

# Рекомендательные системы

## ШАД

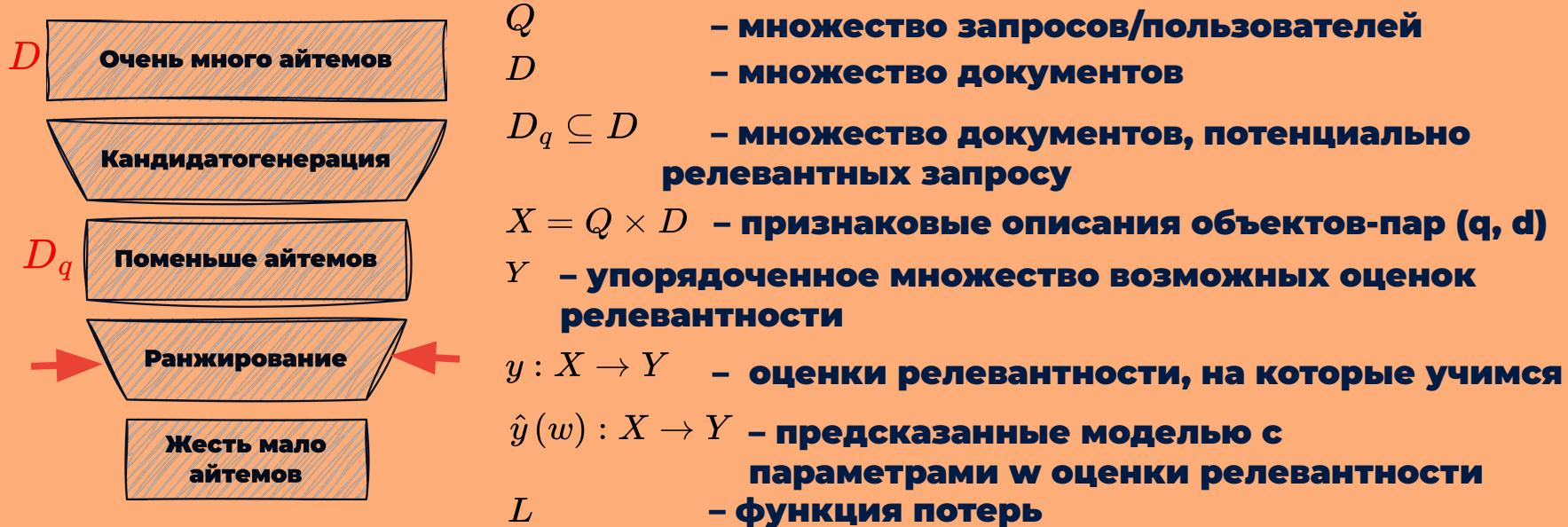
Весна 2025

Лекция 4:  
Ранжирование и  
переранжирование

# План

- ранжирующая модель:
  - постановка задачи
  - подходы
  - лосы
  - пул
- переранжирование
  - разнообразие
  - блендинг и мультитаргет
  - бустинг и пессимизация
  - exploration

# Ранжирование: постановка задачи



**Задача:** найти ранжирующую функцию  $a$ , восстанавливающую правильное отношение порядка

$$i \prec j \equiv y(q, d_i) < y(q, d_j)$$

# Ранжирование: подходы

	Pointwise	Pairwise	Listwise
Фокус	$\hat{y}(q, d_i) \rightarrow y$	$P(d_1 \prec d_2   q)$	$P(\pi   q)$
Учет порядка документов	нет	локально (на уровне пар)	да (на уровне списка)
Сложность вычисления лоса	низкая	средняя	высокая
Примеры	MSE, CrossEntropy	RankSVM, RankNet	SoftRank, ListNet, ListMLE  LambdaRank, LambdaMART, YetiRank

**pointwise**

# Pointwise: сведение к регрессии/классификации

$Y = \mathbb{R}$       или       $Y = \{0, 1, \dots K\}$

- **любая модель  
регрессии/классификации**
- **лосс: например, MSE/CrossEntropy**

$$MSE_q = \frac{1}{|D_q|} \sum_{i=1}^{|D_q|} (y_i - \hat{y}_i)^2 \quad CE_q = -\frac{1}{|D_q|} \sum_{i=1}^N y_i \log(\hat{y}_i)$$

- **а: сортирует Dq по предсказаниям  $\hat{y}_i$**



**pointwise**

**pairwise**

**пограничники**

**listwise**



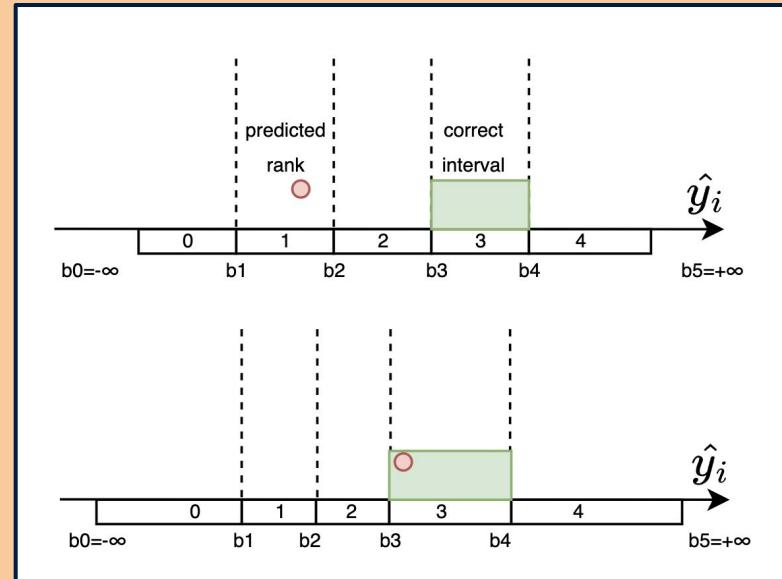
# Pointwise: порядковый таргет

$$Y = \{0, 1, \dots, K\}, 0 < 1 \dots < K$$

**введем пороги значений  
ранжирующей функции  $a$ , которые  
будут также обновляться в процессе  
обучения:**  $b_0 = -\infty, b_1 \leq \dots \leq b_{K-1}, b_K = +\infty$

