

Рекомендательные системы



Проверить, идет ли запись

Меня хорошо видно & слышно?



Тема вебинара

Ранжирование при помощи градиентного бустинга



Бажмин Сергей

ex Intel, Sber, Sbermarket. Сейчас Senior ML Engineer в Research отделе по улучшению поиска и рекомендаций в hh.ru
Магистр по прикладной математике и информатике



bizhman322

Правила вебинара



Активно
участвуем



Off-topic обсуждаем
в учебной группе **OTUS RecSys-2024-10**

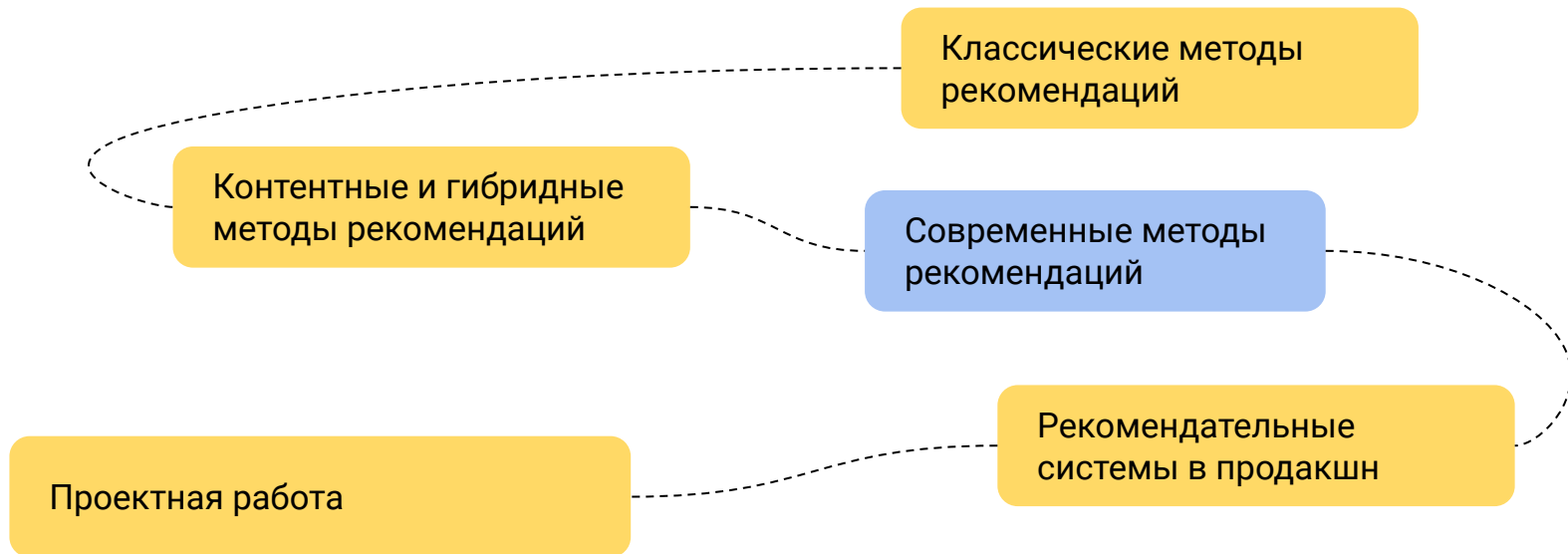


Задаем вопрос
в чат или голосом

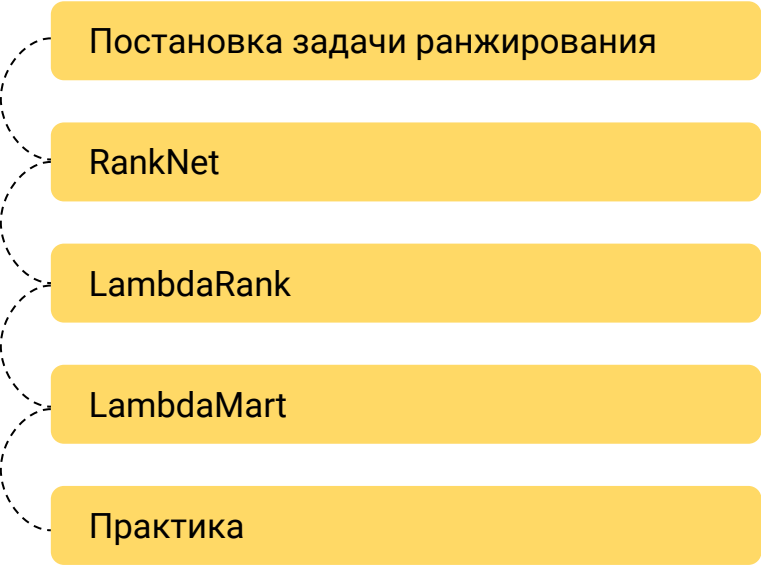


Вопросы вижу в чате,
могу ответить не сразу

Карта курса



Маршрут вебинара



