Московский государственный университет имени М.В. Ломоносова
Факультет Вычислительной математики и кибернетики
Кафедра Математических методов прогнозирования

Кудрявцев Георгий Алексеевич

Построение ансамбля алгоритмов рекомендаций

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
д.ф-м.н., профессор
Дьяконов Александр Геннадьевич

Москва, 2016

Содержание

Аннотация

Данная работа посвящена задаче ранжирования по данным с двоичной релевантностью. Был проведен обзор современных факторизационных методов решения этой задачи, а также метрик, оценивающих качество ранжирования. Проведен анализ качества работы алгоритмов на различных наборах данных. Был предложен метод ансамблирования, который стабильно улучшает качество ранжирования.

1 Введение

С ростом популярности электронной коммерции, возникла задача помощи пользователям в поиске товаров, которые им понравятся. Одними из инструментов, используемые для решения этой проблемы, являются рекомендательные системы.

Коллаборативная фильтрация является одним из методов построения рекомендательных систем. Она использует известные оценки или предпочтения для построения рекомендации пользователям, чьи оценки и предпочтения неизвестны. Идея данного метода состоит в предположении того, что пользователи, имеющие похожие предпочтения в прошлом, будут иметь похожие предпочтения в будущем.

Существует три различных подхода в коллаборативной фильтрации.

Первый подход называется Memory-based. Его идея заключается в вычислении сходства между пользователями или предметами. Соответственно, похожие предметы или пользователи должны иметь похожие оценки или предпочтения.

Второй подход называется Model-based. Его идея заключается в создании моделей при помощи интеллектуального анализа данных и машинного обучения. Модель обучается на реальных данных, например, на истории покупок интернет-магазина, а далее выдает рекомендации для пользователей, чьи предпочтения неизвестны.

Третий подход – Hybrid. Его идея заключается в использовании первого и второго подхода, компенсируя недостатки обоих.

В данной работе будет рассматриваться второй подход.

Также следует упомянуть о таком важном понятии как обратная связь(feedback). Обратной связью некоторого пользователя на предмет называют некоторое событие, по которому можно судить о предпочтении автора. Например, это может быть оценка фильма по десятибалльной шкале. Либо клик на описание товара в интернет-магазине.

В коллаборативной фильтрации выделяют два типа обратной связи: с явным откликом(explicit feedback) и неявным откликом(implicit feedback).

В первом случае пользователь осознано оценивает предмет и производит соответствующий отклик. Например, оценивает работу интернет-магазина по пятибалльной шкале.

Неявный отклик обозначает лишь то, что между пользователем и предметом произошло взаимодействие. Например, покупатель зашел на страницу с описанием товара, либо несколько раз посмотрел видеоролик. Как видно, из этого отклика нельзя выяснить, имеется ли у пользователя положительное или отрицательное предпочтение к предмету, и есть ли оно вообще.

В данной работе будет рассмотрен случай, в котором набор данных состоит

только из неявного отклика, причем он будет в бинарном виде. '1' обозначает наличие взаимодействия с предметом, а '0' его отсутствие. Такая ситуация возможна в некоторых случаях. Например, дружба между пользователями в социальной сети или история встреч на сайте знакомств. Далее будем считать, что неявный отклик показывает положительное предпочтение пользователя к предмету.

В качестве моделей будет рассмотрен класс факторизационных методов. Их основная идея лежит в представлении предпочтения пользователя к предмету в виде скалярного произведения их латентных векторов [5]. Данные методы хорошо себя показали в известном конкурсе Netflix Prize [6].

Также в решении победителей не малую роль сыграл ансамбль алгоритмов, поэтому в данной работе было решено исследовать, как можно улучшить качество современных факторизационных методов в задаче ранжирования при помощи построения ансамблей.

2 Цель работы

Целью работы является изучение и сравнение существующих факторизационных алгоритмов ранжирования, а также создание новых, более эффективных методов при помощи построения ансамблей.

Данная работа разделена на несколько частей.

1. Ввод необходимых понятий и обозначений.
2. Обзор современных методов ранжирования.
3. Построение различных ансамблей методов ранжирования.
4. Тестирование и сравнение методов и ансамблей на реальных данных.

3 Обозначения

Набор данных R представлен в виде матрицы размером $|U| \times |I|$, где U - множество пользователей, I - множество предметов. В дальнейшем обозначим $M = |U|$ и $N = |I|$

Если между пользователем u и предметов i произошел неявный отклик, то $R_{ui} = 1$. В противном случае $R_{ui} = 0$. Будем считать, что предмет i релевантен пользователю u, если $R_{ui} = 1$

Обозначим за $rank(u, i)$ номер позиции предмета i в упорядоченном списке, который был получен при помощи метода ранжирования, для пользователя u.

Введем функцию $rel(u,k)$ такую что, $rel(u,k)=1$, если предмет, стоящий на k позиции в упорядоченном списке предметов для пользователя u, релевантен. В противном случае $rel(u,k)=0$.

Пусть

P - множество профилей пользователей,

Q - множество профилей предметов,

P_u - профиль(латентный вектор) пользователя u,

Q_i - профиль(латентный вектор) предмета i.

Прогноз f_{ui} предпочтения пользователя u предмета i представлен в виде скалярного произведения латентных векторов P_u и Q_i, т.е. $f_{ui}=\langle P_u, Q_i\rangle$

Также в дальнейшем будет часто использоваться сигмоида. Обозначим ее за $\sigma(x)=\frac{1}{1+e^{-x}}$.

4 Критерии качества

В задаче ранжирования не существует однозначно правильного функционала качества, поэтому было решено использовать сразу несколько.

4.1 $P@n$

Определим эту метрику для одного пользователя.

$$P@n(u)=\frac{1}{N}\sum_{k=1}^{n} rel(u,k)$$

Теперь для всех пользователей.

$$P@n=\frac{1}{M}\sum_{u=1}^{M} P@n(u)$$

$P@n$ показывает среднюю долю релевантных объектов по всем пользователям. Недостатком этой метрики является то, что она не учитывает порядок предметов. Например, если пользователь получил только один релевантный предмет, то для этой метрики не важно, был ли он в начале списка или в конце.

4.2 $1call@n$

Определим эту метрику для одного пользователя.

$$1call@n(u)=[\sum_{k=1}^{n} rel(u,k)>0]$$

Теперь для всех пользователей.

$$P@n = \frac{1}{M}\sum_{u=1}^{M} 1call@n(u)$$

$1call@n$ показывает долю пользователей, у которых был хотя бы один релевантный предмет. Метрика не учитывает ни порядок, ни количество релевантных предметов.