**результат работы ранжирующего алгоритма:**

$$a(x, w, b) = y, \text{ если } b_{y-1} < \hat{y}(x, w) \leq b_y$$



**pointwise**

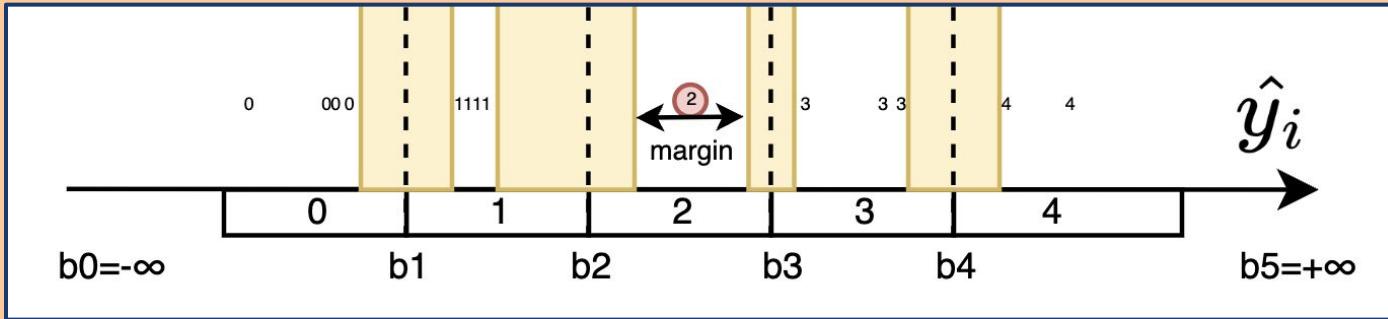
**pairwise**

**пограничники**

**listwise**



# Pointwise: порядковый таргет, SVOR



## Support Vector Ordinal Regression (SVOR) [1]

- **отступ объекта:**  $M_{ij} = (\hat{y}_i(w) - b_j)$
- **невозрастающая функция отступа:**  $f(M_{ij}) = (1 - M_{ij})_+$
- **лосс - сумма по двум порогам:**  
$$L = \sum_{i=1}^n ((1 - \langle x_i, w \rangle + b_{y_i-1})_+ + (1 + \langle x_i, w \rangle - b_{y_i})_+) + \frac{1}{2C} \|w\|^2$$



pointwise

pairwise

пограничники

listwise



**pairwise**

# Pairwise:

- **pointwise отступ**  $M_{ij} = (\hat{y}_i(w) - b_j)$   
-> **pairwise отступ**  $M_{ij} = (\hat{y}_j(w) - \hat{y}_i(w))$
- $L = \sum_{i \prec j} f(M_{ij})$  где  $f(M_{ij})$  – **невозрастающая функция**
  - $f(M_{ij}) = (1 - M_{ij})_+$  – **RankSVM**
  - $f(M_{ij}) = e^{-M_{ij}}$  – **RankBoost**
  - $f(M_{ij}) = \log(1 + e^{-M_{ij}})$  – **RankNet**



**pointwise**

**pairwise**

**пограничники**

**listwise**

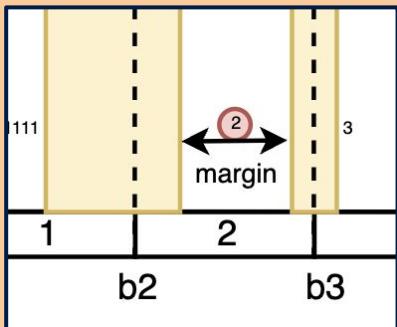


# Pairwise: RankSVM

$$f(M) = (1 - M)_+$$

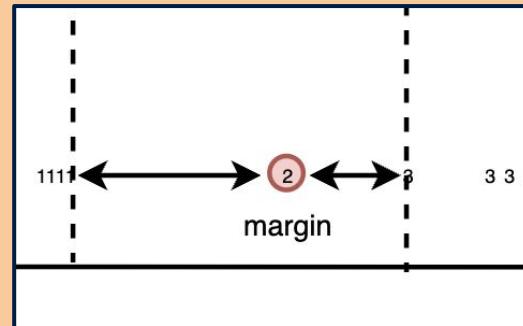
## SVOR

$$M_{ij} = (\hat{y}_i(w) - b_j)$$



## RankSVM [2]

$$M_{ij} = (\hat{y}_j(w) - \hat{y}_i(w))$$



$$\begin{aligned} L_q &= \frac{1}{2C} \|w\|^2 + \\ &+ \sum_{i=1}^{|D_q|} ((1 - \langle x_i, w \rangle + b_{y_i-1})_+ + (1 + \langle x_i, w \rangle - b_{y_i})_+) \end{aligned}$$

$$\begin{aligned} L_q &= \frac{1}{2C} \|w\|^2 + \\ &+ \sum_{i \prec j} \left( 1 - \langle x_j - x_i, w \rangle \right)_+ \end{aligned}$$



pointwise

pairwise

пограничники

listwise



# Pairwise: RankNet [3], 2005

**Лосс:**  $L = \sum_{i \prec j} f(M_{ij}) = [f(M) = \log(1 + e^{-\sigma M})] = \sum_{i \prec j} \log \left(1 + e^{-\sigma \langle w, x_i - x_j \rangle}\right)$

