3. Матричные разложения

Введение

В этой лекции мы поговорим о матричных разложении.
И эволюции их применения к рекомендательным системам.

В прошлой серии

Заключение

- Коллаборативная фильтрация – наиболее используемый вид рекомендательной системы
- Рассмотрели Memory-based модели
- kNN-алгоритм (недостаток: нужно хранить всю матрицу R)
- Метрики близости

Резюме по Memory-based моделям

Преимущества для бизнес-приложений:

- Легко понять
- Легко реализовать

Недостатки:

- Не хватает теоретического обоснования:
  - придумано много способов оценить сходство. . .
  - придумано много гибридных (item-user-based) методов. . .
  - . . . и не ясно, что лучше
- Все методы требуют хранения огромной матрицы R
- Проблема «холодного старта»

Далее:

Латентные модели — лишены этих недостатков (но об этом в следующий раз ☺)

План

1. Сведения из линейной алгебры
2. SVD-разложения
3. NMF
4. FM
5. Заключения

Сведения из линейной алгебры

Матричные разложения

Существует много видов матричных разложений, например LU- и RQ- разложения.
Общая идея матричного разложения состоит в том, чтобы представить матрицу A в виде линейной комбинации (чаще всего произведения) других матриц с целью оптимизации каких либо вычислений.

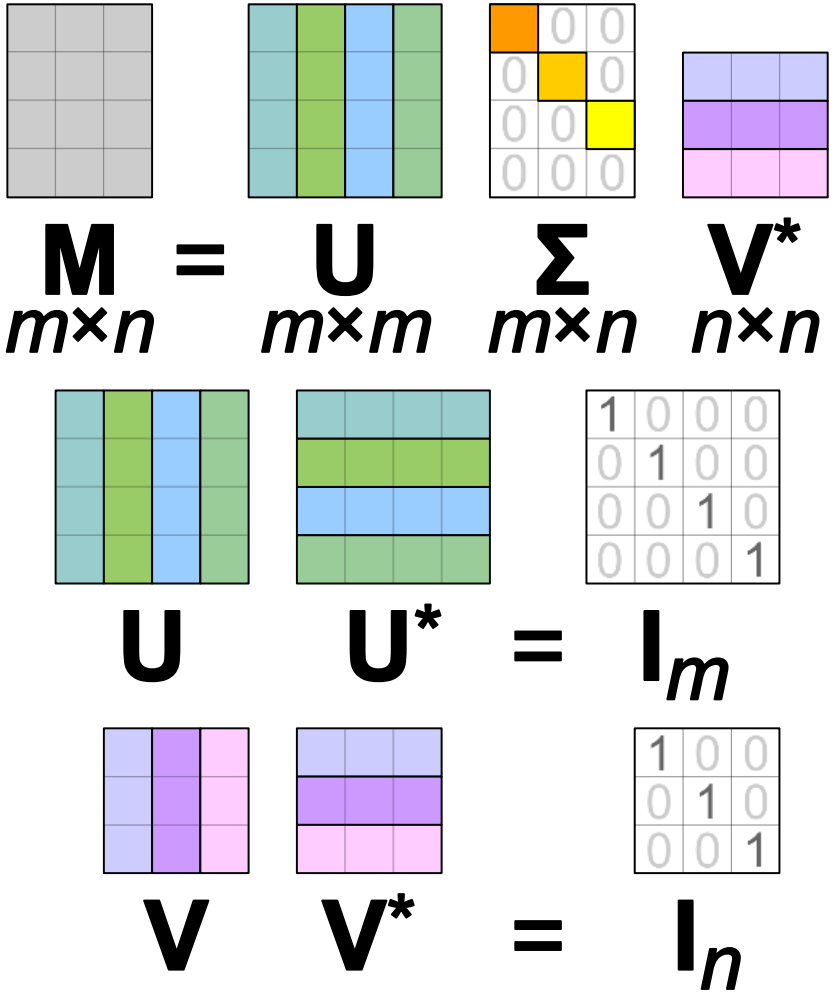
https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B

Сингулярное разложение

Singular value decomposition (SVD) – один из специальных видов матричных разложений.

$$\mathrm{M} = U\Sigma V^*$$

где Σ — матрица с неотрицательными элементами, у которой элементы, лежащие на главной диагонали — это сингулярные числа (а все элементы, не лежащие на главной диагонали, являются нулевыми), а матрицы U и V — это две унитарные матрицы, состоящие из левых и правых сингулярных векторов соответственно (а V^*— это сопряжённо-транспонированная матрица к V).



