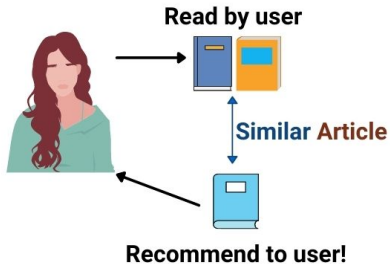


Рекомендательные системы

Виктор Китов

v.v.kitov@yandex.ru



Содержание

- 1 Задача построения рекомендаций
- 2 Доступная информация
- 3 Алгоритмы построения рекомендаций
- 4 Оценка и бизнес-особенности рекомендаций

Постановка задачи

- Задача: рекомендовать пользователю приобрести новые товары/услуги по его интересам.
- Примеры рекомендаций:

сервис	предмет рекомендаций
YouTube, Netflix	видео
last.fm, pandora	музыка
amazon, ozon	товары
Яндекс.Дзен	новости
вконтакте	группы, друзья, посты
TripAdvisor	достопримечательности

Постановка задачи

- Информационный поиск (information retrieval)
 - пользователь знает, что ищет.
 - задача: уточнение поиска
- Рекомендательные системы (recommender systems)
 - пользователь не знает, что ищет.
 - задача: расширение кругозора

Цели рекомендаций

Цели рекомендаций:

Цели рекомендаций

Цели рекомендаций:

- ↑ user experience, ↑ лояльность сервису
 - продать то, что нужно; рекомендовать интересное
- ↑ прибыль
 - продать подороже и доп. опции
- распродать остатки на складе
 - рекомендуем их в первую очередь
- изучить пользователя/товар
 - даём более случайные рекомендации
 - для продвинутых систем - всегда присутствует exploration-exploitation tradeoff.

Виды рекомендаций

Виды рекомендаций:

- Привязанные / не привязанные к пользователю
 - др. название: персональные / не персональные
- Привязанные / не привязанные к текущему товару
 - могут учитывать и более сложный контекст (время, локация, история запросов)

Виды рекомендаций

Виды рекомендаций:

- Привязанные / не привязанные к пользователю
 - др. название: персональные / не персональные
- Привязанные / не привязанные к текущему товару
 - могут учитывать и более сложный контекст (время, локация, история запросов)

В привязке к товару рекомендации бывают:

- более продвинутой версии товара (up-selling)
- более простой версии товара (down-selling)
- дополняющих товаров др. категорий (cross-selling)
- товаров, часто покупаемых вместе

Содержание

- 1 Задача построения рекомендаций
- 2 Доступная информация
- 3 Алгоритмы построения рекомендаций
- 4 Оценка и бизнес-особенности рекомендаций

Доступная информация

Доступная информация:

- о пользователе: анкетные данные, время регистрации, браузер, ОС, взаимодействие с системой
- о товаре: описание, цена, категория, характеристики, отзывы пользователей
- о контексте: время, погода, запрос пользователя, локация запроса

Доступная информация

Информация о взаимодействии пользователей и товаров:

- бинарная:
- целочисленная:
- вещественная:

Доступная информация

Информация о взаимодействии пользователей и товаров:

- бинарная:
 - искал, смотрел, добавил в корзину, купил, написал отзыв
- целочисленная:
 - поставил рейтинг, количество покупок
- вещественная:
 - объем потраченных денег, время просмотра, количество скачанных данных

Примеры матрицы рейтингов

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
u_1	1			5		2
u_2		5			4	
u_3	5	3		1		
u_4			3			4
u_5				3	5	
u_6	5		4			

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
u_1	1			1		1
u_2		1			1	
u_3	1	1		1		
u_4			1			1
u_5				1	1	
u_6	1		1			

- Могут быть и вещественные значения: время просмотра видео, объем скачанной информации.
- По сути задача рекомендации \sim заполнение пропусков в матрице рейтингов (missing value estimation)
 - но матрица очень разреженная

Виды информации о взаимодействии

Виды информации о взаимодействии:

- явные оценки товаров (explicit)
 - оценка, отзыв
- неявные оценки товаров (implicit)
 - бинарные:
 - посмотрел, положил в корзину, купил
 - ранговые:
 - в списке товаров кликнул на предпочтительный

Другая доступная информация

Также могут быть доступны

- взаимодействия пользователей
 - обмениваются комментариями
 - похожие интересы, паттерны поведения
- взаимодействия товаров:
 - субституты
 - более простые/сложные версии
 - дополняющие друг друга товары
 - частота совместной покупки

Содержание

- 1 Задача построения рекомендаций
- 2 Доступная информация
- 3 Алгоритмы построения рекомендаций
 - Контентные рекомендации
 - Коллаборативная фильтрация
 - Усредняющий алгоритм
 - User-based рекомендации
 - Item-based рекомендация
 - Использование матричных разложений
 - Учёт неявных откликов
- 4 Оценка и бизнес-особенности рекомендаций

Виды рекомендательных систем

Виды рекомендательных систем

- неперсональные:
 - summary-based: основаны на общей популярности товаров
 - product-based: отталкиваются от текущего товара
 - по смысловой совместимости с товаром
 - по частоте совместной покупки с товаром
- персональные:
 - content-based: основаны на сочетаемости данных о пользователе и товаре
 - collaborative filtering: используют только матрицу рейтингов
 - context-aware: дополнительно учитывают контекст (запрос, время, локацию)
 - knowledge-based: учитывают характеристики товаров
 - hybrid: ансамбль из разных типов рек. систем
 - самый точный подход

