Нейросетевые рекомендации

RecSys

lecturer: Mollaev D. E.
Sber AI Lab

План лекции

- SLIM
- Factorization Machines
- Content-based рекомендации

План лекции

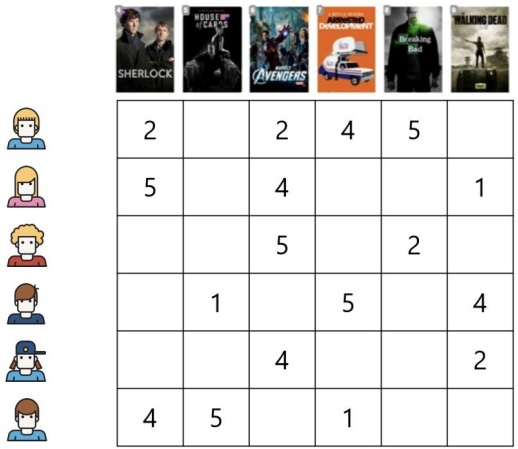
- SLIM
- Factorization Machines
- Content-based рекомендации

SLIM - Sparse LInear Methods

- A - бинарная матрица M * N user-item взаимодействий

Фильм

Пользователь



2		2	4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

SLIM - Sparse LInear Methods

- A - бинарная матрица M * N user-item взаимодействий
- Будем определять взаимодействие как взвешивание событий из прошлого:

$$\hat{a}_{ui} = \sum_{j=1}^{N} w_{ij} a_{uj}$$

SLIM - Sparse LInear Methods

- A - бинарная матрица M * N user-item взаимодействий
- Будем определять взаимодействие как взвешивание событий из прошлого:

$$\hat{a}_{ui} = \sum_{j=1}^{N} w_{ij} a_{uj}$$

- Веса wij≥ 0, то есть модель учитывает похожие айтемы. Например, для фотографии с котиком намного проще сказать, кто на нее больше всего похож, чем кто на нее меньше всего похож

SLIM - Sparse LInear Methods

- A - бинарная матрица M * N user-item взаимодействий
- Будем определять взаимодействие как взвешивание событий из прошлого:

$$\hat{a}_{ui} = \sum_{j=1}^{N} w_{ij} a_{uj}$$

- Веса w_ij ≥ 0, то есть модель учитывает похожие айтемы. Например, для фотографии с котиком намного проще сказать, кто на нее больше всего похож, чем кто на нее меньше всего похож
- Причем w_ii = 0 – позволяет явно избежать элементарного решения

$$W = I_N;$$

SLIM - Sparse LInear Methods

- A - бинарная матрица M * N user-item взаимодействий
- Будем определять взаимодействие как взвешивание событий из прошлого:

$$\hat{a}_{ui} = \sum_{j=1}^{N} w_{ij} a_{uj}$$

- Таким образом, w_ij - некоторая мера похожести товара i на товар j

SLIM - Sparse LInear Methods

- Оптимизируется MSE loss с L_1, L_2 регуляризациями

$$\frac{1}{2}\sum_{u,i}\left(a_{ui}-\sum_{j}w_{ij}a_{uj}\right)^2+\lambda\sum_{i,j}|w_{ij}|+\frac{\beta}{2}\sum_{i,j}(w_{ij})^2\to\min_{W}$$

- Заметим, что по строчкам W_i задача разбивается на m независимых:

$$\frac{1}{2}\sum_{u}\left(a_{ui}-\sum_{j}w_{ij}a_{uj}\right)^2+\lambda\sum_{j}|w_{ij}|+\frac{\beta}{2}\sum_{j}(w_{ij})^2\to\min_{w_{i1},\ldots,w_{iN}}\qquad(\forall i)$$

SLIM - Sparse LInear Methods

- Каждую задачу можно решить покоординатным спуском:
  - фиксируется все W_j кроме одной координаты w_ij
  - переходим в оптимум по w_ij
  - переходим к следующей координате
  - повторяем до сходимости

SLIM - Sparse LInear Methods

$$\hat{a}_{ui} = \sum_{j=1}^{N} w_{ij} a_{uj}$$

- Процесс построения рекомендаций:
  - Берем вектор взаимодействия пользователя A_u
  - Считаем a_ui для всех непросмотренных айтемов
  - Сортируем непросмотренные айтемы по a_ui и берем топ товаров с наибольшим значением