4.3 MRR

MRR – Mean Reciprocal Rank

Пусть $firstrank(u)$ - номер позиции первого релевантного предмета в ранжированном списке для пользователя u. Номер позиции в списке начинается с 1.

$$MRR = \frac{1}{M}\sum_{u=1}^{M} \frac{1}{firstrank(u)}$$

Эта метрика используется, если рекомендательной системе важнее подать пользователю один релевантный предмет в начало списка.

4.4 $NDCG@n$

NDCG – Normalized Discounted Cumulative Gain

$$G(u,k) = 2^{rel(u,k)} - 1$$
$$D(k) = \frac{1}{log_2(k+1)}$$
$$DCG@n(u) = \sum_{k=1}^{n} G(u,k)D(k)$$
$$NDCG@n(u) = \frac{DCG@n(u)}{maxDCG@n}$$

Эта метрика является популярной в информационном поиске. Она учитывает и порядок, и количество релевантных предметов. Также большим плюсом является то, что $NDCG$ работает в случае различных уровней релевантности.

4.5 MAP

MAP – Mean Average Precision

$$AP@n(u) = \frac{1}{n}\sum_{k=1}^{n} rel(u,k)P@k(u)$$

$$MAP@n = \frac{1}{M}\sum_{k=1}^{M} AP@n(u)$$

Аналогично предыдущей метрике, MAP является достаточно популярной, учитывает и порядок, и количество релевантных предметов.

5 Существующие методы

В ходе работы были рассмотрены факторизационные методы ранжирования. Их основная идея заключается в представлении пользователей и предметов в виде векторов латентных векторов P_u и Q_i. Величина $f_{ui} = \langle P_u, Q_i \rangle$ показывает заинтересованность пользователя u в предмете i. Следовательно, по величине f_{ui} можно ранжировать предметы для конкретного пользователя.

Далее приведен список факторизационных методов с их кратким описанием.

1. **CLiMF** – Факторизационный метод, который оптимизирует сглаженную версию метрики MRR. [1]

2. **MPR_MF** – Факторизационный метод, который оптимизирует AUC. [2]

3. **TFMAP** – Факторизационный метод, который оптимизирует сглаженную метрику MAP. [3]

4. **iMF** – Факторизационный метод, который оптимизирует взвешенную квадратичную ошибку. [4]

5. **PopRec** – Простой метод, который ранжирует предметы по убыванию количества пользователей, для которых данный предмет является релевантным. В результате метод выдает для каждого пользователя один и тот же ответ.

5.1 CLiMF

Данный метод использует в качестве функционала качества MRR. Заметим, что MRR можно переписать в другом виде.

$$MRR = \frac{1}{M}\sum_{u=1}^{M}\sum_{i=1}^{N}\frac{R_{ui}}{rank(u,i)}\prod_{k=1}^{N}(1 - R_{uk}[rank(u,k) < rank(u,i)])$$

Но MRR не является гладкой функцией, поэтому авторы метода решили оптимизировать сглаженную версию этой метрики. В качестве регуляризатора был взят L2-регуляризатор. В итоге получаем следующий функционал качества.

$$F(P,Q) = \sum_{u=1}^{M}\sum_{i=1}^{N}[R_{ui}(\ln(\sigma(f_{ui})) + \sum_{k=1}^{N}\ln(1 - R_{uk}\sigma(f_{uk} - f_{ui})))] - \frac{\lambda}{2}(||U||^2 + ||V||^2)$$

Далее этот функционал оптимизируется при помощи стохастического градиентного спуска.

Алгоритм 1 обучение метода CLiMF

Вход: набор данных R, параметр регуляризации λ, скорость обучение γ, максимальное число итерации maxiter, длина латентных векторов K

Выход: обученные латентные векторы P, Q

1: **Цикл** i = 1 .. M **выполнять**
2: $N_u = \{i | R_{ui} > 0, 1 \leq i \leq N\}$
3: **Конец цикла**
инициализируем $U^{(0)}$ и $V^{(0)}$ случайными значениями. t = 0.
4: **Повторять**
5: **Цикл** $u = 1..M$ **выполнять**
6: $P_u^{(t+1)} = P_u^{(t)} + \gamma\frac{\partial F}{\partial P_u^{(t)}}$
7: **Цикл** $i \in N_u$ **выполнять**
8: $Q_i^{(t+1)} = Q_i^{(t)} + \gamma\frac{\partial F}{\partial Q_i^{(t)}}$
9: **Конец цикла**
10: **Конец цикла**
11: **Пока** $t \leq itermax$

5.2 BRP_MF

Авторы данного метода применили байесовский подход к решению задачи ранжирования. Для каждого пользователя u предметы разбиты на 3 класса: релевантные и нерелевантные предметы, а также предметы с неизвестной релевантностью. Собственно, для последнего класса и нужно строить ранжирование.

Пусть θ - параметр метода. В нашем случае это P и Q. I_u^+ – множество релевантных предметов пользователя u. I_u^- – множество нерелевантных предметов пользователя u. $>_u$ - это ранжированный список предметов для пользователя u. Требуется максимизировать вероятность $p(\theta| >_u)$.

$$
\begin{aligned}
&p(\theta| >_u) \propto p(>_u |\theta)p(\theta) \\
&p(>_u |\theta) = \prod_{i,j:\ i\in I_u^+, j\in I_u^-} p(i >_u j|\theta) \\
&p(\theta) \sim N(0, \lambda I) \\
&p(i >_u j|\theta) = \sigma(f_{ui} - f_{uj})
\end{aligned}
$$

В итоге имеем следующий функционал качества.

$$
F(P,Q) = \ln(p(\theta| >_u)) = \ln(p(>_u |\theta)p(\theta)) = \sum_{u,i,j:u\in U\ \ i\in I_u^+\ \ j\in I_u^-} \ln(\sigma(f_{ui}-f_{uj}))-\lambda||\theta||^2
$$

Нормальное априорное распределение задает L2-регуляризацию, а сигмоида позволяет легко рассчитывать производные для данного функционала. Авторами было показано, что данный алгоритм оптимизирует AUC.

