Машинное обучение ранжированию (Learning to Rank)

Информационный поиск. Лекция 8

О преподавателях

Андрей Кривой

Руководитель группы ранжирования

Telegram: @mt6qmzaotzn3

Владимир Суловьев

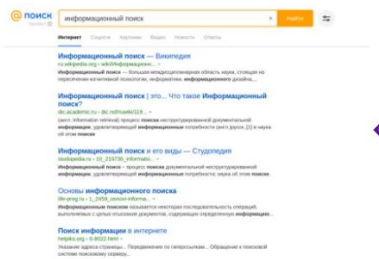
Программист-исследователь в команде поиска по постам

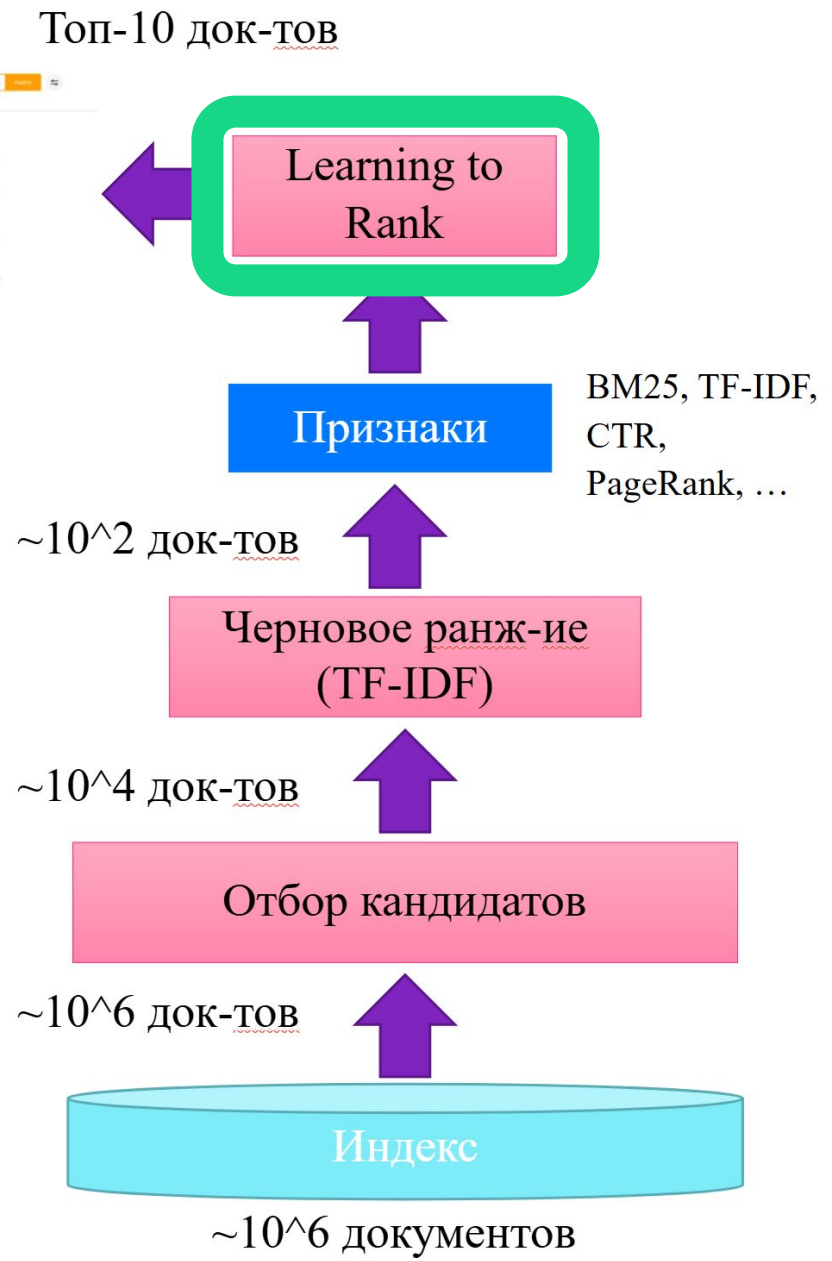
Telegram: @mb_nexttime

О чем будем говорить

Постановка задачи ранжирования

Мы сейчас тут 👀





Вспомним некоторые постановки задачи машинного обучения с учителем

1

Классификация – учимся предсказывать класс(-ы) объекта

2

Регрессия – предсказываем некоторое действительное число(-а)

3

Ранжирование – ?

