Яндекс

Нейросетевой отбор кандидатов

ШАД, курс по рекомендательным системам, весна 2025

Кирилл Хрыльченко

Структура лекции

1 | Введение

2 | Двухбашенные модели

3 | На что учить генерацию кандидатов

4 | Функции похожести

5 | Согласованность с ранжированием

6 | Айтемные башни

7 | Пользовательские башни

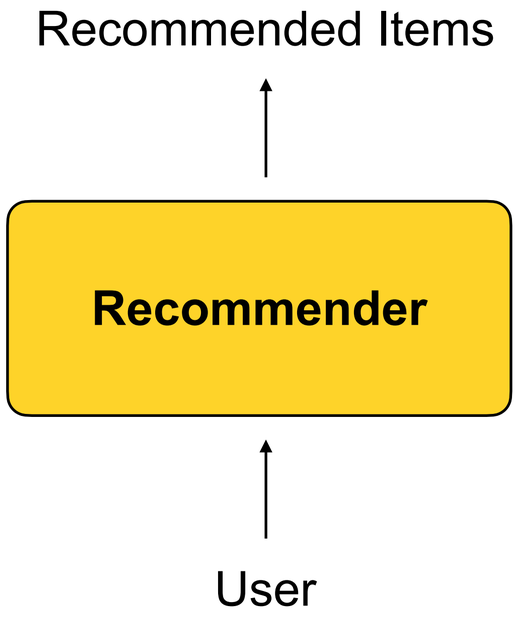
1

Введение

Идеальная рексистема

Три свойства идеальной рексистемы:

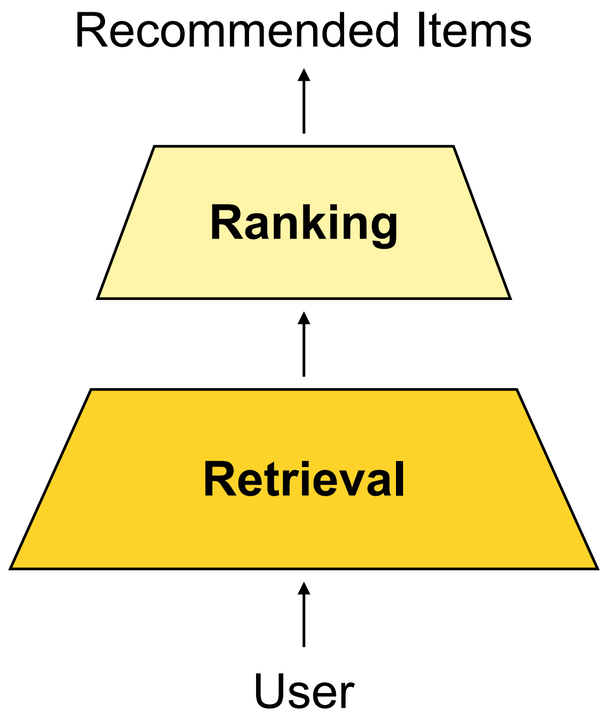
- **Одностадийность** – модель сразу оценивает все айтемы без этапа отбора кандидатов.
- **Максимальная выразительность** – архитектура модели настолько мощная, что может выразить любую зависимость.
- **Учится на свой фидбек** – на то, что сама показала пользователю, то есть impression-aware.



Идеальная рексистема

На практике приходится отказываться от всех трёх свойств:

- **Сложные модели ❌ :** ограничены по времени и ресурсам, нужны простые быстрые модели
- **Многостадийность 💀** : добавление второй стадии с более сложными моделями растит качество
- **Обучение на фидбек 👽**: Прямая обратная связь есть только для верхней стадии, для остальных нужно выдумывать эвристики и суррогатные лоссы



Генерация кандидатов

Генерация кандидатов (она же retrieval) – нижняя стадия. Примеры классических методов:

- **Топ популярного:** нет персонализации, плох для новых айтемов
- **История пользователя:** имеет смысл не на всех доменах, нет discovery
- **Item2item:** меньше персонализации и discovery, может быть плох для новых айтемов; не контекстуален
- **User2item**: плох и для новых айтемов, и для новых пользователей; не контекстуален
- **ALS**: тоже плох для новых айтемов и пользователей; не контекстуален

… нейросети могут все: персонализация, индуктивность, контекстуальность, рост базового качества.

2

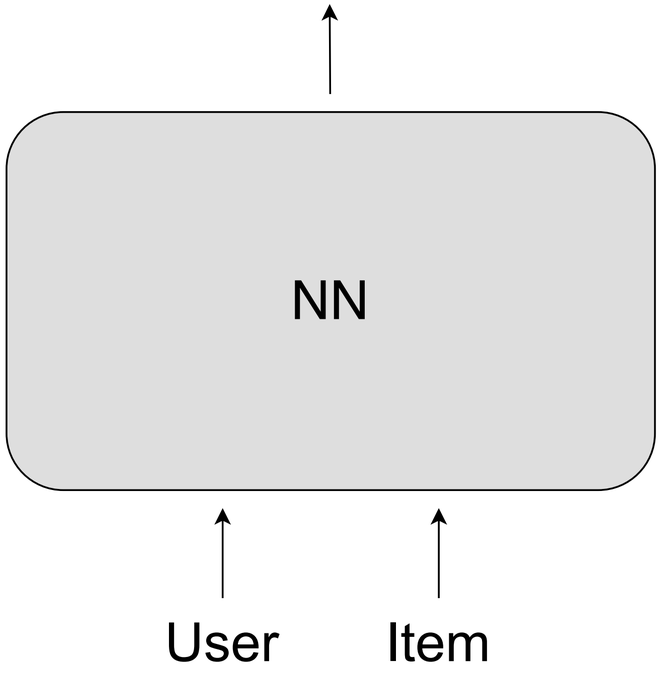
Двухбашенные модели

Почему не ранжирующие нейросети?

В ранжировании:

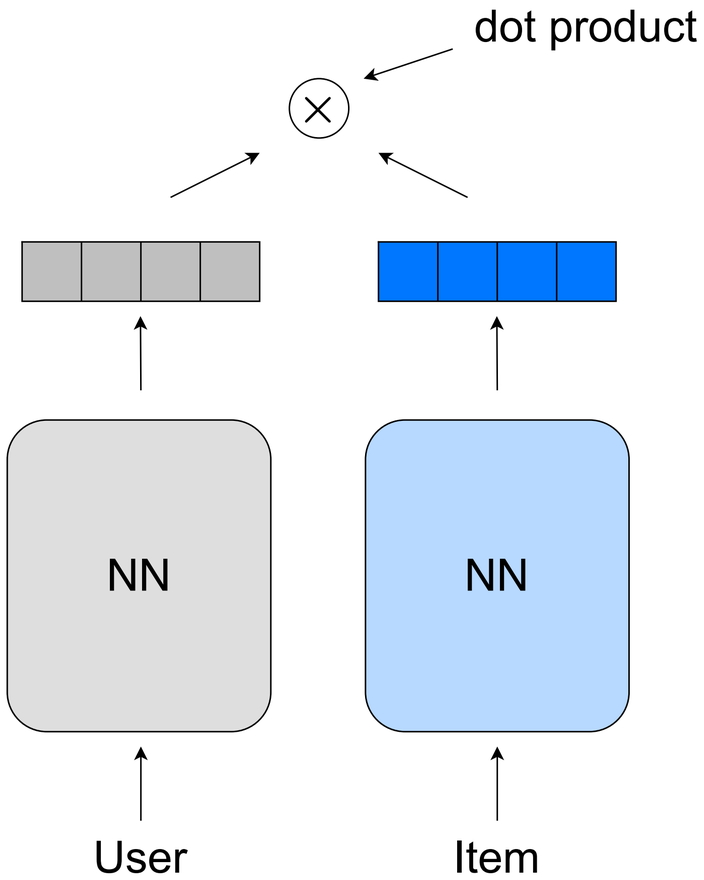
- **Раннее связывание** информации про пользователя и айтем
- Нужно повторно прогонять модель для каждого айтема из каталога
- При больших каталогах это невозможно сделать за разумное время

… нужно сделать более эффективную модель – факторизовать вычисления по пользователю и айтему.



