# Коллаборативная фильтрация и матричные разложения

К. В. Воронцов
vokov@forecsys.ru

Этот курс доступен на странице вики-ресурса
http://www.MachineLearning.ru/wiki
«Машинное обучение (курс лекций, К.В.Воронцов)»

МФТИ • 29 марта 2024

# Содержание

# Определения и обозначения

$U$ — множество клиентов (субъектов/пользователей — users)
$I$ — множество объектов (товаров/предметов — items)

**Типы исходных данных:**

- $D = (u_t, i_t, y_t)_{t=1}^{T} \in U \times I \times Y$ — транзакционные данные, $Y$ — пространство описаний транзакций
- $R = \left(r_{ui}\right)_{U \times I}$ — матрица отношений (или кросс-табуляции) $r_{ui} = \text{aggr}\left\{(u_t, i_t, y_t) \in D \;\middle|\; u_t = u,\ i_t = i\right\}$
- $r_{ui} \in \{0, 1\}$ — бинарные данные
- $r_{ui} \in \{1, 2, \ldots, M\}$ — рейтинги (порядковые или целые)

**Задачи** в рекомендательных системах:

- *заполнение пропусков* (missing values) в ячейках $r_{ui}$
- *ранжирование* списка top-$n$ рекомендаций для $u$ или для $i$

# Пример 1. Рекомендательная система для e-commerce

$U$ — клиенты интернет-магазина

$I$ — товары (книги, видео, музыка, курсы, мероприятия и т.п.)

$r_{ui} = [\text{клиент } u \text{ посмотрел/заказал/купил товар } i]$

**Задачи персонализации предложений:**

- заполнение пропусков в ячейках $r_{ui}$
- ранжирование списка top-$n$ рекомендаций для клиента $u$
- поиск коллабораций (collaborative filtering)
- таргетирование предложений (cross-selling, up-selling)

| Category | Recommender systems |
|---|---|
| **Video** | Netflix, Hulu, Youtube, Youku, iQiyi, Tudou, Ku6 |
| **News** | Google news, ifeng, Toutiao, NetEase news, Digg |
| **Music** | Yahoo! Music, Pandora, Douban music, QQ music, Google play, Last.fm |
| **Socia networking services** | Facebook, Twitter, sina weibo, QQ, Linkedin |
| **E-business** | Amazon, eBay, taobao, JD, Suning |

*Rui Chen et al.* A survey of collaborative filtering-based recommender systems: from traditional methods to hybrid methods based on social networks. 2018.

# Пример 2. Рекомендательная система для web-страниц

$U$ — пользователи Интернет
$I$ — текстовые документы (сайты/страницы/новости и т.п.)
$W$ — словарь слов (токенов/термов), образующих документы

$r_{ui} = [$пользователь $u$ посетил/лайкнул документ $i]$
$n_{iw} =$ частота слова $w$ в документе $i$

*Web Content Mining* — анализ данных о контенте $(n_{iw})$
*Web Usage Mining* — анализ данных об использовании $(r_{ui})$

**Основная гипотеза WUM:** действия пользователя характеризуют его интересы, возможности, привычки, вкусы

**Задачи персонализации:** найти релевантные документы $i$ для пользователя, документа или подборки документов

**Примеры:** Я.Музыка, YouTube, Дзен, МирТесен, SurfingBird.ru

# Пример 3. Рекомендательная система на основе рейтингов

$U$ — клиенты интернет-магазина

$I$ — товары (книги, видео, музыка, курсы, мероприятия и т.п.)

$r_{ui}$ = рейтинг, который клиент $u$ поставил товару $i$

**Пример: конкурс Netflix [www.netflixprize.com]**

- 2 октября 2006 — 21 сентября 2009; главный приз — \$$10^6$
- $|U| = 480\,189$ фильмов, $|I| = 17\,770$ клиентов
- $10^8$ рейтингов фильмов по шкале $\{1, 2, 3, 4, 5\}$
- точность прогнозов оценивалась по тестовой выборке $D'$

$$\text{RMSE}^2 = \frac{1}{|D'|} \sum_{(u,i) \in D'} \left(r_{ui} - \hat{r}_{ui}\right)^2$$

- задача: уменьшить RMSE с 0.9514 до 0.8563 (на 10%)

---

*Y. Koren*. The BellKor solution to the Netflix Grand Prize. August 2009.

*A. Töscher, M. Jahrer*. The BigChaos solution to the Netflix Grand Prize. 2009.