**Градиент:**  $\nabla_w L(w) = -\sigma \sum_{i \prec j} \frac{x_i - x_j}{1 + e^{\sigma \langle w, x_i - x_j \rangle}}$

**Обозначим**  $\lambda_{ij} = \frac{-\sigma}{1 + e^{\sigma \langle w, x_i - x_j \rangle}}$ ,  $\lambda_i = \sum_{j: x_i \succ x_j} \lambda_{ij} - \sum_{j: x_i \prec x_j} \lambda_{ij}$

$$\nabla_w L(w) = \sum_{i < j} \lambda_{ij} (x_i - x_j) = \sum_i \lambda_i x_i$$

**Тогда итерацию градиентного спуска можно записать в виде:**

$$w := w - \eta \sum_i \lambda_i \nabla_w \langle w, x_i \rangle$$



pointwise

pairwise

пограничники

listwise



**pairwise → listwise**

# Pairwise → Listwise: LambdaRank [4], (2007)

Пусть  $\tilde{G}$  – негладкая метрика качества ранжирования,

$\Delta\tilde{G}_{ij}$  – изменение  $\tilde{G}$  при перестановке  $(x_i, x_j)$

Домножение градиента  $\nabla_w L(w) = \sum_i \lambda_i x_i$

на  $|\Delta\tilde{G}_{ij}|$  приводит к приближенной оптимизации  $\tilde{G}$

$$\tilde{\lambda}_{ij} = \frac{-\sigma}{1 + e^{\sigma \langle w, x_i - x_j \rangle}} |\Delta\tilde{G}_{ij}|$$

LambdaMart	t-1	RankNet
non-rel	non-rel	non-rel
rel1	non-rel	non-rel
non-rel	non-rel	rel1
non-rel	non-rel	non-rel
non-rel	rel1	non-rel
non-rel	non-rel	non-rel
non-rel	non-rel	non-rel
non-rel	non-rel	rel2
rel2	non-rel	non-rel
non-rel	rel2	non-rel
non-rel	non-rel	non-rel



pointwise

pairwise

пограничники

listwise



# Pairwise → Listwise: LambdaMART [5], (2010)

**LambdaMart = MART +  $\lambda$ -градиенты из LambdaRank**

**MART (Multiple Additive Regression Trees)** - семейство моделей градиентного бустинга на решающих деревьях

	<b>MART (e.g. regression)</b>	<b>LambdaMART</b>
<b>Метрика</b>	<b>MSE</b> $\frac{2}{n} \sum_{i=1}^n (y_i - \hat{y}_i) \cdot x_i$	<b>NDCG</b> $\sum_i \tilde{\lambda}_i x_i$
<b>Градиент</b>		
<b>Следующее дерево аппроксимирует</b>	$(y_i - \hat{y}_i)$ <b>ошибки</b>	$\lambda$ <b>-градиенты</b>



**pointwise**

**pairwise**

**пограничники**

**listwise**



# Pairwise → Listwise: YetiRank [6], (2010)

**YetiRank = Ranknet + MART + weights**

$$L_q = - \sum_{i=1}^{|D_q|} \sum_{j=1}^{|D_q|} w_{ij} \log \frac{e^{x_i}}{e^{x_i} + e^{x_j}}$$

- **Веса:**
  - **важнее те пары, которые ближе к топу**
  - **менее критичны ошибки на парах с близкими/шумными оценками**
- **Веса меняются в процессе обучения**



**pointwise**

**pairwise**

**пограничники**

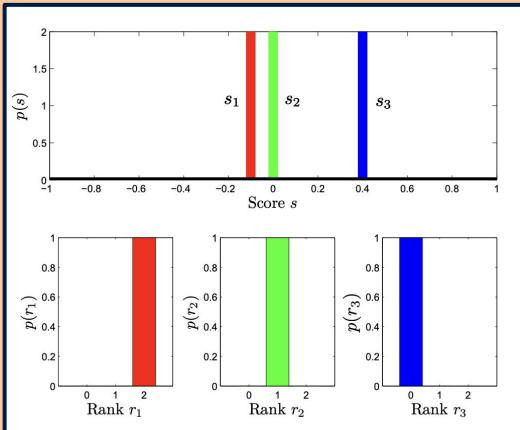
**listwise**



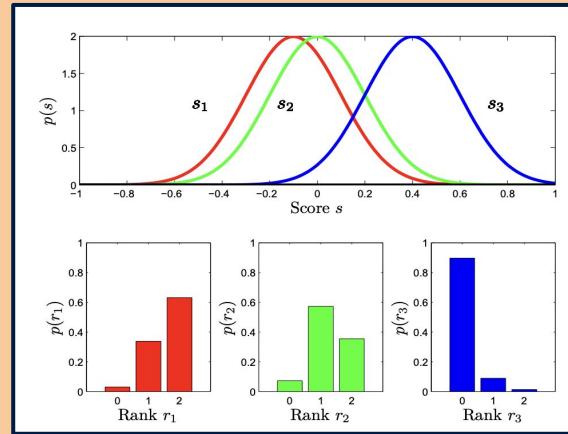
**listwise**

# Listwise: SoftRank [7], (2008)

Идея: сгладить nDCG, чтобы сделать ее дифференцируемой и напрямую использовать в функции потерь