Двухбашенные нейросети

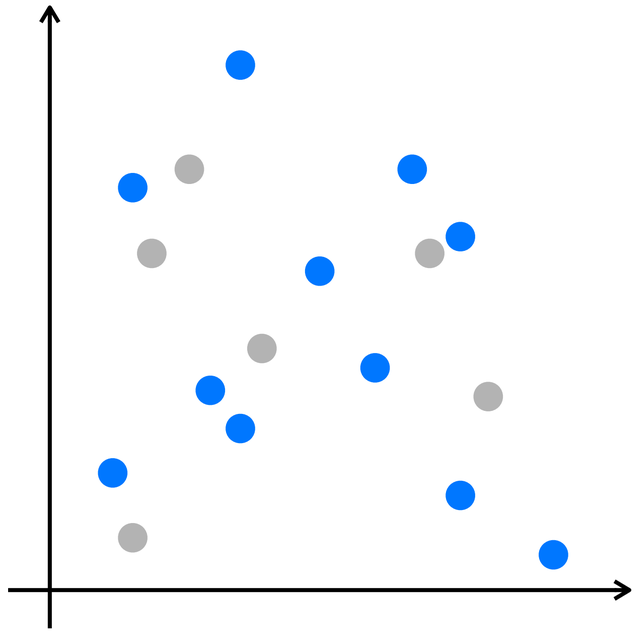
- Строим для пользователей и айтемов **эмбеддинги** с помощью отдельных нейросетевых **башен**
- Делаем **позднее связывание** этих эмбеддингов через простую функцию похожести – скалярное произведение: $f_\theta(u, i) = \langle v_u, v_i \rangle$
- YoutubeDNN[1] – <пользователь, айтем>
- DSSM[2] – <запрос, документ>



1 Deep Neural Networks for YouTube Recommendations
2 Learning deep structured semantic models for web search using clickthrough data

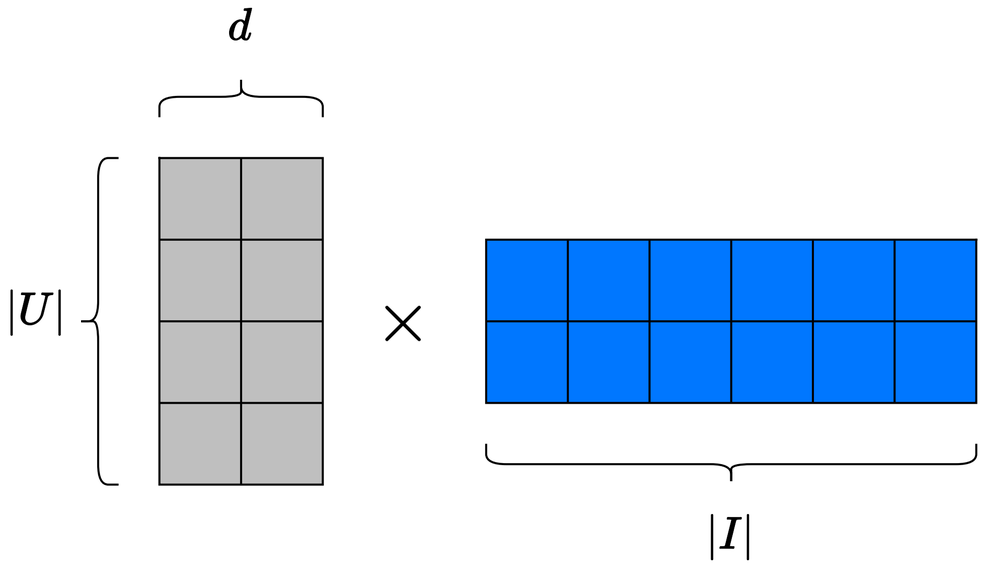
Семантическое пространство

- Обучаем **семантическое пространство** пользователей и айтемов, в котором близость объектов отражает их похожесть
- **MIPS (maximum inner product search):** Умеем быстро искать для пользователя айтемы с большим скалярным произведением
- Векторы айтемов не нужно перевычислять для каждого пользователя (можем закэшировать)
- Векторы пользователей считаем один раз на запрос (или даже тоже кэшируем)



Двухбашенная модель как низкоранговое разложение

- Пусть есть матрица релевантности $R \in \mathbb{R}^{|U| \times |I|}$, определяющая интерес пользователей к айтемам
- Двухбашенная модель учит низкоранговое разложение $R = V_U V_I^T$, где $V_U \in \mathbb{R}^{|U| \times d}$, $V_I \in \mathbb{R}^{|I| \times d}$
- Размерность эмбеддинга d – информационный боттлнек; ранг разложения
- Чем выше d, тем больше ранг аппроксимации



Двухбашенная модель как низкоранговое разложение

Достаточно широкий эмбеддинг позволит выучить любую функцию похожести (матрицу релевантности):

- Вектор пользователя – one hot представление $v_u = (0,\dots 1,\dots 0) \in \mathbb{R}^{|U|}$
- Вектор айтема – строка из R, соответствующая айтему: $v_i = (R_{iu})_{u=1}^{|U|}$
- Тогда $score(u,i) = \langle v_u, v_i \rangle = R_{iu}$

Но на практике не можем позволить себе слишком большие размерности из-за ограничений по памяти и скорости, используем сотни.

3

На что учить генерацию кандидатов

Напоминание: лоссы ранжирования

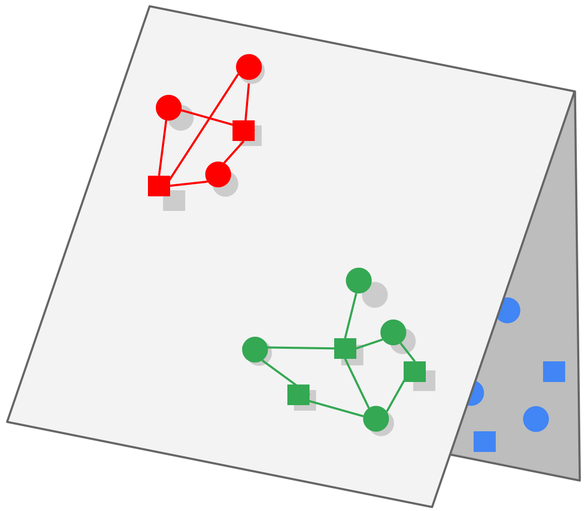
В лекции про нейросетевое ранжирование лоссы не обсуждались: все также, как для бустингов:

- **Impression-aware:** учимся на своем фидбеке, то есть на показанных выдачах
- **Pointwise training:** поточечная кросс-энтропия (будет клик на айтем или нет)
  $-y \log \sigma(x) - (1 - y) \log \left(1 - \sigma(x)\right)$
- **Pairwise training:** попарное ранжирование (вероятность правильно упорядочить пару показанных айтемов с т.з. положительного сигнала только на одного из них)
  $-\log \sigma(x_1 - x_2)$, где $y_1 > y_2$
- Есть всякие модификации (а еще и listwise лоссы)

Folding

Ранжирующий impression-aware лосс для кандгена подходит плохо:

- Если рексистема достаточно хороша, то в выдачах не будет случайных айтемов
- Возникают непересекающиеся user-item группы (e.g., испанцам – испанские видео, итальянцам – итальянские)
- Ранжирующий лосс учит делать локальные сравнения в рамках выдач, то есть ранжирование в рамках этих непересекающихся групп
- Возникает **folding** – разные группы могут “накладываться” (англ. “to fold”) друг на друга в векторном пространстве



Из Google for Developers.Recommendation Systems

При применении нужно уметь сравнивать айтемы из всего каталога, не только обычный “фолд”, соответствующий пользователю. Нужна способность **глобального сравнения!**

Folding: Why Good Models Sometimes Make Spurious Recommendations

Softmax Model

Перейдем от $P(\text{engagement} \mid \text{user, item}) \rightarrow P(\text{item} \mid \text{user, engagement})$

Это задача экстремальной классификации, в которой классы – это айтемы:

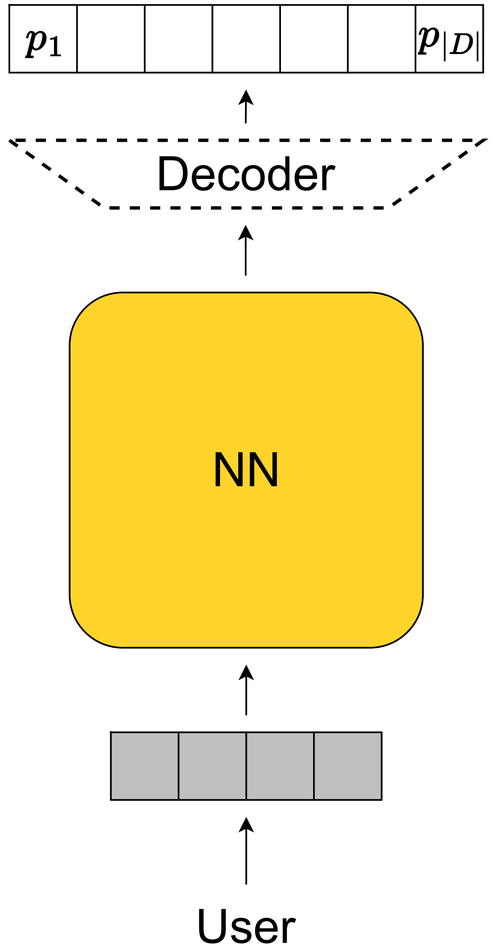
- Входной объект – пользователь / запрос
- На выходе – вероятностное распределение по всем айтемам (вероятность провзаимодействовать с тем или иным айтемом)

Для моделирования вероятностей используем softmax:

$$P(d \mid u) = \frac{e^{f_\theta(d,u)}}{\sum_{d' \in D} e^{f_\theta(d',u)}}, \text{ где } f_\theta(d,u) \text{ – функция похожести.}$$

В качестве лосса используем кросс-энтропию. Для пары $u \in U, p \in D$:

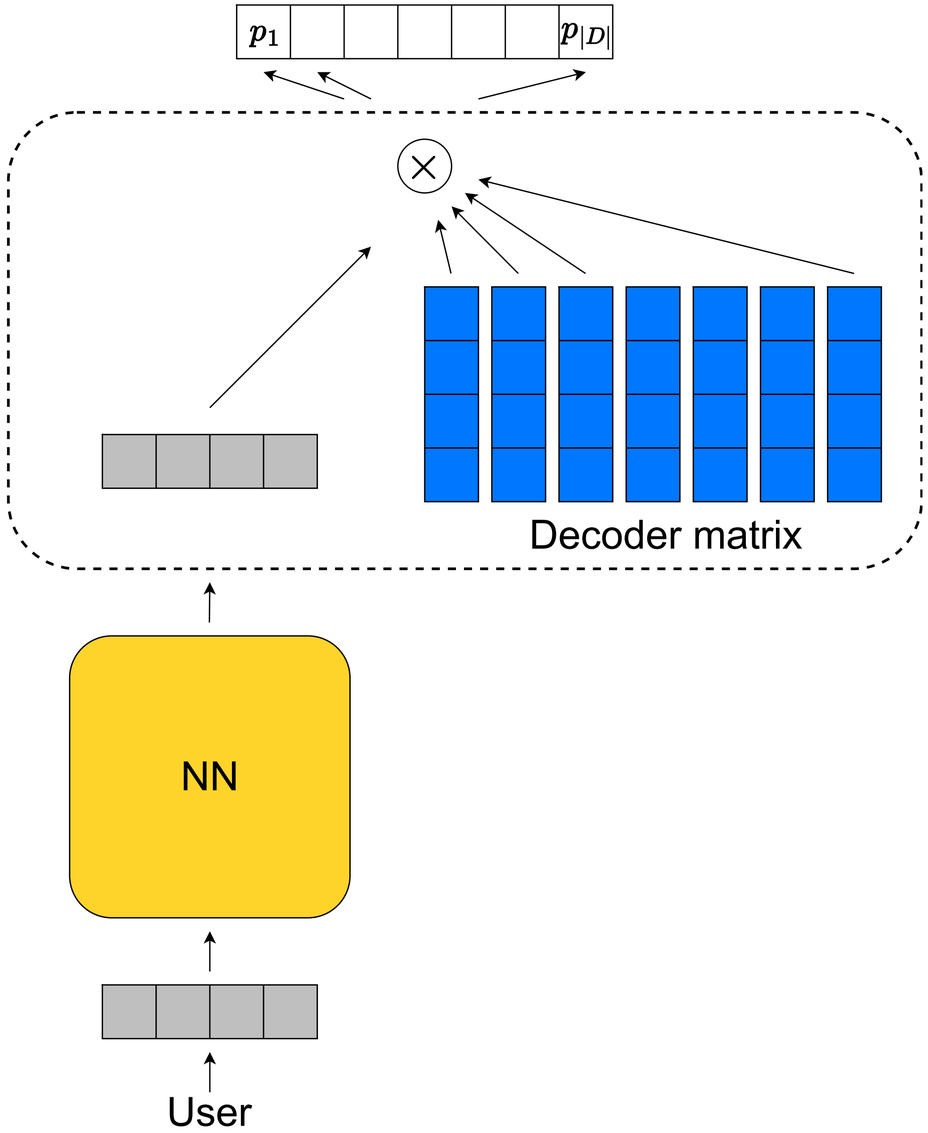
$$L(u,p) = -\log \frac{e^{f_\theta(u,p)}}{\sum_{d' \in D} e^{f_\theta(u,d')}}$$



Softmax Model

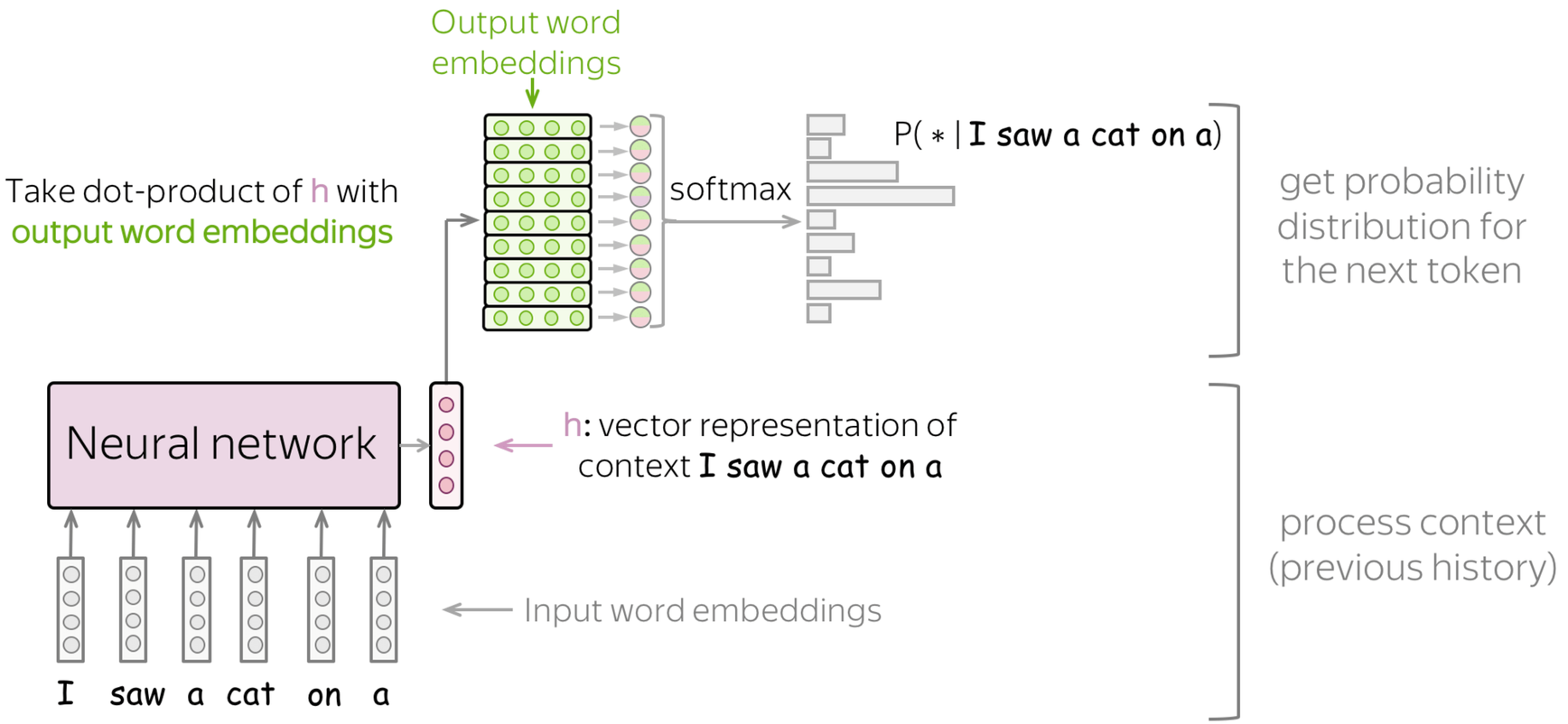
Это двухбашенная модель!

