# Рекомендательные системы

Боровко Никита Алексеевич

8 апреля 2022 г.

## Содержание

# 1 Введение

Рекомендательные системы - это область в машинном обучении, цель которой - определить набор товаров, которые будут интересны определенному пользователю, используя информацию о большей группе пользователей. Они повсеместно используются в популярных музыкальных сервисах, интернет-магазинах, социальных медиа-платформах.

Существует два разных подхода к решению такой задачи: фильтрация на основе содержания и коллаборативная фильтрация. В первом случае используется вся информация о товаре, например, если это товар в магазине, то можно включить в модель его технические характеристики и параметры. Кроме этого мы можем использовать дополнительную информацию о пользователе: его возраст, пол. В случае коллаборативной фильтрации в расчет берется только история оценок группы пользователей рассматриваемых товаров, и на основе только этого ищется зависимость.[9] Помимо этого существуют гибридные подходы, объединяющие в себе оба метода, но они остаются вне рамок рассмотрения данной работы.

Мы сосредоточимся именно на коллаборативной фильтрации, рассматривая различные методы решения задачи, в первую очередь - факторизационные машины.

# 2 Постановка задачи

Пусть у нас имеются два множества: $U$ - множество пользователей, $I$ - множество товаров, помимо этого известен результат взаимодействия некоторых пар вида $r_{ui}$ - оценка, поставленная пользователем $u$ товару $i$. Это означает, что наша учебная выборка будет иметь вид:

$$K = \{(u, i) | \exists r_{ui}, u \in U, i \in I\}$$

Тогда смысл задачи в нахождении аппроксимирующей функции:

$$f(x, y) = \hat{r}_{ui} \approx r_{ui}$$

При этом, как правило, необходимо минимизировать функционал вида:

$$\sum_{(u,i \in K)} (r_{ui} - \hat{r}_{ui})^2 + \lambda \sum_{\theta \in \Theta} ||\theta||^2$$

Основной алгоритм в коллаборативной фильтрации - факторизационные машины и их разновидности. В этом случае аппроксимирующая матрица представляется в виде:

$$\hat{R} = PQ,$$

где $\hat{R} \subset \mathbb{R}^{U \times I}$, $P \subset \mathbb{R}^{U \times \text{latent factors}}$, $Q \subset \mathbb{R}^{\text{latent factors} \times I}$.

# 3 Инструменты

В качестве набора данных для тренировки моделей мы будем использовать датасет MovieLens - оценки 600 пользователями 9000 фильмов, всего более 100000 оценок[2]. Оценки здесь имеют разброс от 0.5 до 5.0 с шагом 0.5, эти данные непрерывно собираются и обновляются.

Основной метрикой качества у нас будет RMSE - Root-mean-square deviation, рассчитываемый как среднее квадратичное отклонение, деленное на количество элементов.

# 4 Обзор существующих методов

## 4.1 Факторизационная машина

Факторизационная машина - это одна из основных идей, используемых в рассматриваемой задаче. Другие подходы, например, SVD, зачастую являются лишь частным случаем этого. Мы предполагаем, что имеет место полиномиальная регрессия 2-ого порядка, т.е. целевая функция зависит не только от рассматриваемых признаков, но еще и от результата их попарного взаимодействия[7]:

$$\hat{r}_{ui} = \omega_0 + \sum_{i=1}^{n} \omega_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} v_i^T v_j x_i x_j$$

где параметры модели установлены следующим образом:

$$\omega_0 \in \mathbb{R}, \omega \in \mathbb{R}, V \in \mathbb{R}^{n \times k}$$

Утварждается, что модель можно обучить достаточно быстро при помощи стохастического градиентного бустинга в силу того, что число параметров и время на настройку и обучение линейны. Важнейшим достоинством факторизационных машин является их способность предсказывания даже при очень высокой разреженности исходной матрицы. В частности, это можно обобщить и на ненаблюдаемые взаимодействия[7].

Для работы с этим методом мы воспользуемся готовой реализацией из библиотеки pyFM. Рассмотрим 4 модели, учитывающие 5, 10, 15 и 20 латентных факторов соответственно, в каждой из них поставим 50 эпох для обучения.