Постановка задачи ранжирования в общем случае

Дано:

- X – множество объектов.
- $X^l = \{ x_1, x_2, \ldots, x_l \}$ – обучающая выборка
- $i \lhd j$ — правильный частичный порядок на парах $(i, j) \in \{ 1, \ldots, l \}^2$

  "Правильность" зависит от конкретной постановки задачи.

Задача:

- Построить ранжирующую функцию $\alpha: X \to R$ такую, что: $i \lhd j \Rightarrow \alpha(x_i) < \alpha(x_j)$.

Постановка задачи ранжирования в поиске

- D – коллекция текстовых документов.
- Q – множество запросов.
- $D_q \subseteq D$ – множество документов, найденных по запросу q.
- $X = Q \times D$ – объектами являются пары (запрос, документ).

Постановка задачи ранжирования в поиске

- Y – упорядоченное множество рейтингов.
- $y: X \to Y$ – оценки релевантности: чем выше оценка $y(q,d)$, тем релевантнее документ d по запросу q.
- Правильный порядок определен только между документами, найденными по одному и тому же запросу q: $(q,d) \prec (q,d') \Leftrightarrow y(q,d) < y(q,d')$.
- Релевантные ответы запросу q – это список документов d, упорядоченных с помощью функции ранжирования $\alpha(q,d)$.

Откуда взять признаки?

Раз объект это пара из документа и запроса, нужно подобрать признаки, которые будут отображать:

1. Соответствие документа запросу – хотим давать пользователю запрашиваемую информацию
2. Общую “привлекательность” документа – хотим давать качественную информацию

Тип признака	Примеры
Текстовые	- TF-IDF - BM25 - Длина документа - …
Поведенческие	- Число кликов/просмотров - ctr - …
Нейрофичи	- Косинус между эмбеддингами запроса и документа - …
…	…

Откуда взять таргет?

Разобрались с тем, что является объектами – откуда будем брать релевантность?

Самый показательный вариант получить релевантность – составить инструкцию по ее оценке и использовать асессоров.

*но есть и множество других вариантов (действия пользователей)

суп пюре без картошки

Россия [188]

Позиция	Серп	Оценка
1	http://dzen.ru/a/ZMO8UArLp1wBHapW	3
2	http://dzen.ru/a/XHuN7HzyvwCy_F03	3
3	http://dzen.ru/a/XtTh0eD9Qk7U35md	3
4	http://dzen.ru/a/Y7Hx8w9kbUua6CB9	3
5	http://dzen.ru/a/ZDTbzy_Bu0hZ1WvN	1
6	http://dzen.ru/a/XJ-UzlMjmgCzpydd	3
7	http://dzen.ru/a/Ys752RXWeIlvXHzv	1
8	http://dzen.ru/a/Wsm3AWEEk4W2X0DM	1
9	http://dzen.ru/a/XIOEaNI1AwC0fM_Q	1
10	http://dzen.ru/a/X9YVDjPtQgw_Rzwh	1

Метрики оценивания качества ранжирования

Как оценить результат?

Предположим, мы построили алгоритм(-ы), как-то ранжирующий документы для запроса.

Как понять, хорош ли построили алгоритм или какую из версий нам выбрать?

Вариант 1: посмотреть на поведенческие метрики (клики пользователей, просмотры и тд.) – непозволительная роскошь при обучения модели машинного обучения.

Хотим иметь возможность быстро оценить качество, имея только датасет.

Как оценить результат?