SLIM - Sparse LInear Methods

$$\hat{a}_{ui} = \sum_{j=1}^{N} w_{ij} a_{uj}$$

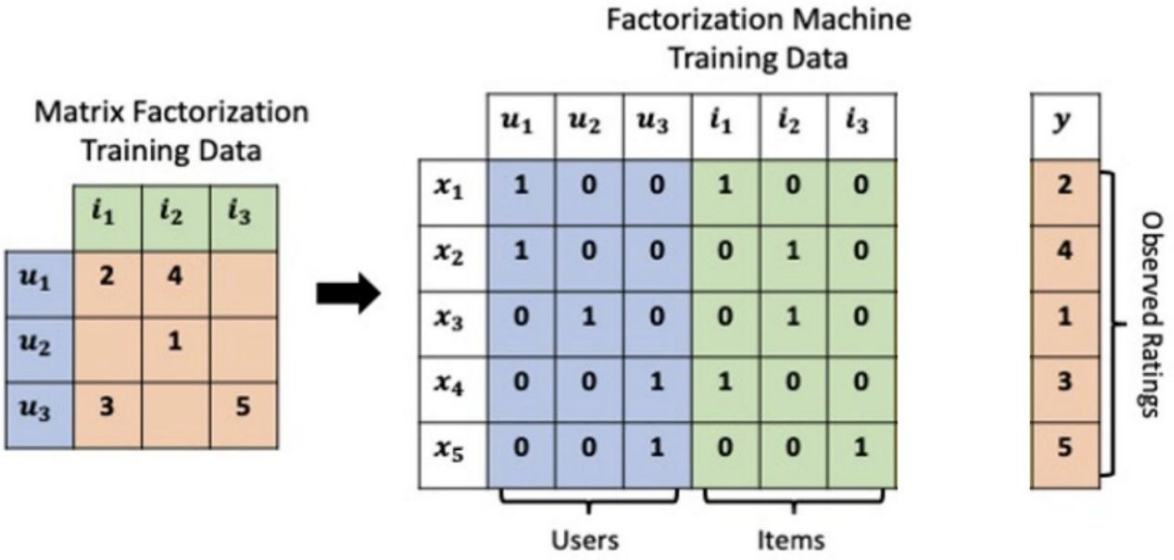
- Процесс построения рекомендаций:
  - Берем вектор взаимодействия пользователя A_u
  - Считаем a_ui для всех непросмотренных айтемов
  - Сортируем непросмотренные айтемы по a_ui и берем топ товаров с наибольшим значением
- Благодаря наличию L_1 регуляризации матрица получается разреженной, матрица событий A тоже разреженная -> что позволяет существенно ускорить асимптотику применения модели

План лекции

- SLIM
- Factorization Machines
- Content-based рекомендации

FM – Factorization Machines

- Пусть x — one-hot вектор взаимодействия пары user-item, где 1 стоит на месте соответствующих пользователя и товара (n = |I| + |U|):



FM – Factorization Machines

- Рассмотрим в данной постановке регрессионную модель:

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i$$

- Добавим в нашу регрессионную модель взаимодействия второго порядка, что позволит учитывать более сложные соотношения между признаками:

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

- В полученной модели $n(n+1)/2 + n + 1$ параметр;

- Так как $n = |I| + |U|$, размер модели становится слишком большим.

FM – Factorization Machines

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

- Сопоставим каждому признаку x_i вектор $v_i \in \mathbb{R}^k$ и представим модель в виде:

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

FM – Factorization Machines

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

- Сопоставим каждому признаку x_i вектор $v_i \in \mathbb{R}^k$ и представим модель в виде:

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

- Число параметров модели снизилось до $nk + n + 1$;

FM – Factorization Machines

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

- Сопоставим каждому признаку x_i вектор $v_i \in \mathbb{R}^k$ и представим модель в виде:

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

- Число параметров модели снизилось до $nk + n + 1$;
- Последнее слагаемое можно посчитать за $O(nk)$:

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \frac{1}{2} \left\| \sum_{i=1}^{n} v_i x_i \right\|_2^2 - \frac{1}{2} \sum_{i=1}^{n} \|v_i\|_2^2 x_i^2$$

FM – Factorization Machines

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

- Сопоставим каждому признаку x_i вектор $v_i \in \mathbb{R}^k$ и представим модель в виде:

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

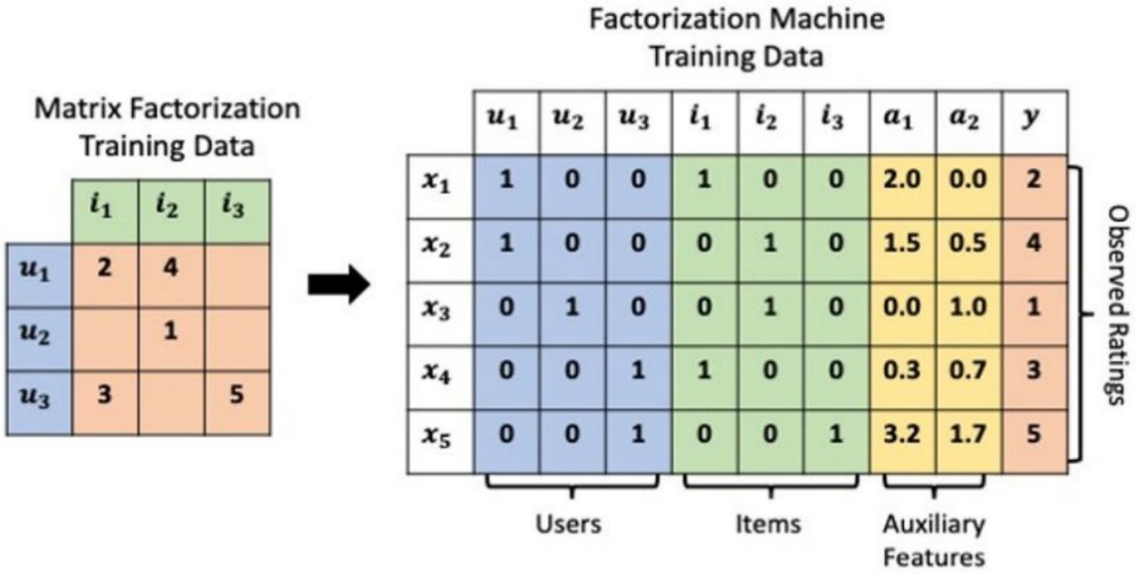
- Число параметров модели снизилось до $nk + n + 1$;
- Последнее слагаемое можно посчитать за $O(nk)$:

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \frac{1}{2} \left\| \sum_{i=1}^{n} v_i x_i \right\|_2^2 - \frac{1}{2} \sum_{i=1}^{n} \| v_i \|_2^2 x_i^2$$

- Такая модель и называется факторизационной машиной.

FM – Factorization Machines

- Помимо one-hot закодированного взаимодействия, можно добавить в вектор x контентные признаки пользователя или товара:



	i_1	i_2	i_3
u_1	2	4	
u_2		1	
u_3	3		5

	u_1	u_2	u_3	i_1	i_2	i_3	a_1	a_2	y
x_1	1	0	0	1	0	0	2.0	0.0	2
x_2	1	0	0	0	1	0	1.5	0.5	4
x_3	0	1	0	0	1	0	0.0	1.0	1
x_4	0	0	1	1	0	0	0.3	0.7	3
x_5	0	0	1	0	0	1	3.2	1.7	5