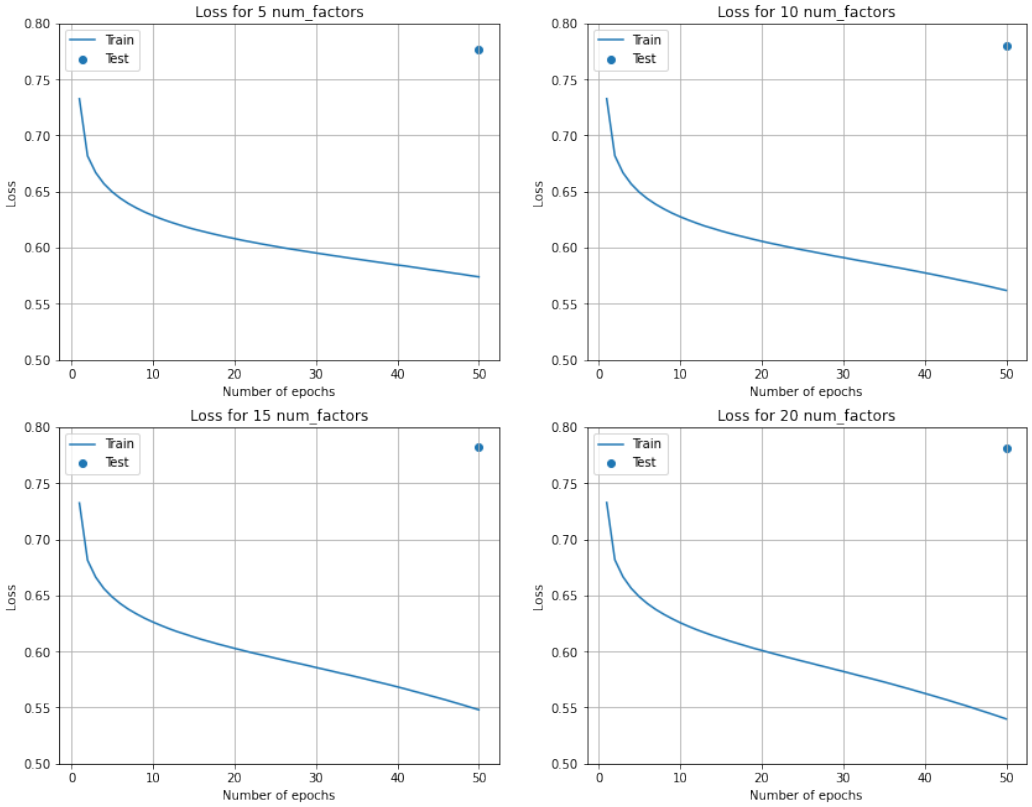


Рис. 1: Зависимость loss от число эпох при разном числе латентных факторов

Анализируя графики, можно заметить, что факторизационные машины дают очень хороший результат в сравнении с другими подходами,

но слишком сильного улучшения за счет увеличения числа латентных факторов не происходит.

## 4.2 SVD и SVD++

Данный метод основан на известном сингулярном разложении:

$$R = U\Sigma V^T$$

Далее мы можем сделать малоранговую аппроксимацию, оставив только k самых больших сингулярных значений:

$$R \approx \hat{R} = \hat{U}\hat{\Sigma}\hat{V}^T$$

Поскольку нам надо аппроксимировать исходную матрицу двумя другими, сделаем такой шаг:

$$R \approx \hat{R} = (\hat{U}\hat{\Sigma})(\hat{V})^T = P^T Q,$$

где

$$P = (\hat{U}\hat{\Sigma})^T, Q = \hat{V}^T$$

Следовательно:

$$R \approx P^T Q$$

Данный метод, как и другие, можно улучшить, рассмотрев еще и смещение, которое позволяет лучше реагировать на те случае, когда к товару относятся предвзято. В поэлементной записи это улучшение имеет следующий вид:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i,$$

где

$\mu$ - глобальное смещение, $b_u$ - смещение
для пользователя u, $b_i$ - смещение для товара i