- Распределение получаем с помощью умножения на **декодер** матрицу V: $p = \text{softmax}(f(u)^T V)$, то есть $f_\theta(i, u) = \langle f(u), v_i \rangle$
- Декодер – матрица эмбеддингов всех айтемов!
- Softmax модель – это двухбашенная модель; левая часть – пользователь, правая – айтем
- Айтемы кодируются через обучаемый эмбеддинг



LLM as Two-Tower Model

LLM – это softmax модель: левая часть – уже обработанное предложение, правая – следующий токен.



Из Lena Voita's YSDA NLP Course

Проблема полного софтмакса

- Вспомним формулу полного софтмакса:

$$L_{\text{softmax}}(u, p) = -\log \frac{e^{f_\theta(u,p)}}{\sum_{d' \in D} e^{f_\theta(u,d')}}$$

- Важное отличие рекомендательных систем от NLP – уникальных токенов в NLP мало (десятки-сотни тысяч), а айтемов в рекомендательных системах – очень много (вплоть до миллиардов).

- Не можем считать “полный” софтмакс из-за суммы по всему каталогу в знаменателе софтмакса

… будем сэмплировать!

Sampled Softmax

Было:

$$L_{\text{softmax}}(u, p) = -\log \frac{e^{f_\theta(u,p)}}{\sum_{d' \in D} e^{f_\theta(u,d')}}$$

Стало – **sampled softmax loss**:

$$L_{\text{sampled}}(u, p) = -\log \frac{e^{f_\theta(u,p)}}{e^{f_\theta(u,p)} + \sum_{i=1}^{n} e^{f_\theta(u,d_i)}}, \qquad d_i \sim Q(d),$$

где $u \in U, p \in D$ – позитив, $\{d_i\}_{i=1}^{n}$ – сэмплированные негативы.

Negative Sampling

> Полный софтмакс “майнит” хард негативы по всему каталогу – вклад в градиент больше у айтемов, в которых модель ошибается

- Если не можем взять все айтемы, надо постараться взять те, у которых был бы большой вклад в градиент

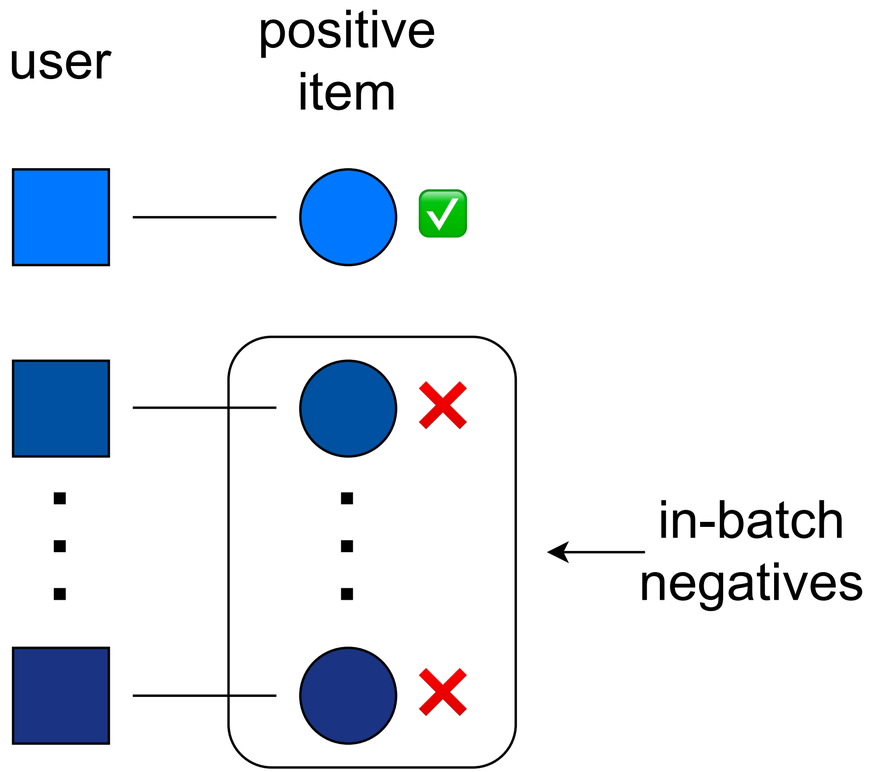
Uniform sampling – равномерное сэмплирование по каталогу, $d \sim \text{Uniform}(D)$:

- очень легкие негативы; дают небольшой вклад в градиент
- нужно большое количество, чтобы хоть что-то выучить

In-batch Sampling

Пусть в батч попали n пар (user, positive item). Для каждой пары будем использовать чужие позитивы в качестве своих негативов:

- Айтемы в батче распределены не равномерно, а **униграммно** (пропорционально популярности)
- Более сложные негативы, чем юниформные
- Можно собирать негативы с других видеокарточек (**cross-device**) и с прошлых батчей (**memory queue**[1,2])
- Sampled softmax с in-batch негативами – популярная техника для self-supervised learning[3]



1 Momentum Contrast for Unsupervised Visual Representation Learning

2 Cross-Batch Negative Sampling for Training Two-Tower Recommenders

3 A Simple Framework for Contrastive Learning of Visual Representations

LogQ Коррекция

Из-за униграммного распределения, модель в качестве негативов чаще видит популярные айтемы и учится их штрафовать за популярность.

Решение – **logQ-коррекция**[1]:

$$L_{\text{logQ}}(u, p) = -\log \frac{e^{f_\theta(u,p) - \log Q(p)}}{e^{f_\theta(u,p) - \log Q(p)} + \sum_{i=1}^{n} e^{f_\theta(u,d_i) - \log Q(d_i)}}.$$

- $Q(d)$ – частота айтема в данных (его популярность)
- Корректируем на величину штрафа. Модели не нужно штрафовать популярные объекты, мы уже это сделали.

1 Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations

Importance sampling

Importance Sampling:

$$\mathbb{E}_{P(x)}[f(x)] = \int_x P(x)f(x)dx = \int_x Q(x)\frac{P(x)}{Q(x)}f(x)dx = \mathbb{E}_{Q(x)}\left[\frac{P(x)}{Q(x)}f(x)\right] \approx$$

$$\approx \{\text{Monte-Carlo sampling}\} \approx \frac{1}{n}\sum_{i=1}^{n}\frac{P(x_i)}{Q(x_i)}f(x_i), \quad x_i \sim Q(x)$$

Как с помощью него оценить знаменатель софтмакса:

$$\sum_{d\in D} e^{f_\theta(u,d)} = |D|\,\mathbb{E}_{d\sim \text{Unif}(D)}\left[e^{f_\theta(u,d)}\right] = \{\text{Importance Weighting}\} =$$

$$= |D|\,\mathbb{E}_{d\sim Q(d)}\left[\frac{1/|D|}{Q(d)}e^{f_\theta(u,d)}\right] = \mathbb{E}_{d\sim Q(d)}\left[e^{f_\theta(u,d)-\log Q(d)}\right] \approx$$

$$\approx \{\text{Monte-Carlo sampling}\} \approx \frac{1}{n}\sum_{i=1}^{n} e^{f_\theta(u,d_i)-\log Q(d_i)}, \quad d_i \sim Q(d)\,.$$

Importance sampling

Пример. Для равномерного распределения Q(d):

$$\sum_{d \in D} e^{f_\theta(u,d)} = \frac{1}{n} \sum_{i=1}^{n} e^{f_\theta(u,d_i) - \log Q(d_i)} = \{Q(d) = \text{Unif}(d)\} =$$

$$= \frac{1}{n} \sum_{i=1}^{n} e^{f_\theta(u,d_i) - \log \frac{1}{|D|}} = |D| \cdot \frac{1}{n} \sum_{i=1}^{n} e^{f_\theta(u,d_i)}.$$

Вывод logQ-коррекции

Не можем посчитать градиент для полного softmax loss'а:

$$\nabla_\theta L_{\text{softmax}}(u, p).$$

Хотим такой L_{sampled}, чтобы была максимально хорошая оценка для полного градиента:

$$\nabla_\theta L_{\text{sampled}}\left(u, p, \{d_i\}_{i=1}^n\right), \quad d_i \sim Q(d).$$

Например, **несмещенность**:

$$\mathbb{E}_{d_i \sim Q(d)}\left[\nabla_\theta L_{\text{sampled}}\right] = \nabla_\theta L_{\text{softmax}}\,.$$

… или хотя бы состоятельность (мат.ожидание стремится к истинному градиенту).