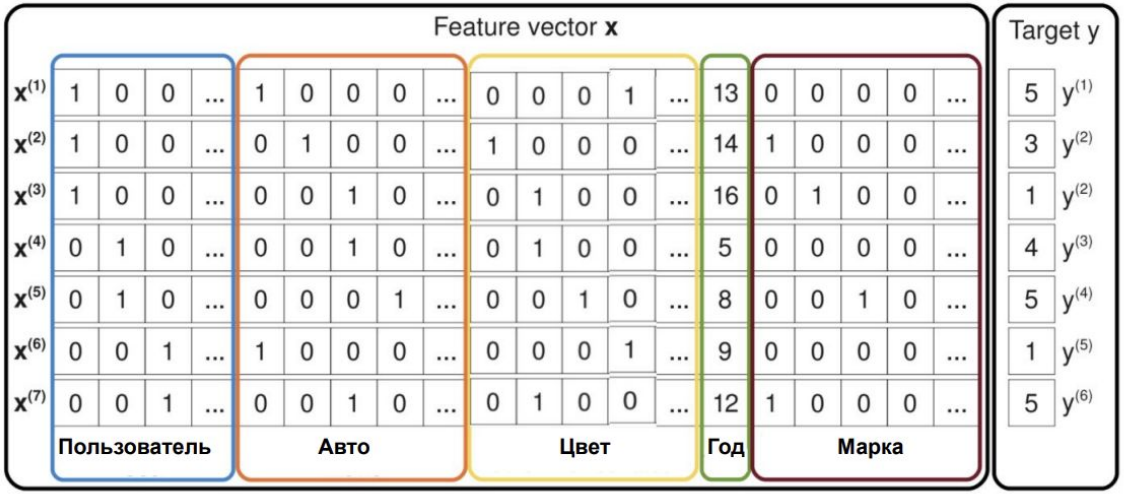
FM – Factorization Machines

$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

- Модель обучается градиентным спуском;
- Основной смысл факторизационных машин в том, что веса при парных взаимодействиях признаков факторизованы;
- Кроме того, предсказания модели можно явно считать линейно, что дает неплохое время работы алгоритма.

FFM - Field-aware Factorization Machines

- Пример: есть 3 разных по своей природе признака: год выпуска, цвет и марка автомобиля;
- В FM модели для учета взаимодействия год-цвет и год-марка используется один и тот же вектор для года;
- Но так как эти признаки разные по смыслу, то и характер их взаимодействия может отличаться;
- Идея: использовать 2 разных вектора для признака “год выпуска” при учете взаимодействий год-цвет и год-марка;



FFM - Field-aware Factorization Machines

- Разобьем признаки на группы, пусть f_i — индекс группы i-того признака;;
- Тогда модель FFM выглядит как:

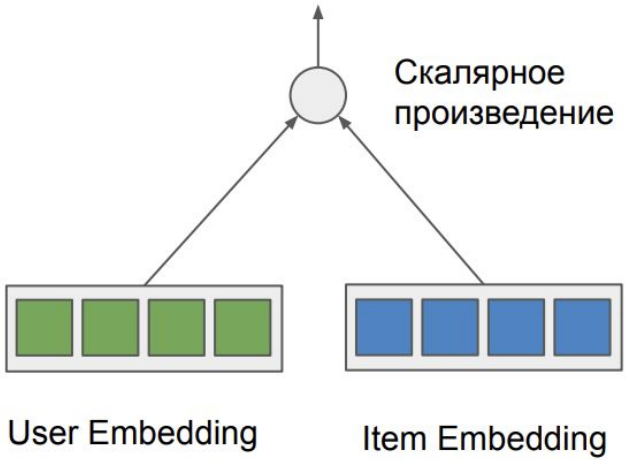
$$a(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_{i,f_j}, v_{j,f_i} \rangle x_i x_j$$

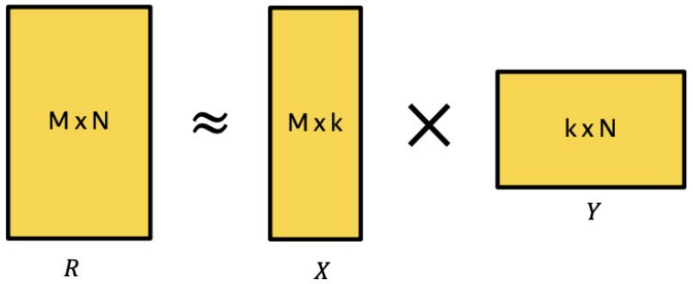
- Обучается градиентным спуском, аналогично FM;
- Аналогичным образом квадратичную сумму можно вычислять линейно по n;
- Работают лучше всего с группами вида “категориальный признак большой кардинальности”;