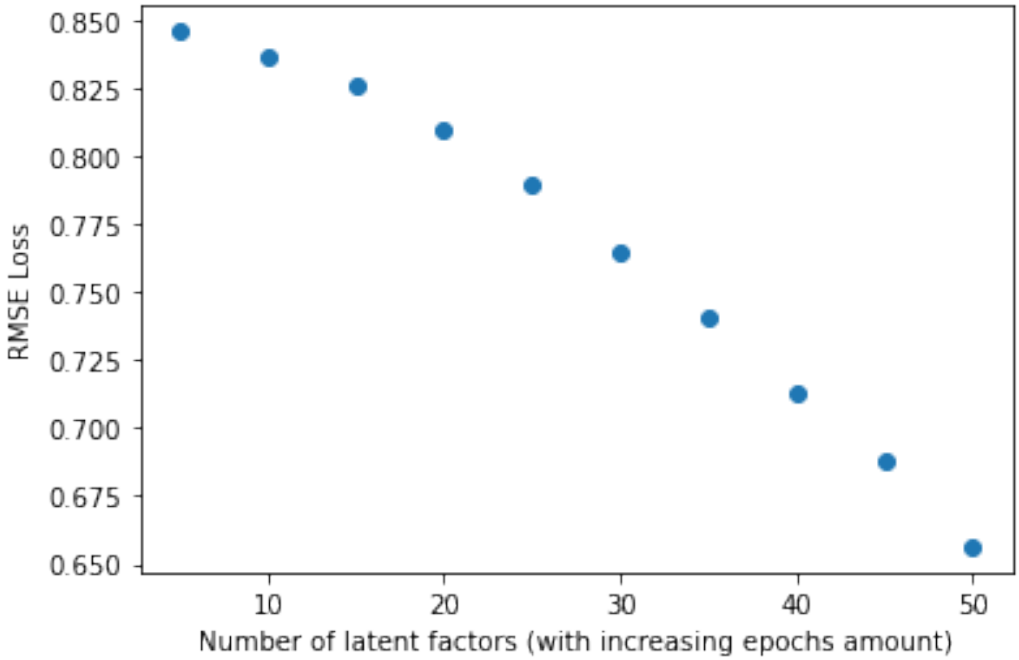


Рис. 2: Зависимость loss от числа латентных параметров

Следует отметить, что существует улучшенная версия данной модели - SVD++. В ней рассматривается еще и неявный отклик пользователей, это означает, что будут учитываться взаимодействия пользователя и товара такие, что мы не знаем их оценку, но знаем про сам факт их наличия[5]. Добавляя это в формулу для SVD мы получим:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T(p_u + |N(u)|^{-\frac{1}{2}} + \sum_{j \in Nu} y_j),$$

где

$N(u)$ - множество товаров, с которыми пользователь u провзаимодействовал неявно.

Метод SVD++ на практике несколько выигрывает у SVD, но разница RMSE остается весьма небольшой, при 50, 100 и 200 латентных факторах разница в RMSE у этих алгоритмов состовляет порядка 1%[5].

## 4.3 Slope One

Семейство алгоритмов Slope One[6] представляет собой иной подход к задаче, здесь вводится функция отклонения средней оценки товара j от i:

$$\text{dev}_{j,i} = \sum_{u \in S_{j,i}(R)} \frac{u_j - u_i}{|S_{j,i}(R)|},$$

где

$S_{j,i}(R)$ - множество пользователей, оценивших товары i и j.

Тогда предполагаемая оценка пользователем u товара i высчитывается с использованием знания о каждом другом оцененном товаре и отклонения его оценки от оценки рассматриваемого товара:

$$\hat{r}_{ui} = \frac{1}{|R_i(u)|} \sum_{j \in R_i(u)} (\text{dev}_{i,j} + u_i)$$

Исследователи выяснили, что если данные достаточно плотные, т.е. почти для каждой пары товаров существуют пользователи, оценившие их вместе, то допустима следующая аппроксимация:

$$\hat{r}_{ui} \approx \mu_u + \frac{1}{|R_i(u)|} \sum_{j \in R_i(u)} \text{dev}_{i,j},$$

где

$\mu_u$ - средняя оценка пользователя u.

Таким образом, в итоговой модели на результат влияет только средняя оценка пользователя, а не его оценка других предметов, что несколько облегчает расчет[6].

Этот подход отличается своей простотой и вполне приемлимым качеством, также в какой-то степени решена проблема холодного старта, когда для новых пользователей трудно делать рекомендации из-за небольшого числа их оценок. Однако Slope One может проигрывать более сложным алгоритмам, так как улавливает зависимости слишком примитивно.

