе .

Лемма 1. При фиксированном G оптимальный дискриминатор D равен

$$D_G^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}.$$

Доказывается для произвольных функций, а не для параметрических, таких как нейронные сети.

При фиксированном G имеем

$$V(G,D) = \int_x P_r(x)\log(D(x))\,dx + \int_z P_z(z)\log(1-D(g(z)))\,dz = \int_x [P_r(x)\log(D(x)) + P_g(x)\log(1-D(x))]\,dx.$$

Максимум этого функционала достигается, когда достигается максимум подынтегрального выражения. Вычисляя производную по $D(x)$, получим

$$\frac{P_r(x)}{D(x)} - \frac{P_g(x)}{1-D(x)} = 0$$

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

Замечаем, что при фиксированном G функционал потерь для D представляет собой максимизацию логарифмического правдоподобия условной вероятности:

$$\max_D E_{x\sim P_r}[\log D_G^*(x)] + E_{z\sim P_z}[1-\log D_G^*(G(z))] = E_{x\sim P_r}[\log D_G^*(x)] +$$
$$E_{x\sim P_g}[1-\log D_G^*(x)] = E_{x\sim P_r}\left[\log\frac{P_r(x)}{P_r(x)+P_g(x)}\right] + E_{x\sim P_g}\left[1-\log\frac{P_g(x)}{P_r(x)+P_g(x)}\right] \quad (2.2.2).$$

Лемма 2. Глобальный оптимум при оптимальном дискриминаторе D достигается тогда и только тогда, когда $P_g = P_r$, где значение функционала (2.2.1) равно $-log4$.

При $P_g = P_r$ $D_G^*(x) = \frac{1}{2}$ $\max_D V(D,G) = -\log 4$. Убедимся, что $-log4$ это единственное оптимальное значение, достигнутое при $P_g = P_r$. Отнимем $E_{x\sim P_r}[-\log 2] + E_{x\sim P_g}[-\log 2] = -\log 4$ и добавим к (2.2.2):

$$\max_D V(D,G) = -\log 4 + KL\left(P_r||\frac{P_g+P_r}{2}\right) + KL\left(P_g||\frac{P_g+P_r}{2}\right) = -\log 4 + JS(P_r||P_g)$$

Последнее слагаемое это расстояние Йенсена-Шеннона между P_r и P_g, и так как оно не может быть меньше нуля, значит $-log4$ есть оптимальное значение.

И так, мы разобрались с критерием обучения (2.2.1) и увидели, что он неявным образом минимизирует расстояние Йенсена-Шеннона. В [2] доказывается сходимость этого метода для случая произвольных функций.

2.3 Неустойчивость обучения классических GAN.

С момента появления GAN они все больше и больше находят применения в разных задачах машинного обучения, особенно в задачах компьютерного зрения, такие как: генерация человеческих лиц, перевод изображение в изображение [24], шумоподавление и улучшение качества изображений [6], и т.д. (идея последней статьи и применяется в данной работе). Со всеми значительными успехами применения GAN, они наделены и недостатком неустойчивости обучения из-за чего не всегда получается добиться желаемых результатов. В [5] делается тщательный теоретический анализ трудностей, возникающих при обучении GAN, мы здесь перечислим главные из них и дадим их интуитивное описание.

1. Низкая размерность носителя функции распределения P (low dimensional manifold).
   Когда распределение P концентрируется на небольшом сгустке $supp\,P(x) = \{x \in R^n;\ P(x) \neq 0\}$ пространства R^n, для большинства точек из R^n $P(x) = 0$, и это способствует неустойчивости обучения GAN. Мы еще вернемся к этой проблеме в следующей главе, когда будем рассматривать Вассерштайн GAN, так как разобраться в ней нам необходимо для понимания предлагаемого метода.
2. Проблема затухающего градиента генератора (vanishing gradient).
   Когда реальные и поддельные объекты идеально различаются дискриминатором, мы имеем $D(x) = 1,\ \forall x \sim P_r$ и $D(x) = 0,\ \forall x \sim P$ и т. о. (2.2.1) равняется нулю и не дает полезных градиентов для обучения генератора.

3. Коллапс мода (mode collapse).

   Во время обучения генератор может застрять на небольшом множестве генерируемых значений и выдавать только одинаковые объекты. Например, при обучении генерации рукописных цифр генератор может выдавать только число 9. Генератор хоть и может перехитрить дискриминатор, но он не может научиться представлять сложные многомерные распределения.
4. Отсутствие хорошей метрики качества.