План лекции

- SLIM
- Factorization Machines
- Content-based рекомендации

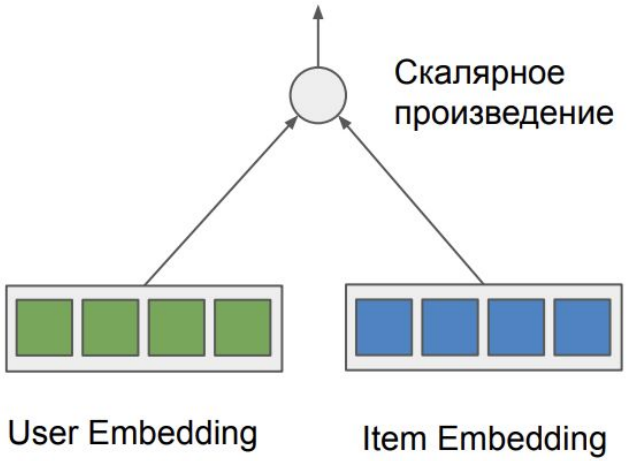
SVD своего рода нейронная сеть



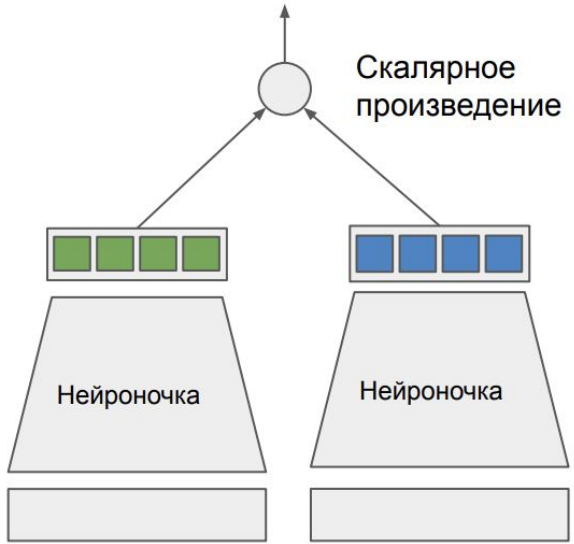


Обучаем с помощью SGD

SVD своего рода нейронная сеть

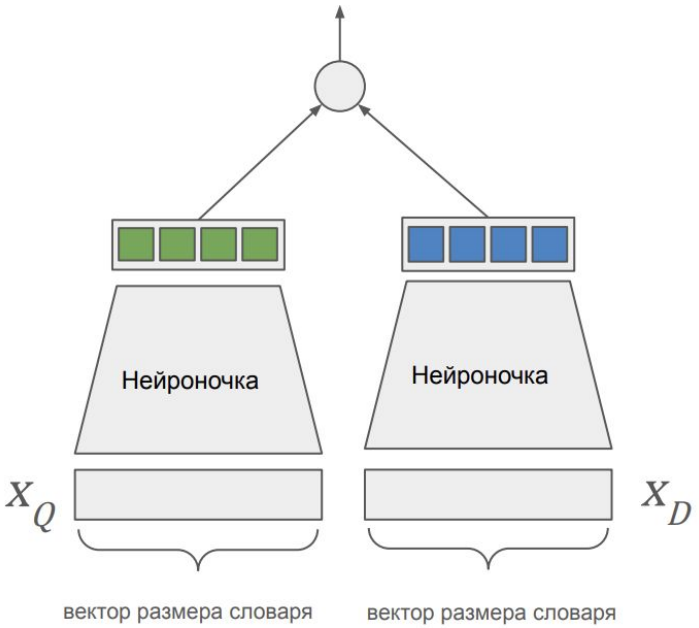


Обучаем с помощью SGD



DSSM - Deep Structured Semantic Models

$$R(Q, D) = \text{cosine}(y_Q, y_D) = \frac{{y_Q}^T y_D}{\|y_Q\| \|y_D\|}$$



DSSM – классическая модель поиска и ранжирования

- Q – текстовый запрос, D – документ
- x_Q и x_D – их представления, например, в виде bag of words (~100k)
- Запрос и документ нейросетями переводим в эмбеддинги небольшого размера (~300)
- Между ними считаем функцию близости, косинус, скалярное произведение или др.
- По значению близости ранжируем документы

Deep Structured Semantic Models: как учить?

Будем считать условную вероятность клика по документу D при условии запроса Q.

$$P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{D} \exp(\gamma R(Q,D))} \quad , \text{ где} \quad R(Q,D) = \text{cosine}(y_Q, y_D) = \frac{{y_Q}^T y_D}{\|y_Q\| \|y_D\|}$$

Здесь

γ – к-т сглаживания, устанавливается эмпирически

D – множество всех документов

Deep Structured Semantic Models: как учить?