## 4.4 Alternating Least Squares

Этот метод, пожалуй, является самым интересным из преставленных, теперь мы будем рассчитывать значения предпочетния (preference) и уверенности в нем (confidence):

$$p_{ui} = \begin{cases} 1, r_{ui} > 0 \\ 0, r_{ui} = 0 \end{cases}$$

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

Таким образом, изначально $p_{ui}$ рассчитывается как бинарное представление оценок, а $c_{ui}$ показывает, насколько мы уверены в достоверности наблюдения для данной пары $(u, i)$. Экспериментально[4] было утановлено, что значение $\alpha = 40$ хорошо справляется с задачей.

Переобозначим искомые матрицы $P$ и $Q$ за $X$ и $Y$, чтобы избежать путаницы. Тогда смысл задачи будет в нахождении векторов $x_u$ для каждого пользователя и $y_i$ для каждого товара, которые будут минимизировать функцию:

$$min \sum_{x^*, y^*} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u ||x_u||^2 + \sum_i ||y_i||^2)$$

Здесь второе слагаемое - регуляризатор, используемой для избежания переобучения на данном наборе данных. Затем путем дифференцирования находятся подходящие минимизирующие выражения для пользователей и предметов:

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i p(i)$$

Эти выражения можно оптимизировать, вынося $X^T X$ и $Y^T Y$, эти слагаемые не зависят от $u$ и $i$, поэтому их можно найти предрасчетом, сэкономив вычислительное время:

$$x_u = (Y^T Y + Y^T (C^u - I) Y + \lambda I)^{-1} Y^T C^u p(u)$$

$$y_i = (X^T X + X^T (C^i - I) X + \lambda I)^{-1} X^T C^i p(i)$$

Тогда для вычисления $RMSE$ на каждом шаге итерации потребуется вычисление полученной матрицы, которое с учетом вычисляемого сдвига для большей точности будет выглядеть так:

$$\hat{R} = XY^T + X_{bias} + Y_{bias}$$

Метод ALS был написан нами вручную, и были произведены настройки моделей для различного числа латентных параметров, результаты с валидацией показаны на Рис. 3.

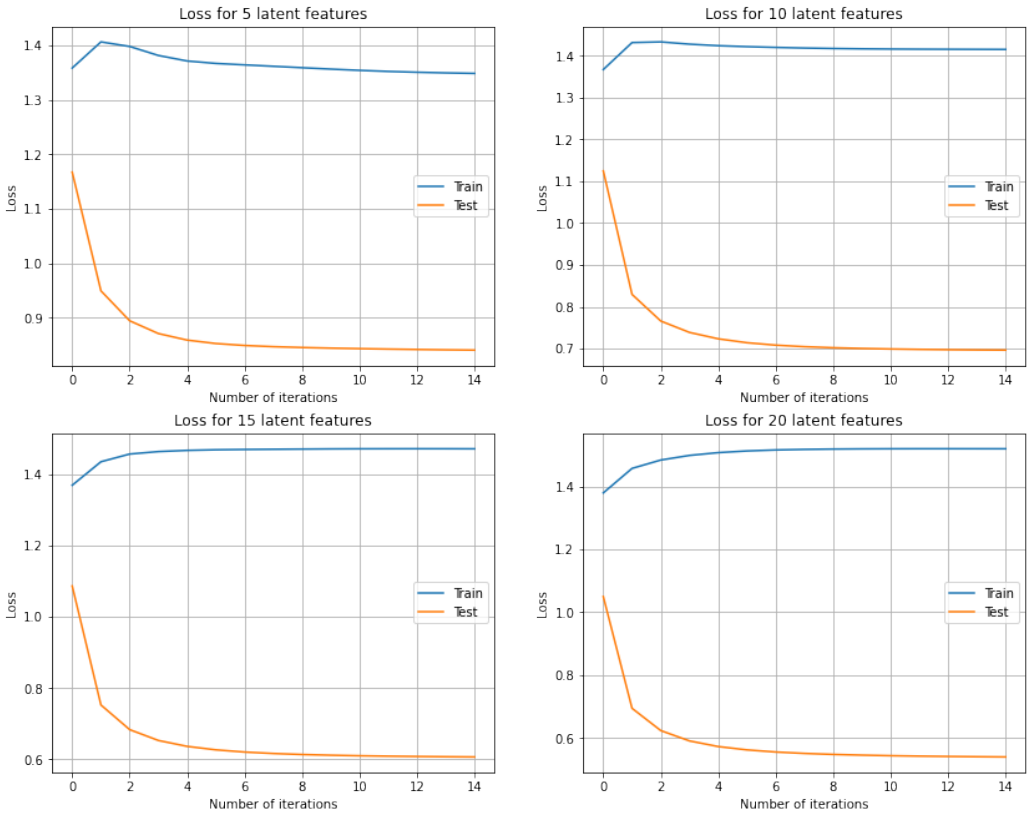


Рис. 3: Зависимость loss на Train и Test от числа итераций при разном числе латентных факторов