$$\hat{y}_i \rightarrow \mathcal{N}(\hat{y}_i, \sigma^2)$$



$$nDCG@k = \frac{1}{IDCG@k} \sum_{i=1}^k \frac{2^{\hat{y}_i}}{\log_2(i+1)} \rightarrow$$

$$nDCG_{soft}@k = \frac{1}{IDCG@k} \sum_{i=1}^k \frac{2^{\hat{y}_i}}{\mathbb{E}(\log_2(i+1))}$$



pointwise

pairwise

пограничники

listwise



# Listwise: ListNet [8], (2007)

**Вероятность перестановки**  $\pi = (d_1, \dots, d_{|D_q|})$

$$P(\pi|\vec{y}) = \prod_{i=1}^{|D_q|} \frac{\exp(y_{\pi(i)})}{\sum_{k=i}^{|D_q|} \exp(y_{\pi(k)})}$$

**Можно использовать KL-дивергенцию, но различных перестановок  $|D_q|!$**

**Top-One Probability:** смотрим на распределение индекса первого документа в выдаче.

$$P(y_i) = \frac{\exp(y_i)}{\sum_{k=1}^{|D_q|} \exp(y_k)}$$

**Лосс:**

$$L_q = - \sum_{i=1}^{|D_q|} P(y_i) \log P(\hat{y}_i)$$



**pointwise**

**pairwise**

**пограничники**

**listwise**



# Listwise: ListMLE [9], (2008)

**Максимизируем вероятность правильной  
перестановки**

$$L = -\log P(\pi|\hat{y}) = - \sum_{i=1}^{|D_q|} \left( \hat{y}_{\pi(i)} - \log \sum_{j=i}^{|D_q|} \exp(\hat{y}_{\pi(j)}) \right)$$



**pointwise**

**pairwise**

**пограничники**

**listwise**



**Сбор пула**

# Сбор пула: виды фидбека

## Explicit

- **действия, которые отражают явное отношение к айтему**
  - **рейтинг**
  - **отзыв**

границы сервисо-специфичны

## Implicit

- **все остальные события, которые могут выступать в качестве прокси к explicit**
  - **покупка**
  - **время просмотра**
- **таких данных, как правило, больше**

# Сбор пула: кандидатогенерация vs ранжирование

## Кандидатогенерация :

- **Две возможные цели**
  - **полнота**
  - **попадание в топ на ранжировании**
- **Нужна репрезентативность** всего пространства
- **Чаще используют позитивные взаимодействия**

## Ранжирование :

- **Фокус на порядок - поднять самые релевантные айтемы на первые позиции**
- **Важны как позитивные, так и негативные сигналы**
- **можно использовать более жирные модели с большим количеством фичей (в т.ч. вычисленных на этапе кандидатогенерации)**
- **Задачи бизнеса нужно решать здесь!**

# Сбор пула: bias

- Важно следить за толстыми:
  - пользователями - часто и много взаимодействуют с контентом
  - айтемами - часто показываются, собирают кучу фидбека
- Если ничего не делать, то
  - переобучимся под них и даже не заметим этого в метриках
  - будем усиливать feedback loop
- А что делать то?
  - ограничивать кол-во взаимодействий на одного юзера/айтема
  - взвешивать
  - стратифицировать и считать метрики по сегментам
  - блендить - отдельные модели / мета-модель, которая решает, какую стратегию использовать



# Самый полезный слайд во всем курсе

(в этот раз на самом деле)

- сплит на **train → valid → test** нужно осуществлять

## по времени

- айтемы устаревают
- интересы пользователей меняются
- много всего происходит
- и особое внимание на **фичи, зависящие от времени**
  - подклеиваем в пул именно те значения, которые были на момент события
  - следим за тем, чтобы распределение фичей при обучении было таким же, как и при применении



# **Переранжирование**

# Разнообразие

- айтемы с **близкой релевантностью** , как правило, **похожи контентно**
- если отранжировать строго по **скору релевантности** - можем получить **однообразную** выдачу
- скучно и странно
- если не попали с тематикой, то:
  - показываем сразу много нерелевантных айтемов
  - теряем место, чтобы показать релевантное



# Разнообразие: по категориям

- **Throwback: одна из метрик Diversity**

$$\text{CategoryCoverage}@k = \frac{|\text{UniqueCategories}@k|}{|\text{Categories}@k|}$$

- **Алгоритм переранжирования:**  
**В цикле**
  - засемплировать скоры всех оставшихся айтемов из Dq
  - выбрать айтем с максимальным скором
  - пенализировать все оставшиеся айтемы, которые относятся к той же категории