Алгоритм общей популярности

- Алгоритм общей популярности (summary-based).
 - неперсональные рекомендации, основанные на средней оценке товара.
- Если оценок мало - доверие к ним меньше, поэтому варианты сортировки:

$$\frac{1}{N} \sum_{i=1}^N r_i \rightarrow \frac{1}{N + \alpha} \sum_{i=1}^N r_i, \quad \alpha \sim 100$$

$$\bar{r} - \beta \sqrt{\text{Var}[\bar{r}]}, \quad \beta > 0$$

Product-based: правилый алгоритм

- Правилый алгоритм рекомендует товар, часто покупаемый с заданными
 - неперсональный
 - использует поиск ассоциативных правил вида: (association rules mining)
 - $\{A, B, C\} \Rightarrow D$ (были в одной сессии/чеке)
 - $A \rightarrow B \rightarrow C \Rightarrow D$ (были последовательно перед)
 - $A \rightarrow \dots \rightarrow B \rightarrow \dots \rightarrow C \rightarrow \dots \Rightarrow D$ (встретились до)
 - отбор правил:
 - следствие возникает с большой условной вероятностью
 - правило часто встречалось (достоверность)

3 Алгоритмы построения рекомендаций

- Контентные рекомендации
- Коллаборативная фильтрация
- Усредняющий алгоритм
- User-based рекомендации
- Item-based рекомендация
- Использование матричных разложений
- Учёт неявных откликов

Контентные рекомендации

Контентные рекомендации (content-based) - используют

- признаки о пользователе $f(u)$
 - как минимум: о просмотренных/купленных товарах
- признаки о товаре $f(i)$
- признаки о прошлом взаимодействии $f(u, i)$
 - например факт просмотра товара
 - hybrid: подставляем оценки рейтинга/эмбединги из коллаборативной фильтрации

Прогноз рейтинга - по обучаемой модели:

- (классификация, логистическая регрессия, ordered probit)

$$\hat{r}_{ui} = G_{\theta}(f(u), f(i), f(u, i))$$

Простейший пример

- Простейший пример контентных рекомендаций:
 - $f(i)$: TF-IDF описания товара
 - $f(u)$: TF-IDF конкатенации описаний ранее купленных товаров

$$\hat{r}_{ui} = \text{cos-sim}(f(u), f(i))$$

Напоминание TF-IDF:

$$w_{x,y} = \text{tf}_{x,y} \times \log\left(\frac{N}{\text{df}_x}\right)$$

TF-IDF

Вес слова x в описании товара y

$\text{tf}_{x,y}$ = частота слова x в описании товара y

df_x = количество товаров, содержащих слово x

N = общее количество товаров

- Здесь вообще не нужна матрица рейтингов.
Если есть - лучше настроить $\hat{r}_{ui} = G_{\theta}(f(u), f(i), f(u, i))$

Анализ контентных рекомендаций

Анализ контентных рекомендаций:

- \oplus : не нужно статистики взаимодействия, сразу строим прогноз по описанию
- \ominus : нужен хороший контент (музыка, видео - сложно)
- \ominus : однообразные рекомендации (описание товара статично)
- \ominus : хуже коллаборативной фильтрации при наличии статистики

3 Алгоритмы построения рекомендаций

- Контентные рекомендации
- **Коллаборативная фильтрация**
- Усредняющий алгоритм
- User-based рекомендации
- Item-based рекомендация
- Использование матричных разложений
- Учёт неявных откликов

Обозначения

- $R = \{r_{u,i}\}_{i \in I, u \in U}$ - матрица рейтингов
 - $r_{u,i}$ - рейтинг товара i пользователем u
 - u - отдельный пользователь
 - i - отдельный товар
- $(u, i) \in R$ означает, что известен r_{ui} .
- $\hat{r}_{u,i}$ - предсказанный рейтинг
- U - множество пользователей (users)
- I - множество товаров (items)
- I_u - множество товаров, оцененных пользователем u
- U_i - множество пользователей, оценивших товар i .

Соревнование Netflix - предшественник kaggle

- Netflix - сервис по онлайн-аренде DVD и доступа к цифровым каналам.
- Октябрь 2006 - сентябрь 2009: выложил данные для рекомендаций фильмов клиентам.
 - коллаборативная фильтрация с
 - 480.189 пользователями
 - 17.770 фильмами
 - оценками: 1,2,3,4,5.
 - призовой фонд: 1.000.000 \$
- Формат данных (коллаборативная фильтрация+время):
< пользователь, фильм, датаоценки, оценка >

Соревнование Netflix - предшественник kaggle

- Netflix - сервис по онлайн-аренде DVD и доступа к цифровым каналам.
- Октябрь 2006 - сентябрь 2009: выложил данные для рекомендаций фильмов клиентам.
 - коллаборативная фильтрация с
 - 480.189 пользователями
 - 17.770 фильмами
 - оценками: 1,2,3,4,5.
 - призовой фонд: 1.000.000 \$
- Формат данных (коллаборативная фильтрация+время):
< пользователь, фильм, датаоценки, оценка >
- Привлечены аналитики со всего мира.
- Лучший алгоритм - ансамбль большого количества хороших решений (не был внедрен).