Вывод logQ-коррекции

1. Переход к мат. ожиданию:

$$\nabla_\theta L_{\text{softmax}}(u,p) = \nabla_\theta \left[-\log \frac{e^{f_\theta(u,p)}}{\sum_{d\in D} e^{f_\theta(u,d)}} \right] = \nabla_\theta \left[-f_\theta(u,p) + \log \sum_{d\in D} e^{f_\theta(u,d)} \right] =$$

$$= -\nabla_\theta f_\theta(u,p) + \frac{1}{\sum_{d\in D} e^{f_\theta(u,d)}} \sum_{d\in D} e^{f_\theta(u,d)} \nabla_\theta f_\theta(u,d) = -\nabla_\theta f_\theta(u,p) + \sum_{d\in D} \frac{e^{f_\theta(u,d)}}{\sum_{d'\in D} e^{f_\theta(u,d')}} \nabla_\theta f_\theta(u,d) =$$

$$= -\nabla_\theta f_\theta(u,p) + \sum_{d\in D} \mathrm{P}_\theta(d \mid u) \nabla_\theta f_\theta(u,d) = -\nabla_\theta f_\theta(u,p) + \mathbb{E}_{d\sim \mathrm{P}_\theta(d|u)} \left[\nabla_\theta f_\theta(u,d) \right].$$

2. Importance sampling для получения несмещенной оценки

$$= -\nabla_\theta f_\theta(u,p) + \mathbb{E}_{d\sim Q(d)} \left[\frac{\mathrm{P}_\theta(d \mid u)}{Q(d)} \nabla_\theta f_\theta(u,d) \right] \approx \{ \text{ Monte-Carlo Sampling} \} \approx$$

$$\approx -\nabla_\theta f_\theta(u,p) + \frac{1}{n} \sum_{i=1}^{n} \frac{\mathrm{P}_\theta\left(d_i \mid u\right)}{Q\left(d_i\right)} \nabla_\theta f_\theta\left(u, d_i\right) = -\nabla_\theta f_\theta(u,p) + \frac{1}{n} \sum_{i=1}^{n} \frac{e^{f_\theta\left(u, d_i\right) - \log Q\left(d_i\right)}}{\sum_{d'\in D} e^{f_\theta\left(u, d'\right)}} \nabla_\theta f_\theta\left(u, d_i\right).$$

Вывод logQ-коррекции

3. Importance sampling для знаменателя (**важно**: оценка перестает быть несмещенной):

$$-\nabla_\theta f_\theta(u,p) + \frac{1}{n}\sum_{i=1}^{n} \frac{e^{f_\theta(u,d_i)-\log Q(d_i)}}{\sum_{d'\in D} e^{f_\theta(u,d')}} \nabla_\theta f_\theta\left(u,d_i\right) \approx -\nabla_\theta f_\theta(u,p) + \frac{1}{n}\sum_{i=1}^{n} \frac{e^{f_\theta(u,d_i)-\log Q(d_i)}}{\frac{1}{n}\sum_{k=1}^{n} e^{f_\theta(u,d_k)-\log Q(d_k)}} \nabla_\theta f_\theta\left(u,d_i\right) =$$

$$= -\nabla_\theta f_\theta(u,p) + \sum_{i=1}^{n} \frac{e^{f_\theta(u,d_i)-\log Q(d_i)}}{\sum_{k=1}^{n} e^{f_\theta(u,d_k)-\log Q(d_k)}} \nabla_\theta f_\theta\left(u,d_i\right).$$

4. Нужный градиент получается при следующей функции потерь:

$$L_{\text{logQ}}(u,p) = -\log \frac{e^{f_\theta(u,p)}}{\sum_{i=1}^{n} e^{f_\theta(u,d_i)-\log Q(d_i)}}, \quad d_i \sim Q(d).$$

Альтернативный вывод logQ-коррекции

Weighted importance sampling. Если $P(x) = \frac{h(x)}{Z}$, где $Z = \int_x h(x)dx$, то:

$$\mathbb{E}_{P(x)} f(x) \approx \frac{1}{n} \sum_{i=1}^{n} \frac{P(x_i)}{Q(x_i)} f(x_i) = \frac{1}{n} \sum_{i=1}^{n} w_i f(x_i), \text{ где } w_i = \frac{h(x_i)/Q(x_i)}{\sum_{j=1}^{n} h(x_j)/Q(x_j)}.$$

Аппроксимация градиента softmax loss с помощью weighted importance sampling:

$$\nabla_\theta L_{\text{softmax}}(u, p) = -\nabla_\theta f_\theta(u, p) + \mathbb{E}_{\mathrm{P}_\theta(d|u)} \nabla_\theta f_\theta(u, d) \approx \left\{ h(d) = e^{f_\theta(u,d)} \Rightarrow w_i = \frac{e^{f_\theta(u, d_i) - \log Q(d_i)}}{\sum_{k=1}^{n} e^{f_\theta(u, d_k) - \log Q(d_k)}} \right\} =$$

$$= -\nabla_\theta f_\theta(u, p) + \sum_{i=1}^{n} \frac{e^{f_\theta(u, d_i) - \log Q(d_i)}}{\sum_{k=1}^{n} e^{f_\theta(u, d_k) - \log Q(d_k)}} \nabla_\theta f_\theta\left(u, d_i\right).$$

LogQ Коррекция

Обычно еще добавляют позитив в знаменатель (это очень важно для качества):

$$L_{\text{logQ2}}(u,p) = -\log \frac{e^{f_\theta(u,p)}}{\sum_{i=1}^{n} e^{f_\theta(u,d_i)-\log Q(d_i)}} \quad \Rightarrow \quad -\log \frac{e^{f_\theta(u,p)-\log Q(p)}}{e^{f_\theta(u,p)-\log Q(p)} + \sum_{i=1}^{n} e^{f_\theta(u,d_i)-\log Q(d_i)}}.$$

И убирают для него logQ-коррекцию:

$$L_{\text{logQ3}}(u,p) = -\log \frac{e^{f_\theta(u,p)}}{e^{f_\theta(u,p)} + \sum_{i=1}^{n} e^{f_\theta(u,d_i)-\log Q(d_i)}}.$$

Коррекция для позитива в числителе на градиент не влияет, а вот ссылки на честный вывод для знаменателя нет.