Будем считать условную вероятность клика по документу D при условии запроса Q.

$$P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{\boldsymbol{D}} \exp(\gamma R(Q,D))} \quad , \text{ где} \quad R(Q,D) = \text{cosine}(y_Q, y_D) = \frac{{y_Q}^T y_D}{\|y_Q\| \|y_D\|}$$

Здесь

γ – к-т сглаживания, устанавливается эмпирически

D – множество всех документов

Negative sampling опции:

1. Равновероятно выбирать подмножество документов из некликнутых
2. С большей вероятностью выбирать те некликнутые документы, популярность которых выше
3. На каждой эпохе обучения выбирать некликнутые документы с максимальным скором (скор берётся с предыдущей эпохи)

Deep Structured Semantic Models: как учить?

С учётом negative sampling вероятность клика в документ описывается формулой

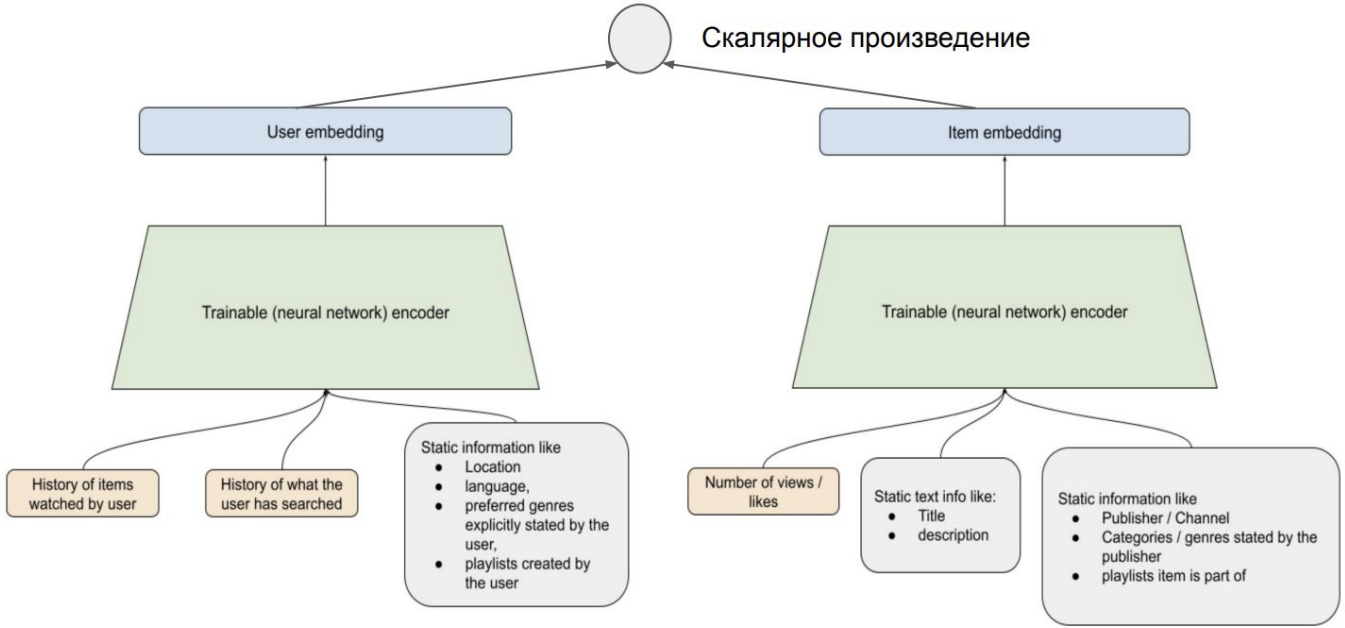
$$P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\exp(\gamma R(Q,D)) + \sum_{d \in \boldsymbol{D}^-} \exp(\gamma R(Q,d))}$$

В процессе обучения будем максимизировать правдоподобие выборки, или, что то же самое, минимизировать лосс:

$$L(\Lambda) = -\log \prod_{(Q,D^+)} P(D^+|Q) \qquad L(\Lambda) = -\log \prod_{(Q,d \in \mathbf{D}^+)} P(d|Q)$$

где Λ – параметры слоёв
$\boldsymbol{D}^+$ – множество кликнутых документов
$\boldsymbol{D}^-$ – множество negative sampling документов