Нормы векторов

$$\|x\|_1 = \sum_{i=1}^{m} |x_i|$$

$$\|x\|_2 = \sqrt{\sum_{i=1}^{m} (x_i)^2}$$

Нормы матриц

$$\|X\| = \sum_{i=1}^{m}\sum_{j=1}^{n}|x_i|$$

$$\|X\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(x_{ij})^2}$$

$$\|X\|_1 = \max_{1\le j\le n}\sum_{i=1}^{m}|x_i|$$

$\|.\|_F$ — норма Фробениуса.

MAE и RMSE

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|\widehat{x_i} - x_i|$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\widehat{x_i} - x_i)^2}$$

Метрики качества

MAE и RMSE удобны для оценки качества прогноза. Они дают нам представление о том, насколько точны наши прогнозы и, в свою очередь, насколько точны наши рекомендации.

Матричная факторизация

Netflix Prize

Netflix Prize - это конкурс, в котором требовалось спрогнозировать оценку пользователями фильмотеки Netflix.
Это была задача с явными рейтингами, оценки ставились по шкале от 1 до 5.

Были доступны следующие данные:

- Обучающие данные (training data set) содержат 100.480.507 оценок, которые 480.189 клиентов поставили 17.770 фильмам
- Названия и годы выхода в прокат всех 17.770 фильмов.

Нужно было предсказать какие оценки поставит пользователь тому или иному фильму.

Netflix Prize

Этот конкурс породил бум рекомендательных систем!

Определение победителя:
На скрытой части оказалось, что точность у этих команд совпадает до четвертого знака после запятой, поэтому победителя определила разница коммитов в 20 минут.

Победивший ансамбль использовал модели следующих классов:

- Регрессионная модель, основанная на средних оценках
- collaborative filtering — коллаборативная фильтрация
- Random Forests

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43

Уроки Netflix Prize

Netflix Prize - это конкурс, в котором требовалось спрогнозировать оценку пользователями фильмотеки Netflix. Обучающие данные содержат 100 млн. оценок (от 1 до 5), которые 0,5 млн. клиентов поставили к 17 000 фильмам.

Точность прогноза оценивалась по RMSE.

Метрика:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\widehat{x_i} - x_i)^2}$$

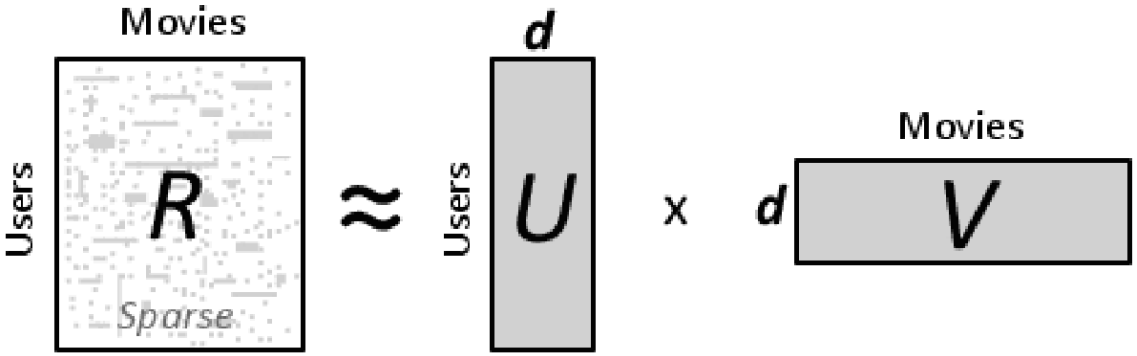
x_i - истинное значение

$\widehat{x_i}$ - оценка

SVD-разложения

Funk MF

Исходный алгоритм, предложенный Саймоном Функом, факторизовал матрицу рейтинга user-item как произведение двух матриц меньшей размерности, в первой из которых есть строка для каждого пользователя, а во второй – столбец для каждого элемента. Строка или столбец, связанные с конкретным пользователем или элементом, называются скрытыми факторами.



$$\hat{r}_{ui} = \langle \boldsymbol{p}_u, \boldsymbol{q}_i \rangle$$

https://sifter.org/~simon/journal/20061211.html

Funk MF

Прогнозируемые рейтинги можно рассчитать как:

$\tilde{R} = HW$, где $\tilde{R} \in \mathbb{R}^{users \times items}$ — матрица рейтинга user-item
$H \in \mathbb{R}^{users \times latent\ factors}$ — скрытые факторы пользователя
$W \in \mathbb{R}^{users \times latent\ factors}$ — скрытые факторы предмета

В частности, прогнозируемая оценка, которую пользователь u поставит элементу i, вычисляется как:

$$\tilde{r}_{ui} = \sum_{f=0}^{n\ factors} H_{u,f} W_{f,i}$$

Funk MF

Настроить качество модели можно, изменив количество скрытых факторов. Увеличение количества скрытых факторов улучшит персонализацию и, следовательно, качество рекомендаций, пока количество факторов не станет слишком большим, после чего модель начнет переобучаться, и качество рекомендаций снизится. Распространенной стратегией, позволяющей избежать переобучения, является добавление членов регуляризации к целевой функции.

Funk MF был разработан как задача прогнозирования рейтингов, поэтому он использует явные числовые рейтинги в качестве взаимодействий между пользователем и элементом.

Учитывая все обстоятельства, Funk MF минимизирует следующую целевую функцию:

$$arg\min_{H,W}(\left\|R - \tilde{R}\right\|_F + \alpha\|H\| + \beta\|W\|)$$

Funk MF с точки зрения сингулярного разложения

Из курса линейной алгебры мы умеем делать «классическое» SVD разложение.

$\mathrm{R} = U\Sigma V^*$ (это точное разложение). Мы хотим получить приближенное.

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

Можно получить таком образом:

$$P = \left(\hat{U}\hat{\Sigma}\right)^*$$
$$Q = \hat{V}^*$$

Тогда:

$$\hat{R} = P^*Q$$

$$\hat{\mathrm{r}}_{\mathrm{ij}} = \mathrm{q}_{\mathrm{i}}^{\mathrm{T}}\mathrm{p}_{\mathrm{u}}$$

SVD-разложение

Для улучшения FUNK MF удобно использовать смещения (bais) по Items и User.
А также μ как средний как средний известный рейтинг. Тогда разложение можно записать следующим образом:

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

https://www.hindawi.com/journals/mpe/2017/1975719/

Funk MF как задача оптимизации

Давайте попробуем сформулировать разложение $\hat{R} = P^* Q$ как задачу оптимизации:

$$\sum_{i,j} (\widehat{r_{i,j}} - r_{i,j})^2 = (\widehat{\mathrm{q}_\mathrm{i}^\mathrm{T} \mathrm{p}_\mathrm{u}} - r_{i,j})^2 \to \min$$

Подходы к построению разложения

Есть 2 основных подхода:

- Стохастический градиентный спуск (SGD)
- Altering Least Squares (ALS)

SQD

Метод стохастического градиента:

Случайным образом выбираем пару (u,i) и делаем каждый раз градиентный шаг для задачи $\min_{p_u, q_i} \sum_{i,j} (\widehat{q_i^T p_u} - r_{i,j})^2 = \varepsilon_{ui}$

Обновляем градиент стандартно:

$$p_u := p_u + \eta \varepsilon_{ui} q_u$$
$$q_i := q_i + \eta \varepsilon_{ui} p_i$$

Метод попеременных наименьших квадратов

Идея Altering Least Squares (ALS):
Если зафиксировать либо P, либо Q, то задача становится выпуклой (проверьте это!).
Оптимальное значение P(или Q) при фиксированном Q(или P) можно выписать аналитически!

$p_u := \left(\sum_i q_i q_i^T\right)^{-1} \sum_i r_{ui} q_i$ (по тем i где существует r_{ui})

$q_i := \left(\sum_i p_i p_i^T\right)^{-1} \sum_i r_{ui} p_i$ (по тем i где существует r_{ui})

p_u q_i - это столбцы матрицы P и Q

Особенности SVD

SVD хорошо работает с рейтингами.
Но если матрица задана бинарными значения (0,1), как например в случае, когда матрица заполняется покупками, SVD, как правило, показывает плохие результаты.

С этой целью был разработан **NNMF**.