LogQ Коррекция

- Коэффициенты для logQ можно преподсчитать по всему датасету, либо использовать count-min sketch для подсчета на лету
- LogQ-коррекцию можно делать не на обучении, а на применении (но работает хуже и нет теоретического обоснования)
- На практике используем **mixed negative sampling**[1] – смешиваем два источника негативов, равномерные и in-batch

1 Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations

Альтернативные лоссы

Почему все-таки softmax (cross-entropy) loss:

- **Softmax loss** – “какой айтем среди всего каталога лучший для конкретного пользователя?”. Учит персонализированное глобальное ранжирование.
- **BCE** – “этот айтем хорош”? Учит сравнивать все пары (user, item) между собой, нам это не нужно.
- **BPR** – “кто среди этих двух айтемов лучше”? Хуже “майнит” негативы, так как приходится усреднять лосс по негативам. Вклад каждого негатива ограничен (out of scope для лекции).
- У нас на практике softmax для генерации кандидатов работает лучше.
- Есть статьи в академии с похожими выводами.[1]

1 BPR: Bayesian Personalized Ranking from Implicit Feedback

2 On the Effectiveness of Sampled Softmax Loss for Item Recommendation

4

Функции похожести

Косинус

Функции близости в двухбашенных моделях:

- $\langle u, i \rangle = \cos(u, i)\|u\|\|i\|$
- $\cos(u, i) = \left\langle \frac{u}{\|u\|}, \frac{i}{\|i\|} \right\rangle$, то есть скалярное произведение двух l2-нормализованных векторов

Если использовать скалярное произведение, нормы векторов выучивают популярности:

$$\nabla_{\|p\|} L_{\text{softmax}}(u, p) = \nabla_{\|p\|} \left[-\log \frac{e^{\langle u,p \rangle}}{\sum_{d \in D} e^{\langle u,d \rangle}} \right] = \nabla_{\|p\|} \left[-\cos(u, p)\|u\|\|p\| + \log \sum_{d \in D} e^{\langle u,d \rangle} \right] =$$

$$= -\cos(u, p)\|u\| + P_\theta(p \mid u) \cdot \cos(u, p)\|u\| = -\cos(u, p)\left[1 - P_\theta(p \mid u)\right].$$

Косинус

Шаг спуска:

$$\|p\|_{t+1} = \|p\|_t + \alpha \cos(u, p)\left[1 - P_\theta(p \mid u)\right].$$

- Если модель правильно предсказывает угол ($\cos(u, p) > 0$), она растит свою уверенность в айтеме $\|p\|$.
- У популярных айтемов по определению почти со всеми правильный угол.
- Для популярных айтемов чаще делаем шаги спуска.

Косинус убирает этот эффект, и в целом учится стабильней.

Температура

При использовании косинуса, значения функции близости ограничены от -1 до 1:

- Если у нас n негативов в софтмаксе, то при идеальных скорах получим вероятность

$$P(p \mid u) = \frac{e^1}{e^1 + (n-1)e^{-1}} = \{\text{В случае } n = 100\} \approx 0.07$$

Чтобы это исправить, добавляется **температура**[1] τ:

$$f(u, i) = \frac{\cos(u, i)}{\tau}.$$

- При использовании фиксированной температуры важно ее правильно подобрать
- Альтернатива – **обучаемая** температура

1 – в исходном распределении Больцмана, использовавшем софтмакс, тоже есть температура.

Температура

При использовании обучаемой температуры:

- $\tau \to 0$ – модель очень уверена
- $\tau \to \infty$ – модель совсем не уверена, выдает всем айтемам одинаковые вероятности
- На практике используем `clip(exp(learnable scalar), 0.01, 100)` и инициализируем `learnable scalar` нулём

Пусть $\gamma = \tau^{-1}$ – уверенность модели, тогда градиент по γ:

$$\nabla_\gamma L_{\text{softmax}}(u, p) = \nabla_\gamma \left[-\log \frac{e^{\cos(u,p)\cdot\gamma}}{\sum_{d \in D} e^{\cos(u,d)\cdot\gamma}} \right] = -\cos(u, p) + \mathbb{E}_{d \sim P_\theta(d|u)} \cos(u, d)\,.$$

- Если $\cos(u, p) < \mathbb{E}_d \cos(u, d)$, то $\gamma \downarrow$
- Если $\cos(u, p) > \mathbb{E}_d \cos(u, d)$, то $\gamma \uparrow$

5

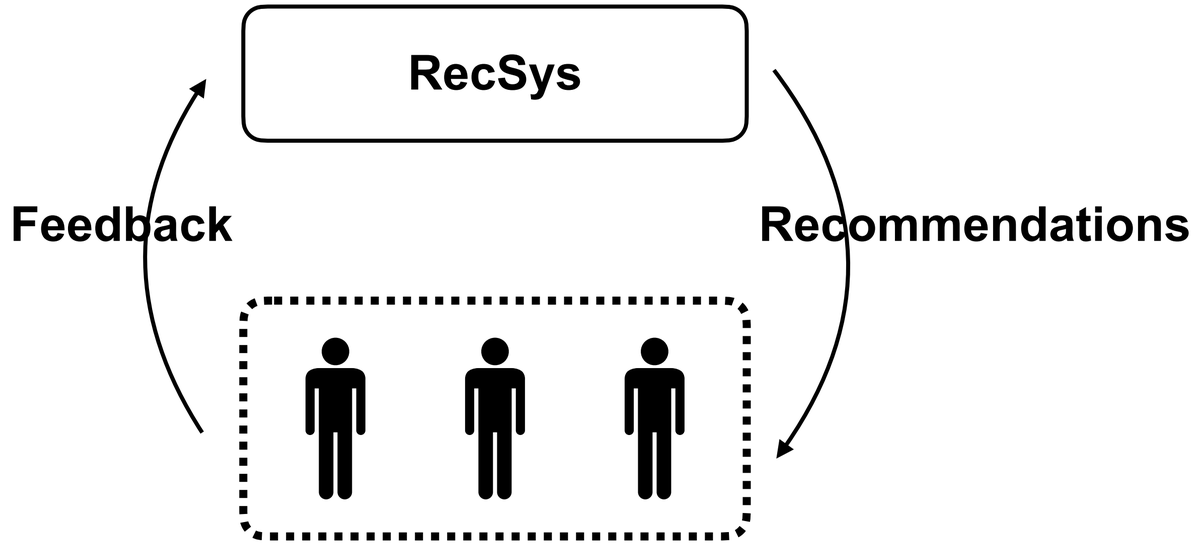
Согласованность с ранжированием

Рекомендательная система как RL

Задачу рекомендательной системы можно формализовать через **обучение с подкреплением**:

- Агент – рекомендательная система
- Действие – показать айтем (рекомендация)
- Окружение (среда) – пользователи
- Политика – распределение по айтемам
- Награда – клики, покупки, лайки, подписка, ретеншн (LTV превращается в кумулятивную награду)

… это самая правильная, фундаментальная модель рекомендаций!



Top-K Off-Policy Correction for a REINFORCE Recommender System

Многостадийная рексистема как RL

Подход 1: Multi-agent система

- Генератор кандидатов и ранжирование – разные агенты с общей целью
- Требуется согласованность: первый агент должен помогать второму

Подход 2: Action space reduction

- В RL при огромном пространстве действий применяются эвристики для его сужения
- Генератор кандидатов – эвристика, которая этим занимается
- Важна полнота (recall), но не относительно позитивов, а относительно того что нравится ранжированию

Согласованность с ранжированием

Генератор кандидатов – помощник ранжирования

- Он не должен “соревноваться” с ранкером, а должен выдавать понятных ему кандидатов
- Если есть несколько источников кандидатов и один из них “умнее” ранжирования (не согласован с ним), то ранкер будет его игнорировать

> **Цель: выбирать таких кандидатов, которые понравятся ранкеру.**

… С точки зрения оптимизации:

- логично учить кандген на согласованность с ранжированием
- не просто угадывать позитивные айтемы, а предсказывать что бы выбрал ранкер

6

Айтемные башни

Обучаемые эмбеддинги

В softmax-модели естественным образом получили двухбашенную модель с обучаемыми эмбеддингами для айтемов.