Алгоритм 2 обучение метода BRP_MP

Вход: набор данных R, параметр регуляризации λ, скорость обучение γ, максимальное число итерации maxiter, длина латентных векторов K

Выход: обученные латентные векторы P, Q

1: Инициализируем θ случайными значениями, t = 0
2: $|R|$ - общее число взаимодействий пользователей с предметами.
3: **Повторять**
4: берем случайную тройку (u, i, j), где $u \in U, i \in I_u^+, j \in I_u^-$
5: $\theta = \theta + \alpha\big(\frac{1}{1+e^{(f_{ui}-f_{uj})}}\frac{\partial}{\partial\theta}(f_{ui} - f_{uj}) + \lambda\theta\big)$
6: t = t + 1
7: **Пока** $t \leq itermax \cdot |R|$

5.3 iMF

За основу iMF был взят оригинальный SVD, функционал качества которого выглядит следующим образом.

$$F(P,Q) = \sum_{R_{ui}-\text{известно}} (R_{ui} - f_{ui})^2 + \lambda(||P||^2 + ||Q||^2)$$

Недостаток SVD заключается в том, что он показывает плохое качество ранжирования в поставленной задаче. Чтобы преодолеть данную проблему, авторы метода поменяли функционал качества на следующий.

$$F(P,Q) = \sum_{u,i} c_{ui}(g_{ui} - f_{ui})^2 + \lambda(||P||^2 + ||Q||^2)$$

$$g_{ui} = \begin{cases} 0 & \text{если } R_{ui} = 0 \\ 1 & \text{если } R_{ui} > 0 \end{cases}$$

$$c_{ui} = 1 + \alpha R_{ui}$$

Переменная g_{ui} отвечает за неявный отклик между пользователем u и предметом i. Т.е. iMF пытается определить не уровень предпочтения пользователя, а неявный отклик. Хотя в нашем случае это одно и тоже.

Переменная c_{ui} является весом каждого квадратного слагаемого. Чем больше предпочтение, тем больше вес.

Метод обучается при помощи ALS [5].

Пусть Q – матрица профилей предметов размера $N \times K$, где K - размерность латентного вектора. Каждая i-ая строка равна латентному вектору предмета i. C^u – диагональная матрица размера $N \times N$, в которой $C^u_{ii} = c_{ui}$. S_u – вектор размера N, в котором $S_{ui} = R_{ui}$. Тогда латентный вектор пользователя u обновляется по следующей формуле.

$$P_u = (Q^T C^u Q + \lambda I)^{-1} Q^T C^u S_u$$

Заметим, что $(Q^T C^u Q + \lambda I) = Q^T Q + Q^T(C^u - I)Q$. Матрицу $Q^T Q$ можно вычислить один раз перед обновлением всех латентных векторов пользователей, а в матрице $C^u - I$ количество ненулевых элементов равно числу взаимодействий пользователя u с предметами.

Для латентных векторов рассуждения аналогичны.

Алгоритм 3 обучение метода iMF

Вход: набор данных R, параметр регуляризации λ, скорость обучение γ, параметр весовых коэффициентов α, максимальное число итерации maxiter, длина латентных векторов K

Выход: обученные латентные векторы P, Q

Инициализируем P и Q случайными значениями, t = 0
Повторять
    вычислить матрицу Q^TQ
    Цикл $u = 1..M$ **выполнять**
        обновить P_u
    Конец цикла
    вычислить матрицу P^TP
    Цикл $i = 1..N$ **выполнять**
        обновить Q_i
    Конец цикла
Пока $t \leq itermax$

5.4 TFMAP

Данный метод использует в качестве функционала качества MAP. Заметим, что MAP можно переписать в следующем виде.

$$MAP = \frac{1}{M}\sum_{u=1}^{M}\frac{\sum_{i=1}^{N}\frac{R_{ui}}{rank(u,i)}\sum_{j=1}^{N}R_{uj}[rank(u,j) \leq rank(u,i)]}{\sum_{i=1}^{N}R_{ui}}$$

Далее проводятся рассуждения аналогичные CLiMF. Метрика TFMAP не является гладкой, следовательно, будем оптимизировать приближенную гладкую версию это метрики. Также добавим L2-регуляризатор.

В итоге получаем следующую формулу.

$$F(P,Q) = \sum_{u=1}^{M}\frac{1}{\sum_{i=1}^{N}R_{ui}}\sum_{i=1}^{N}R_{ui}\sigma(f_{ij}) \times \sum_{j=1}^{N}R_{mj}\sigma(f_{uj}-f_{ui}) - \frac{1}{2}\lambda(||P||^2 + ||Q||^2)$$

Далее возникает проблема подсчета частной производной по Q_i. Она вычисляется при помощи следующей формулы.

$$\frac{\partial F}{\partial Q_i} = \sum_{u=1}^{M}\frac{R_{ui}P_u}{\sum_{i=1}^{N}R_{ui}}\sum_{j=1}^{N}\Big(\sigma'(f_{ui})\sigma(f_{uj}-f_{ui}) + (\sigma(f_{uj})-\sigma(f_{ui}))\sigma'(f_{uj}-f_{ui})\Big)R_{ui} - \lambda Q_i$$

Сложность вычисления этого выражения – $O(KN|R|)$, где K - размерность латентных векторов, $|R|$ - количество взаимодействий пользователей с предметами. На практике подсчет такой производной несет большие вычислительные затраты. Для ускорения вычисления выражения авторы заменили $\sum_{i=1}^{N}$ на $\sum_{i \in B_u}$, где B_u - множество предметов специального вида. Далее приведен алгоритм построения этого множества.

Алгоритм 4 построение множества B_u

Вход: Q_i и f_{ui} для всех i, размер выборки n, P_u

Выход: B_u

1: $B_u = \emptyset$
2: $B_u = B_u \cup \{i|R_{ui} = 1\}$
3: $n_u = |B_u|$
4: $p = \min_{i \in B_u} f_{ui}$
5: $S = \{i|R_{ui} = 0\} \cap \{i|f_{ui} > p\}$
6: Случайно выбираем подмножество $L \subset S$ размера n.
7: Отсортируем предметы $i \in L$ по убыванию f_{ui}
8: Выбираем первые n_u предметов из полученного списка. Обозначим это множество предметов за B^-
9: $B_u = B_u \cup B^-$

Метод обучается при помощи стохастического градиентного спуска.

6 Эксперименты

6.1 Наборы данных

В исследовании были использованы 4 разных набора данных.

- **Epinion**[1] - социальная сеть, в которой публикуются покупательские отзывы и рецензии на товары и услуги. Каждый участник решает кому он "доверяет". Следовательно, если пользователь u "доверяет"пользователю i, то $R_{ui} = 1$

- **Slashdot**[2] - сайт, который предоставляет различные новости в сфере IT. Пользователи сами публикуют новости, в то время как другие пользователи их оценивают и обсуждают. Также данный сайт предоставляет возможность пользователям объявлять друг друга "врагом"или "другом". $R_{ui} = 1$, если пользователь u является "другом"или "врагом"пользователя i.

[1] https://snap.stanford.edu/data/soc-Epinions1.html

[2] https://snap.stanford.edu/data/soc-Slashdot0811.html

- **MovieLens**[3] - сайт, в котором пользователи рекомендуют различные фильмы друг другу. Movielens предоставляет возможность ставить оценки фильмам. $R_{ui} = 1$, если пользователь u поставил оценки фильму i. В данной работе использованы два различных набора данных MovieLens: в первом 100000 оценок, во втором 1000000.

Перед тем как использовать наборы данных в экспериментах, из них были удалены пользователи, которые взаимодействовали с менее 25 предметами. Это было сделано для преодоления проблемы холодного старта, которым страдают факторизационные методы.