Использование R

- Пользователи различаются по средним оценкам и разбросу (пессимисты/оптимисты):

$$u_1 = (1, 3, 1, 3)$$

$$u_2 = (6, 9, 6, 9)$$

- Также товары различаются по средним оценкам и разбросу (модные/не модные)
- Можно нормализовать R перед обработкой, потом денормализовать
 - по μ_u (самое важное)
 - по μ_u, σ_u
 - по $\mu_u, \sigma_u, \mu_i, \sigma_i$

Пример нормализации

- Пример нормализации по u :
 - для каждого u :
 - $\mu_u = \text{mean}(r_{u:})$, $\sigma_u = \text{std}(r_{u:})$; $r_{u:} := \frac{r_{u:} - \mu_u}{\sigma_u}$
 - считаем похожести u, i ; строим прогнозы \hat{r}_{ui}
 - для каждого u :
 - $\hat{r}_{u:} := \sigma_u \hat{r}_{u:} + \mu_u$

3 Алгоритмы построения рекомендаций

- Контентные рекомендации
- Коллаборативная фильтрация
- **Усредняющий алгоритм**
- User-based рекомендации
- Item-based рекомендация
- Использование матричных разложений
- Учёт неявных откликов

Простейшие базовые алгоритмы

Простейшие базовые алгоритмы:

- $\hat{r}_{u,i} = \mu$ ($\mu = \frac{1}{K} \sum_{u,i} r_{u,i}$, $K = |\{(u, i) : \exists r_{u,i}\}|$)
- $\hat{r}_{u,i} = \bar{r}_u = \frac{1}{|I_u|} \sum_{i \in I_u} r_{u,i}$
- $\hat{r}_{u,i} = \bar{r}_i = \frac{1}{|U_i|} \sum_{u \in U_i} r_{u,i}$

Усредняющий алгоритм

- Прогноз базового алгоритма:

$$b_{u,i} := \hat{r}_{u,i} = \mu + \Delta_u + \Delta_i$$

$$\Delta_u = \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u,i} - \mu)$$

$$\Delta_i = \frac{1}{|U_i|} \sum_{u' \in U_i} (r_{u',i} - \mu - \Delta_{u'})$$

- Интуиция:
 - Δ_u насколько u оценивает товары выше среднего
 - Δ_i насколько i в моде.

Усредняющий алгоритм с регуляризацией

- Усредняющий алгоритм с регуляризацией (with damping):

$$\hat{r}_{u,i} = \mu + \Delta_u + \Delta_i$$

$$\Delta_u = \frac{1}{|I_u| + \alpha} \sum_{i \in I_u} (r_{u,i} - \mu)$$

$$\Delta_i = \frac{1}{|U_i| + \beta} \sum_{u' \in U_i} (r_{u',i} - \mu - \Delta_{u'})$$

- $\alpha > 0, \beta > 0$ - сила регуляризации, $\alpha = \beta \approx 25$.

Усредняющий алгоритм с регуляризацией

- Усредняющий алгоритм с регуляризацией (with damping):

$$\hat{r}_{u,i} = \mu + \Delta_u + \Delta_i$$

$$\Delta_u = \frac{1}{|I_u| + \alpha} \sum_{i \in I_u} (r_{u,i} - \mu)$$

$$\Delta_i = \frac{1}{|U_i| + \beta} \sum_{u' \in U_i} (r_{u',i} - \mu - \Delta_{u'})$$

- $\alpha > 0, \beta > 0$ - сила регуляризации, $\alpha = \beta \approx 25$.
- Интуиция: доверяем Δ только когда выборка велика.

$$\Delta = \frac{1}{N + \alpha} \sum_{n=1}^N z_n = \begin{cases} \approx 0 & \text{для малых } N \\ \approx \frac{1}{N} \sum_{n=1}^N z_n & \text{для больших } N \end{cases}$$

Применения усредняющего алгоритма

Применения усредняющего алгоритма:

- R слишком разреженная для более сложных моделей
- заполнение пропусков для dense SVD
- $\hat{r}_{u,i}$ - доп. признак для content-based модели
- сложная модель предсказывает $r_{u,i} - \hat{r}_{u,i}$ вместо $r_{u,i}$
 - концентрируется на сложных случаях

Неявные отклики: алгоритм YouTube

- В YouTube

$$r_{u,i} = \begin{cases} 1, & u \text{ посмотрел } i \\ 0, & u \text{ не посмотрел } i \end{cases}$$

- Рассматриваются просмотры в последние 24 часа.
- Первый рекомендательный алгоритм системы.
- Считаем похожесть видео i и j :

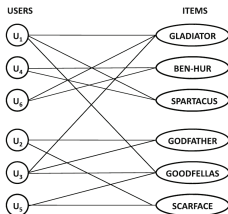
$$\text{sim}(i, j) = \frac{\sum_u r_{u,i} \cdot r_{u',j}}{(\sum_u r_{u,i}) \cdot (\sum_{u'} r_{u',j})}$$

- Пусть S_u - множество просмотренных u видео, $R(S_u)$ - похожие на них.
- Рекомендации - по принципу "друг моего друга - мой друг":

$$R(S) \cup R(R(S)) \cup \dots$$

Рекомендация на основе графа

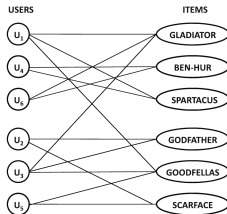
- Строим двудольный граф пользователи-товары.
- Ребро между (u, i) , если $r_{ui} > 0$ (r_{ui} - сила связи)



- Запускаем процесс случайного блуждания.
- Обозначим $P(a|b)$ = вероятность оказаться в вершине a , если стартовали в b (Personalized PageRank)
- Предсказываем предпочтения, основываясь на

$$\hat{r}_{ui} = P(i|u)$$

Рекомендация на основе графа



- Также можно считать (для user-based и item-based)

$$\text{sim}(u, u') = \frac{1}{2} (P(u'|u) + P(u|u'))$$

$$\text{sim}(i, i') = \frac{1}{2} (P(i'|i) + P(i|i'))$$

3 Алгоритмы построения рекомендаций

- Контентные рекомендации
- Коллаборативная фильтрация
- Усредняющий алгоритм
- **User-based рекомендации**
- Item-based рекомендация
- Использование матричных разложений
- Учёт неявных откликов

User-based алгоритм

Определим функцию близости между пользователями $s(u_1, u_2)$.