> У обучаемых эмбеддингов есть возможность выучить любую структуру семантического пространства, это буквально **самая выразительная модель.**

- Их можно инициализировать эмбеддингами из любой другой модели
- При этом в обратную сторону так не сделаешь (разве что дистилляцией)

… это все при условии что объект во времени не меняется (иначе нужно для каждого момента времени строить свой обучаемый эмбеддинг)

Минусы обучаемых эмбеддингов

Это те же эмбеддинги категориального признака очень высокой кардинальности (айдишника),
со всеми вытекающими минусами:

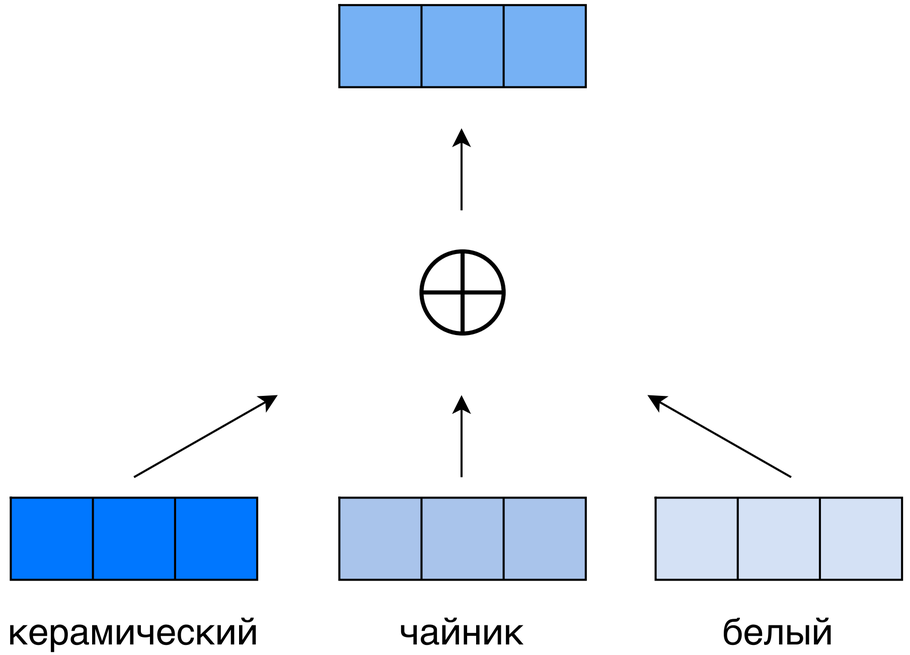
- **Трансдуктивность**: не умеем обрабатывать unseen айтемы
- **Плохое качество на тяжелом хвосте айтемов**, меморизация и переобучение
- **Память**: миллиарды айтемов – это очень большие embedding-таблицы
- **Distribution drift**: не учитываем изменения объекта во времени

Другой подход: Мешок слов

Для айтема-товара эмбеддинг можно составить через название:

> Токенизируем название → извлекаем для каждого токена обучаемый эмбеддинг → усредняем, чтобы получить векторное представление товара

- Товары с похожими названиями будут иметь похожие векторы
- Не учитывается порядок слов



Inductive Bias

> Inductive bias – ограничения на модель, **увеличивающие обобщающую способность**:

- Увеличиваем bias, уменьшаем variance (bias-variance tradeoff)
- Увеличиваем генерализацию, уменьшаем меморизацию
- Уменьшаем переобучение
- Нужно меньше данных для получения адекватных результатов
- Для двухбашенных моделей – ограничиваем структуру семантического пространства (e.g., мешок слов)

Inductive Bias

Вопрос: Как увеличить inductive bias, не выходя за рамки модели обучаемых эмбеддингов?

Inductive Bias

Вопрос: Как увеличить inductive bias, не выходя за рамки модели обучаемых эмбеддингов?

Ответ: Уменьшить размерность, добавить l2-нормализацию

Inductive Bias

Вопрос: Есть

- Модель без фичей (обучаемый эмбеддинг)
- Модель с одной фичой A
- Модель с двумя фичами A и B

Где больше inductive bias?

Inductive Bias

Вопрос: Есть

- Модель без фичей (обучаемый эмбеддинг)
- Модель с одной фичой A
- Модель с двумя фичами A и B

Где больше inductive bias?

Ответ: 1 < 3 < 2

Контентное кодирование

> Вместо обучаемых id-based эмбеддингов – **используем содержимое айтемов.**

Примеры контента:

- Текст названия, описание, картинка (неструктурированная информация)
- Артист, альбом, жанр, язык, год релиза; цена, магазин, цвет (признаки, мета-данные)

Обучаем энкодер, превращающий объект в эмбеддинг:

- Мешок слов над текстом (возможен и над признаками)
- Трансформер над текстом, CNN над картинкой, или даже мультимодальный энкодер
- DCN / MLP над признаками, как в прошлой лекции
- Какая-то комбинация вышеперечисленного

Если энкодер учится на данных всех пользователей, то **это гибридная модель** (и контентная, и коллаборативная).

Плюсы контентного кодирования

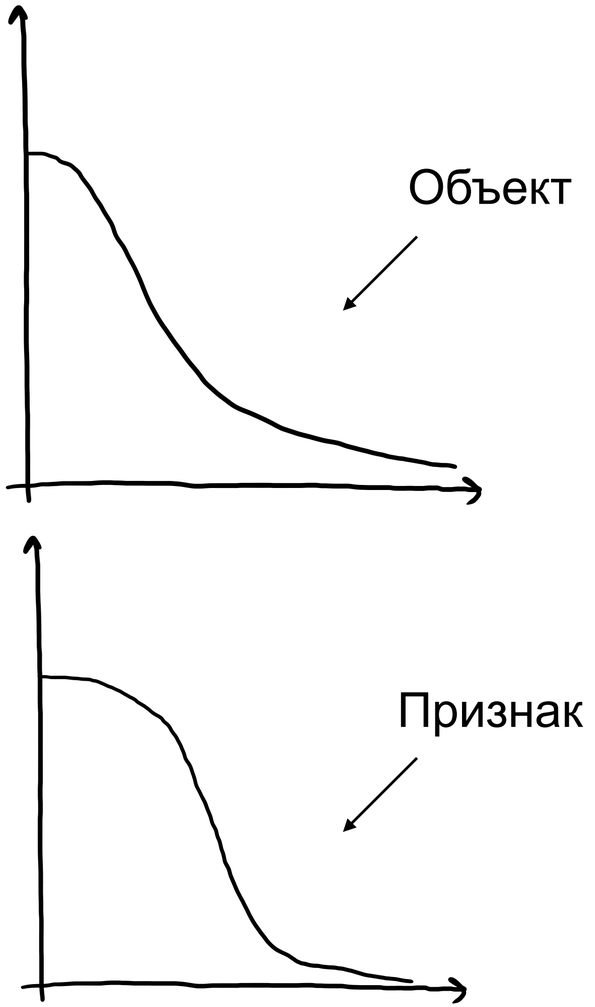
Контентное кодирование – это inductive bias:

- **Индуктивность**: умеем кодировать новые объекты
- **Меньше переобучения**, меньше меморизации, больше обобщающей способности
- **Лучше качество на тяжелом хвосте**
- **Требуют меньше памяти**, не нужна большая embedding-таблица
- **Учитываем изменения контента**

Тяжелый хвост и контентное кодирование

Почему контентое кодирование помогает с тяжелым хвостом объектов:

- Признаки всегда **повторяются** между объектами
- Хвост по признакам **строго легче**, чем хвост по ID объекта
- Пример: у редких айтемов может совпадать жанр / автор
- Экстремальный случай: если у каждого объекта значение признака уникальное, то этот признак можно использовать как ID объекта
- Кодируя признаки, улучшаем обобщающую способность, уменьшаем меморизацию



7

Пользовательские башни

Кодирование пользователя

Почему **не хотим кодировать пользователя через user id**, то есть использовать обучаемые эмбеддинги:

- Пользователи меняются со временем, не статичны
- Нужна реактивность (особенно на негативный фидбек)
- Холодные и неактивные пользователи очень важны: хотим конвертировать их в активных
- Важен контекст: время, запрос, seed item, где делаем рекомендацию (e.g., настройки моей волны)

Кодирование пользователя

Будем кодировать пользователя через его историю взаимодействий с айтемами:

- Событие в истории пользователя: пользователю в определенном **контексте** показали рекомендацию **айтема**, он оставил **фидбек** (explicit / implicit)
- Еще могут быть органические взаимодействия, вне рекомендательного трафика
- Могут быть события, не связанные с айтемами (запросы, кросс-доменные события)

… А еще кодирование через историю пользователя удовлетворяет Марковскому свойству для MDP, если не ограничиваем длину истории:

$$P(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots s_1, a_1) = P(s_{t+1} \mid s_t, a_t)$$

$s_t = \{h_1, h_2, \ldots h_t\}$, где h_i – событие в истории пользователя

Пример: SGD-based embedding

Рассмотрим простую модель:

- Для пользователей и айтемов используются обучаемые эмбеддинги
- Функция близости – скалярное произведение
- Учимся на полный softmax loss
- Используем SGD

Тогда градиент по эмбеддингу пользователя равен:

$$\nabla_u L_{\text{softmax}} = \nabla_u \left[-\log \frac{e^{\langle u,p \rangle}}{\sum_{d \in D} e^{\langle u,d \rangle}} \right] = -p + \sum_{d \in D} \frac{e^{\langle u,d \rangle}}{\sum_{d' \in D} e^{\langle u,d' \rangle}} d = -\left(1 - \mathbb{P}(p \mid u)\right) p + \sum_{d \in D / \{p\}} \mathbb{P}(d \mid u) d \,.$$

И шаг градиентного спуска выглядит как:

$$u_{t+1} = u_t + \alpha \left(1 - \mathbb{P}(p \mid u)\right) p - \alpha \sum_{d \in D / \{p\}} \mathbb{P}(d \mid u) d = u_t + \alpha_p p - \sum_n \alpha_n n \,.$$

То есть эмбеддинг пользователя – это сумма эмбеддингов айтемов, в которой позитивы суммируются, а негативы вычитаются.