Далее приведены статистические характеристики каждого набора данных после предобработки.

Таблица 1: Статистические характеристики

Набор данных	Epinion	Slashdot	MovieLens100k	MovieLens1m
Число ненулевых элементов	326114	573578	96963	989202
Число пользователей	4405	6992	806	5549
Число предметов	34777	63730	1682	3702
Плотность матрицы	0.21%	0.13%	7.15 %	4.81 %
Среднее число предметов у пользователя	51	50	81	106
Максимальное число пользователей у одного предмета.	1801	2508	737	2314

6.2 Сравнение методов

Сравним качество работы методов ранжирования друг с другом. Для этого проведем два различных эксперимента. Первый предложен авторами метода CLiMF. Второй является n-карманной кросс-валидацией.

Параметры методов были настроены при помощи кросс-валидации на наборе данных Epinion.

Таблица 2: Параметры методов

Название метода	Параметры
CLiMF	$K = 10$, $\lambda = 0.005$, $\gamma = 0.001$, maxiter = 15
BPR_MF	$K = 10$, $\lambda = 0.0025$, $\gamma = 0.05$, maxiter = 300
iMF	$K = 10$, $\lambda = 0.015$, $\alpha = 1$, maxiter = 15
TFMAP	$K = 10$, $\lambda = 0.001$, $\gamma = 0.01$, $n = 100$, maxiter = 15

[3] `http://grouplens.org/datasets/movielens/`

6.2.1 Первый эксперимент

Разобьем набор данных на тренировочную и тестовую выборку случайным образом. В тренировочной выборке у каждого пользователя будет trainK предметов. Остальные предметы лежат в тестовой выборке. Алгоритмы обучаются на тренировочной выборке, а качество работы измеряется на тестовой выборке. Далее такой эксперимент повторяется maxiter раз. Конечным результатом является средняя величина качества работы по метрикам и алгоритмам.

В таблицах 3, 4, 5 и 6 показано качество работы алгоритмов на различных данных и параметрах эксперимента.

Таблица 3: Набор данных Epinion

	trainK = 5, maxiter = 5					trainK = 10, maxiter = 5				
	PopRec	CLiMF	BRP_MF	iMF	TFMAP	PopRec	CLiMF	BRP_MF	iMF	TFMAP
P@5	**0.1987**	0.1962	0.1960	0.1876	0.1807	0.1837	0.1844	0.1846	**0.2613**	0.1808
1call@5	**0.5632**	0.5448	0.5450	0.5509	0.5448	0.5189	0.5293	0.5241	**0.6383**	0.5212
NDCG@5	**0.2222**	0.2205	0.2205	0.1980	0.2104	0.2061	0.2064	0.1909	**0.2725**	0.2041
MAP@5	0.3813	0.3769	0.3774	0.3356	**0.3873**	0.3568	0.3572	0.3120	**0.4121**	0.3597
MRR	**0.4387**	0.4368	0.4383	0.3818	0.4306	0.4113	0.4112	0.3480	**0.4609**	0.4114
AUC	0.8307	0.7512	**0.8347**	0.6915	0.6407	0.8558	0.8143	**0.8591**	0.8105	0.7310

Таблица 4: Набор данных Slashdot

	trainK = 5, maxiter = 5					trainK = 10, maxiter = 5				
	PopRec	CLiMF	BRP_MF	iMF	TFMAP	PopRec	CLiMF	BRP_MF	iMF	TFMAP
P@5	0.1225	**0.1227**	0.1210	0.1049	0.1153	0.1119	0.1118	0.1128	**0.1358**	0.1128
1call@5	**0.3765**	0.3774	0.3761	0.3419	0.3592	0.3528	0.3528	0.3531	**0.3928**	0.3544
NDCG@5	**0.1319**	0.1316	0.1309	0.1097	0.1244	0.1210	0.1210	0.1207	**0.1424**	0.1215
MAP@5	0.2295	0.2289	**0.2297**	0.1949	0.2194	0.2146	0.2147	0.2110	**0.2374**	0.2148
MRR	**0.2765**	0.2755	0.2761	0.2343	0.2567	0.2602	0.2598	0.2558	**0.2794**	0.259429
AUC	0.7770	0.6897	**0.7850**	0.5908	0.5973	0.8127	0.7582	**0.8182**	0.6925	0.6662

Таблица 5: Набор данных MobieLens 100k

	trainK = 5, maxiter = 5					trainK = 10, maxiter = 5				
	PopRec	CLiMF	BRP_MF	iMF	TFMAP	PopRec	CLiMF	BRP_MF	iMF	TFMAP
P@5	**0.5309**	0.5131	0.5303	0.4625	0.2624	0.4841	0.4821	0.4738	**0.5307**	0.4738
1call@5	**0.9431**	0.9334	0.9374	0.8970	0.7461	0.9146	0.9141	0.9074	**0.9292**	0.8937
NDCG@5	**0.5341**	0.5171	0.5319	0.4756	0.3055	0.4934	0.4958	0.4725	**0.5369**	0.4802
MAP@5	**0.6703**	0.6512	0.6636	0.6341	0.5452	0.6492	0.6568	0.6117	**0.6741**	0.6212
MRR	**0.7045**	0.6970	0.7019	0.6818	0.5931	0.6885	0.6984	0.6474	**0.7139**	0.6660
AUC	0.8274	0.7572	**0.8288**	0.6785	0.6366	0.8530	0.8405	**0.8531**	0.8285	0.7634

Таблица 6: Набор данных MobieLens 1m

	trainK = 5, maxiter = 5					trainK = 10, maxiter = 5				
	PopRec	CLiMF	BRP_MF	iMF	TFMAP	PopRec	CLiMF	BRP_MF	iMF	TFMAP
P@5	0.4805	**0.4835**	0.4797	0.4739	0.4138	0.4903	0.4903	0.4835	**0.5739**	0.4906
1call@5	0.8805	0.8704	**0.8826**	0.8707	0.8538	0.8610	0.8610	0.8479	**0.9302**	0.8587
NDCG@5	**0.4935**	0.4952	0.4923	0.4850	0.4515	0.5011	0.5011	0.4918	**0.5838**	0.5014
MAP@5	0.6489	**0.6534**	0.6484	0.6285	0.6626	0.6557	0.6557	0.6151	**0.7088**	0.6594
MRR	**0.6941**	0.6916	0.6930	0.6728	0.7019	0.6903	0.6903	0.6597	**0.7530**	0.6897
AUC	**0.8503**	0.8286	0.8488	0.7185	0.7476	**0.8561**	0.8514	0.8556	0.8279	0.8043

Видно, что качество работы методов сильно зависит от параметров проводимого эксперимента.