Особенности SVD

Хотя Funk MF может обеспечить очень хорошее качество рекомендаций, его способность использовать только явные числовые рейтинги в качестве взаимодействий между пользователями и элементами представляет собой ограничение. Современные рекомендательные системы должны использовать все доступные взаимодействия, как явные (например, числовые рейтинги), так и неявные (например, лайки, покупки, пропущенные, добавленные в закладки).

С этой целью был разработан SVD++ так, чтобы также учитывать неявные взаимодействия.

Модификации SVD

SVD++

Можно добавлять и дополнительную информацию в эту модель. Например, введем дополнительный набор факторов для y_j (который будут характеризовать пользователя на основе того, что он просматривал, но не оценивал).
Модель после этого принимает вид

$$\hat{r}_{ui} = \mu + b_i + b_u + q_u^T \left(p_u + \frac{1}{\sqrt{|V(u)|}} \sum_{j \in V(u)} y_j \right)$$

где $V(u)$ - множество продуктов, которые просматривал этот пользователь ($\frac{1}{\sqrt{|V(u)|}}$ - контролирует «дисперсию»)

Очень долго работает на практике!

Групповой SVD

Групповой SVD может быть эффективным решением проблемы холодного старта. Он группирует пользователей .
Затем, когда появляется новый пользователь или элемент, мы можем присвоить ему метку группы и аппроксимировать его латентный фактор групповыми эффектами (соответствующей группы). Хотя рейтинги, связанные с новым пользователем или элементом, не обязательно могут быть доступны, групповые эффекты обеспечивают немедленные и эффективные прогнозы.

$$\tilde{r}_{ui} = \sum_{f=0}^{n\,factors} (H_{u,f} + S_{v_u f,i})\,(W_{f,i} + T_{f,j_i})$$

Здесь представлены метки группы пользователя u и элемента i, соответственно, которые идентичны для всех членов одной группы.
T и S – матрицы групповых эффектов

timeSVD++

Обобщение с учетом времени:

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

где

$$b_i(t) = b_i + b_{i,Bin(t)}$$

$$b_u(t) = b_u + \alpha_u dev_u(t) + b_{u,t}$$

$$p_{u,f}(t) = p_{u,f} + \alpha_{u,f} dev_u(t) + p_{u,f} + \frac{1}{\sqrt{|V(u)|}} \sum_{j \in V(u)} y_j$$

$$dev_u(t) = sign(t - t_u)|t - t_i|^{\beta}$$

Регрессия по признакам

Также можно использовать регрессионный подход для SVD модели.

Строить модель регрессии по пользователя x_u и продукта x_i мы получаем модель

$$\hat{r}_{ui} \sim \mu + b_{user}(x_u) + b_{item}(x_i) + q_i^T p_u(t)$$

где

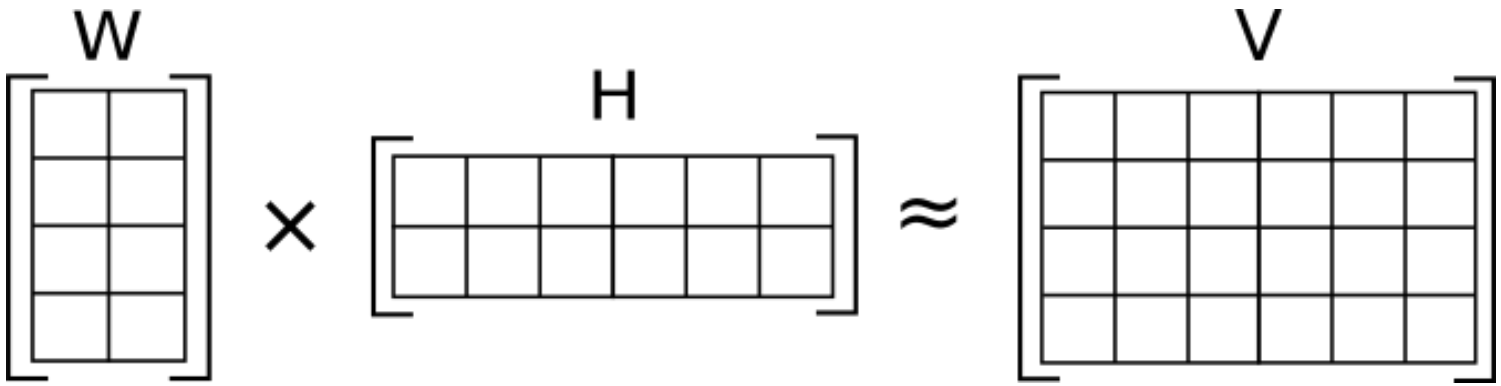
$$b_{user}(x_u) \sim \mathcal{N}(g(x_u), \sigma_g^2)$$
$$b_{item}(x_i) \sim \mathcal{N}(v(x_i), \sigma_v^2)$$