Самая простая модель пользователя

Average pooling по айтемам в истории пользователя:

- На примере YoutubeDNN[1]: складываем вектора 50 просмотренных айтемов
- Напоминает item2item подход: $\left\langle u, i_{\text{target}} \right\rangle = \sum_k \left\langle i_k, i_{\text{target}} \right\rangle$
- Можно сделать time decay или взять только последние n событий

Айтемные эмбеды можно получать разными способами:

- Из какой-то другой модели
- Обучаемые (YoutubeDNN)
- Более сложное кодирование, которое учится одновременно с самим пулингом (end-to-end): e.g., мешок слов

Все еще плохо учитываем порядок событий.

1 Deep Neural Networks for YouTube Recommendations

Самая сложная модель пользователя

> История – это последовательность событий. Закодируем ее с помощью **трансформера**!

Про каждое событие знаем:

- Айтем
- Тип события (click, view, like)
- Позиция
- (Опционально) контекст

Можно отдельно сформировать эмбеддинги для всех четырех сущностей, и объединить в единый эмбеддинг суммой / конкатом / MLP / DCN.

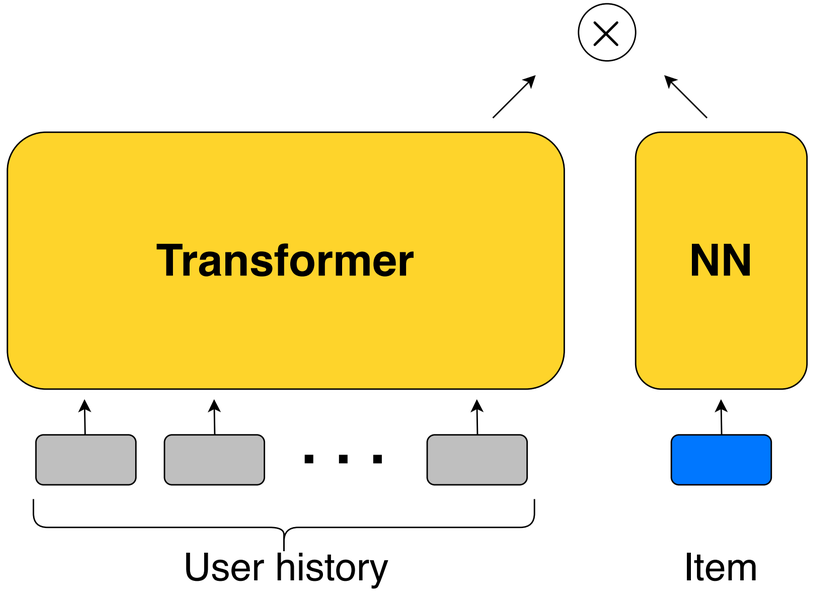
Позиционное кодирование:

- Любая схема из NLP: абсолютные позиционные обучаемые эмбеддинги, alibi, rotary, etc
- Timestamp-based кодирование

Трансформеры

Детали:

- Энкодер или декодер (двунаправленная маска или каузальная)
- Можно использовать любые улучшения трансформеров из NLP (prenorm, swiglu, rmsnorm)[1]
- В статьях используют 2-4 слоя, у нас на практике до 20
- Пулинг (сворачивание пользователя в один вектор) может быть с помощью отдельного CLS-токена, последнего токена, или контекста

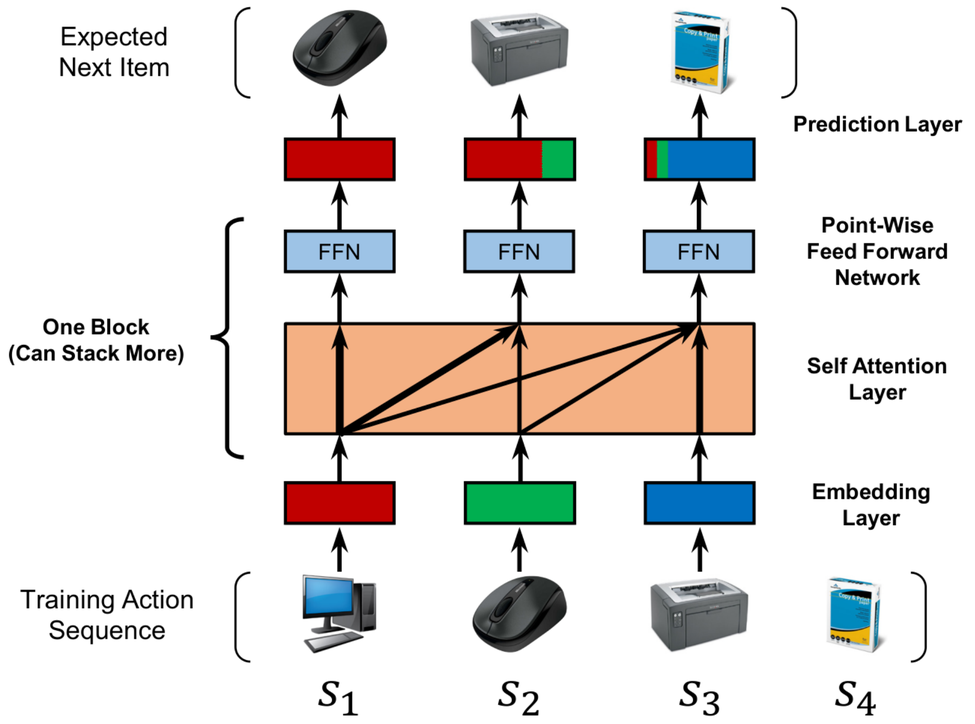


1 Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations

Next Item Prediction

Учим модель предсказывать следующий айтем в истории пользователя:

- Softmax по каталогу P(item | history)
- Эвристика (суррогатный лосс)
- Можно учить как авторегрессивный декодер (как LLM)
- Sequential Recommenders: Caser[1], GRU4Rec[2], SASRec[3], BERT4Rec[4,5]



Из статьи Self-Attentive Sequential Recommendation

1 Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding

2 Session-based Recommendations with Recurrent Neural Networks

3 Self-Attentive Sequential Recommendation

4 BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

5 Turning Dross Into Gold Loss: is BERT4Rec really better than SASRec?

Summary

Обсудили:

- зачем нужна генерация кандидатов
- зачем нужна нейросетевая генерация кандидатов
- двухбашенные модели
- на что учить генерацию кандидатов
- какую функцию близости использовать
- почему важна согласованность с ранжированием
- как могут выглядеть айтемные и пользовательские башни

Honorable Mentions

Учим модель предсказывать следующий айтем в истории пользователя:

- GNN как способ кодирования айтемов и пользователей
- Pinnerformer[1] как пример индустриального sequential recommender’a
- Context-aware трансформер с отдельными токенами под контекст
- Мультивекторные модели, lifelong моделирование, etc (из-за bitter lesson[2] будут не нужны)
- Semantic IDs[3], generative retrieval[4], mixture of logits[5] – в следующей лекции

1 PinnerFormer: Sequence Modeling for User Representation at Pinterest

2 http://www.incompleteideas.net/IncIdeas/BitterLesson.html

3 Better Generalization with Semantic IDs: A Case Study in Ranking for Recommendations

4 Recommender Systems with Generative Retrieval

5 Retrieval with Learned Similarities