При trainK = 5 в среднем лучше работает PopRec. Во многом высокое качество работы PopRec связано с тем, что во всех представленных наборах данных существует большое количество популярных предметов, которые есть почти у каждого пользователя. Например, очередной блокбастер или популярный блог. С данной проблемой сталкивались авторы других статей [1, 7].

CLiMF, BRP_MF и TFMAP всегда имеют качество работы близкое к PopRec. Но в отличие от PopRec, факторизационные методы имеют разные прогнозы предпочтения для разных пользователей.

В результатах экспериментов также не было замечено какой-либо хорошей работы CLiMF и TFMAP на метриках MRR и MAP@n соответственно. Причем не один из данных методов не имел самое высокое качество по своему функционалу.

BRP_MF почти всегда имеет самый высокий AUC. Но по эксперименту при параметре trainK = 10 видно,что такой функционал абсолютно не подходит. iMF имеет самое высокое качество на втором эксперименте по всем метрикам, кроме AUC.

6.2.2 Второй эксперимент

Минусом первого эксперимента является то, что в тренировочной выборке присутствует лишь малая часть от всех данных. Следовательно, имела место ситуация, в которой некоторые методы показывали недостаточно хорошее качество из-за того, что они недообучились. Поэтому было решено проверить качество работы при помощи n-карманной кросс-валидации.

В таблицах 7, 8, 9 и 10 показано качество работы алгоритмов на различных данных и параметрах эксперимента.

Таблица 7: Набор данных MovieLens 100k

	n = 10					n = 5				
	PopRec	CLiMF	BRP_MF	iMF	TFMAP	PopRec	CLiMF	BRP_MF	iMF	TFMAP
P@5	0.1399	0.1429	0.2672	**0.2972**	0.1478	0.2401	0.2468	0.3945	**0.4476**	0.257
1call@5	0.4516	0.4565	0.6947	**0.7332**	0.4354	0.6625	0.6550	0.8461	**0.8883**	0.6153
NDCG@5	0.1549	0.1592	0.2837	**0.3220**	0.1652	0.2591	0.2611	0.4118	**0.4782**	0.2748
MAP@5	0.2853	0.2934	0.4511	**0.5043**	0.2882	0.4245	0.4179	0.5815	**0.6651**	0.4169
MRR	0.3413	0.3451	0.5025	**0.5568**	0.3384	0.4806	0.4711	0.6335	**0.7205**	0.4624
AUC	0.8597	0.8470	**0.9317**	0.9310	0.8503	0.8566	0.8417	**0.9290**	0.9283	0.8471

Таблица 8: Набор данных Epinion

	n = 5					n = 10				
	PopRec	CLiMF	BRP_MF	iMF	TFMAP	PopRec	CLiMF	BRP_MF	iMF	TFMAP
P@5	0.0301	0.0294	0.0557	**0.06940**	0.0273	0.0564	0.0545	0.0999	**0.1248**	0.0548
1call@5	0.1327	0.1300	0.2267	**0.2726**	0.1213	0.2301	0.2237	0.3532	|**0.4251**	0.2221
NDCG@5	0.0338	0.0332	0.0586	**0.0742**	0.0312	0.0636	0.0619	0.1052	**0.1316**	0.0628
MAP@5	0.0767	0.0757	0.1210	**0.1515**	0.0716	0.1375	0.1353	0.1973	**0.2438**	0.1363
MRR	0.0999	0.0990	0.1541	**0.1864**	0.0946	0.1703	0.1680	0.2385	**0.2853**	0.1687
AUC	0.8690	0.7685	**0.9192**	0.9175	0.8607	0.8700	0.7576	**0.9185**	0.9159	0.8615

Таблица 9: Набор данных Slashdot

	n = 10					n = 5				
	PopRec	CLiMF	BRP_MF	iMF	TFMAP	PopRec	CLiMF	BRP_MF	iMF	TFMAP
P@5	0.0154	0.0154	0.0247	**0.0393**	0.0139	0.0294	0.0295	0.0420	**0.0722**	0.0289
1call@5	0.0721	0.0725	0.1045	**0.1582**	0.0662	0.1307	0.1322	0.1608	**0.2626**	0.1295
NDCG@5	0.0179	0.0179	0.0265	**0.0430**	0.0144	0.0335	0.0334	0.0439	**0.0781**	0.0324
MAP@5	0.0428	0.0427	0.0566	**0.0901**	0.0325	0.0764	0.0764	0.0867	**0.1522**	0.0738
MRR	0.0597	0.0589	0.0775	**0.1151**	0.0490	0.1024	0.1013	0.1150	**0.1864**	0.0989
AUC	0.8457	0.7213	**0.8710**	0.8703	0.8376	0.8460	0.7073	0.8631	**0.8665**	0.8354

Таблица 10: Набор данных Movie Lens 1m

	n = 10					n = 5				
	PopRec	CLiMF	BRP_MF	iMF	TFMAP	PopRec	CLiMF	BRP_MF	iMF	TFMAP
P@5	0.1336	0.1157	0.2237	**0.2600**	0.0792	0.2265	0.212507	0.3345	**0.3972**	0.1405
1call@5	0.4112	0.3692	0.6141	**0.6520**	0.2870	0.5779	0.5595	0.7531	**0.8010**	0.4420
NDCG@5	0.1439	0.1198	0.2400	**0.2799**	0.0838	0.2387	0.2181	0.3501	**0.4175**	0.1504
MAP@5	0.2564	0.2091	0.3942	**0.4424**	0.1603	0.3780	0.3399	0.5085	**0.5767**	0.2689
MRR	0.3780	0.3399	0.5085	**0.5767**	0.2689	0.4189	0.3856	0.5588	**0.6255**	0.3142
AUC	0.8573	0.8489	**0.9244**	0.9171	0.8491	0.8579	0.8523	**0.9228**	0.9157	0.8502

CLiMF и TFMAP показывают такое же качество, что и PopRec.

BPR_MF стабильно показывает более высокое качество, чем PopRec, а iMF работает значительно лучше по всем метрикам, кроме AUC.

Далее в работе будет показано, что несмотря на значительную разницу в качестве работы методов, при помощи ансамблирования можно добиться более лучшего результата.

6.3 Составление линейного ансамбля

Пусть имеется множество базовых алгоритмов $b_i(x)$. Необходимо подобрать такие веса α_i, чтобы линейная комбинация алгоритмов $\hat{b}(x) = \sum_i \alpha_i b_i(x)$ показывала лучший результат по какому-нибудь заданному функционалу.

В нашем случае b_i - методы ранжирования описанные ранее. Каждый метод возвращает ранжированный список предметов, который неудобно использовать в задаче ансамблирования. Удобнее использовать величину f_{ui}, которую можно интерпретировать как меру предпочтения. Обозначим за f_{ui}^m значение f_{ui} m-ого базового метода.