Постановка задачи ранжирования

RankNet

LambdaRank

LambdaMart

Практика

Цели вебинара

К концу занятия вы сможете

1. Изучить базовые принципы ранжирования и первые подходы
2. Применить ранкер на основе градиентного бустинга на практике

Смысл

Зачем вам это изучать

1. Изучить основу ранжирования
 2. Уметь применить градиентный бустинг для ранжирования
-

Ранжирование

Постановка задачи

Не понравившиеся фильмы
(с отрицательным ранком)

Еще не просмотренные

Понравившиеся фильмы
(с положительным ранком)

Прометей



Китайский синдром



Меланхолия



Притяжение

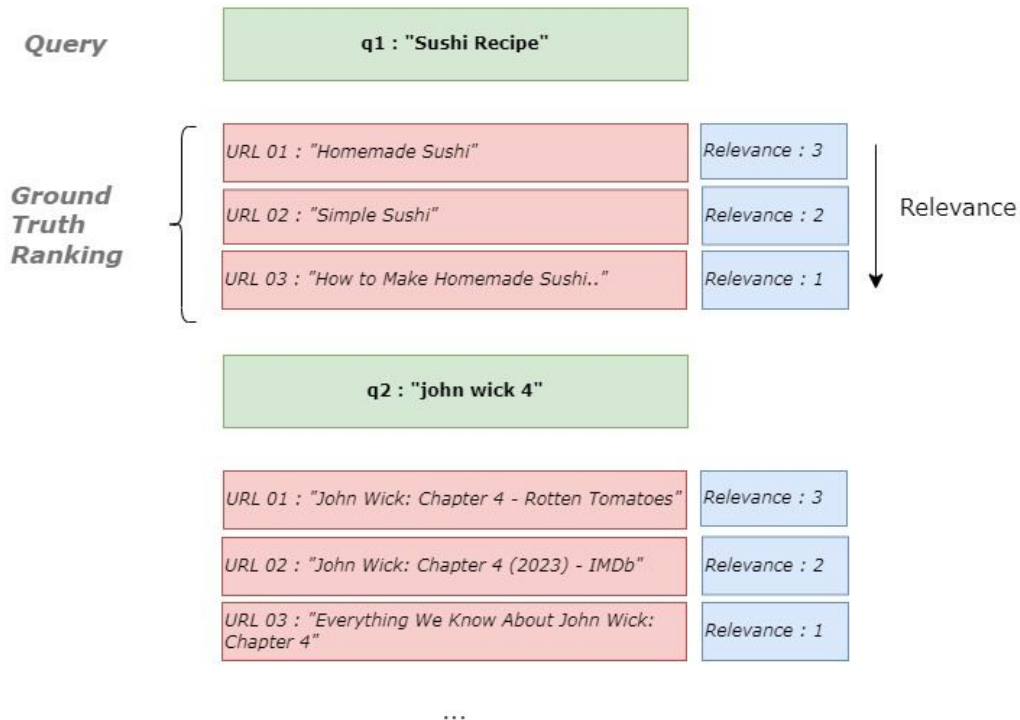


Интерстеллар



...