Построения прогноза $\hat{r}_{u,i}$:

- 1 Найдем подмножество пользователей U_i , оценивших товар i .
- 2 Используя $s(u_1, u_2)$ найдем похожих на u пользователей N_u .
- 3 Прогноз-средний рейтинг среди пользователей $U_i \cap N_u$.

User-based алгоритм

Определим функцию близости между пользователями $s(u_1, u_2)$.

Построения прогноза $\hat{r}_{u,i}$:

- 1 Найдём подмножество пользователей U_i , оценивших товар i .
- 2 Используя $s(u_1, u_2)$ найдём похожих на u пользователей N_u
 - альтернатива: кластеризуем пользователей
 - тогда N_u - все пользователи кластера, где u
 - при кластеризации не учитываем пропуски ($r_{ui} = 0$)
- 3 Прогноз-средний рейтинг среди пользователей $U_i \cap N_u$.
 - лучше взвешенное среднее

User-based алгоритм

Базовый user-based прогноз:

$$\hat{r}_{u,i} = \frac{\sum_{u' \in U_i \cap N_u} s(u, u') r_{u',i}}{\sum_{u' \in U_i \cap N_u} |s(u, u')|}$$

User-based алгоритм

Базовый user-based прогноз:

$$\hat{r}_{u,i} = \frac{\sum_{u' \in U_i \cap N_u} s(u, u') r_{u',i}}{\sum_{u' \in U_i \cap N_u} |s(u, u')|}$$

явный учет пользовательских смещений
(оптимисты/пессимисты):

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{u' \in U_i \cap N_u} s(u, u') (r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in U_i \cap N_u} |s(u, u')|}$$

User-based алгоритм

Базовый user-based прогноз:

$$\hat{r}_{u,i} = \frac{\sum_{u' \in U_i \cap N_u} s(u, u') r_{u',i}}{\sum_{u' \in U_i \cap N_u} |s(u, u')|}$$

явный учет пользовательских смещений
(оптимисты/пессимисты):

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{u' \in U_i \cap N_u} s(u, u') (r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in U_i \cap N_u} |s(u, u')|}$$

явный учет смещений и разбросов:

$$\hat{r}_{u,i} = \bar{r}_u + \sigma_u \frac{\sum_{u' \in U_i \cap N_u} s(u, u') (r_{u',i} - \bar{r}_{u'}) / \sigma_{u'}}{\sum_{u' \in U_i \cap N_u} |s(u, u')|}$$

μ_u, σ_u - среднее и std. отклонение пользователя u .

Выбор пользователей, похожих на u

Выбор пользователей N_u , похожих на u :

- использовать всех: $U \setminus \{u\}$
- использовать K самых похожих на u (обычно $K \in [20, 50]$)
- использовать $\{u' : s(u', u) \geq \text{threshold}\}$

Похожесть пользователей

$$s(u, v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u) (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}}$$

- Учитывает только линейную связь (можно считать ранговую корреляцию),
- Если есть нейтральная оценка: $\bar{r} \rightarrow r_{neutral}$
- Можно штрафовать похожесть за малое пересечение по товарам:

$$s'(u, v) = s(u, v) \min\{|I_u \cap I_v|/50, 1\}$$

- Для более сильного учёта мнений пользователей, оценивших мало товаров, можно предварительно

$$\mathbf{r}_u \leftarrow \mathbf{r}_u / f(\|\mathbf{r}_u\|) \text{ для некоторой } \uparrow f(\cdot)$$

$\uparrow \mathbf{r}_u$ при малом $\|\mathbf{r}_u\|$.

Похожесть пользователей

- Можно сравнивать u, v по мере Жаккарда:

$$s(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}$$

- В чем недостаток?

Похожесть пользователей

- Можно сравнивать u, v по мере Жаккарда:

$$s(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}$$

- В чем недостаток?
- Отсутствует учет величины рейтингов. Воспользуемся взвешенной мерой Жаккарда:

$$s(u, v) = \frac{\sum_{i \in I_u \cap I_v} \min \{r_{u,i}, r_{v,i}\}}{\sum_{i \in I_u \cap I_v} \max \{r_{u,i}, r_{v,i}\}}$$

- Применима только для $r_{u,i} \geq 0$. Позволяет оценивать похожесть множеств по степени представленности элементов в множествах.

3 Алгоритмы построения рекомендаций

- Контентные рекомендации
- Коллаборативная фильтрация
- Усредняющий алгоритм
- User-based рекомендации
- **Item-based рекомендация**
- Использование матричных разложений
- Учёт неявных откликов

Item-based алгоритм

Определим похожесть товаров $s(i_1, i_2)$.

Алгоритм определения $\hat{r}_{u,i}$:

- ① Определим подмножество товаров I_u , оцененных u .
- ② Используя $s(i_1, i_2)$, определим подмножество товаров S_i , похожих на i .

- ③ Прогноз=средний рейтинг u по товарам $I_u \cap S_i$:

$$\hat{r}_{u,i} = \frac{\sum_{i' \in I_u \cap S_i} s(i, i') r_{u,i'}}{\sum_{i' \in I_u \cap S_i} |s(i, i')|}$$

+поправка на среднее & разброс:

$$\hat{r}_{u,i} = \mu_i + \sigma_i \frac{\sum_{i' \in I_u \cap S_i} s(i, i') (r_{u,i'} - \mu_{i'}) / \sigma_{i'}}{\sum_{i' \in I_u \cap S_i} |s(i, i')|}$$

Item-based алгоритм

Определим похожесть товаров $s(i_1, i_2)$.