# Основные подходы в коллаборативной фильтрации (CF)

**По используемой математической технике:**

- *корреляционные модели* (Memory-Based CF): корреляции строк/столбцов исходной матрицы $R$
- *латентные семантические модели* (Latent Model-Based CF): векторные представления (эмбединги) клиентов и объектов

**По типам используемых данных:**

- 2D: только матрица $R$
- 3D: контекстно-зависимые (context-aware, field-aware)
- учёт содержимого (content-aware, гибридные модели)
- учёт связей доверия между клиентами (trust-aware)
- учёт времени (time-aware)

---

*F.Ricci, L.Rokach, B.Shapira.* Recommender Systems Handbook. Springer. 2015.
https://shuaizhang.tech/posts/2019/08/blog-post-2

# Непараметрическая регрессия для восстановления пропусков

**Оценка рейтинга по схожим клиентам** (User-Based CF):

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in U(u)} S(u,v)(r_{vi} - \bar{r}_v)}{\sum_{v \in U(u)} S(u,v)}$$

$U(u)$ — *коллаборация*, множество клиентов, схожих с $u$
$\bar{r}_u$ — средний рейтинг клиента $u$
$S(u,v)$ — функция сходства пары клиентов $(u,v)$

**Оценка рейтинга по схожим объектам** (Item-Based CF):

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in I(i)} S(i,j)(r_{uj} - \bar{r}_j)}{\sum_{j \in I(i)} S(i,j)}$$

$I(i)$ — множество объектов, схожих с $i$
$\bar{r}_i$ — средний рейтинг объекта $i$
$S(i,j)$ — функции сходства пары объектов $(i,j)$

# Функции сходства для рейтинговых данных

$I(u)$ — множество объектов, которые клиент $u$ отрейтинговал
$I(u,v)$ — множество объектов, которые отрейтинговали $u$ и $v$

**Корреляция Пирсона:**

$$S(u,v) = \frac{\sum_{i \in I(u,v)} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I(u,v)} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I(u,v)} (r_{vi} - \bar{r}_v)^2}}$$

**Косинусная мера сходства:**

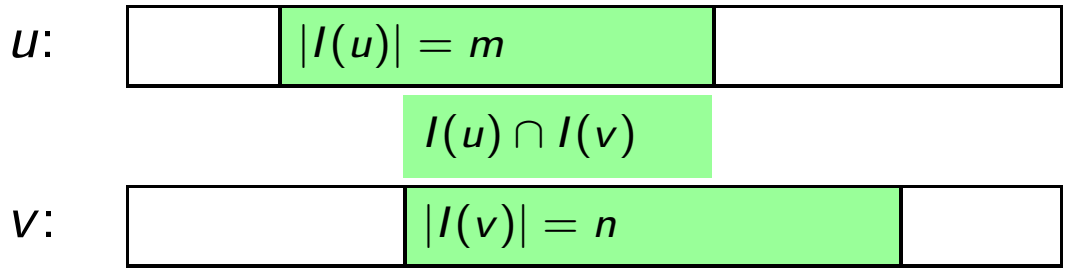
$$S(u,v) = \frac{\sum_{i \in I(u,v)} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I(u)} r_{ui}^2} \sqrt{\sum_{i \in I(v)} r_{vi}^2}},$$

где неявно предполагается, что $r_{ui} = 0$, если $i \neq I(u)$

Функция сходства $S(i,j)$ пар объектов определяется аналогично

# Функции сходства для бинарных данных

Чем больше $I(u) \cap I(v)$, тем более схожи клиенты $u$ и $v$:




**Мера близости Жаккара** (Jaccard similarity):

$$S(u,v) = \frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|}$$

**Точный тест Фишера** (Fisher's Exact Test, FET)
Вероятность пересечения оценок при *нулевой гипотезе*, что клиенты $u$ и $v$ совершают свой выбор независимо:

$$S(u,v) = -\log P\{|I(u) \cap I(v)| = i\} = -\log \frac{C_m^i C_{|I|-m}^{n-i}}{C_{|I|}^n}$$

# Разреженная линейная модель (Sparse LInear Method, SLIM)

**Item-Based:** оцениваем $\hat{r}_{ui}$ по $r_{uj}$ других объектов, используя обучаемые параметры $w_{ij}$ вместо функции сходства $S(i,j)$:

$$\hat{r}_{ui} = \sum_{j \in I \setminus i} r_{uj} w_{ij} = \langle r_u, w_i \rangle$$

В матричной записи: $|I|$ линейных регрессий $\underset{U\times 1}{\hat{r}_i} = \underset{U\times I\times 1}{R\, w_i}$, $i \in I$