   Трудно построить хорошую метрику для отслеживания качества GAN или для сравнения двух GAN.

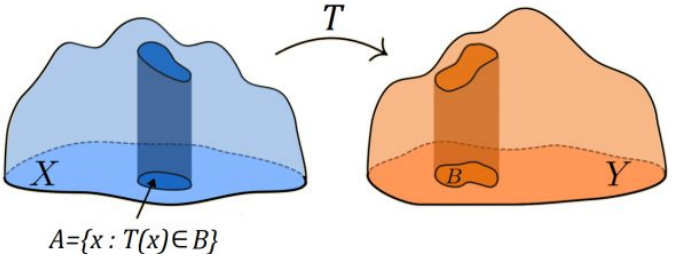
В последние годы регулярно появляются разные вариации GAN, которые в какой-то мере решают эти проблемы, но идеального решения не существует. Наверное один из самых качественных (особенно для нашей задачи) GAN, это так называемый Вассерштайн GAN, который особенно хорошо справляется с первой проблемой, упомянутой выше поэтому и перейдем к его описанию.

2.4 Вассерштейн GAN

Вассерштайн GAN описывается в [1]. Здесь мы покажем основные его факты, которые лежат в основе выбора именно этого GAN для настоящей работы.

Расстояние Вассерштайна (называемое также как EM- Earth-Mover) - это мера близости между двумя функциями распределения, как и упомянутые ранее расстояния Кульбака-Лейблера, Йенсена-Шеннона и другие.

Расстояние EM происходит от оптимальной транспортной задачи [18], [19], его можно интуитивно представить как минимальную стоимость перемещения и преобразование кучи грунта в форме одного распределения в форму другого (поэтому и называется earth mover distance).



изображение из [18]

Проблема была впервые формализована французским математиком Гаспаром Монжем в 1781 г, её развитие во время Великой Отечественной Войны продолжил советский математик Леонид Канторович.

Формулировка Монжа. Пусть даны $\mu \in P(x)$ и $\nu \in P(y)$, транспортная задача формулируется как минимизация

$$min \int_x c\left(x, T\left(x\right)\right) d\mu\left(x\right),$$

где c некоторая функция расстояния : $(x \times y) \to R$.

Формулировка Канторовича

$$\min \int_{x \times y} c\left(x, y\right) d\pi\left(x, y\right),$$

где $\pi(x, y) \in \Pi(x, y)$ множество всевозможных совместных распределений.

На второй формулировке и основывается расстояние Вассерштайна

$$W\left(P_r, P_g\right) = \inf_{\gamma \in \Pi(P_a, P_y)} E_{(x,y)\sim\gamma}\left[\|x - y\|\right] \tag{2.4.1}$$

Стоит заметить, что главное отличие метрики Вассерштайна от остальных известных нам метрик, таких как KL, JS, TV, состоит в том что, в расстоянии между двумя функциями распределения учитывается и само расстояние между объектами, кроме того вычисление этого расстояния само по себе является задачей оптимизации. В разных источниках (как в [13] или [14]) можно найти простые примеры транспортной задачи для дискретных случаев, при которых она является задачей линейного программирования.

Рассмотрим важный пример из [1], показывающий как Вассерштайн GAN справляется с проблемой низкой размерности, описанный в (2.2.1), где GAN, основанные на других метриках проваливаются.

Перечислим основные метрики, используемые при вычислении расстояния между функциями распределения.

1. Total Variation - TV :

$$TV\left(P_r, P_g\right) = \sup_A \left|P_r\left(A\right) - P_g\left(A\right)\right|$$

2. Kullback-Leibler - KL :

$$KL\left(P_r || P_g\right) = \int_x \log\left(\frac{P_r\left(x\right)}{P_g\left(x\right)}\right) P_r\left(x\right) dx$$

Не определено, если $P_g(x) = 0$ и $P_r(x) > 0$

3. Jensen-Shannon - JS:

$$JS\left(P_r, P_g\right) = \frac{1}{2}\left(KL\left(P_r||P_m\right) + KL\left(P_g||P_m\right)\right)$$