Предположим, мы построили алгоритм(-ы), как-то ранжирующий документы для запроса.

Как понять, хорош ли построили алгоритм или какую из версий нам выбрать?

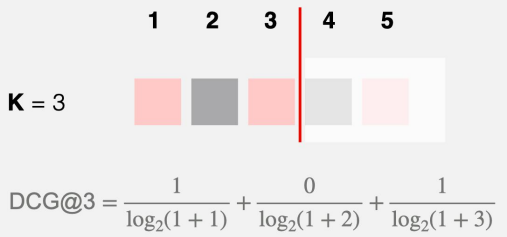
Вариант 2: оффлайн метрики качества поиска!

Вспомним некоторые из них:

Рекап метрик

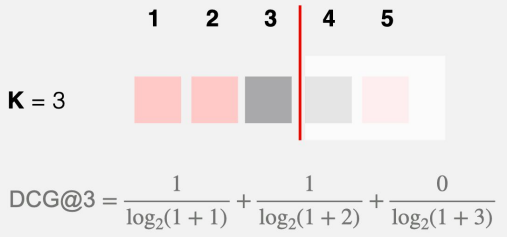
Вспомним, какие метрики мы используем для оценки качества поиска:

$$\text{DCG@K} = \sum_{k=1}^{K} \frac{rel_i}{\log_2(i+1)}$$



$$\text{DCG@3} = 1 + 0 + 0.5 = 1.5$$

$$\text{NDCG@K} = \frac{DCG@K}{IDCG@K}$$

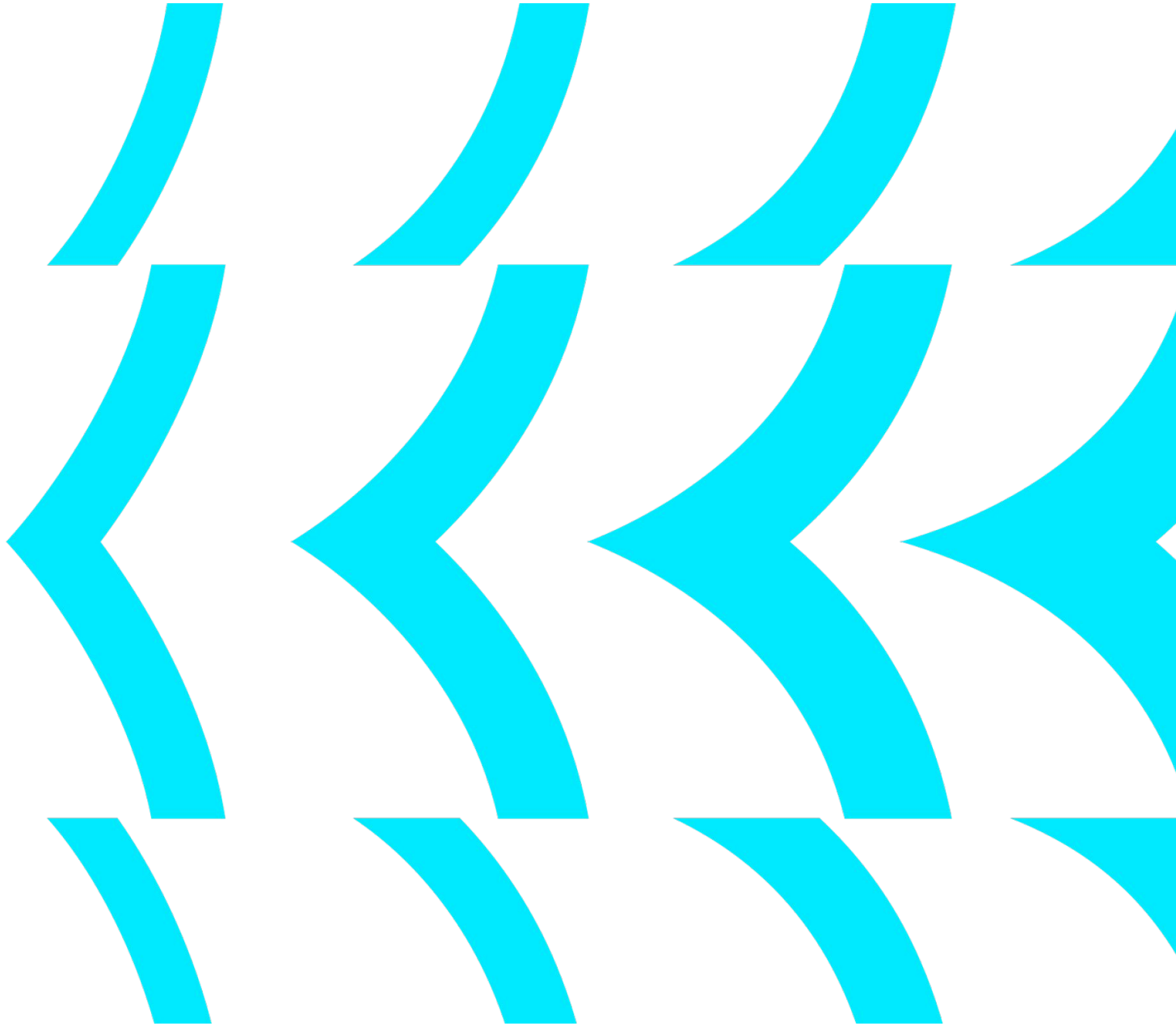


$$\text{DCG@3} = 1 + \frac{1}{1.585} + 0 \approx 1.63$$

(картинки взяты с сайта evidentlyai.com)

Подходы к решению задачи ранжирования

Почему не оптимизировать сразу DCG/NDCG?



Pointwise

Будем оптимизировать не метрику напрямую, а некоторый лосс, оптимизируя который мы будем улучшать MAP, DCG, NDCG и тд.

Обучаемся напрямую на паре запрос-документ, не обращая внимания на остальные документы для этого запроса.

Плюсы

- Подход работает: плохие документы оказываются снизу, а хорошие – сверху.

Минусы

- Никак не учитывается, что нужно предсказать порядок объектов, а не оценки.

Минусы Pointwise наглядно

Pointwise подход рассматривает абсолютные значения оценок, хотя они имеют разный смысл:

1 – вообще не релевантно, а 2 и 3, например, имеют разные степени релевантности.

Пример:

query: VK

выдача / скор модели:

- vk-stadium.ru: 4.55
- vk.com: 4.5

настоящие оценки:

- vk.com: 5
- vk-stadium.ru: 4

MSE: vk.com – 0.25, vk-stadium.ru – 0.3
Ошибка и там и там маленькая – модель считает, что все хорошо, однако ранжирование неверное.

Pairwise

Все еще оптимизируем не метрику напрямую, а прокси лосс.

Обучаемся на парах документов в рамках запроса.

Примеры: RankNet, LambdaRank, LambdaMART, YetiRank

Бонус – кликовый датасет
Пример таргетов для произвольного запроса “*vk*”:

vk.com	1
vk.company	0
ru.wikipedia.org/wiki/VK	0

Плюсы

- Акцент именно на правильное ранжирование документов.
- Не обязательно иметь все релевантности.

Минусы

- Не учитываем различное число документов для каждого запроса: количество документов в запросе порождает квадратичное число пар.

Listwise

С помощью вероятностных эвристик пытаемся оптимизировать DCG напрямую.

Обучаем модель на ранжированных списках.

На практике – несколько сложных алгоритмов, у каждого своя фишка

Плюсы

- Редко переобучается.
- Оптимизируем метрику напрямую.

Минусы

- Долгое обучение по сравнению с другими подходами.
- Долгий инференс.
- Не можем применить к одному документу.