Алгоритм определения $\hat{r}_{u,i}$:

- ① Определим подмножество товаров I_u , оцененных u .
- ② Используя $s(i_1, i_2)$, определим подмножество товаров S_i , похожих на i .
 - альтернатива: кластеризуем товары, S_i -все товары кластера, где i . При кластеризации не учитываем пропуски.
- ③ Прогноз=средний рейтинг u по товарам $I_u \cap S_i$:

$$\hat{r}_{u,i} = \frac{\sum_{i' \in I_u \cap S_i} s(i, i') r_{u,i'}}{\sum_{i' \in I_u \cap S_i} |s(i, i')|}$$

+поправка на среднее & разброс:

$$\hat{r}_{u,i} = \mu_i + \sigma_i \frac{\sum_{i' \in I_u \cap S_i} s(i, i') (r_{u,i'} - \mu_{i'}) / \sigma_{i'}}{\sum_{i' \in I_u \cap S_i} |s(i, i')|}$$

Особенность item-based алгоритма

- Необходимо быстро пересчитывать рекомендации по динамически наполняемой корзине товаров в магазине.
- Использовать user-based или item-based алгоритм?

Особенность item-based алгоритма

- Необходимо быстро пересчитывать рекомендации по динамически наполняемой корзине товаров в магазине.
- Использовать user-based или item-based алгоритм?
- Профиль пользователя динамически меняется.
 - User-based: нужно пересчитывать похожих пользователей, долго.
 - Item-based: $s(i, i') \approx const$, предсчитаем их вместе с S_i $\forall i$. Меняется только I_u и $r_{u,i'}$, поэтому item-based быстро пересчитать.

Похожесть товаров

$$s(i, j) = \frac{\langle \mathbf{r}_i, \mathbf{r}_j \rangle}{\|\mathbf{r}_i\| \|\mathbf{r}_j\|} = \frac{\sum_{u \in U_i \cap U_j} r_{u,i} r_{u,j}}{\sqrt{\sum_{u \in U_i \cap U_j} r_{u,i}^2} \sqrt{\sum_{u \in U_i \cap U_j} r_{u,j}^2}}$$

$$s(i, j) = \frac{\sum_{u \in U_i \cap U_j} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_i \cap U_j} (r_{u,j} - \bar{r}_j)^2}}$$

Можем использовать корреляцию между рангами, штрафовать малые $|U_i \cap U_j|$.

Комментарии

- User-based или item-based более применим в онлайн-режиме?

Комментарии

- User-based или item-based более применим в онлайн-режиме?
- \oplus : простые методы, не надо обучать
- \ominus : много вычислений для $s(u, u')$, $s(i, i')$
- \ominus : приходится хранить всю матрицу рейтингов

3 Алгоритмы построения рекомендаций

- Контентные рекомендации
- Коллаборативная фильтрация
- Усредняющий алгоритм
- User-based рекомендации
- Item-based рекомендация
- **Использование матричных разложений**
- Учёт неявных откликов

Метод матричных разложений

- Будем строить эмбединг для каждого u и i в \mathbb{R}^d , $d \sim 100$:

$u \rightarrow \mathbf{p}_u \in \mathbb{R}^d$ интерес пользователя к категориям

$i \rightarrow \mathbf{q}_i \in \mathbb{R}^d$ представленность категорий в товаре

- Прогноз - соответствие интересующих и представленных категорий

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

- В матричной записи ($\mathbf{p}_u, \mathbf{q}_i$ - строки P, Q):

$$\hat{R} = PQ^T$$

- Прогнозирование свелось к задаче низкорангового разложения R .
 - можем использовать SVD, pLSA и др.

Метод pure SVD

- Метод pure SVD использует trunkated SVD:

$$R \approx \hat{R} = \underbrace{(U\Sigma)}_P \underbrace{V^T}_Q$$

- Заполнение пропусков r_u :

$$\hat{\mathbf{r}}_u = V \left(\underbrace{V^T \mathbf{r}_u}_{\mathbf{p}_u} \right)$$

⊖ : SVD не допускает пропуски

- можно заполнить: 0, средним, оценкой другой моделью
- но это вызывает смещение оценок

Метод матричных разложений

- С поправкой на средние оценки пользователя и товара

$$\hat{r}_{ui} = w_u + w_i + \langle \mathbf{p}_u, \mathbf{q}_i \rangle$$

- Функция потерь (невыпуклая):

$$\sum_{(u,i) \in R} (w_u + w_i + \langle \mathbf{p}_u, \mathbf{q}_i \rangle - r_{ui})^2 + \alpha \sum_{u \in U} \|\mathbf{p}_u\|^2 + \beta \sum_{i \in I} \|\mathbf{q}_i\|^2$$

$$\rightarrow \min_{\mathbf{p}_u, \mathbf{q}_i, w_u, w_i}$$

- Оценивание - SGD

- повторять до сходимости
 - сэмплируем случайную пару $(u, i) \in R$, обновляем¹:

$$\mathbf{p}_u := \mathbf{p}_u - \varepsilon \frac{\partial \mathcal{L}(u, i)}{\partial \mathbf{p}_u}; \quad \mathbf{q}_i := \mathbf{q}_i - \varepsilon \frac{\partial \mathcal{L}(u, i)}{\partial \mathbf{q}_i}; \quad \dots$$

¹Посчитайте аналитически.