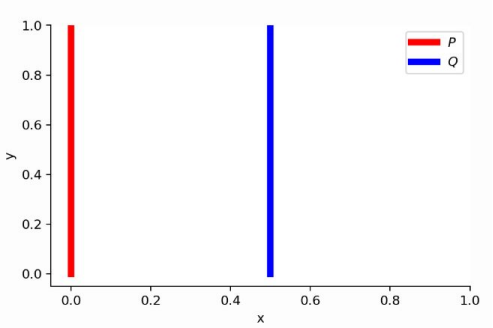
где $P_m = \frac{P_r + P_g}{2}$

4. Wasserstein - W:

$$W\left(P_r, P_g\right) = \inf_{\gamma \in \Pi(P_a, P_y)} E_{(x,y)\sim\gamma}\left[\|x - y\|\right]$$

Пусть имеется два распределения P и Q,

$\forall (x,y) \in P,\ x = 0,\ y \sim U(0,1),\ \forall (x,y) \in Q,\ x = \theta,\ y \sim U(0,1),$ где $U(0,1)$ равномерное распределение на отрезке (0,1).



Пересечение пусто, если $\theta \neq 0$.

Если $\theta = 0$

$$TV\left(P, Q\right) = KL\left(P||Q\right) = KL\left(Q||P\right) = JS\left(P, Q\right) = W\left(P, Q\right) = 0$$

в противном случае

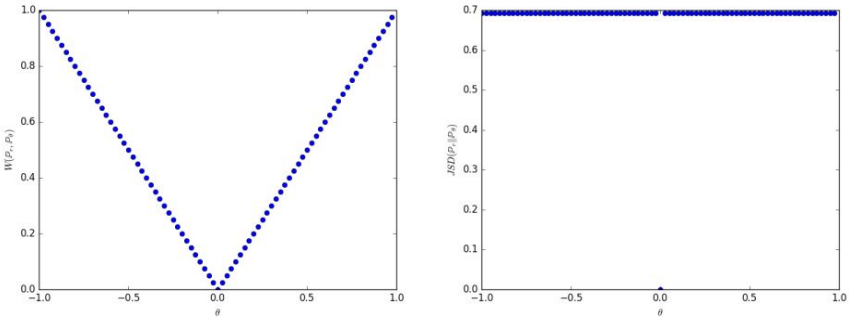
$$TV\left(P, Q\right) = 1$$

$$KL\left(P||Q\right) = KL\left(Q||P\right) = \sum_{x,y\sim U(0,1)} \log\frac{1}{0} \cdot 1 = \infty$$

$$JS\left(P, Q\right) = \frac{1}{2}\log\frac{P\left(x,y\right)}{\frac{P\left(x,y\right) + Q\left(x,y\right)}{2}} P\left(x,y\right) + \frac{1}{2}\log\frac{Q\left(x,y\right)}{\frac{P\left(x,y\right) + Q\left(x,y\right)}{2}} Q\left(x,y\right) =$$

$\log 2$

$$W\left(P, Q\right) = |\theta|$$



Слева показан график функции $\theta \to W(P_r, P_\theta)$, а справа $\theta \to JS(P_r, P_\theta)$, при $P_r = P$, $P_\theta = Q$

Как видим на этом примере, GAN, основанные на оптимизации метрик как TV, KL, JS, не могут обучаться, так как градиенты этих функций равны нулю и невозможно сделать градиентный шаг, однако в отличие от остальных метрик, метрика Вассерштайна сходится, имея почти везде градиенты не равные нулю. Однако, прямая оптимизация (2.4.1) практически невозможна, так как невозможно вычислять всевозможные совместные распределения, поэтому авторы статьи предлагают перейти к двойственной задаче Канторовича-Рубинштейна [19]

$$W(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} E_{x \sim P_r}[f(x)] - E_{x \sim P_\theta}[f(x)]$$

где sup берется по всем К-Липшиц функциям:

$|f(x_1) - f(x_2)| \leq k\,|x_1 - x_2|, \ \forall x_1, x_2 \in R^n, \ k \in R$.