Итого

Метод	Pointwise	Pairwise	Listwise
Плюсы	• Наиболее быстрый • Хорошее оказывается сверху, плохое – снизу	• Фокусируемся именно на правильном ранжировании • Работает с кликами!	• Оптимизирует напрямую целевую метрику
Минусы	• Плохо ранжируем хорошие относительно хорошего	• Нужно следить за тем, чтобы у запросов было одинаковое количество документов	• Медленнее остальных • Не работает с одним документом

RankNet

RankNet

- RankNet – первая идея pairwise-подхода.

- Подход к решению: давайте обучать функцию, которая по данному вектору атрибутов x $\in \mathbb{R}^n$ выдаёт f(x) и ранжирует документы по значению f(x).

RankNet

Итак, для тестовых примеров x_i и x_j модель считает

$s_i = f(x_i)$ и $s_j = f(x_j)$, а затем оценивает

$$p_{ij} = p(x_i \succ x_j) = \frac{1}{1 + e^{-\alpha(s_i - s_j)}}$$

Таргеты – это $q_{ij} = x_i \succ x_j$

принимают, например, точные значения из $\{0, 1\}$.

Разумная функция ошибки – кросс-энтропия:

$$C = -q_{ij} log(p_{ij}) - (1 - q_{ij}) log(1 - p_{ij})$$

RankNet

$$C = -q_{ij} log(p_{ij}) - (1 - q_{ij}) log(1 - p_{ij})$$

Рассмотрим случай

для $S_{ij} \in \{-1, 0, 1\}$

$q_{ij} = (1 + S_{ij})/2$

получаем

$$C = \frac{1}{2}(1 - S_{ij})\alpha(s_i - s_j) + log(1 + e^{-\alpha(s_i - s_j)})$$

т.е.

$$C = \begin{cases} log(1 + e^{-\alpha(s_i - s_j)}) & \text{for} \quad S_{ij} = 1 \\ log(1 + e^{-\alpha(s_j - s_i)}) & \text{for} \quad S_{ij} = -1 \end{cases}$$

RankNet

$$C = \frac{1}{2}(1 - S_{ij})\alpha(s_i - s_j) + log(1 + e^{-\alpha(s_i - s_j)})$$

Посчитаем градиент по s_i

$$\frac{\partial C}{\partial s_i} = \alpha(\frac{1 - S_{ij})}{2} - \frac{1}{1 + e^{\alpha(s_i - s_j)}}) = -\frac{\partial C}{\partial s_j}$$

Осталось использовать это для подсчета градиента по весам

$$\frac{\partial C}{\partial w_k} = \sum_i \frac{\partial C}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \sum_j \frac{\partial C}{\partial s_j}\frac{\partial s_j}{\partial w_k}$$

RankNet

Продолжаем упрощать:

$$\frac{\partial C}{\partial w_k} = \frac{\partial C}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \frac{\partial C}{\partial s_j}\frac{\partial s_j}{\partial w_k} = \lambda_{ij}\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right)$$

Где $\lambda_{ij} = \alpha\left(\frac{1 - S_{ij}}{2} - \frac{1}{1 + e^{\alpha(s_i - s_j)}}\right)$

Переупорядочив пары так, чтобы всегда было $x_i \succ x_j$
и $S_{ij} = 1$

Получим:

$$\lambda_{ij} = -\frac{\alpha}{1 + e^{\alpha(s_i - s_j)}}$$

RankNet

$$\lambda_{ij} = -\frac{\alpha}{1 + e^{\alpha(s_i - s_j)}}$$

Итак суммарный апдейт веса будет:

$$\Delta w_k = -\eta \sum_{(i,j)\in I} (\lambda_{ij} \frac{\partial s_i}{\partial w_k} - \lambda_{ij} \frac{\partial s_j}{\partial w_k}) = -\eta \sum_i \lambda_i \frac{\partial s_i}{\partial w_k}$$