**Метод наименьших квадратов** с регуляризаторами $L_2$ и $L_1$, с ограничением неотрицательности коэффициентов и условием, что объект не должен оцениваться по самому себе:

$$\frac{1}{2}\|r_i - Rw_i\|^2 + \frac{\beta}{2}\|w_i\|_2^2 + \lambda\|w_i\|_1 \;\to\; \min_{w_i}$$
$$w_i \geqslant 0, \quad w_{ii} = 0$$

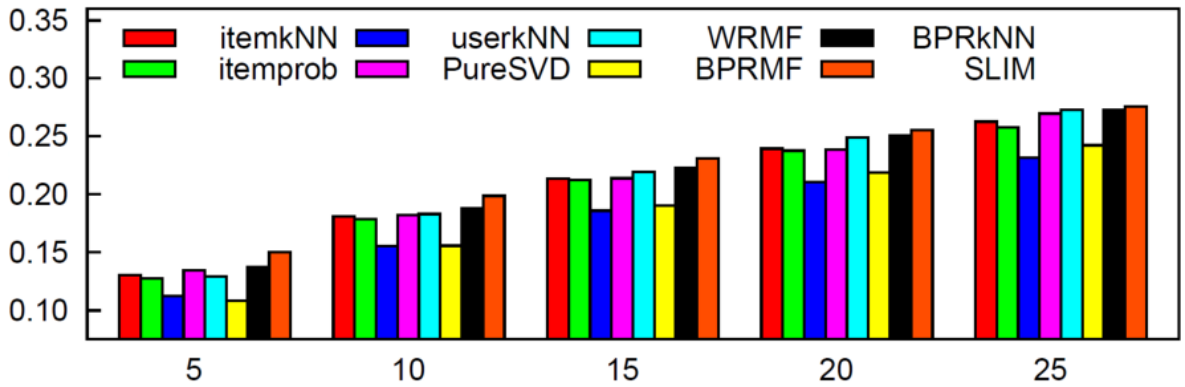
---

*Xia Ning, George Karypis.* SLIM: sparse linear methods for top-N recommender systems. 2011.

# Преимущества SLIM (Sparse LInear Method)

- Это стандартная задача линейной регрессии (ElasticNet)
- Решение распараллеливается по столбцам $r_i$, $i \in I$
- $L_1$-регуляризация делает решение разреженным
- Дополнительный отбор признаков по косинусной мере: для каждого $i$ берём 10 признаков $j$ с наибольшими $S(i,j)$
- Успешный метод по качеству ранжирования и скорости

**Пример:** зависимость LOOCV HitRate от длины списка (на данных Netflix)




*Xia Ning, George Karypis.* SLIM: sparse linear methods for top-N recommender systems. 2011.

# Резюме по Memory-Based методам

**Преимущества для бизнес-приложений:**

- легко понимать и объяснять:
  «те, кто купил эту книгу, также покупали...» [Amazon.com]
- легко реализовать
- метод SLIM — простой и до сих пор один из лучших

**Недостатки:**

- требуется хранение огромной разреженной матрицы $R$
- проблема «холодного старта»
  — не ясно, что рекомендовать новым клиентам
  — не ясно, как и кому рекомендовать новые объекты
- иногда рекомендации тривиальны
  — предлагается всё наиболее популярное

**Далее:** как *латентные модели* устраняют эти недостатки

# Низкоранговые матричные разложения (matrix factorization)

$T$ — множество интересов (тем): $|T| \ll |U|, \ |T| \ll |I|$

$p_{ut}$ — *векторное представление* клиента $u$, $P = \left(p_{ut}\right)_{U \times T}$

$q_{it}$ — *векторное представление* объекта $i$, $Q = \left(q_{it}\right)_{I \times T}$

**Задача:** найти разложение $\hat{r}_{ui} = \sum\limits_{t \in T} p_{ut} q_{it} = \langle p_u, q_i \rangle$

**Матричная запись:** $\hat{R} = PQ^{\mathrm{T}}$, критерий $\|R - PQ^{\mathrm{T}}\| \to \min\limits_{P,Q}$

**Вероятностная интерпретация:** $\underbrace{p(i|u)}_{r_{ui}} = \sum\limits_{t \in T} \underbrace{q(i|t)}_{q_{it}} \underbrace{p(t|u)}_{p_{ut}}$

**Методы решения:**

- Сингулярное разложение (singular value decomposition, SVD)
- Неотрицательное матричное разложение (NNMF): $p_{ut}, q_{it} \geqslant 0$
- Стохастическое матричное разложение (Topic Modeling)

# Сингулярное разложение (singular value decomposition, SVD)

Постановка задачи: $\|R - PQ^{\mathrm{T}}\|^2 \to \min\limits_{P,Q}$

Сингулярное разложение: $\hat{R} = \underbrace{V\sqrt{D}}_{P}\underbrace{\sqrt{D}U^{\mathrm{T}}}_{Q^{\mathrm{T}}}$, $U^{\mathrm{T}}U = I$, $V^{\mathrm{T}}V = I$