Окончательный вид $W(P_r, P_\theta)$ получается после замены

$f(x)$ − ов на параметрические функции $f_w(x)$ и sup на max :

$$\max_{w \in W} E_{x \sim P_r}[f_w(x)] - E_{z \sim p(z)}[f_w(g_\theta(z)] \qquad (2.4.2)$$

При такой постановке дискриминатор больше не делает прямые вероятностные предсказания, а обучается находить K -Липшиц функции для приближенного вычисления (2.4.2). По мере уменьшения критерий (2.4.2) выходные данные генератора становятся похожими на объекты реального распределения P_r. В процессе обучения надо обеспечить условие К-Липшица, чтобы все это сработало. Авторы статьи используют простой, но очень практичный прием: после каждого обновления градиентов делается weight clipping (большие веса дискриминатора отрезаются) на отрезок (-0.01,0.01). Процесс обучения состоит из следующих трех шагов:

1. Фиксируются параметры θ, обучается критик (дискриминатор) f_w, что в свою очередь дает приближенное значение $W(P_r, P_\theta)$.
2. Как только нашли оптимальный f_w, то вычисляется градиент по θ $\nabla_\theta W = -E_{z \sim P(z)}[\nabla_\theta f_w(g_\theta(z))]$ для n штук z.

3. Делается градиентный спуск для θ и продолжается цикл обучения.

2.5 Генерация дискретных распределений

Все метрики, описанные в 2.4 предполагают, непрерывные распределения (непрерывность в смысле машинного обучения, а не в строго математическом смысле), поэтому и GAN особенно хорошо обучаются, если объекты $\{x_i\}_{i=1}^{n}$ изображения. Восстановление сложных дискретных распределений (такие как матрицы коллаборативных фильтрации) с помощью GAN непростая задача и отсутствие качественных моделей таких GAN ограничивает их применение в других областях, вне компьютерного зрения. Тем не менее в последние годы активно развивается теория дискретных GAN и есть успешные работы как [3], [17]. Так как модель из [3] основана на дискретной модификации Вассерштайн GAN, предполагается, что при обучении на дискретных данных в какой-то мере будут сохраняться преимущества метрики Вассерштайна, особенно ее способность обучаться при низкой размерности носителя распределений (пример из 2.4). Учитывая эти предположения, в настоящей работе за основу берется модель из [3] и модифицируется под нашу задачу, поэтому здесь стоит изложить основные факты этой модели, названной как Discrete Wasserstein Generative Adversarial Network (DWGAN).

2.5.1 DWGAN

Как показано в [3], прямое применение (2.4.1) на дискретных данных невозможно, поэтому и авторы, следуя идее [15], предлагают нижеприведенную модель.

Пусть $x = (x_1, x_2, \dots, x_n)$ дискретная случайная величина, где $x_i = \{0, 1, \dots, k\}$, пусть P_r и P_g два распределения над значениями x_i, тогда (дискретное) расстояние Вассерштайна между P_r и P_g выглядит так

$$W(P_r, P_g) = \min_{\gamma \in \Pi(P_r, P_g)} E_{(x,x') \sim \gamma} [d(x, x')] = \min_{\gamma \in \Pi(P_r, P_g)} \sum_i \sum_j \gamma(x_i, x_j)\, d(x_i, x_j)$$

где $\Pi(P_r, P_g)$ - множество всевозможных совместных распределений $\gamma(x, x')$, $x \sim P_r$, $x' \sim P_g$, а $d(x_i, x_j)$ - расстояние между x_i и x_j, в статье предлагается расстояние Хэмминга:

$$d(x_i, x_j) = \sum_k I(x_i(k) \neq x_j(k))$$

Соответствующая двойственная задача будет

$$\begin{cases} \max_f E_{x \sim p_r} [f(x)] - E_{x \sim P_g.} [f(x)] \\ f(x_i) - f(x_j) \leq d(x_i, x_j), \forall i, j = \{0, 1, \dots m\} \end{cases}$$

Последнее неравенство заменяет условие Липшица.

Авторы предлагают заменить разность $h(x,y) = f(x) - f(y)$:

$$\begin{cases} \max_h E_{(x,x')\sim(P_r,P_g)}[h(x,x')] \\ h(x_i,x_j) \le d(x_i,x_j), \forall i,j = \{0,1,\dots m\} \end{cases} \quad (2.5.1.1)$$

Пусть $y \in [0,1]^{nxk}$ one-hot представление вектора $x \in R^m$, а k-число классов. Критик $h(y,y')$ принимает два значения $y \sim P_r$ и $y' \sim P_g$, В архитектуре h определяется другая функция $\rho_w(y,y') = v \quad \rho : [0,1]^{2\times n\times k} \to [-1,1]^n$, определяется фильтр вектор $u \in R^n$: $u(i) = I(x(i) \neq \tilde{x}'(i))$, где $\tilde{x}' = I(x' > 0.5)$ бинаризованное значение выхода генератора. Критик h определяется как $h_w = u^T v = u \cdot \rho_w(y,y')$. Умножение на фильтр-вектора u и обеспечивает неравенство $h(x_i,x_j) \le d(x_i, x_j)$ в ходе обучения.