Оценивание методов ALS

$$\sum_{(u,i) \in R} (w_u + w_i + \langle \mathbf{p}_u, \mathbf{q}_i \rangle - r_{ui})^2 + \alpha \sum_{u \in U} \|\mathbf{p}_u\|^2 + \beta \sum_{i \in I} \|\mathbf{q}_i\|^2$$
$$\rightarrow \min_{\mathbf{p}_u, \mathbf{q}_i, w_u, w_i}$$

Метод ALS (alternating least squares):

- повторять по сходимости:
 - При фикс. P, W_u - найдем $w_i, q_i, i \in I$
 - При фикс. Q, W_i - найдем $w_u, p_u, u \in U$

Каждый раз - гребневая регрессия на $(u, i) \in R$

- аналит. решение, глобальный минимум

Метод матричных разложений

- Рассмотрим: при фикс. Q, W_i - нахождение $w_u, p_u, u \in U$
 - решаем для каждого u независимо
 - параллелизация
 - не нужно хранить все P
 - удобно для нового u
 - легко учитывать всю новую информацию о u
- p_u, q_i можно использовать:
 - для прогноза
 - как признаки в content-based модели
- q_i можно найти независимо как эмбединги i
 - удобно для новых товаров (без статистики) - строим их эмбединги по контенту (описание, музыка, текст, видео)

SVD++

- SVD++: для учёта не только явных оценок $r_{u,i}$, но и неявных (напр. просмотры).
- Вводится отдельный эмбединг для просмотренных товаров j : \mathbf{y}_j .
- Прогноз:

$$\sum_{(u,i) \in R} \left(w_u + w_i + \left\langle \mathbf{p}_u + \frac{1}{\sqrt{|\text{view}(u)|}} \sum_{j \in \text{view}(u)} \mathbf{y}_j, \mathbf{q}_i \right\rangle - r_{ui} \right)^2$$

$$+ \alpha \sum_{u \in U} \|\mathbf{p}_u\|^2 + \beta \sum_{i \in I} \|\mathbf{q}_i\|^2 + \gamma \sum_{i \in I} \|\mathbf{y}_i\|^2$$

$$\rightarrow \min_{\mathbf{p}_u, \mathbf{q}_i, \mathbf{y}_j, w_u, w_i}$$

- Нормировка почему-то на $\sqrt{|\text{view}(u)|}$.

Учёт социальных связей²

- Люди часто формируют рейтинги под влиянием друзей.
- Метод Social Regularization позволяет точнее оценивать профиль u , используя информацию о его круге друзей $\mathcal{F}(u)$.
- Предлагается метод матричных разложений с социальным регуляризатором:

$$R_1 = \sum_{u \in U} \left\| p_u - \frac{1}{|\mathcal{F}(u)|} \sum_{f \in \mathcal{F}(u)} p_f \right\|^2$$

$$R_2 = \sum_{u \in U} \sum_{f \in \mathcal{F}(u)} \|p_u - p_f\|^2$$

- В чём недостаток?

²Recommender Systems with Social Regularization

Учёт социальных связей

- Поскольку друзья могут различаться по вкусам, то правильнее их учитывать с учётом похожести:

$$R_1 = \sum_{u \in U} \left\| p_u - \frac{1}{|\mathcal{F}(u)|} \frac{\sum_{f \in \mathcal{F}(u)} \text{sim}(u, f) p_f}{\sum_{f \in \mathcal{F}(u)} \text{sim}(u, f)} \right\|^2$$

$$R_2 = \sum_{u \in U} \sum_{f \in \mathcal{F}(u)} \text{sim}(u, f) \|p_u - p_f\|^2$$

- Даёт прирост качества, лучше всего работала с R_2 и похожестью=корреляции.

Факторизационные машины^{3,4}

- Факторизационные машины (FM, factorization machines) применимы для регрессии, бинарной классификации, ранжирования и рек. систем.
 - для рек. систем - часто занимала топовые места на соревнованиях.
- Для рекомендаций - вектор признаков x_i в виде (вариант):

Feature vector x																	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	$y^{(3)}$
$x^{(4)}$	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	$y^{(6)}$
A B C ... User				TI NH SW ST ... Movie				TI NH SW ST ... Other Movies rated				Time	TI NH SW ST ... Last Movie rated									

$$\hat{y}(x) = \tilde{w}_0 + \sum_{i=1}^D \tilde{w}_i x^i + \sum_{i=1}^N \sum_{j=i+1}^N \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

³<https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf>

⁴Реализация: <http://www.libfm.org>

Билинейная модель

- Обычная билинейная модель
 - напр. SVM+poly-kernel[d=2]

$$\hat{y}(x) = \tilde{w}_0 + \sum_{i=1}^D \tilde{w}_i x^i + \sum_{i=1}^D \sum_{j=i}^D w_{ij} x_i x_j$$

- Много параметров за счёт $W \in \mathbb{R}^{D \times D}$, переобучается
- Требует неразреженных данных, иначе не обучится.
- $O((\alpha D)^2)$ - прогноз и обучение
 - α - доля ненулевых признаков.
- Любая $W = W^T$ допускает спектральное разложение:

$$W = P \Sigma P^T = P \Sigma P^T = P \Sigma^{1/2} \left(\Sigma^{1/2} \right)^T P^T = S S^T$$

Отличие факторизационных машин

- В FM W приближается с помощью:

$$W \approx VV^T, \quad V \in \mathbb{R}^{D \times K}, \quad K \ll D$$

Факторизационные машины (FM):

$$\hat{y}(x) = \tilde{w}_0 + \sum_{i=1}^D \tilde{w}_i x^i + \underbrace{\sum_{i=1}^N \sum_{j=i+1}^N \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j}_{\text{почти } \frac{1}{2} x^T V V^T x = \frac{1}{2} \langle V^T x, V^T x \rangle}$$

- Сложность прогноза и сложность обновления всех весов через SGD: $O(\alpha DK)$
 - α - доля ненулевых признаков.
- FM лучше учатся на разреженных данных за счёт факторизации (с малым K)