Two tower

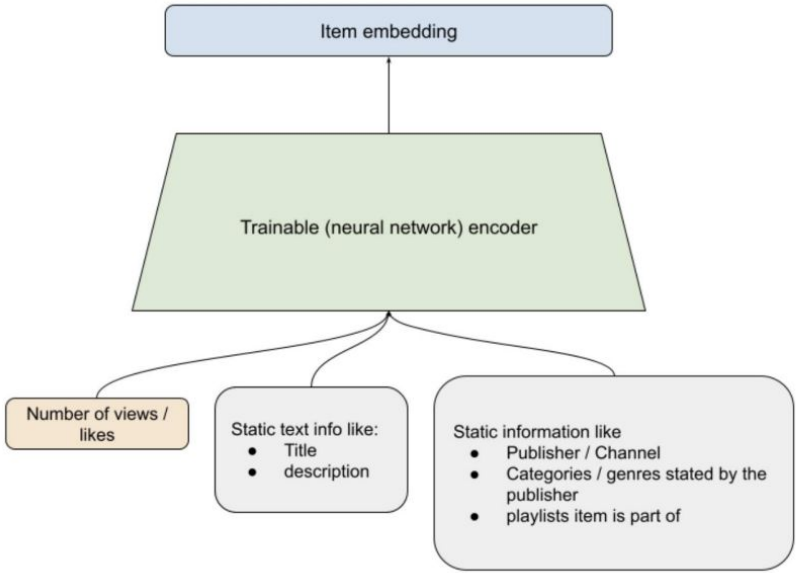


Поиск релевантных айтемов можно представить как задачу ранжирования, где в качестве запроса – пользователь, его история и фичи.

Two tower: Признаки айтема

В качестве признаков можно использовать:

1. стандартные статистики документа: количество лайков, кликов, подписок
2. признаки автора: количество подписчиков, жанр
3. неструктурированные данные: текст документа (можно использовать BOW формат, а можно воспользоваться предобученными эмбедиингами), видео и картинки (также предварительно представить их в виде эмбеддинга)

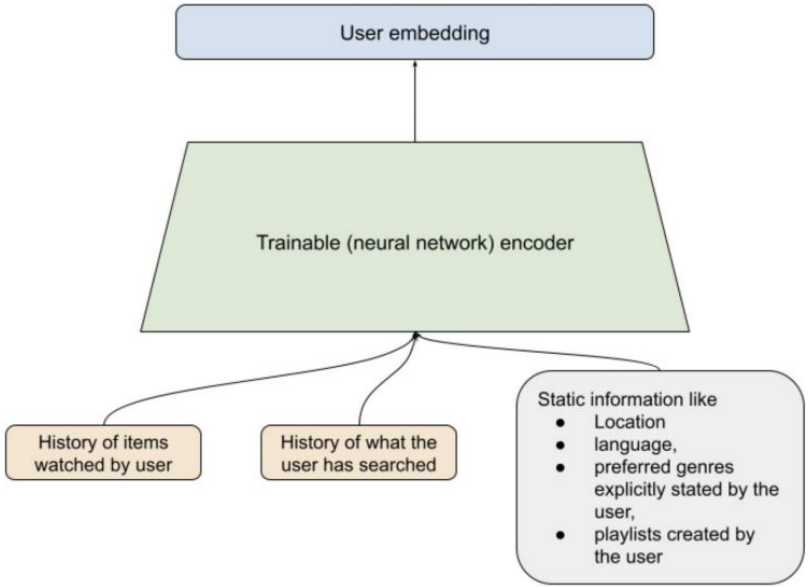


Two tower: Признаки пользователя

В качестве признаков можно использовать:

1. информацию про пользователя: возраст, пол, язык, насколько долго пользуется сервисом
2. информацию про контекст запроса: с какого устройства был сделан, в какое время
3. информацию про друзей/подписчиков пользователя и их взаимодействия

Имеет смысл использовать историю пользователя как усреднённый эмбеддинг тех статей, которые он читал. Или обучить RNN или Transformer на истории и результат конкатенировать к остальным фичам.