**Достоинства:**

- можно применять готовые библиотеки линейной алгебры
- хорошее ранжирование предложений на некоторых данных

**Недостатки:**

- если $r_{ui}$ не известно, то полагаем $r_{ui} = 0$
  (неявно считаем, что если клиент $u$ никогда не выбирал объект $i$, то он ему, скорее всего, не интересен)
- ортогональность (собственных) векторов $p_t$, $q_t$
- неинтерпретируемость компонент векторов $p_u$, $q_i$

---

*Cremonesi P., Koren Y., Turrin R.* Performance of recommender algorithms on top-n recommendation tasks. RecSys 2010.

# Метод чередующихся наименьших квадратов (ALS)

Постановка задачи:

$$\Delta(P,Q) = \|R - PQ^{\mathsf{T}}\|^2 + \lambda\|P\|^2 + \mu\|Q\|^2 \;\to\; \min_{P,Q}$$

Метод Alternating Least Squares (ALS):

$$\frac{1}{2}\frac{\partial\Delta}{\partial P} = (PQ^{\mathsf{T}} - R)Q + \lambda P = 0 \quad\Rightarrow\quad P = RQ(Q^{\mathsf{T}}Q + \lambda I)^{-1}$$

$$\frac{1}{2}\frac{\partial\Delta}{\partial Q} = P^{\mathsf{T}}(PQ^{\mathsf{T}} - R) + \mu Q = 0 \quad\Rightarrow\quad Q = R^{\mathsf{T}}P(P^{\mathsf{T}}P + \mu I)^{-1}$$

Задача распадается на независимые линейные регрессии (вычисления легко распараллеливаются):

$$p_u = (Q^{\mathsf{T}}Q + \lambda I)^{-1}Q^{\mathsf{T}}r_u$$

$$q_i = (P^{\mathsf{T}}P + \mu I)^{-1}P^{\mathsf{T}}r_i$$

# Модель латентных факторов (LFM, Latent Factor Model)

Постановка задачи для случая разреженных данных:

$$\sum_{(u,i)\in D} \underbrace{\left(r_{ui} - \bar{r}_u - \bar{r}_i - \sum_{t\in T} p_{ut} q_{it}\right)}_{\varepsilon_{ui}}^2 \to \min_{P,Q}$$

**Метод стохастического градиента:**
перебираем все $(u,i) \in D$ многократно в случайном порядке и делаем каждый раз градиентный шаг для задачи $\varepsilon_{ui}^2 \to \min_{p_u, q_i}$:

$$p_{ut} := p_{ut} + \eta \varepsilon_{ui} q_{it}, \quad t \in T$$
$$q_{it} := q_{it} + \eta \varepsilon_{ui} p_{ut}, \quad t \in T$$

Можно также обучать $\bar{r}_u$ $(u \in U)$ и $\bar{r}_i$ $(i \in I)$ как параметры

---

*Tacáks G., Pilászy I., Németh B., Tikk D.* Scalable collaborative filtering approaches for large recommendation systems // JMLR, 2009, No. 10, Pp. 623–656.

# Модель латентных факторов (LFM, Latent Factor Model)

Преимущества метода стохастического градиента:

- легко вводится регуляризация:
$$\varepsilon_{ui}^2 + \lambda\|p_u\|^2 + \mu\|q_i\|^2 \to \min_{p_u, q_i}$$
- легко вводятся ограничения неотрицательности:
$$p_{ut} \geqslant 0, \quad q_{it} \geqslant 0 \quad \text{(метод проекции градиента)}$$
- легко вводится обобщение для ранговых данных:
$$\sum_{(u,i)\in D} \left( r_{ui} - \mu\left( \bar{r}_u + \bar{r}_i + \sum_{t\in T} p_{ut} q_{it} \right) \right)^2 \to \min_{P,Q,\mu},$$
где $\mu$ — гладкая монотонная функция с $M$ параметрами, преобразующая оценки $\hat{r}_{ui}$ в шкалу рейтингов $\{1, \ldots, M\}$
- легко реализуются все виды инкрементности — добавление ещё одного: клиента $u$ / объекта $i$ / значения $r_{ui}$
- высокая численная эффективность на больших данных

# NNMF (Non-Negative Matrix Factorization)

Метод *чередующихся наименьших квадратов* (Alternating Least Squares, ALS) для неотрицательного матричного разложения:

$$\Delta = \left\| R - \sum_{t \in T} p_t q_t^{\mathsf{T}} \right\|^2 = \left\| R_t - p_t q_t^{\mathsf{T}} \right\|^2 \to \min_{\{p_t \geqslant 0,\, q_t \geqslant 0\}}$$