Заметим, что для алгоритмов ранжирования важно не само значение f_{ui}, а их порядок друг относительно друг. Поэтому диапазон значений этой величины у каждого алгоритма разный и зависит от функционала качества и регуляризатора. Поэтому, чтобы линейный комбинации были корректны, необходимо нормировать значения f_{ui} для каждого пользователя u. В результате $0 \leq f_{ui} \leq 1$. Далее будем считать, что все f_{ui} нормированы.

Рассмотрим несколько способов создание ансамблей:

- Простое голосование.
- Взвешенное голосование при помощи линейной регрессии.
- Оптимальное взвешенное голосование

Для экспериментов над ансаблями наборы данных были разбиты на 3 выборки: тренировочную, тестовую и валидационную. На тренировочной обучаются факторизационые методы, на валидационной настраиваются веса, а на тестовой вычисляется качество работы ансамбля.

Простое голосование

Один из простейших методов, который представляет из себя линейную комбинацию базовых методов с равными весами.

$$\hat{b}(x) = \frac{1}{T} \sum_{m=1}^{T} b_m(x)$$

В случае ранжирования данная выражение будет выглядеть следующим образом.

$$\hat{f}_{ui} = \frac{1}{T} \sum_{m=1}^{T} f_{ui}^m$$

Взвешенное голосование при помощи линейной регрессии

Линейная регрессия была одним из самых простых способов ансамблирования, которые были реализованы победителями конкурса Netflix. Основная проблема линейной регрессии и всех остальных методов агрегирования из этого конкурса в том, что они оптимизируют совершенно другой функционал. А именно RMSE. Но с другой стороны iMF тоже хорошо справляется с задачей хоть и не оптимизирует на прямую метрики ранжирования.

Пусть у пользователя l предметов лежит в валидационной выборке. Они все релевантные. Далее случайным образом выберем 5l предметов, которые не лежат ни в валидационной, ни в тренировочной выборке, и объявим их нерелевантными. В итоге из этих предметов строим матрицу признаков X и целевой вектор y.

X				y
$f_{ui_1}^{m_1}$	$f_{ui_1}^{m_2}$	$f_{ui_1}^{m_3}$	$f_{ui_1}^{m_4}$	R_{ui_1}
$f_{ui_2}^{m_1}$	$f_{ui_2}^{m_2}$	$f_{ui_2}^{m_3}$	$f_{ui_2}^{m_4}$	R_{ui_2}
$f_{ui_3}^{m_1}$	$f_{ui_3}^{m_2}$	$f_{ui_3}^{m_3}$	$f_{ui_3}^{m_4}$	R_{ui_3}
$f_{ui_4}^{m_1}$	$f_{ui_4}^{m_2}$	$f_{ui_4}^{m_3}$	$f_{ui_4}^{m_4}$	R_{ui_2}
...	...	...	...	...

Таблица 11: Схематичное изображение признаковой матрицы

Далее обучаем линейную регрессию на матрице признаков X и на целевом векторе y. В экспериментах была использована готовая реализация линейной регрессии из библиотеки scikit-learn[4].

Оптимальное взвешенное голосование

Рассмотрим линейную комбинацию двух алгоритмов ранжирования в следующем виде:

$$\hat{f}_{ui} = \alpha f_{ui}^{m_1} + (1-\alpha) f_{ui}^{m_2} \text{ , где } 0 \leq \alpha \leq 1$$

Заметим, что все метрики ранжирования меняются только при изменении порядка предметов в ранжированном списке, а не от самого изменения f_{ui}. Поэтому число различных значений метрики не может быть больше конечного числа. На рис.1 видно, что изменения порядка происходит в точках пересечения линий. Их не более чем $O(|U||I|^2)$.

Идея метода состоит в вычислении качества ансамбля при различных α на валидационной выборке в поисках оптимального параметра [8, 9]. Теоретически

[4] `http://scikit-learn.org/stable/`

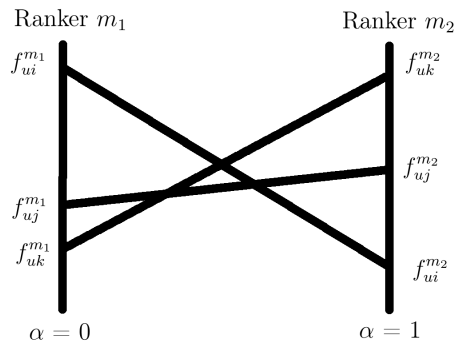

Рис. 1: Две вертикальные линии обозначают размер величины f. Чем выше точка, тем больше величина. Невертикальные линии обозначают линейные комбинации величин f. Если $\alpha = 0$, то линейная комбинация учитывает только Ranker m_1, если $\alpha = 1$, то линейная комбинация учитывает только Ranker m_2

можно перебрать все значения α, при которых значение метрики различны. Но количество различных точек α слишком велико, поэтому было решено брать только малое подмножество точек. Далее было замечено, если брать α равномерно от 0 до 1, то результат получается такой же. В экспериментах был использован последний вариант.

Обозначим за $b(x) = (b_i, b_j)$ ансамбль двух алгоритмов при помощи оптимального взвешенного голосования. Обобщим оптимальное взвешенное голосование для 4 методов двумя способами:

- при помощи бустинга. $(((b_1, b_2), b_3), b_4)$
- при помощи построения дерева. $((b_1, b_2), (b_3, b_4))$

6.4 Сравнение ансамблей

Введем несколько обозначений.

- BSM(best single method) - лучшее значение метрики достигаемое одиночным методом.
- SV(Simple Vote) - простое голосование.
- RV(Regression Vote) - взвешенное голосование при помощи линейной регрессии.
- OVB(Optimal Vote Boosting) - оптимальное взвешенное голосование при помощи бустинга.

- OVT(Optimal Vote Tree) - оптимальное взвешенное голосование при помощи построения дерева.

В качестве оптимизирующей метрики для OVB и OVT была использована мера качества NDCG.

Для сравнения качества методов проведем два эксперимента.

6.4.1 Первый эксперимент

В тренировочной выборке у каждого пользователя будет trainK предметов. В валидационной – validK. Остальные предметы лежат в тестовой выборке. Алгоритмы обучаются на тренировочной выборке, а качество работы измеряется на тестовой выборке. Далее такой эксперимент повторяется maxiter раз. Конечным результатом является средняя величина качества работы по метрикам и алгоритмам.

В таблицах 12, 13, 14 и 15 показано качество работы ансамблей на различных данных и параметрах эксперимента.