Постановка задачи



RankNet

[статья](#)

Пусть есть некоторые объекты i и j . Также имеем такую разметку, что $Y_i > Y_j$, означает объект i релевантнее объекта j .

Пусть X_i - вектор признаков объекта i . Хотим построить такую модель:

$$s_i = f(X_i) > f(X_j) = s_j.$$

Для такой модели функция потерь должна штрафовать некорректные предсказания $s_i - s_j < 0$, и незначительно менять модель при $s_i - s_j > 0$.

RankNet

Попробуем смоделировать вероятность того, что в паре (i, j) i объект релевантнее j-го. Для этого возьмем стандартный BCE:

$$\mathcal{L} = \theta^y (1 - \theta)^{1-y}, \quad y \in 0, 1 \qquad \theta = Pr(y|x)$$

$$\ell\ell = \log(\mathcal{L}) = y \log(\theta) + (1 - y) \log(1 - \theta)$$

$$\ell\ell_{ij} = y_{ij} \log\left(\frac{1}{1 + e^{-\sigma(s_i - s_j)}}\right) + (1 - y_{ij}) \log\left(\frac{e^{-\sigma(s_i - s_j)}}{1 + e^{-\sigma(s_i - s_j)}}\right)$$

$$= -y_{ij} \log(1 + e^{-\sigma(s_i - s_j)}) + \log(e^{-\sigma(s_i - s_j)}) - \log(1 + e^{-\sigma(s_i - s_j)}) - y_{ij} \log(e^{-\sigma(s_i - s_j)}) + y_{ij} \log(1 + e^{-\sigma(s_i - s_j)})$$

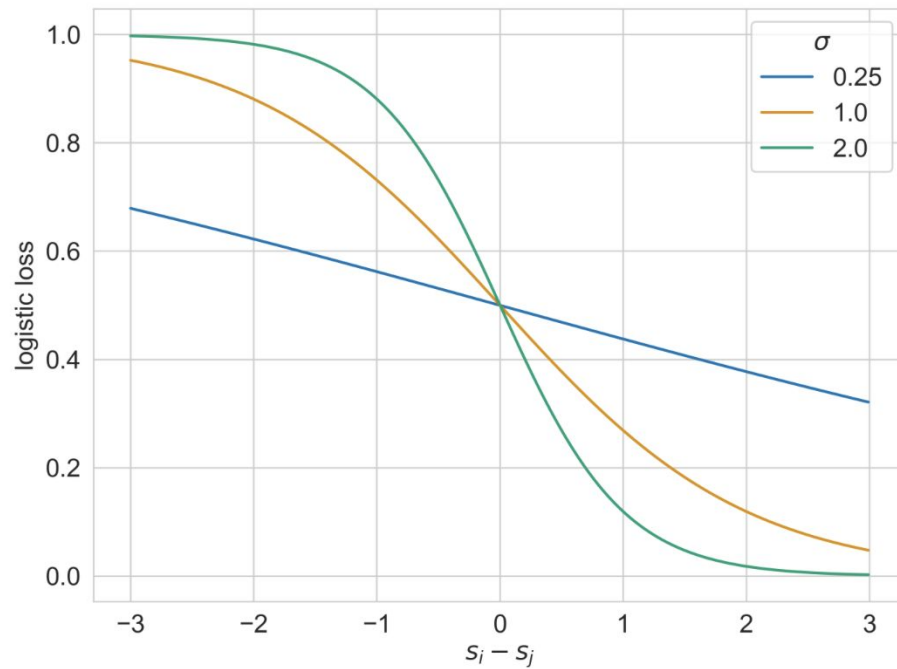
$$= (1 - y_{ij}) \log(e^{-\sigma(s_i - s_j)}) - y_{ij} \log(1 + e^{-\sigma(s_i - s_j)})$$