# Разнообразие: DPP [10]

- Throwback: другая метрика diversity - intra-list similarity

$$ILS@k = \frac{2}{k(k-1)} \sum_{i=1}^k \sum_{j=i+1}^k \text{sim}(i, j)$$

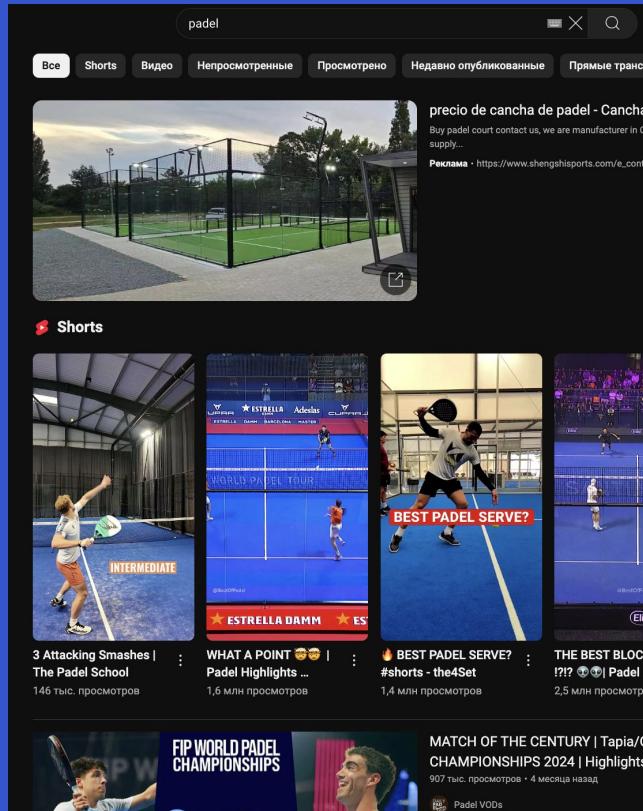
- Determinantal Point Process (DPP)
  - Выбираем подмножество айтемов, максимизирующих определитель матрицы сходства  $L'$

	$x_1$	$x_2$	$x_3$
$x_1$	$y_1^2$	$\text{sim}(x_1, x_2)$	$\text{sim}(x_1, x_3)$
$x_2$	$\text{sim}(x_2, x_1)$	$y_2^2$	$\text{sim}(x_2, x_3)$
$x_3$	$\text{sim}(x_3, x_1)$	$\text{sim}(x_3, x_2)$	$y_3^2$

# Блендинг и мультитаргет

- **блэндинг - объединение разных типов контента, e.g.**
  - **длинные видео и shorts**
  - **текстовые результаты и картинки**
- **мультитаргет - учет нескольких аспектов в таргете, e.g.**
  - **релевантность и конверсионность в поиске**
  - **дискавери и конверсионность в рекомендациях**

**tldr - необходимо учитывать разные сигналы от разных типов контента**



# Блендинг и мультитаргет

a. **единая формула , явно учитывающая разные сигналы (e.g. профицит[11] в Яндексе, EEAT[12] в Google)**

- **оцениваем каждый тип контента по каждому аспекту (одной моделью или отдельными)**
- **формируем итоговую выдачу по финальному скору**

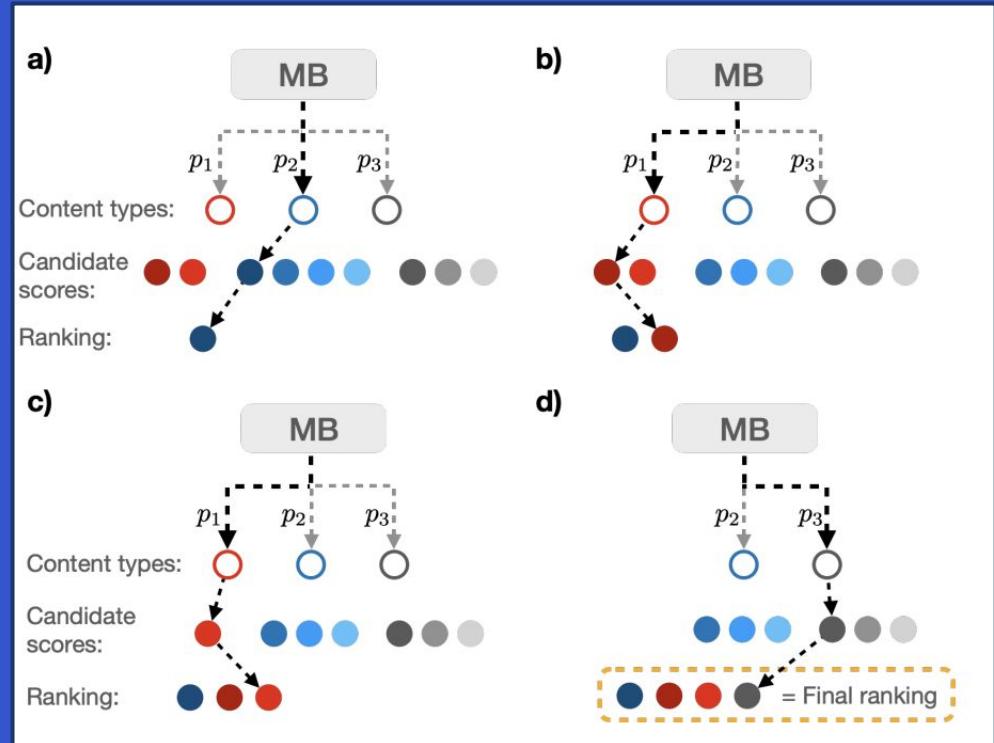
$$Score = \alpha \cdot CTR + \beta \cdot TimeSpent + \gamma \cdot Engagement$$



# Блендинг и мультитаргет

## b. multinomial blending [13]

- ранжируем каждый тип контента / айтемы, полученные из моделей под разный таргет отдельно
- и смешиваем: в цикле
  - выбираем тип контента в соответствии с распределениям  $p = [p_1, p_2, \dots, p_c]$
  - берем первый по скору еще неиспользованный айтем из этой группы