3. Предлагаемая работа

3.1 Постановка задачи и идея решения

Несмотря на весомые успехи годами отточенных методов коллаборативной фильтрации, практически во всех user-item матрицах существуют две проблемы: проблема разреженности матрицы и проблема шума (неинформативных) значений. Второй проблемы мы касаться не будем, наша задача связана с первой проблемой. Для смягчения первой проблемы нередко используются дополнительные данные, такие как пол, возраст и другие характеристики клиентов, а если таких данных нет, то приходится либо прямо применить некоторый метод коллаборативной фильтрации на исходной матрице, либо как-то снизить разреженность матрицы R, после чего применить методы коллаборативной фильтрации, такие как ALS или SGD ((1.2.1), (1.2.2)).

В этом и состоит наша задача:

смягчить разреженность матрицы R, добавляя новые правдоподобные значения (оценки на некоторые товары для некоторых клиентов), при этом добавить в матрицу R как можно меньше шума, после чего прогнать алгоритм коллаборативной фильтрации и улучшить некоторые метрики качества, такие как среднеквадратичное отклонение (MSE), P@K (precision at K), на тестовой выборке.

Для реализации этой идеи строится условный GAN, при мульти условии, используется способность к шумоподавлению (denoising) [6] - все это применяется для модификации DWGAN из (2.5.1). Начиная с 2017 года на ведущих конференциях, таких как RecSys - ACM,

KDD все больше и больше представляются статьи по применению GAN в рекомендательных системах.

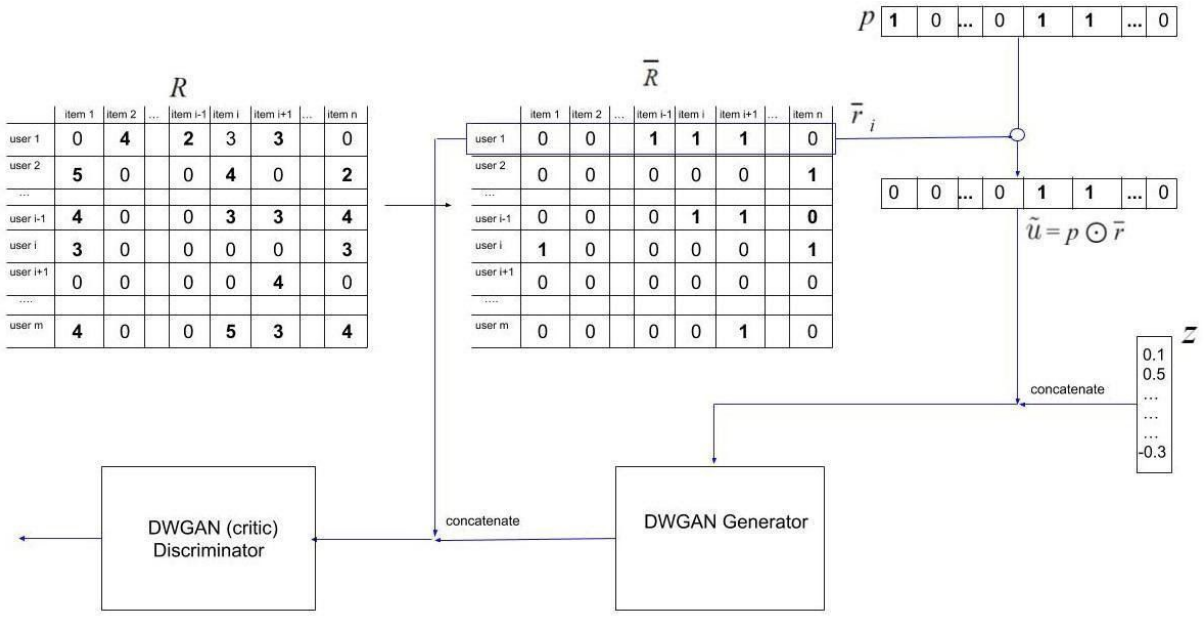
В [7] делается подробный обзор таких статей. Но прямая генерация искусственных объектов (строк или столбцов матрицы R) для аугментации тренировочного датасета (как это делается, например, для задач классификации медицинских изображений [20], [21] или [22], а также в топовых решениях конкурса kaggle для обнаружения диабетической ретинопатии, где тоже используется техника аугментации данных с помощью GAN), невозможна, так как сгенерированные векторы опять будут разреженными и внесут в матрицу R еще больше шума, поэтому и предлагается использовать вместо обычного условный GAN, который будет генерировать новые значения (оценки) для заданного клиента (по сути здесь тоже делаются предсказания на некотором очень маленьком количестве объектов, но здесь ключевую роль играет комбинированный подход использования GAN и коллаборативной фильтрации). Достижение желаемых результатов во многом зависит от выбранной модели GAN. Повторяя сказанное в (2.2.1), выбор модели DWGAN (2.5.1), как каркасной, для нашей задачи, обусловлен способностью Вассерштайн GAN обучаться и при низкой размерности носителя, а данные рекомендательных систем (строки или столбцы матрицы R) именно такие, так как в них очень много нулевых значений.