RankNet

Поскольку $i > j$ означает $y = 1$ (i -ый объект релевантнее j -го), то $i < j$ с таргетом $y = 0$ можно использовать как симметричный вариант $j > i$ с $y = 1$. Поэтому в принципе можно считать функцию потерь только по таргетам $y = 1$:

$$\text{logloss}_{ij} = \log(1 + e^{-\sigma(s_i - s_j)})$$

RankNet



LambdaRank

Статья

$$\begin{aligned}\frac{\partial \log \text{loss}_{ij}}{\partial s_i} &= \frac{-\sigma e^{-\sigma(s_j - s_i)}}{1 + e^{-\sigma(s_i - s_j)}} \\ &= \frac{-\sigma}{1 + e^{\sigma(s_i - s_j)}} \quad \left(\text{mult by } \frac{e^{\sigma(s_i - s_j)}}{e^{\sigma(s_i - s_j)}} \right) \\ &= \lambda_{ij}\end{aligned}$$

This is where the “lambda” in `lambdarank` comes from.

Note a critical shortcoming of the λ_{ij} gradients - **they are completely positionally unaware**. The formulation of the pairwise loss that we’ve used to derive the λ_{ij} treats the loss the same whether we’ve incorrectly sorted two items near the top of the query or near the bottom of the query. For example, it would assign the same loss and gradient to the pairs (1, 3) and (100, 101) when $s_i - s_j$ and Y_i and Y_j are the same. But most eCommerce or Google/Bing users spend like 90% of their time near the top of the query, so it’s much more important to optimize the items appearing within the first few positions. A possible correction proposed by Burges in 2006 [4] was to just scale λ_{ij} by the change in NDCG (a positionally-aware ranking metric) that would be incurred if the incorrect pairwise sorting of items (i, j) was used:

$$\lambda_{ij} = \log(1 + e^{-\sigma(s_i - s_j)}) |\Delta \text{NDCG}_{ij}|$$

эвристика, которая хорошо себя показала

LambdaRank



Fig. 1 A set of urls ordered for a given query using a binary relevance measure. The light gray bars represent urls that are not relevant to the query, while the dark blue bars represent urls that are relevant to the query. Left: the total number of pairwise errors is thirteen. Right: by moving the top url down three rank levels, and the bottom relevant url up five, the total number of pairwise errors has been reduced to eleven. However for IR measures like NDCG and ERR that emphasize the top few results, this is not what we want. The (black) arrows on the left denote the RankNet gradients (which increase with the number of pairwise errors), whereas what we'd really like are the (red) arrows on the right.

LambdaMart

[статья](#)

Вспомним алгоритм градиентного бустинга:

Input: training set $\{(x_i, y_i)\}_{i=1}^n$, a differentiable loss function $L(y, F(x))$, number of iterations M .

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma).$$

2. For $m = 1$ to M :

1. Compute so-called *pseudo-residuals*:

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \dots, n.$$

2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$.
3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

LambdaMart

Градиент из LambdaRank:

$$\lambda_{ij} = \log(1 + e^{-\sigma(s_i - s_j)}) |\Delta NDCG_{ij}|$$

LambdaMart

Ранжирование равносильно упорядочиванию объектов внутри какой-то группы. Под группой можно понимать различные поисковые запросы, пользователей и тд. Поскольку мы определяем скор внутри этой самой группы, то и градиент нужно считать внутри группы. Для объекта i градиентом будет не значение для пары (i, j) , а сумма градиентов всевозможных пар из группы:

$$\lambda_i = \sum_{j:i,j \in I} \lambda_{ij} - \sum_{j:j,i \in I} \lambda_{ij}$$

Получив эти градиенты для объектов внутри группы, мы продолжаем алгоритм бустинга для регрессии. Тем самым мы можем для каждого объекта предсказать скор внутри группы (например, получить скор для документа из поисковой выдачи для какого-то конкретного поискового запроса).