**Идея:** искать поочерёдно то столбцы $p_t$, то столбцы $q_t$ при фиксированных остальных: $R_t = R - \sum_{s \in T \setminus t} p_s q_s^{\mathsf{T}}$.

$$\frac{\partial \Delta}{\partial p_t} = 0 \quad \Rightarrow \quad (p_t q_t^{\mathsf{T}} - R_t) q_t = 0 \quad \Rightarrow \quad p_t = \left( \frac{R_t q_t}{q_t^{\mathsf{T}} q_t} \right)_+$$

$$\frac{\partial \Delta}{\partial q_t} = 0 \quad \Rightarrow \quad p_t^{\mathsf{T}} (p_t q_t^{\mathsf{T}} - R_t) = 0 \quad \Rightarrow \quad q_t = \left( \frac{R_t^{\mathsf{T}} p_t}{p_t^{\mathsf{T}} p_t} \right)_+$$

(положительная срезка — из условий Каруша–Куна–Таккера)

---

*A.Cichocki, R.Zdunek, S.Amari.* Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. 2007

# Мультипликативный алгоритм NNMF

Метод наименьших квадратов с условиями неотрицательности:

$$\left\|R - PQ^{\mathsf{T}}\right\|^2 \to \min_{P,Q}, \quad P \geqslant 0, \quad Q \geqslant 0$$

Из условий Каруша–Куна–Таккера получаем систему уравнений, удобную для применения метода простых итераций:

$$p_{ut} := p_{ut}\frac{(RQ)_{ut}}{(PQ^{\mathsf{T}}Q)_{ut}} \qquad q_{it} := q_{it}\frac{(R^{\mathsf{T}}P)_{it}}{(QP^{\mathsf{T}}P)_{it}}$$

Удобная матричная запись, где $\otimes$ и $\oslash$ обозначают покомпонентное (адамарово) умножение и деление матриц:

$$P := P \otimes (RQ) \oslash (PQ^{\mathsf{T}}Q)$$
$$Q := Q \otimes (R^{\mathsf{T}}P) \oslash (QP^{\mathsf{T}}P)$$

---

*Andrzej Cichocki et al.* Nonnegative matrix and tensor factorizations. 2009.

# Вероятностный латентный семантический анализ (PLSA)

Тематические модели в CF: документ $\leftrightarrow$ клиент, слово $\leftrightarrow$ объект

Максимизация правдоподобия для тематической модели PLSA:

$$\sum_{(u,i)\in D} r_{ui} \ln \sum_{t\in T} q_{it} p_{ut} \to \max_{P,Q}, \quad q_{it} = p(i|t), \quad p_{ut} = p(t|u)$$

EM-алгоритм (результат E-шага $p(t|u,i)$ встроен в M-шаг):

$$p_{ut} := \operatorname*{norm}_{t\in T}\left(\sum_{i\in I} r_{ui}\frac{q_{it}p_{ut}}{(PQ^{\mathrm{T}})_{ui}}\right) \qquad q_{it} := \operatorname*{norm}_{i\in I}\left(\sum_{u\in U} r_{ui}\frac{q_{it}p_{ut}}{(PQ^{\mathrm{T}})_{ui}}\right)$$

Удобная для реализации матричная запись,

где r-norm — нормировка по строкам, c-norm — по столбцам:

$$P := \text{r-norm}\big(P \otimes (R \oslash PQ^{\mathrm{T}})Q\big)$$
$$Q := \text{c-norm}\big(Q \otimes (R^{\mathrm{T}} \oslash QP^{\mathrm{T}})P\big)$$

---

*M.Shashanka et al.* Probabilistic latent variable models as nonnegative factorizations. 2008.

# Модель с учётом неявной информации (implicit feedback)

Явные (explicit) предпочтения $r_{ui}$, более качественные данные:

- покупки товаров в интернет-магазине
- оценки, рейтинги, лайки/дизлайки

Неявные (implicit) предпочтения $s_{ui}$, большой объём данных:

- посещение страницы товара
- просмотр (какой-то части) фильма

**Идея:** предсказываем $s_{ui}$ с весом $c_{ui} = 1 + \alpha r_{ui}$:

$$\sum_{(u,i)\in D} c_{ui}\left(s_{ui} - \bar{s}_u - \bar{s}_i - \sum_{t\in T} p_{ut}q_{it}\right)^2 + \lambda\sum_{u\in U}\|p_u\|^2 + \mu\sum_{i\in I}\|q_i\|^2 \to \min_{P,Q}$$

Модель с неявными предпочтениями победила в Netflix Prize.