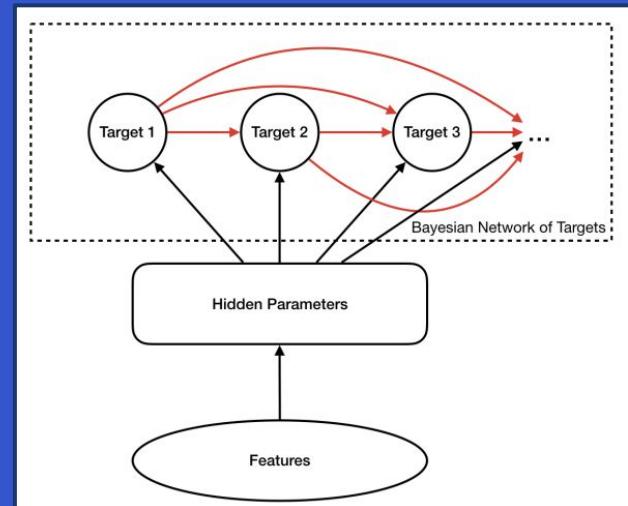
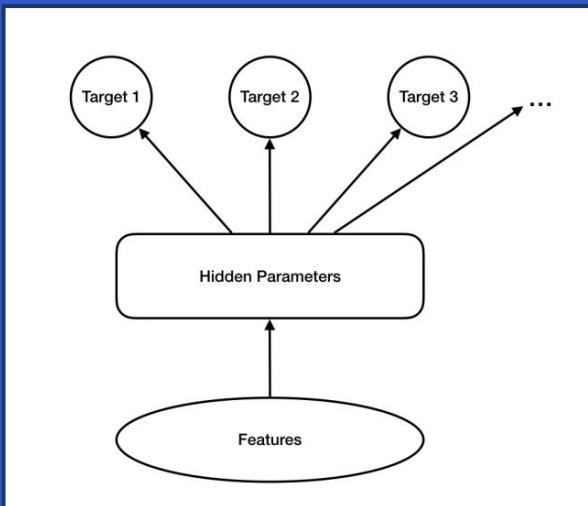
# Блендинг и мультитаргет

с. мультитаргетные модели, e.g. Deep Bayesian Multi-Target Learning (DBMTL)  
[14]

$$P(y_1 + y_2 + \dots y_c | x)$$

$$P(y_1 | x), P(y_2 | x), \dots, P(y_c | x)$$

$$P(y_1 | x) \cdot P(y_2 | x, y_1) \cdot \dots \cdot P(y_c | x, y_1, \dots, y_{c-1})$$



# Бустинг и пессимизация

- **искусственное повышение или понижение позиций определенных айтемов**
  - продвижение определенного автора
  - уклон в продажу высокомаржинальных товаров
  - снижение видимости устаревшего/низкокачественного контента
  - ...
- **how to**
  - **оффлайн:** учет специфичных факторов в таргете
  - **онлайн:**
    - **переранжирование поверх скоров модели**
    - **квотирование (like multinomial blending)**

$$\hat{y}'_i = \hat{y}_i \cdot \sum_{g=1}^{N_g} I[x_i \in g] \cdot PessCoef_g / BoostCoef_g$$

# **Exploration**

- **3 ключевые проблемы рексистем,**  
**recap:**
  - **exploration-exploitation tradeoff**
  - **feedback loop**
  - **cold start**

# Exploration: sampling

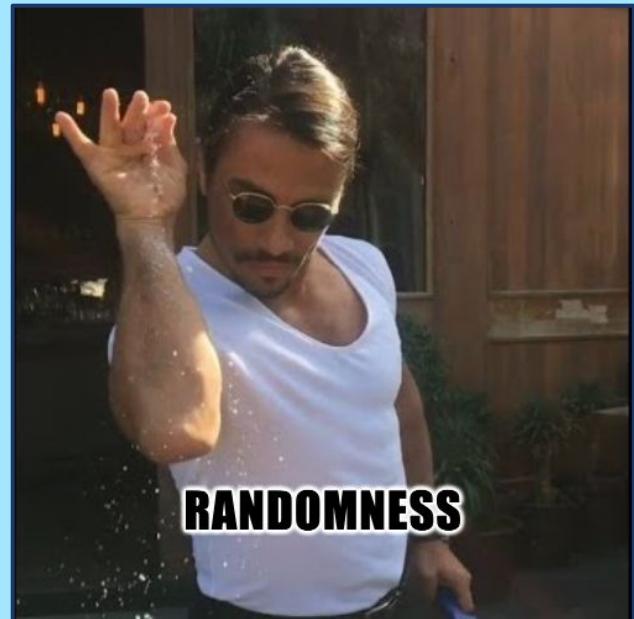
- добавление случайности в рекомендации
- семплировать можно, например:
  - из неизвестных категорий
  - из популярного среди остальных пользователей
  - из менее релевантного (по мнению модели)  
пропорционально релевантности
  - пример: softmax-sampling

$$\text{sample} \left( \frac{\exp(y_i/\tau)}{\sum_{j=1}^N \exp(y_j/\tau)} \right)$$

Gumbel-max trick [15]



$$\arg \max (y_i + \text{Gumbel}(\mu, \beta))$$

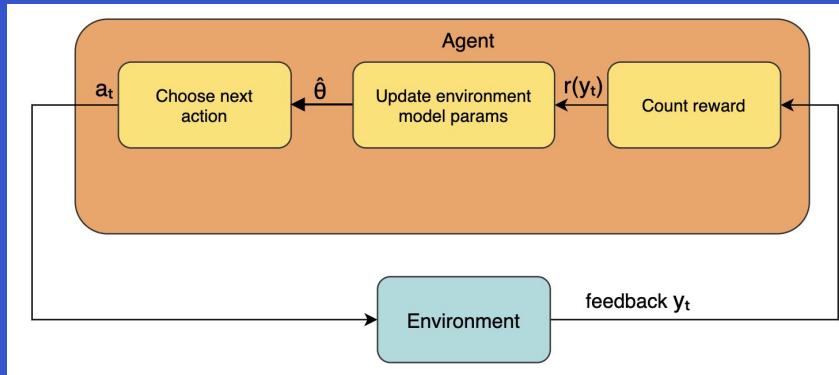


# Exploration: Многорукие бандиты

- есть несколько одноруких бандитов
- каждый бандит с некоторой собственной вероятностью дает фиксированный выигрыш при каждой попытке дернуть за ручку
- цель игрока - выиграть как можно больше: быстро найти наиболее выгодного однорукого бандита и играть в него
- среда не зависит от действий агента
- выбор ручки = выбор из фиксированного набора действий
  - показ пользователю конкретного айтема
  - использование вышеуровневого принципа генерации рекомендаций