и в качестве g и v может выступать любая регрессия

NMF или NNMF

NMF (неотрицательное разложение)

Неотрицательное матричное разложение (Non-negative matrix factorization) – это группа алгоритмов, в которых матрица V разлагается на (обычно) две матрицы W и H, со свойством, что все три матрицы имеют неотрицательные элементы. Эта неотрицательность делает получившиеся матрицы более простыми для исследования.



NMF (неотрицательное разложение)

Идея разложения сохраняется, однако в NMF применяются положительные матрицы.

Недостатком NMF является, что в общем случае построение такого разложение являет NP-трудной задачей.

NMF (неотрицательное разложение)

Пусть матрица V - произведение матриц W и H:

$$\mathrm{V} \approx \mathrm{WH}$$

Чтобы найти приблизительную факторизацию $\mathrm{V} \approx \mathrm{WH}$, нам сначала нужно определить функции затрат, которые количественно определяют качество приближения. Такую функцию затрат можно построить, используя некоторую меру расстояния между двумя неотрицательными матрицами A и B. Одна полезная мера - это просто квадрат евклидова расстояния между A и B:

$$\|A - B\|^2\mathrm{V} = \sum_{ui} (A_{ui} - B_{ui})^2$$

NMF (неотрицательное разложение)

Теперь мы рассмотрим две альтернативные формулировки NMF как задачи оптимизации:

Проблема 1:
Минимизировать $||V - WH||^2$ по отношению к W и H с учетом ограничений $W, H \geq 0$ (положительны по элементам)

Проблема 2:
Минимизировать $\mathrm{D}(V\ ||WH)$ по отношению к W и H с учетом ограничений $W, H \geq 0$ (положительны по элементам)

https://proceedings.neurips.cc/paper/2000/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf

NMF (неотрицательное разложение)

Теорема 1:
Евклидово расстояние $\|V - WH\|$ не увеличивается по правилам обновления.

$$H_{\alpha,\mu} \leftarrow H_{\alpha,\mu} \frac{(W^T V)_{\alpha,\mu}}{(W^T WH)_{\alpha,\mu}}; \qquad W_{i\alpha} \leftarrow W_{i\alpha} \frac{(VH^T)_{i\alpha}}{(WHH^T)_{i\alpha}}$$

Евклидово расстояние инвариантно при этих обновлениях тогда и только тогда, когда W и H находятся в стационарной точке расстояния.

NMF (неотрицательное разложение)

Теорема 2:
Дивергенция $\mathrm{D}(V||WH)$ не увеличивается по правилам обновления.

$$H_{\alpha,\mu} \leftarrow H_{\alpha,\mu} \frac{\sum_i W_{i\alpha} V_{i\alpha} / (WH)_{i,\mu}}{\sum_k W_{k\alpha}}; \qquad W_{i\alpha} \leftarrow W_{i\alpha} \frac{\sum_\mu H_{\alpha\mu} V_{i\mu} / (WH)_{i,\mu}}{\sum_v H_{\alpha v}}$$

Дивергенция инвариантна относительно этих обновлений тогда и только тогда, когда W и H находятся в стационарной точке дивергенции.

Мультипликативные обновления

Методы с мультипликативными обновлениями очень популярны, потому что они:

- просты в реализации
- хорошо масштабируются и легко приспосабливаются к работе с разреженными матрицами
- были предложены в самой первой работе по NMF

Известно, что скорость их сходимости невелика, однако ее можно существенно увеличить, если обновлять A и X по несколько раз подряд

Метод попеременных неотрицательных наименьших квадратов

Altering Nonnegative Least Squares (ANLS): на каждом шаге точно находится покомпонентный минимум в неотрицательной области

$$Q \leftarrow \underset{a \geq 0}{\operatorname{argmin}} \|R - PQ\|_F$$

Покомпонентные минимумы можно находить с помощью методов активных ограничений, проекции градиента, квазиньютоновских методов, **оптимального градиентного метода Нестерова**.