---

*Bell R. M., Koren Y., Volinsky C.* The BellKor 2008 solution to the Netflix Prize.

# Автокодировщики для коллаборативной фильтрации

$f: X \to Z$ — кодировщик (encoder), кодовый вектор $z = f(x, \alpha)$
$g: Z \to X$ — декодировщик (decoder), реконструкция $\hat{x} = g(z, \beta)$

Суперпозиция $\hat{x} = g(f(x))$ должна восстанавливать исходные $x_i$:

$$\sum_{i=1}^{\ell} \big(g(f(x_i, \alpha), \beta) - x_i\big)^2 \to \min_{\alpha,\beta}$$

Два варианта автокодировщика, user-based и item-based:

$$\text{uAE:} \quad \sum_{u \in U} \big(g(f(r_u, \alpha), \beta) - r_u\big)^2 + \rho(\alpha, \beta) \to \min_{\alpha,\beta}$$

$$\text{iAE:} \quad \sum_{i \in I} \big(g(f(r_i, \alpha), \beta) - r_i\big)^2 + \rho(\alpha, \beta) \to \min_{\alpha,\beta}$$

где $r_i$ — столбцы $R$, $r_u$ — строки $R$, $\rho(\alpha, \beta)$ — регуляризатор

# Матричное разложение как линейный автокодировщик

**Матричное разложение** (MF, matrix factorization)

$$\|R - PQ^{\mathsf{T}}\|^2 + \lambda\|P\|^2 + \mu\|Q\|^2 \;\to\; \min_{P,Q}$$

ALS с $L_2$-регуляризацией, без ограничений неотрицательности:

$$P = RQ(Q^{\mathsf{T}}Q + \lambda I)^{-1} \qquad Q = R^{\mathsf{T}}P(P^{\mathsf{T}}P + \mu I)^{-1}$$

**Автокодировщик клиентов** $r_u \xrightarrow{f} p_u \xrightarrow{g} \hat{r}_u$ (user-based AE):

$$\text{uAE:} \quad \begin{cases} p_u = f(r_u, Q) = (Q^{\mathsf{T}}Q + \lambda I)^{-1} Q^{\mathsf{T}} r_u \\ \hat{r}_u = g(p_u, Q) = Qp_u \end{cases}$$

**Автокодировщик объектов** $r_i \xrightarrow{f} q_i \xrightarrow{g} \hat{r}_i$ (item-based AE):

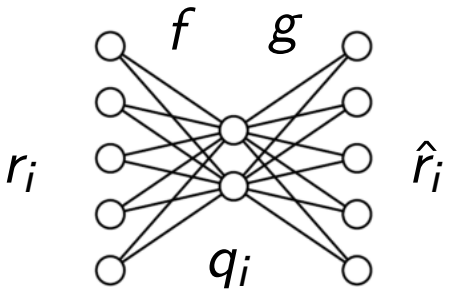
$$\text{iAE:} \quad q_i = f(r_i, P) = (P^{\mathsf{T}}P + \mu I)^{-1} P^{\mathsf{T}} r_i$$
$$\hat{r}_i = g(q_i, P) = Pq_i$$

MF = матрица параметров $\times$ матрица кодовых векторов

# Нейросетевой автокодировщик для CF

Два варианта автокодировщика, user/item-based:

$$\text{uAE:}\quad \hat{r}_u = \sigma\big(W_2\,\sigma(W_1 r_u + b_1) + b_2\big)$$
$$\text{iAE:}\quad \hat{r}_i = \sigma\big(W_2\,\sigma(W_1 r_i + b_1) + b_2\big)$$




**Шумоподавляющий автокодировщик** (Denoising iAE):
на вход $\hat{r}_i(r_i, W)$ подаются векторы $\tilde{r}_i$ с шумом, в которых некоторые координаты обнулены, $B_{ui} = [\tilde{r}_{ui} = 0 \neq r_{ui}]$

$$\sum_{(u,i)\in D} \big(\alpha B_{ui} + \beta \bar{B}_{ui}\big)\big((\hat{r}_i(\tilde{r}_i, W))_u - r_{ui}\big)^2 + \lambda\|W\|^2 \to \min_W$$

где штраф за ошибки реконструкции: $\alpha$ с шумом, $\beta$ без шума

- в экспериментах оптимальными оказались $\alpha = 1$, $\beta = 0.65$
- iAE работает немного лучше, чем uAE

---

*F.Strub et al.* Hybrid recommender system based on autoencoders. 2017.