Two tower: виды лоссов

1. Cross Entropy Loss (CE)
2. Pairwise loss
3. Full Product Softmax loss (он же: InfoNCE, InfoMAX, SIMCLR)

Two-tower neural network: cross entropy loss

Вероятность того, что пользователь u кликнет на айтем i можно представить в виде:

$$\widehat{p}_{ui} = \sigma(R(u,i)) = \sigma(\mathrm{dot}(y_u, y_i))$$

где

y_i – эмбеддинг айтема

y_u – эмбеддинг пользователя

$\sigma(x) = \frac{1}{1+e^{-x}}$

Тогда лосс имеет вид ($r_{u,i} \in \{0,1\}$ – рейтинг):

$$L = -\sum_{u,i}(r_{ui} \log \widehat{p}_{u,i} + (1 - r_{u,i}) \log(1 - \widehat{p}_{u,i}))$$

Two-tower neural network: pairwise loss

Рассмотрим пару айтемов, в которой i_1 - положительный, i_2 - отрицательный, есть несколько вариантов попарного лосса:

a. $L(R(u, i_1), R(u, i_2)) = \text{CrossEntropy}(1.0,\ \sigma(R(u, i_1) - R(u, i_2)))$

сеть учится ранжировать положительные примеры выше отрицательных

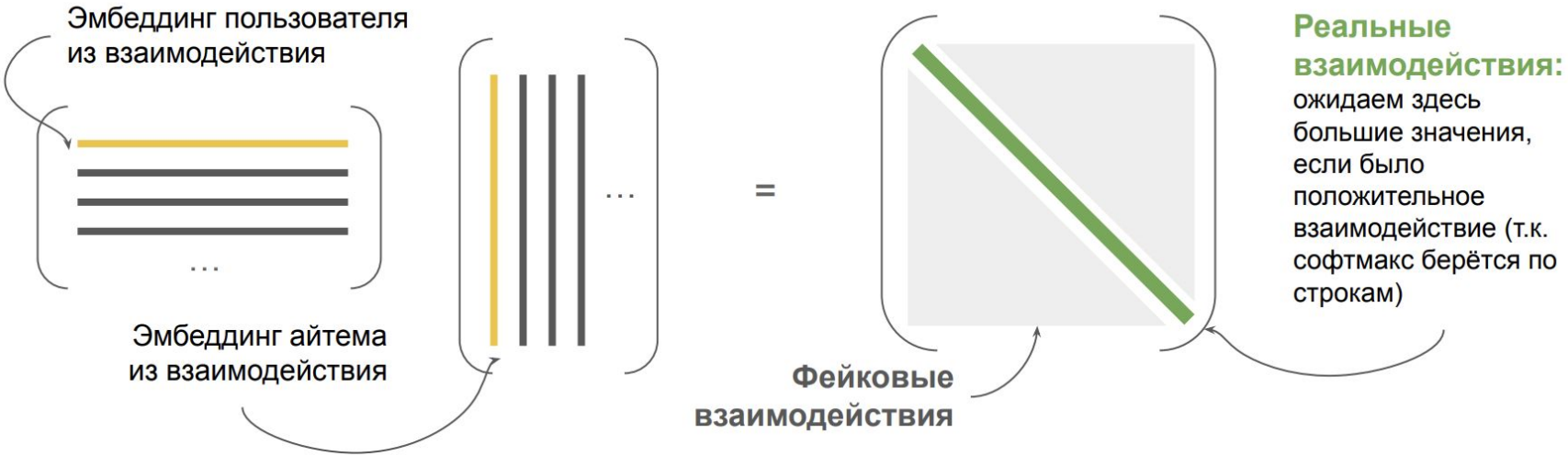
b. $L(R(u, i_1), R(u, i_2)) = \max(0, \alpha - R(u, i_1) + R(u, i_2))$

сеть делает так, чтобы положительный и отрицательный примеры как можно больше отличались (известен как triplet loss, который используется для обучения сиамских сетей)

Full Product Softmax loss

Рассмотрим батч взаимодействий размера m, состоящий из
- Матрицы эмбедингов пользователей $U \in \mathbb{R}^{m \times d}$
- Матрицы эмбедингов айтемов $I \in \mathbb{R}^{m \times d}$
- Вектора таргетов $r \in \mathbb{R}^m$

Рассмотрим матрицу $\mathrm{Softmax}(\alpha \cdot UI^T + \beta)$, $UI^T \in \mathbb{R}^{m \times m}$, где софтмакс берётся по строкам.



Full Product Softmax loss

Рассмотрим лосс вида

$$L = -(r > 0)^T \cdot \log(\mathrm{diag}(\mathrm{softmax}(\alpha \cdot UI^T + \beta)))$$

Лосс делает так, чтобы диагональные элементы матрицы были больше остальных элементов: так в датасете с уникальными пользователями и документами на диагонали оптимальной матрицы будут стоять $r > 0$