3.2 Реализация идеи

Пусть дана разреженная user-item матрица $R \in \mathrm{R}^{mxn}$ явной или неявной оценки (эксперименты сделаны на датасетах movielens-100k и movielens-1m явной оценки, где оценки $y = \{0, 1, 2, 3, 4, 5\}$). Если у нас задача неявной оценки, то этот шаг пропускается, в противном случае - берутся только негативные (или аналогично позитивные, без потери общности изложим дальнейший материал для негативного случая) объекты, т.е. делается бинаризация: $\forall (i,j),\ \overline{R}(i,j) = I(R(i,j) > 0\ \&\ R(i,j) < 4)$, т.е. объекты, которые не понравились клиенту обозначены единицами, а объекты, которые либо понравились, либо еще не были оценены - нулями. При обучении на вход генератору вместе со случайным шумом z подается и маскированная строка негативных оценок $\tilde{u}_i = p \odot \overline{r}_i$, где $p \in R^n$ случайный вектор нулей и единиц, в котором на каждом шаге доля нулей берется случайным образом, в диапазоне 10%-50% , $\overline{r}_i$ – строка матрицы $\overline{R}_i$, а дискриминатору D подается строка бинаризованной матрицы R : $\overline{r}_i$ как подлинный объект. Таким образом генератор научиться предсказывать те объекты, которые с высокой вероятностью не понравятся рассматриваемому клиенту.

После обучения нашего GAN дискриминатор нам уже не нужен, а с помощью генератора проходимся по всем строкам матрицы $\overline{R}$, (уже не маскированной), для каждого клиента находим объекты, которые с большой вероятностью не понравятся этому клиенту (так как на последнем слое дискриминатора стоит функция softmax, берем, только значения больше некоторого порогового значения, в экспериментах как пороговое значение используется 0.8, чтобы добавить только значения с высокой вероятностью и не добавлять много шума) и добавляем к исходной матрице R. Так как сгенерированные значения единицы, и они представляют собой негативные оценки пользователя, мы должны добавить значения 1, 2 либо 3, если рассматриваемая задача это задача явной оценки (в противном случае добавляются единицы). В наших экспериментах мы добавляем $\delta = \{1, 2, 3\}$ случайным образом, по мере их появления в исходной матрице R.

Процесс обучения нашего GAN (Discrete Conditional Collaborative Filtering Wasserstein GAN) выглядит следующим образом



На этапе предсказания $\tilde{u}_i = \overline{r}_i$, фиксируем пороговое значение, например $t = 0.8$ и проходимся по каждой строке матрицы R.

3.3 Некоторые замечания.

Хоть и такой метод кажется более приемлемым для задач неявных оценок, (так как при добавлении негативной аугментированной оценки $\delta = \{1, 2, 3\}$ мы вносим шум в исходную матрицу) но эксперименты на датасетах MovieLens-100k и MovieLens-1M показали, что при надлежащем выборе δ, порогового значения *t* и гиперпараметров обучения GAN, хорошее улучшение MSE и P@K достигаются и для задач явных оценок.