# Exploration: Многорукие бандиты



$a_t$  – действие агента на  $t$ -м шаге

$y_t \sim \mathbb{Y}_t$  – ответ среды

$r_t$  – функция награды

$\theta, \hat{\theta}$  – модель зависимости ответов среды от совершенных действий и ее приближение, которое нам и нужно восстановить

Пример: 3 товара - купят / не купят



$Bern(\theta_1)$

$Bern(\theta_2)$

$Bern(\theta_3)$

модель среды:

$$\theta = \{\theta_1, \theta_2, \theta_3\}$$

возможные действия:

$$a_t = \{1, 2, 3\}$$

# Многорукие бандиты: $\epsilon$ -greedy

суть:

- как выбираем ручку : на основе уже собранной статистики:
  - с вероятностью  $\epsilon$  – с самым большим матожиданием выигрыша (*exploitation*)
  - с вероятностью ( $1 - \epsilon$ ) – любую другую (*exploration*)
- как обновляем модель среды :  
пересчитываем матожидания

но: не учитывается

- по мере поступления данных некоторые ручки становятся менее перспективными для исследования . все они получают равные шансы при решении идти в *exploration*
- уровень уверенности в текущих оценках



# Многорукие бандиты: учесть недоисследованность

Переходим от точечных оценок параметров модели среды к их вероятностному распределению

Формула Байеса:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Перекладываем на модель среды:  $p(\theta | Y, X) = \frac{p(Y | \theta, X) \cdot p(\theta)}{\int p(Y | \theta, X) p(\theta) d\theta}$

$p(\theta)$

– априорное распределение: текущие представления о модели

$p(Y|\theta, X)$

Правдоподобие: ответы среды при текущей модели

$p(\theta|Y, X)$

– апостериорное распределение: обновлённые представления о модели

Интуиция: наибольшие значения должны получать

- либо распределения с большим матожиданием ( exploitation )
- либо распределения с большой дисперсией ( exploration )

как  
выбираем  
ручку

### Thompson Sampling

- из текущего распределения параметров среды сэмплируем их значения

$$\theta \sim p(\theta | Y, X)$$

- считая их истинными, выбираем действие, при котором матожидание выигрыша наибольшее

$$a_t = \arg \max_{a_t} (r(a_t^i(\theta)))$$

как  
обновляем  
модель  
среды

### Upper Confidence Bound (UCB)

- считаем матожидания выигрыша каждого действия

$$\mathbb{E}_{a_t^i} = \mathbb{E}(r(a_t^i(\theta | Y, X)))$$

- считаем бонус за недоисследованность, e.g

$$b_{a_t^i} = \sqrt{\frac{\ln t}{n_{a_t^i}}}$$

- выбираем действие с максимальной верхней границей

$$a_t = \arg \max_{a_t} (\mathbb{E}_{a_t^i} + w \cdot b_{a_t^i})$$

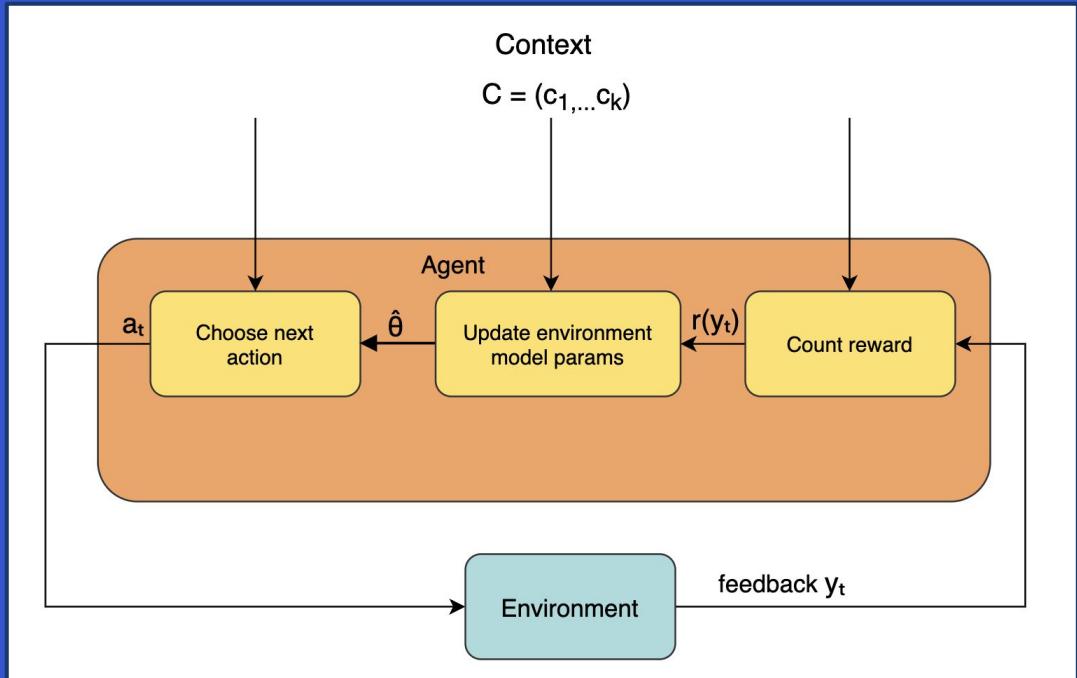
пересчитываем апостериорное распределение параметров  $p(\theta | Y, X)$