# Дополнительные данные в MF и автокодировщиках

$x_u \in \mathbb{R}^n$ — вектор признаков клиента $u \in U$
$y_i \in \mathbb{R}^m$ — вектор признаков объекта $i \in I$
Предсказательная модель с параметрами $p_u, p'_u, q_i, q'_i$:

$$\hat{r}_{ui} = \langle p_u, q_i \rangle + \langle x_u, q'_i \rangle + \langle p'_u, y_i \rangle$$

Матричное разложение: $\|PQ^\mathsf{T} + XQ'^\mathsf{T} + P'Y^\mathsf{T} - R\|^2 \to \min\limits_{P,Q,P',Q'}$

Нейросетевые автокодировщики с дополнительными данными:

$$\text{uAE:} \quad \hat{r}_u = \sigma\big(W_2\,\sigma(W_1 r_u + W'_1 x_u + b_1) + W'_1 x_u + b_2\big)$$
$$\text{iAE:} \quad \hat{r}_i = \sigma\big(W_2\,\sigma(W_1 r_i + W'_1 y_i + b_1) + W'_2 y_i + b_2\big)$$

**Недостаток:** невозможно одновременно учитывать $x_u$ и $y_i$

---

*F.Strub et al.* Hybrid recommender system based on autoencoders. 2017.

# Дополнительные данные в линейной регрессионной модели

$x_{ui} = (x_{ui1}, \ldots, x_{uin})$ — вектор признакового описания $(u, i)$

Примеры признаков:

- для фильмов: текст описания, теги, артисты
- для музыки: жанр, исполнитель, теги
- для событий: текст описания, геолокация, отзывы
- унитарный код (one-hot encoding) клиента $u$
- унитарный код (one-hot encoding) объекта $i$

Линейная регрессионная модель для $r_{ui}$:

$$\hat{r}_{ui} = w_0 + \sum_{j=1}^{n} w_j x_{uij}.$$

Она не описывает связи между клиентами и объектами

# Дополнительные данные в квадратичной регрессионной модели

Добавим взаимодействия между признаками:

$$\hat{r}_{ui} = w_0 + \sum_{j=1}^{n} w_j x_{uij} + \sum_{j=1}^{n} \sum_{k=1}^{n} w_{jk} x_{uij} x_{uik}$$

Представим веса $w_{jk}$ низкоранговым матричным разложением:

$$w_{jk} = v_j^{\mathsf{T}} v_k, \qquad v_i \in \mathbb{R}^{|T|}$$

- регулируемое число параметров
- если нет дополнительных признаков, то получаем LFM
- настраивается с помощью SGD или ALS
- наиболее мощный инструмент — библиотека libFM

---

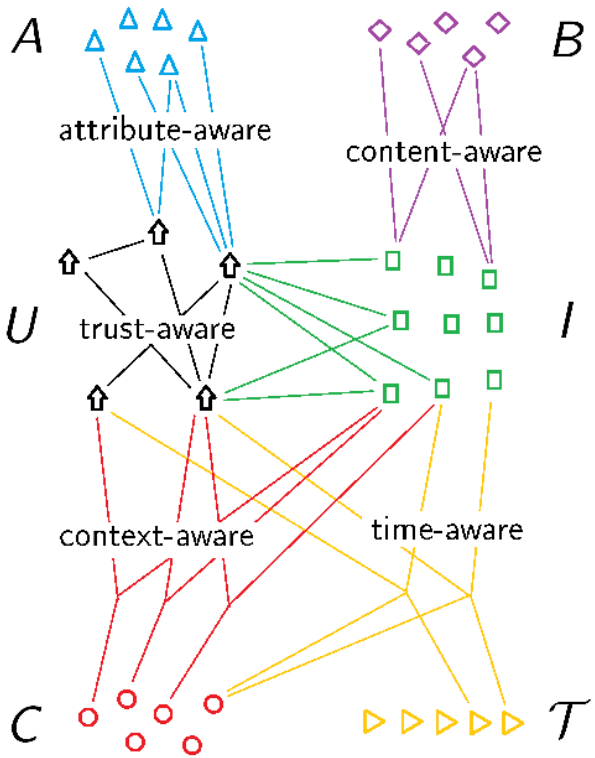
*Steffen Rendle*. Factorization machines with libFM. 2012.