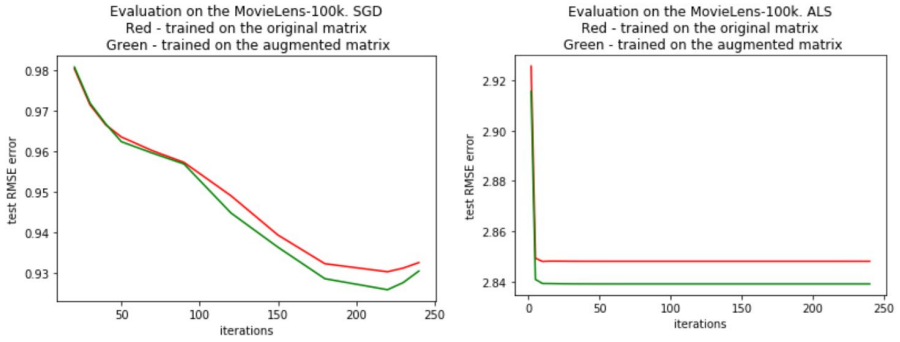
Заметим, что можно было подстроить эту архитектуру и метрику под дискретные данные, (так чтобы генератор выдавал значения $\{0, 1, \ldots, n\}$, а не только бинарные значения как в нашем случае), и избавиться от δ и t, но наши эксперименты показали, что бинарный подход, дает более качественные результаты.

Замечаем, что такой подход можно было и реализовать с помощью шумоподавляющего автокодировщика (denoising autoencoder) (подобная работа описана в [8], но для топ N рекомендаций), но теоретически такие автокодировщики и GAN решают совершенно разные задачи. Как улучшение предложенного метода предлагается аналогичным образом сделать аугментацию и для положительных оценок. Можно аугментированные значения учитывать с другими весами в функции потерь методов коллаборативных фильтрации. Еще можно скомбинировать с автокодировщиком: параллельно обучить шумоподавляющий автокодировщик, и взять пересечение множеств значений (наши эксперименты показали, что пересечение значений автокодировщика и нашей системы большое). Отдельное аугментирование автокодировщиком не дает значительное улучшение, по сравнению с предлагаемым методом.

Как дополнение планируется присоединение процессов обучения коллаборативной фильтрации и предлагаемого метода аугментирования в один процесс, но в таком случае аугментация тесно связывается с алгоритмом коллаборативной фильтрации и в некоторых случаях это может быть невозможно. Весь код экспериментов выложен в github.

3.4 Результаты экспериментов

3.4.1. MoviLens-100k



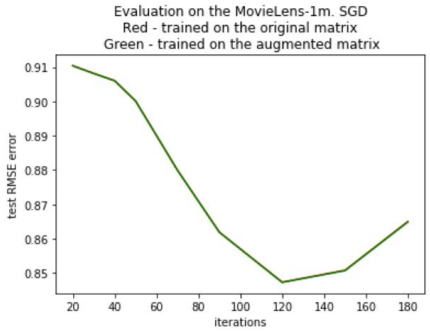
Бейслайн - MSE ALS, обученный на исходной матрице R - 2.848008

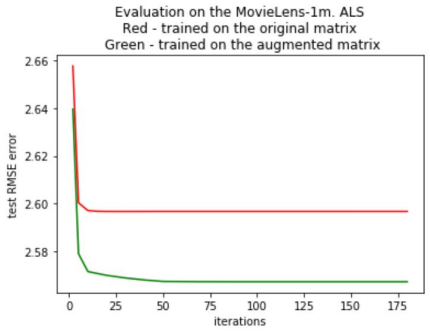
MSE ALS, обученный на аугментированной матрице R' - **2.839090**

Бейслайн - MSE SGD, обученный на исходной матрице R - 0.930387

MSE SGD, обученный на аугментированной матрице R' - **0.925967**

3.4.2. MovieLens-1m





Бейслайн - RMSE ALS, обученный на исходной матрице R - 2.596734

RMSE ALS, обученный на аугментированной матрице R' - **2.567347**

Бейслайн - RMSE SGD, обученный на исходной матрице R - 0.847364

RMSE SGD, обученный на аугментированной матрице R' - **0.847294**

3.5 Существующие методы

Так как описанная теория появилась совсем недавно, все существующие методы максимум 2-3 летней давности и их немного. Все существующие методы так или иначе слегка отличаются от нашего подхода, например в [22] аугментация делается с помощью дополнительной информаций, в [10] предлагается метод на комбинации коллаборативного шумоподовляющего автокодировщика (CDAE [8]) и GAN коллаборативной фильтрации (CFGAN[12]), для сбалансирования оценок. По моему убеждению ключ к качественной реализации предложенного метода состоит в оптимизации дискретного расстояния Вассерштайна с соответствующим дискретным ограничением Липшица, а действующие методы оптимизируют (явным или неявным образом) расстояния JS или KL, а именно: они основаны на классическом определении GAN из 2.2.