Таблица 12: Набор данных Epinion

	trainK = 5, validk=5, maxiter = 5					trainK = 10, validk = 5, maxiter = 5				
	BSM	SV	RV	OVB	OVT	BSM	SV	RV	OVB	OVT
P@5	0.1811	0.1986	0.2033	**0.2108**	0.2095	0.2364	0.2120	0.1935	**0.2438**	0.2387
1call@5	0.5334	0.5622	0.5707	**0.5852**	0.5847	0.6004	0.5651	0.5243	**0.6190**	0.6177
NDCG@5	0.2028	0.2190	**0.2230**	0.2305	0.2291	0.2456	0.2271	0.2060	**0.2564**	0.2529
MAP@5	0.3590	0.3734	0.3784	**0.3892**	0.3872	0.3775	0.3670	0.3339	0.3995	0.**4011**

Таблица 13: Набор данных Slashdot

	trainK = 5, validk=5, maxiter = 5					trainK = 10, validk = 5, maxiter = 5				
	BSM	SV	RV	OVB	OVT	BSM	SV	RV	OVB	OVT
P@5	0.1098	0.1137	**0.1155**	0.1121	0.1127	0.1272	0.1137	0.1111	**0.1333**	0.1322
1call@5	0.3507	0.3631	**0.3729**	0.3560	0.3646	0.3796	0.3565	0.3760	0.4017	**0.4071**
NDCG@5	0.1192	0.1214	**0.1225**	0.1209	0.1212	0.1340	0.1175	0.1250	**0.1407**	0.1397
MAP@5	0.2134	0.2143	**0.2177**	0.2147	0.2166	0.2277	0.2073	0.2207	0.2410	**0.2419**

Таблица 14: Набор данных MovieLens 100k

	trainK = 5, validk=5, maxiter = 5					trainK = 10, validk = 5, maxiter = 5				
	BSM	SV	RV	OVB	OVT	BSM	SV	RV	OVB	OVT
P@5	0.5142	0.5243	0.5124	**0.5321**	0.5296	0.4930	0.5298	**0.5622**	0.5513	0.5573
1call@5	0.9317	0.9272	0.9280	**0.9372**	0.9347	0.9143	0.9346	**0.9346**	0.9230	0.9230
NDCG@5	0.5202	0.5392	0.5118	**0.5455**	0.5416	0.5116	0.5424	0.5727	0.5647	**0.5754**
MAP@5	0.6616	0.6842	0.6435	**0.6918**	0.6868	0.6708	0.6907	0.7060	0.6979	**0.7133**

Таблица 15: Набор данных MovieLens 1m

	trainK = 5, validk=5, maxiter = 5					trainK = 10, validk = 5, maxiter = 5				
	BSM	SV	RV	OVB	OVT	BSM	SV	RV	OVB	OVT
P@5	0.4700	0.4957	0.4947	**0.5104**	0.5028	0.5484	0.5312	0.4926	0.5709	**0.5772**
1call@5	0.8587	0.8812	0.8835	0.8942	**0.8990**	0.9131	0.8835	0.8574	0.9201	**0.9225**
NDCG@5	0.4832	0.5099	0.5101	**0.5268**	0.5183	0.5603	0.5344	0.4990	0.5847	**0.5909**
MAP@5	0.6442	0.6606	0.6568	**0.6728**	0.6684	0.6922	0.6633	0.6410	0.7136	**0.7187**

6.4.2 Второй эксперимент

В тестовой выборке у каждого пользователя будет ptest% выборки. В валидационной pvalid% выборки. А в тренировочной все остальные предметы. Далее эксперимент повторяется maxiter раз. Конечным результатом является средняя величина качества работы по метрикам и алгоритмам.

В таблицах 20, 21, 22 и 23 показано качество работы ансамблей на различных данных и параметрах эксперимента.

Таблица 16: Набор данных MovieLens 100k

	ptest = 10%, pvalid=10%, maxiter = 5					ptest = 20%, valid = 10%, maxiter = 5				
	BSM	SV	RV	OVB	OVT	BSM	SV	RV	OVB	OVT
P@5	0.2965	0.1602	0.2528	**0.3069**	0.3019	0.4451	0.2531	0.3831	**0.4498**	0.4464
1call@5	0.7344	0.4950	0.6712	**0.7580**	0.7555	0.8706	0.6203	0.8039	**0.8784**	0.8734
NDCG@5	0.3201	0.1778	0.2783	**0.3307**	0.3257	0.4705	0.2738	0.4070	**0.4770**	0.4752
MAP@5	0.5035	0.3239	0.4599	**0.5185**	0.5152	0.6480	0.4261	0.5763	0.6555	**0.6562**

Таблица 17: Набор данных Epinion

	ptest = 10%, pvalid=10%, maxiter = 5					ptest = 20%, valid = 10%, maxiter = 5				
	BSM	SV	RV	OVB	OVT	BSM	SV	RV	OVB	OVT
P@5	0.0716	0.0295	0.0534	**0.0764**	0.0760	0.1283	0.0550	0.1000	0.1328	**0.1363**
1call@5	0.2710	0.1303	0.2183	**0.2921**	0.2908	0.4179	0.2242	0.3595	0.4372	**0.4438**
NDCG@5	0.0775	0.0334	0.0572	**0.0830**	0.0822	0.1384	0.0615	0.1062	0.1425	**0.1468**
MAP@5	0.1561	0.0759	0.1193	**0.1688**	0.1672	0.2536	0.1316	0.2035	0.2607	**0.2686**

Таблица 18: Набор данных Slashdot

	ptest = 10%, pvalid=10%, maxiter = 5					ptest = 20%, valid = 10%, maxiter = 5				
	BSM	SV	RV	OVB	OVT	BSM	SV	RV	OVB	OVT
P@5	0.0431	0.0186	0.0291	**0.0438**	0.0436	0.0746	0.0362	0.0519	0.0757	**0.0758**
1call@5	0.1644	0.0866	0.1272	**0.1683**	0.1671	0.2570	0.1577	0.2038	0.2631	**0.2653**
NDCG@5	0.0473	0.0201	0.0312	**0.0481**	**0.0481**	0.0803	0.0397	0.0556	0.0819	**0.0822**
MAP@5	0.0959	0.0455	0.0684	0.0976	**0.0978**	0.1506	0.0877	0.1140	0.1550	**0.1562**

Таблица 19: Набор данных Movie Lens 1m

	ptest = 10%, pvalid=10%, maxiter = 5					ptest = 20%, valid = 10%, maxiter = 5				
	BSM	SV	RV	OVB	OVT	BSM	SV	RV	OVB	OVT
P@5	0.2601	0.1065	0.2374	0.2628	**0.2639**	0.3936	0.1974	0.3616	**0.3983**	0.3944
1call@5	0.6541	0.3424	0.6186	0.6566	**0.6593**	0.7985	0.5255	0.7714	**0.8075**	0.8066
NDCG@5	0.2795	0.1104	0.2476	0.2830	**0.2836**	0.4127	0.2007	0.3710	**0.4192**	0.4157
MAP@5	0.4384	0.1936	0.3859	0.4440	**0.4438**	0.5729	0.3127	0.5166	**0.5839**	0.5822

По результатам эксперимента видно, что SV работает хорошо, если качество методов примерно равны. Если один из методов работает намного лучше другого, то он хуже BSM. Аналогичная ситуация с RV.