Метод матричных разложений

- $\hat{r}_{ui} = w_u + w_i + \langle \mathbf{p}_u, \mathbf{q}_i \rangle = w_u + w_i + \|\mathbf{p}_u\| \|\mathbf{q}_i\| \cos \phi$
 - $\|\mathbf{q}_i\| \gg 1 \Rightarrow$ часто $|\hat{r}_{ui}| \gg 1$ (популярные) можно хранить только их
 - чтобы не заикливаться на популярных, можно подмешивать товары по cos-sim $(\mathbf{p}_u, \mathbf{q}_i)$
- Neural collaborative filtering: нейросетевое обобщение $\langle \mathbf{p}_u, \mathbf{q}_i \rangle \rightarrow F_\theta(\mathbf{p}_u, \mathbf{q}_i)$

$$\sum_{(u,i) \in R} (w_u + w_i + F_\theta(\mathbf{p}_u, \mathbf{q}_i) - r_{ui})^2 + \alpha \sum_{u \in U} \|\mathbf{p}_u\|^2 + \beta \sum_{i \in I} \|\mathbf{q}_i\|^2$$

$$\rightarrow \min_{\mathbf{p}_u, \mathbf{q}_i, w_u, w_i, \theta}$$

- но нейросеть вычисляется долго и не так хорошо способна выучить $\langle \mathbf{p}_u, \mathbf{q}_i \rangle^5$.

⁵<https://arxiv.org/pdf/2005.09683.pdf>

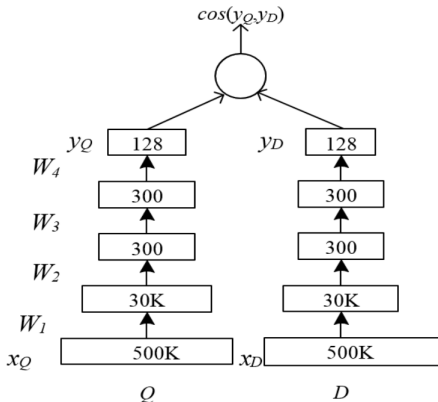
Модель DSSM⁶

- Модель DSSM (Deep Structured Semantic Model)
вычисляют соответствие
 - пользователя и товара (рек. система)
 - документа и запроса (ранжирование)
 - p_u, q_i можно брать content-based
 - вариант: $p_u = \text{mean}_{r_{ui} > 0} \{q_i\}$
 - либо подставлять p_u и q_i из др. модели

⁶ A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems.

Модель DSSM

- Используется сиамская сеть, отображающее все в общем семантическое пространство.
- $\cos\text{-sim}$ используется для оценки похожести



3 Алгоритмы построения рекомендаций

- Контентные рекомендации
- Коллаборативная фильтрация
- Усредняющий алгоритм
- User-based рекомендации
- Item-based рекомендация
- Использование матричных разложений
- **Учёт неявных откликов**

Модель Implicit CF

- Большинство рек. систем работают с явными оценками товаров (explicit feedback).
- Модель Implicit CF⁷ ориентирована на неявные оценки.
 - время просмотра, количество покупок, потраченные ср-ва (чем больше, тем достовернее связь)
- Прогноз - интерпретируемый в виде item-based рекомендации с автонастраиваемой похожестью товаров
 - похожесть своя для каждого u

⁷Collaborative Filtering for Implicit Feedback Datasets.

Модель Implicit CF

- Пусть r_{ui} - неявная вещественная оценка (напр. время просмотра видео)
- Определим

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

- Силу взаимодействия закодируем через вес

$$c_{ui} = 1 + \alpha r_{ui}, \quad \alpha \sim 40.$$

- Предлагается решать⁸

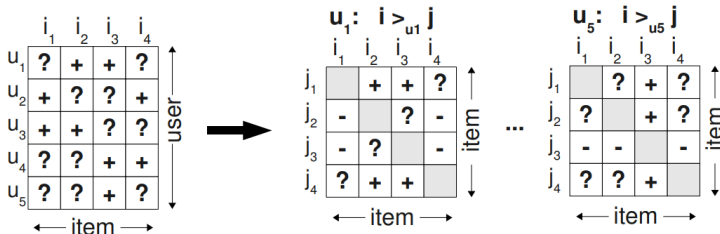
$$\sum_{(u,i) \in R} c_{ui} (w_u + w_i + \langle \mathbf{p}_u, \mathbf{q}_i \rangle - s_{ui})^2 + \alpha \sum_{u \in U} \|\mathbf{p}_u\|^2 + \beta \sum_{i \in I} \|\mathbf{q}_i\|^2$$

$$\rightarrow \min_{\mathbf{p}_u, \mathbf{q}_i, w_u, w_i}$$

⁸В оригинальной статье - без w_u, w_i и $\alpha = \beta$.

Рекомендации с помощью ранжирования

- Модель BPR (Bayesian Personalized Ranking)⁹ настраивается на $r_{u,i} \in \{0, 1\}$ (implicit feedback)
- Используются потери из ранжирования.
- Если $r_{ui} = 1$ и $r_{uj} = 0$, то предполагаем, что для u $i \succ_u j$.