Factorization Machines (FM)

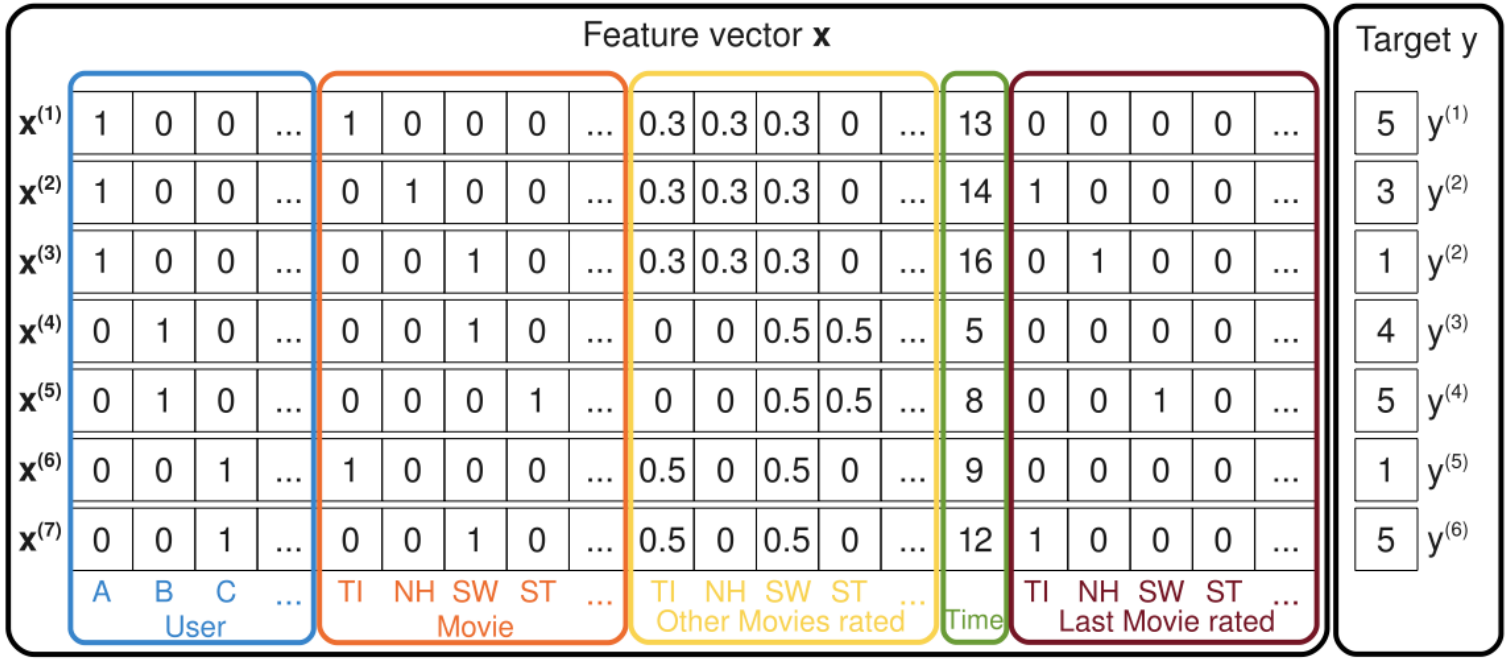
Factorization Machines (FM)

В качестве обобщения разных подходов была предложена факторизационная машина.
Авторы в своей статье показывают, что SVD, SVD++ — это лишь частные случаи FM.
Модель пытается все возможные взаимодействия, например, 2-ых (и выше) порядков.

https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf

Factorization Machines (FM)

Идея FM на картинке:



	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...	Target y
$x^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5 $y^{(1)}$
$x^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3 $y^{(2)}$
$x^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1 $y^{(2)}$
$x^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4 $y^{(3)}$
$x^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5 $y^{(4)}$
$x^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1 $y^{(5)}$
$x^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5 $y^{(6)}$

Первые 4 столбца (синие) - индикаторы для активного пользователя. Следующие 5 (красных) переменных индикатора для **активного** элемента.

Следующие 5 столбцов (желтые) содержат дополнительные неявные индикаторы (т.е. другие фильмы, которые оценил пользователь).(Зеленый) представляет время до оценки. Последние 5 столбцов (коричневые) содержат индикаторы для последнего фильма, который пользователь оценил до **активного**.