# Дополнительные данные, представимые в виде (гипер)графа

$A$ — словарь атрибутов клиентов (соцдем, регион, хобби...)
$B$ — словарь свойств объектов (слова в текстовых объектах)
$C$ — конечное множество (словарь) ситуативных контекстов
$\mathcal{T}$ — конечное множество (словарь) моментов времени

**Виды дополнительных данных:**

$r_{ui}$ — клиент $u$ выбрал объект $i$
$n_{ua}$ — клиент $u$ имеет атрибут $a$
$n_{ib}$ — объект $i$ имеет свойство $b$
$n_{uv}$ — клиент $u$ доверяет клиенту $v$
$n_{uib}$ — клиент $u$ отметил $i$ тэгом $b$
$n_{uic}$ — клиент $u$ выбрал объект $i$
в ситуативном контексте $c$
$n_{uic\tau}$ — клиент $u$ выбрал объект $i$
в контексте $c$ в момент времени $\tau$



# Мультимодальная (гипер)графовая тематическая модель

**Общая идея** графовых эмбедингов и тематических моделей:
наблюдаемые рёбра объясняются скрытыми векторами вершин

$$\sum_{u,i} r_{ui} \ln \sum_{t\in T} p(t|u)\,p(t|i) : p(t)$$
$$+\lambda_1 \sum_{i,b} n_{ib} \ln \sum_{t\in T} p(t|i)\,p(t|b) : p(t)$$
$$+\lambda_2 \sum_{u,a} n_{ua} \ln \sum_{t\in T} p(t|u)\,p(t|a) : p(t)$$
$$+\lambda_3 \sum_{u,i,c} n_{uic} \ln \sum_{t\in T} p(t|i)\,p(t|u)\,p(t|c) : p^2(t)$$
$$+\lambda_4 \sum_{u,i,c,t} n_{uic\tau} \ln \sum_{t\in T} p(t|i)\,p(t|u)\,p(t|c)\,p(t|\tau) : p^3(t) \;\to\; \max$$

Оптимизация по всем эмбедингам — векторам вида $p(t|\bullet)$
Для вероятностных тематических эмбедингов — EM-алгоритм
Для обычных эмбедингов — стохастический градиент

# Измерение качества рекомендаций

RMSE — точность предсказания рейтингов (как в NetflixPrize):

$$\text{RMSE}^2 = \sum_{(u,i)\in D} \left(r_{ui} - \hat{r}_{ui}\right)^2$$

Точность предсказаний не гарантирует хороших рекомендаций

$R_u(k) \subset I$ — первые $k$ рекомендаций для клиента $u$
$L_u \subset I$ — истинные предпочтения клиента $u$

Более адекватные метрики качества рекомендаций:

- $\text{precision@}k = \frac{|R_u(k) \cap L_u|}{|R_u(k)|}$ — точность
- $\text{recall@}k = \frac{|R_u(k) \cap L_u|}{|L_u|}$ — полнота
- меры качества ранжирования: MAP, NDCG и др.

---

*Guy Shani, Asela Gunawardana*. Evaluating Recommendation Systems. 2017.

# Измерение качества рекомендаций

Многокритериальность в рекомендательных системах:

- *Разнообразие* (diversity): число рекомендаций из разных категорий или степень различия рекомендаций между сессиями клиента
- *Новизна* (novelty): сколько среди рекомендаций объектов, новых для данного клиента
- *Покрытие* (coverage): доля объектов, которые хотя бы раз побывали среди рекомендованных
- *Догадливость* (serendipity): способность угадывать неожиданные нетривиальные предпочтения клиентов
- *Предвзятость* (propensity): прошлые рекомендации влияют на выбор клиентов, смещая последующие рекомендации

Можно оптимизировать линейную комбинацию критериев, либо оптимизировать один при ограничениях на остальные.

# Оффлайн- и онлайн- измерения качества рекомендаций

Типичная схема эксперимента:

- разбиваем выборку сессий на обучение и тест
- оптимизируем оффлайн-метрику качества на обучении
- оцениваем качество на тесте и выбираем модель
- внедряем модель в рекомендательный сервис
- проводим AB-тестирование, измеряем онлайн-метрику (деньги или число кликов)

Онлайн- и оффлайн-метрики могут быть слабо связаны:

- в оффлайне не известно, что пользователь мог бы купить
- в оффлайне не известно, что он купил бы без рекомендаций

**Выводы из экспериментов:** для улучшения онлайн-качества нужно оптимизировать разные аспекты качества в оффлайне.

# Резюме

*Коллаборативная фильтрация* (Collaborative Filtering) — это набор методов для построения рекомендательных систем (Recommender Systems)

*Корреляционные модели* — простые, но устаревшие

*Латентные модели* на основе матричных разложений или автокодировщиков обладают рядом преимуществ:

- сокращается объём хранимых данных
- привлекаются внешние данные для «холодного старта»
- неотрицательные и вероятностные эмбединги интерпретируются как векторы «интересов» или «тем»
- легко добавляются регуляризаторы для многокритериальной оптимизации качества рекомендаций