с учетом правдоподобия по вновь полученным оценкам  $p(Y | \theta, X)$

считая предыдущее апостериорное распределение за априорное  $p(\theta)$

# Многорукие бандиты: добавим контекста

- время
- сегмент
- пользователя (по полу/возрасту)
- тип приложения



# Контекстные многорукие бандиты: LinUCB

**Теперь для каждой ручки не один параметр, а вектор параметров:**

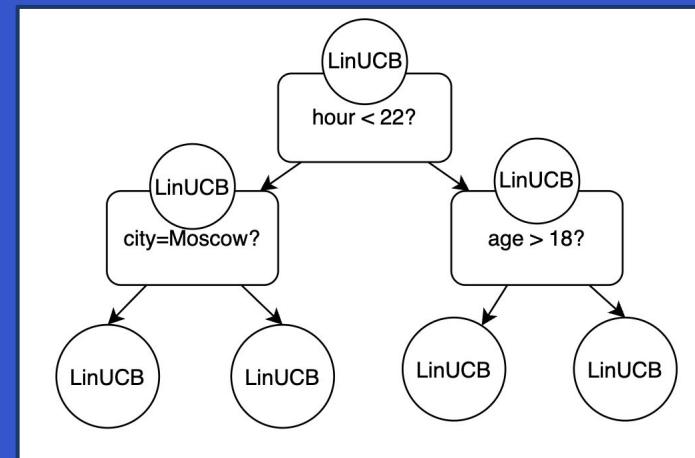
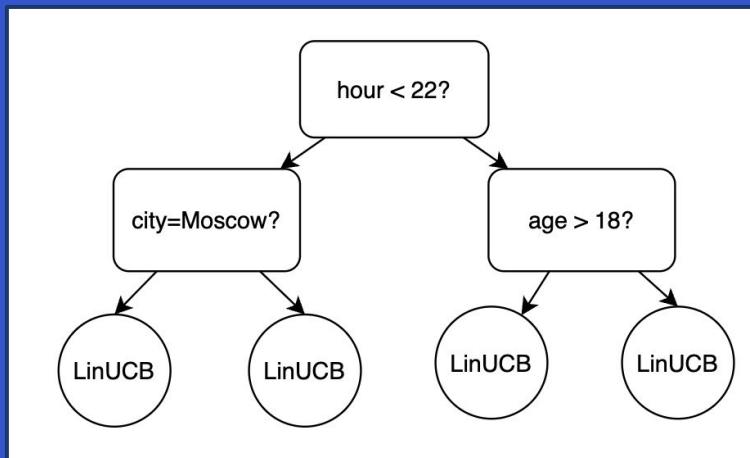
$$\theta_k = (\theta_k^1, \dots, \theta_k^v)$$

**Предполагается, что ожидаемая награда линейна от контекста:**

$$r(a_t^i(\theta, C)) = r(a_t^i(C \cdot \theta^T)) = \frac{1}{1 + \exp(-\theta_i^T \cdot C)}$$

# Контекстные многорукие бандиты: Tree Based LinUCB

- сегментируем контекст
- в каждом листе - локальная LinUCB



# Контекстные многорукие бандиты: NN UCB[18]

- **добавляем нейронку**  $r(a_t^i, \theta, C) = f_\theta(a_t^i, C)$
- **поверх - все тот же UCB с наградой и бонусом за неуверенность**

$$a_t = \arg \max_{a_t} \left( \mathbb{E}_{a_t^i} + w \cdot b_{a_t^i} \right)$$

- **как вычислять неуверенность:**
  - **дропаут**
  - **ансамблирование**
  - **байесовский слой**

# Бандиты: применимость в RT

- **много составных частей = большие риски**
  - **скорость доезда статистик и их объемы**
  - **теоремы могут оказаться применимыми**
  - **все может долго сходиться (а айтемы могут вообще выйти из обращения, пока мы пытались что-то выучить)**
- **сложно проверить в АВ новых бандитов**
  - **для чистоты нужно обнулить накопленные статистики и в текущих продовых бандитах**
  - **но можно напилить и не пустой старт**
  - **или построить фейковую модель среды**
- **на практике**
  - **многие не строят технически сложные пайплайны, собирают статистики с большими задержками**
  - **чаще выбирают томпсоновское семплирование**

# **Спасибо!**

Алена Зайцева,  
Служба ML-сервисов Лавки,  
Белград 2025

- [1] **SVOR**
- [2] **RankSVM**
- [3] **RankNet**
- [4] **LambdaRank**
- [5] **LambdaMART**
- [6] **YetiRank**
- [7] **SoftRank**
- [8] **ListNet**
- [9] **ListMLE**
- [10] **Determinantal Point Process**
- [11] **Профицит**
- [12] **EEAT**
- [13] **Ranking Across Different Content Types**
- [14] **Deep Bayesian Multi-Target Learning for Recommender Systems**
- [15] **Gumbel-max trick**
- [16] **Tree Based LinUCB**
- [17] **Многорукие бандиты в Яндекс Лавке**
- [18] **Neural Contextual Bandits**