OVB и OVT стабильно показывают хорошее качество во всех поставленных ситуациях.

6.4.3 Сравнение разных метрик для оптимальных комбинаций

Одним из параметров OVB и OVT является оптимизирующий функционал. Исследуем какой функционал лучше выбирать в качестве оптимизирующего.

В таблицах 20, 21, 22 и 23 показано качество работы ансамблей на различных данных и параметрах второго эксперимента. В качестве ансамбля был взят OVT.

Таблица 20: Набор данных Movie Lens 100k

	test = 0.1, valid=0.1, maxiter = 5					test = 0.2, valid = 0.1, maxiter = 5				
	BSM	P@5	1call@5	NDCG@5	MAP@5	BSM	P@5	1call@5	NDCG@5	MAP@5
P@5	0.2866	0.3024	0.3007	0.2987	**0.3037**	0.4466	0.4522	0.4476	0.4466	**0.4526**
1call@5	0.7406	0.7717	0.7729	0.7630	**0.7667**	0.8672	0.8707	0.8660	0.8672	**0.8709**
NDCG@5	0.3163	0.3292	0.3266	0.3281	**0.3303**	0.4721	0.4790	0.4748	0.4721	**0.4794**
MAP@5	0.5156	0.5242	0.5269	**0.5271**	0.5223	0.6448	0.6530	**0.6483**	0.6448	0.6530

Таблица 21: Набор данных Epinion

	test = 0.1, valid=0.1, maxiter = 5					test = 0.2, valid = 0.1, maxiter = 5				
	BSM	P@5	1call@5	NDCG@5	MAP@5	BSM	P@5	1call@5	NDCG@5	MAP@5
P@5	0.0766	0.0784	0.0753	**0.0779**	0.0786	0.1264	0.1307	0.1310	**0.1321**	0.1317
1call@5	0.2854	0.2935	0.2871	**0.2942**	0.2935	0.4152	0.4242	0.4241	0.4330	**0.4338**
NDCG@5	0.0836	0.0850	0.0825	0.0853	**0.0855**	0.1351	0.1386	0.1392	0.1400	**0.1409**
MAP@5	0.1665	0.1704	0.1682	**0.1725**	0.1710	0.2469	0.2501	0.2500	0.2555	**0.2559**

Таблица 22: Набор данных Slashdot

	test = 0.1, valid=0.1, maxiter = 5					test = 0.2, valid = 0.1, maxiter = 5				
	BSM	P@5	1call@5	NDCG@5	MAP@5	BSM	P@5	1call@5	NDCG@5	MAP@5
P@5	0.0408	**0.0421**	**0.0421**	**0.0421**	**0.0421**	0.0732	**0.0757**	**0.0757**	**0.0757**	**0.0757**
1call@5	0.1573	**0.1637**	**0.1637**	**0.1637**	**0.1637**	0.2484	**0.2587**	**0.2587**	**0.2587**	**0.2587**
NDCG@5	0.0450	**0.0465**	**0.0465**	**0.0465**	**0.0465**	0.0790	**0.0820**	**0.0820**	**0.0820**	**0.0820**
MAP@5	0.0921	**0.0955**	**0.0955**	**0.0955**	**0.0955**	0.1476	**0.1550**	**0.1550**	**0.1550**	**0.1550**

Таблица 23: Набор данных Movie Lens 1m

	test = 0.1, valid=0.1, maxiter = 5					test = 0.2, valid = 0.1, maxiter = 5				
	BSM	P@5	1call@5	NDCG@5	MAP@5	BSM	P@5	1call@5	NDCG@5	MAP@5
P@5	0.2570	**0.2614**	0.2604	0.2610	0.2610	0.3981	**0.4020**	**0.4020**	0.4012	**0.4020**
1call@5	0.6482	0.6581	**0.6588**	0.6570	0.6570	0.7985	**0.8122**	**0.8122**	0.8062	**0.8122**
NDCG@5	0.2762	**0.2804**	0.2794	0.2802	0.2802	0.4178	0.4220	0.4220	**0.4221**	0.4220
MAP@5	0.4362	0.4424	0.4422	**0.4426**	**0.4426**	0.5760	0.5819	0.5819	**0.5834**	0.5819

Никакой значительной зависимости качества ансабля от выбора оптимизирующего функционала замечено не было. Более того, на наборе данных Slashdot линейные коэффициенты настроились одинаково на всех метриках.

Из-этого можно сделать вывод, что выбор функционала не играет существенной роли.

7 Заключение

В ходе работы были получены следующие результаты.

1. Составлен обзор основных факторизационных методов для задачи ранжирования для набора данных с двоичной релевантностью.
2. Предложен метод ансамблирования, который стабильной улучшает качество работы ранжирования.
3. Реализованы факторизационные методы и эксперименты на языке python и c++.

Список литературы

[1] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas.

CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering.

RecSys '12 the sixth ACM conference on Recommender systems, pages 139-146, 2012.

[2] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner.

BPR: Bayesian Personalized Ranking from Implicit Feedback.

UAI '09 Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pages 452-461, 2009.

[3] Yue Shia,Alexandros Karatzogloub, Linas Baltrunas.

TFMAP: Optimizing MAP for Top-N Context-aware Recommendation.

35th international ACM SIGIR conference on Research and development in Information Retrieval, pages 155-164, 2012.

[4] Yifan Hu, Yehuda Koren, Chris Volinsky.

Collaborative Filtering for Implicit Feedback Datasets.

8th IEEE International Conference on Data Mining, pages 263-272, 2008.

[5] Y. Koren, R. Bell, C. Volinsky.

Matrix Factorization Techniques for Recommender Systems.

Computer IEEE, Vol.42, pages 30-37, 2009.

[6] Yehuda Koren.

The BellKor Solution to the Netflix Grand Prize.

2009.

[7] Paolo Cremonesi, Yehuda Koren, Roberto Turrin.

Performance of Recommender Algorithms on Top-N Recommendation Tasks.

RecSys '10 Proceedings of the fourth ACM conference on Recommender systems, pages 39-46, 2010.

[8] Christopher J. C. Burges, Krysta M. Svore, Paul N. Bennett.

Learning to Rank Using an Ensemble of Lambda-Gradient Models.

JMLR: Workshop and Conference Proceedings 14, pages 25-35, 2011.

[9] Qiang Wu, Christopher J. C. Burges, Krysta M. Svore.
Adapting Boosting for Information Retrieval Measures.
Information Retrieval, Vol. 13, pages 254 - 270, 2010.