Factorization Machines (FM)

Обычная задача регрессии:

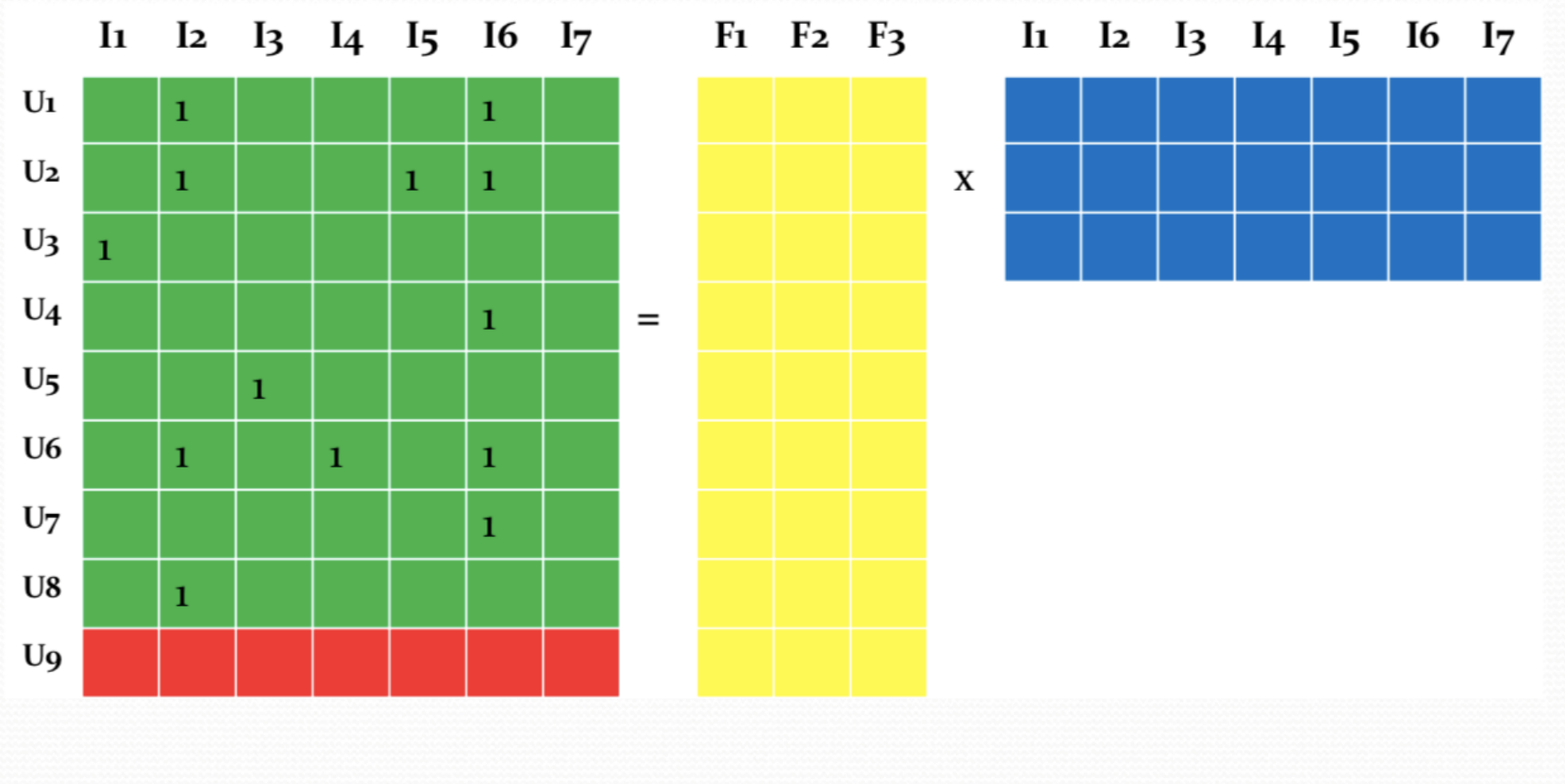
$$y(x) = w_0 + \sum_{i=1} w_i x_i + \sum_{i=1} \sum_{j=i+1} (v_i, v_j) x_i x_j$$

Factorization Machines (FM)