Главный недостаток этого метода связан с масштабируемостью - здесь мы дополнительно учитываем неявные (implicit) наблюдения, а значит,

надо хранить и обрабатывать гораздо больше данных, несмотря на разреженность, что будет вызывать серьезные проблемы в слишком больших системах[4].

# 5 Возникающие проблемы

При работе с рекомендательными системами возникает несколько серьезных проблем. Во-первых, данные с оценками товаров пользователями очень разреженны. Это делает задачу предсказывания отношения пользователя к товару сложнее, однако на классические факторизационные машины это не оказывает сильного влияния из-за их способности улвливать полиномиальную регрессию высших порядков[7].

Похожая проблема - проблема холодного старта, когда появляется новый товар или пользователь, не успевший сделать достаточно числа оценок, в этом случае хорошо себя показывает алгоритм SlopeOne, который рассматривает не оценку пользователем каждого товара, а оценку в среднем[6].

Сложные алгоритмы зачастую подвержены проблеме масштабируемости, в первую очередь, это метод ALS, которому необходимо хранить и обрабатывать информацию и о неявных откликах[4], но на несложные методы типа SlopeOne эта проблема не оказывает сильного влияния.

# 6 Проблемы нейросетевого подхода

Существует много подходов к решению рассматриваемой задачи при помощи методов глубокого обучения, однако обычно эта технология применяется к анализу дополнительной информации о контенте, а когда дело доходит до обработки матрицы оценок, по-прежнему доминирующим инструментом остаются факторизационные машины и их разновидности. В 2017 году появился новый подход к коллаборативной фильтрации на основе нейронных сетей (NCF), использущий многослойный перцептрон[3].

Однако существует[8] обратное мнение, что в случае правильно подобранных гиперпараметров данный подход значительно уступает более простым алгоритмам без применения глубокого обучения. Также говорится, что хотя многослойный перцептрон в теории может аппроксимировать любую функцию, на практике это слишком сложно.

Помимо этого был произведен анализ[1] 26 имеющихся подходов с применением глубоко обучения, и только 12 из них прошли проверку на воспроизводимость. Анализ также выявил различные методологические проблемы, исследователи пришли к выводу, что прогресс в этой области замедлился из-за неправильного подбора *baseline* и недостаточной работы над оптимизацией. Из-за этого часто складывается ситуация, когда даже самый простой и примитивный алгоритм либо не сильно хуже, либо даже лучше сложных нейронных сетей. Текущую ситуацию даже назвали *стагнацией*.

# 7 Заключение

В работе были рассмотрены некоторые популярные алгоритмы коллаборативной фильтрации - SlopeOne, SVD, SVD++, ALS, FMs. У каждого из этих алгоритмов есть свои достоинства и недостатки, и зачастую использование более примитивного алгоритма целесообразнее применения сложной модели. Несмотря на существование подходов с применением глубокого обучения, многие из них не являются достаточно полезными и оптимальными в условиях данной задачи, поэтому при решении реальной задачи вопрос выбора соотвутствующего метода является крайне важной задачей.

# Список литературы


[1] DACREMA, M. F., BOGLIO, S., CREMONESI, P., Politecnico, and JANNACH, D. A troubling analysis of reproducibility and progress in recommender systems research.

[2] Harper, F. M., and Konstan, J. A. The movielens datasets: History and context.

[3] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering.

[4] Hu, Y., Koren, Y., and Volinsky, C. Collaborative filtering for implicit feedback datasets.

[5] Koren, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model.

[6] Lemire, D., and Maclachlan, A. Slope one predictors for online rating-based collaborative filtering.

[7] Rendle, S. Factorization machines.

[8] Rendle, S., Krichene, W., Zhang, L., and Anderson, J. Neural collaborative filtering vs matrix factorization revisited.

[9] Ricci, F., Rokach, L., and Shapira, B. Introduction to recommender systems handbook.