LambdaMart

На самом деле под капотом идет оптимизация методом Ньютона с различными хаками, которая зависит от реализации в конкретной библиотеке

$$\begin{aligned}\frac{\partial^2 \text{logloss}_{ij}}{\partial s_i^2} &= \frac{\sigma^2 e^{-\sigma(s_j - s_i)} |\Delta NDCG_{ij}|}{(1 + e^{-\sigma(s_j - s_i)})^2} \\ &= \frac{-\sigma}{1 + e^{-\sigma(s_j - s_i)}} |\Delta NDCG_{ij}| \cdot \frac{-\sigma e^{-\sigma(s_j - s_i)}}{1 + e^{-\sigma(s_j - s_i)}} \\ &= \lambda_{ij} \frac{-\sigma e^{-\sigma(s_j - s_i)}}{1 + e^{-\sigma(s_j - s_i)}}\end{aligned}$$

LambdaMart

LambdaMART algorithm

set number of trees N , number of training samples m , number of leaves per tree L ,
learning rate η

for $i = 0$ to m **do**

$F_0(x_i) = \text{BaseModel}(x_i)$ //If BaseModel is empty, set $F_0(x_i) = 0$

end for

for $k = 1$ to N **do**

for $i = 0$ to m **do**

$y_i = \lambda_i$

$w_i = \frac{\partial y_i}{\partial F_{k-1}(x_i)}$

end for

$\{R_{lk}\}_{l=1}^L$ // Create L leaf tree on $\{x_i, y_i\}_{i=1}^m$ R_{lk} is data items at leaf node l

$\gamma_{lk} = \frac{\sum_{x_i \in R_{lk}} y_i w_i}{\sum_{x_i \in R_{lk}} w_i}$ // Assign leaf values based on Newton step.

$F_k(x_i) = F_{k-1}(x_i) + \eta \sum_l \gamma_{lk} I(x_i \in R_{lk})$ // Take step with learning rate η .

end for

Функции потерь для ранжирующего бустинга

Lambdarank не единственный вариант для бустинга, [catboost](#) предлагает большой зоопарк других функций, в некоторые реализации можно передать сразу пары для обучения pairwise функций:

QueryRMSE

$$\sqrt{\frac{\sum_{Group \in Groups} \sum_{i \in Group} w_i \left(t_i - a_i - \frac{\sum_{j \in Group} w_j (t_j - a_j)}{\sum_{j \in Group} w_j} \right)^2}{\sum_{Group \in Groups} \sum_{i \in Group} w_i}} \quad \dots$$

PairLogit

$$\frac{- \sum_{p,n \in Pairs} w_{pn} \left(\log \left(\frac{1}{1 + e^{-(a_p - a_n)}} \right) \right)}{\sum_{p,n \in Pairs} w_{pn}}$$

Формат данных для ранжирующего бустинга

тут id для примера, на самом деле здесь будут фичи

по умолчанию часто требуется упорядочить записи последовательно по группам, имея информацию о релевантности. В данном случае мы можем сгруппировать аниме по пользователю, который их просмотрел.

	User Id	Anime Id	Relevance
Group Size = 3	1	192	0.3
	1	29	0.39
	1	282	1
Group Size = 2	2	12	1
	2	20	0.92
Group Size = 1	3	202	0.34

Group_Parameter = [3,2,1]

Вопросы?



Ставим “+”,
если вопросы есть



Ставим “-”,
если вопросов нет

Практика

Вопросы?



Ставим “+”,
если вопросы есть



Ставим “-”,
если вопросов нет

Рефлексия

Рефлексия



С какими впечатлениями уходите с вебинара?



Как будете применять на практике то, что узнали на вебинаре?

Цели вебинара

К концу занятия вы сможете

1. Изучить базовые принципы ранжирования и первые подходы
2. Применить ранкер на основе градиентного бустинга на практике

**Заполните, пожалуйста,
опрос о занятии
по ссылке в чате**