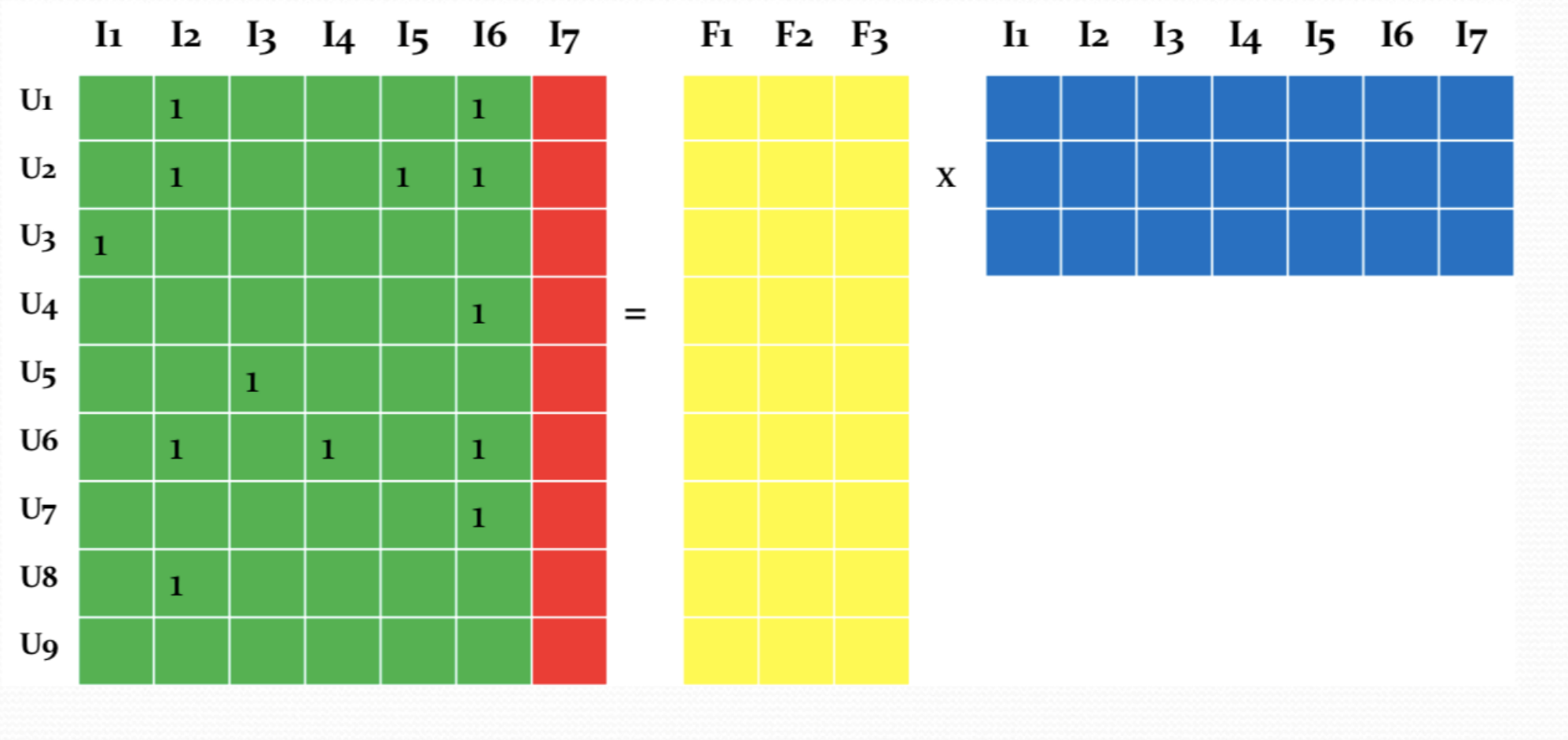
Реализацию FM можно найти в библиотеке libFM.
Не смотря на то что в модели есть некоторая математическая неустойчивость на практике все работает хорошо.

И снова о холодном старте

Проблема холодного старта для нового пользователя



Проблема холодного старта для новой публикации



Вопросы

Заключение

- Матричные разложения позволяют хранить пользователей и объекты в латентном виде
- **FM** обобщает «классические» SVD и SVD-модификации.
- По-прежнему есть проблема холодного старта, которую в явном виде не решают **FM и** SVD.

Семинар матричные разложения