3.6. Заключение

Итак, в 1.1 и 1.2 мы показали 2 классических подхода к задаче низкорангового приближения разреженной матрицы R, широко используемых в задачах коллаборативной фильтрации. В 2-м разделе мы подробно описали классический GAN, проблемы неустойчивости обучения классических GAN, после чего показали улучшенный вариант Вассерштайн GAN. В конце 2

раздела мы увидели трудности восстановления дискретных распределений с помощью GAN и перешли к описанию DWGAN, на основе дискретной интерпретации Вассерштайн GAN. В 3-м разделе предложили метод для снижения разреженности с помощью специально разработанной системы GAN. Предложенный метод можно использовать для аугментации исходной матрицы R, после чего прогнать метод коллаборативной фильтрации уже на аугментированной матрице R'. Надо подчеркнуть, что поставленная задача далеко непростая задача, так как все рассматриваемые методы коллаборативной фильтраций SGD, ALS (и NMF) уже почти идеально восстанавливают пропущенные значения тестовой выборки и все они широко используются в продукции. Несмотря на трудность поставленной задачи, результаты 3.4, полученные на двух широко известных в рекомендательных системах датасетах MovieLens-100k и MovieLens-1M, показывают, что этот метод действительно дает улучшение оценки MSE на тестовой выборке, по сравнению с прямым использованием коллаборативных методов SVD, ALS и sklearn NMF. В 3.3 делаются некоторые замечания и предлагаются идеи для дальнейшего развития работы.

Главный недостаток предложенного метода состоит в его бинарной конфигурации (если метод применяется для задач неявных оценок): подбирая $\delta = \{1, 2, 3\}$, мы вносим шум в исходную матрицу R. Как и подчеркивается в 3.3, самой главной идеей для дальнейшей работы станет разработка дискретного преобразования метрики Вассерштайна (с соответствующим ограничением условия Липшица) для задач коллаборативной фильтрации неявных оценок, со строгим теоретическим фундаментом.

4. Список литературы

[1] Martin Arjovsky. Wasserstein GAN. 2017

[2] Ian Goodfellow. Generative Adversarial Nets. 2014

[3] Discrete Wasserstein Adversarial Networks (DWGAN). Under review 2018.

[4] Yehuda Koren. Matrix Factorization Techniques for Recommendation Systems. 2009

[5] Martin Arjovsky. Towards Principled Methods for Training Generative Adversarial Networks. 2017

[6] Qiaojing Yan. DCGANs for image super-resolution, denoising and deblurring. 2019

[7] Min Gao. Recommender Systems Based on Generative Adversarial Networks: A Problem-Driven Perspective. 2020

[8] Yao Wu. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. 2016

[9] Qinyong Wang. Enhancing Collaborative Filtering with Generative Augmentation. 2019

[10] Dong-Kyu Chae. Rating Augmentation with Generative Adversarial Networks towards Accurate Collaborative Filtering. 2019

[11] Christopher Aberger. Recommender: An Analysis of Collaborative Filtering Techniques. 2014

[12] Dong-Kyu Chae. CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks. 2018

[13] Wasserstein Distance and the Kantorovich-Rubinstein Duality. 2017

[14] IFT 6085 - Lecture 13 Wasserstein GANs. 2019

[15] Ishaan Gulrajani. Improved Training of Wasserstein GANs. 2017

[16] Mehdi Mirza. Conditional Generative Adversarial Nets. 2014

[17] Boundary-Seeking Generative Adversarial Networks. 2018

[18] Matthew Thorpe. Introduction to Optimal Transport. 2018

[19] Cédric Villani. Optimal Transport. 2009

[20] Maayan Frid-Adar. GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification. 2018

[21] Veit Sandfort. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. 2019

[22] Fabio Henrique Kiyoiti dos Santos Tanaka. Data Augmentation Using GANs. 2019

[23] Ruslan Salakhutdinov. Restricted Boltzmann Machines for Collaborative Filtering. 2007

[24] Jun-Yan Zhu. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. 2018