⁹<https://arxiv.org/pdf/1205.2618.pdf>

Рекомендации с помощью ранжирования

$$\sum_{\{u,i,j: i \succ_u j\}} \underbrace{\ln \sigma(\langle \mathbf{p}_u, \mathbf{q}_i \rangle - \langle \mathbf{p}_u, \mathbf{q}_j \rangle)}_{P(i \succ_u j)} - \alpha \sum_{u \in U} \|\mathbf{p}_u\|^2 - \beta \sum_{i \in I} \|\mathbf{q}_i\|^2$$

$$\rightarrow \max_{\mathbf{p}_u, \mathbf{q}_i, w_u, w_i}$$

Похожий метод¹⁰ (ранжирование через hinge):

$$\sum_{\{u,i,j: i \succ_u j\}} \max \{0; 1 + \langle \mathbf{p}_u, \mathbf{q}_j \rangle - \langle \mathbf{p}_u, \mathbf{q}_i \rangle\}$$

$$+ \alpha \sum_{u \in U} \|\mathbf{p}_u\|^2 + \beta \sum_{i \in I} \|\mathbf{q}_i\|^2 \rightarrow \min_{\mathbf{p}_u, \mathbf{q}_i, w_u, w_i}$$

¹⁰ Improving Maximum Margin Matrix Factorization

Содержание

- 1 Задача построения рекомендаций
- 2 Доступная информация
- 3 Алгоритмы построения рекомендаций
- 4 Оценка и бизнес-особенности рекомендаций

Рекомендательные системы на практике

Рекомендательные системы на практике состоят из этапов:

- 1 Отбор кандидатов ~ 3000
- 2 Препроцессинг кандидатов
- 3 Переранжирование кандидатов сложной рек. моделью
- 4 Постпроцессинг ранжирования из бизнес требований.

Рекомендательные системы на практике

Рекомендательные системы на практике состоят из этапов:

- ❶ Отбор кандидатов ~ 3000
 - по пользовательским предпочтениям (подпискам, просмотрам, интересам в анкете, поисковому запросу)
 - дополнения к предыдущим покупкам
 - по близости content-based эмбедингов пользователей и товаров
 - по облегченной рекомендательной системе
 - по товарам с высоким $\|\mathbf{q}_i\|$ (потенциально популярные)
 - близкие к \mathbf{p}_u (существуют м-ды быстрого поиска похожих)
- ❷ Препроцессинг кандидатов
 - удаляем сомнительные
 - удаляем слишком похожие
 - например, кластеризуем и берём по одному из каждого кластера
- ❸ Переранжирование кандидатов сложной рек. моделью
- ❹ Постпроцессинг ранжирования из бизнес требований.

Холодный старт

Холодный старт (cold start) - проблема построения прогнозов

- для нового пользователя:
- для нового товара:

Холодный старт

Холодный старт (cold start) - проблема построения прогнозов

- для нового пользователя:
 - рекомендовать популярное (summary-based)
 - вручную модерируемые подборки
 - опросить в начале, что пользователю интересно
- для нового товара:
 - рекомендовать по тематике
 - нужно гарантировать опр. кол-во просмотров
 - exploration vs. exploitation tradeoff

Контроль качества

- Как разбивать на train/test?

Контроль качества

- Как разбивать на train/test?
 - главное: разбить по времени
 - иначе модель будет знать заранее о трендах моды и вкусов
 - можно стратифицировать по категориям товаров
- Важно в онлайне следить за качеством модели
 - донастраивать при деградации
- Все нововведения должны проверяться через A/B тест

Метрики качества

- Кликовые метрики по top-K рекомендованным:
- Бизнес-метрики:

Метрики качества

- Кликовые метрики по top-K рекомендованным:
 - precision@K: доля кликнутых товаров
 - hitrate@K: 1, если кликнул на что-либо, 0 иначе
 - др. метрики из ранжирования (nDCG, pFound, ...)
- Бизнес-метрики:
 - время просмотра рекомендованных товаров
 - частота покупок рекомендаций
 - прибыль от рекомендаций
 - доля вернувшихся пользователей (retention)

Метрики качества

Что еще важно для качественных рекомендаций?

Метрики качества

Что еще важно для качественных рекомендаций?

- Фильтровать сомнительные рекомендации
 - используя ассессоров или отзывы пользователей
- Нетривиальность рекомендаций, "способность удивлять" (serendipity)
 - рекомендовать масло к хлебу - тривиально
- Обеспечивать разнообразие рекомендаций
 - повышает $\text{hitrate}@K$, почему?

Метрики качества

Что еще важно для качественных рекомендаций?

- Фильтровать сомнительные рекомендации
 - используя ассессоров или отзывы пользователей
- Нетривиальность рекомендаций, "способность удивлять" (serendipity)
 - рекомендовать масло к хлебу - тривиально
- Обеспечивать разнообразие рекомендаций
 - повышает hitrate@K , почему?
 - модель максимизирует для каждого i $p(\text{click on } i|u)$
 - а $\text{hitrate@K} = p(\text{click on any } i \in i_1, i_2, \dots, i_K|u)$

Комментарии

- Получается много метрик m_1, \dots, m_S
- Можно обучить веса наилучшей смеси, макс. коррелирующую с целевой бизнес-метрикой (прибыль, retention, ...)

$$\text{TargetMetric} \approx w_0 + w_1 m_1 + \dots + w_S m_S$$

- Особенность алгоритмов CF:
 - усиление моды (популярные i становятся еще популярнее)
 - усиление сегментации интересов
 - прививочники смотрят видео о пользе прививок
 - антипрививочники - об их возможном вреде

Комментарии

- Важно учитывать время:
 - многие товары имеют сезонный спрос
 - со временем меняется интерфейс системы, вкусы, мода.
 - для некоторых товаров (новости), важна временная свежесть.
- В данных много шума:
 - семейные аккаунты
 - возможны атаки на систему
 - положит. комментарии на "свои" товары
 - отрицат. комментарии на "чужие" товары

Заключение

- Рекомендации могут быть основаны:
 - на характеристиках пользователя и товара (content-based)
 $F : (\text{признаки пользователя, признаки товара}) \rightarrow \text{соответствие}$
 - на матрице рейтингов (collaborative filtering)
 - усредняющий алгоритм
 - memory-based алгоритмы (user/item based)
 - матричная факторизация (потери - поточенные или попарные)
- Итоговые рекомендации учитывают прогноз и др. факторы
 - разнообразие, нетривиальность, приличность и др.