где

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

RankNet

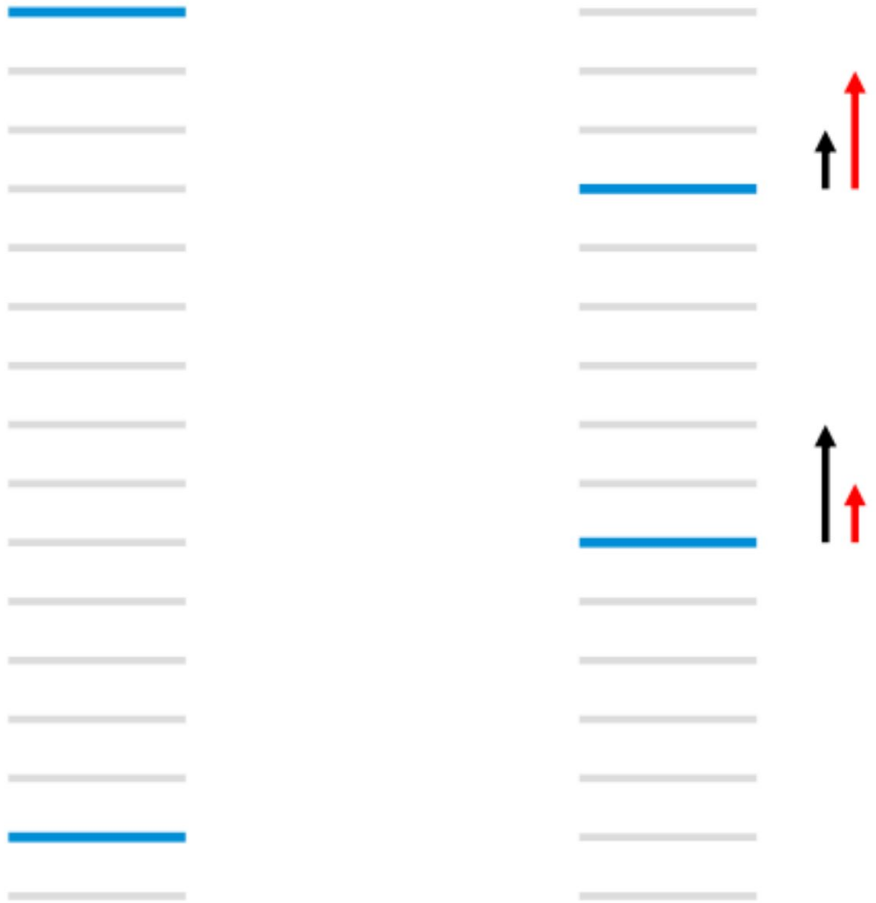
Смысл лямбд – в том, что это как бы “сила”, с которой каждый конкретный документ нужно потянуть либо вниз, либо вверх на каждой итерации.

LambdaRank

LambdaRank

В чем проблема RankNet?

- Мы оптимизируем число попарных ошибок, а это не всегда то, что нужно.
- Градиенты RankNet – это не то же самое, что градиенты NDCG.



LambdaRank

Заметим, что нам сама ошибка не нужна, а нужны только градиенты λ (стрелочки).

$$\lambda_{ij} = \frac{\partial C(s_i - s_j)}{\partial s_i} = -\frac{\alpha}{1 + e^{\alpha(s_i - s_j)}} |\Delta_{NDCG}|$$

$|\Delta_{NDCG}|$ – это абсолютное изменение метрики NDCG, при обмене позиций документов D_i и D_j

То есть мы считаем градиенты уже после сортировки документов по оценкам, и градиенты как будто от NDCG.

MART

MART

Делаем бустинг на регрессионных деревьях.

Итоговая модель будет искать

$$F_M(x) = \sum_{m=1}^{M} \alpha_m f_m(x)$$

где $f_m(x)$ задается регрессионным деревом,

а $\alpha_m \in \mathbb{R}$ – веса бустинга.

Строим очередное дерево так, чтобы $f_{n+1}(x)$ было бы производной функции ошибки по отношению к текущей модели, посчитанной в каждой точке обучающей выборки.

LambdaMart

LambdaMART

- Комбинация идей LambdaRank и Mart.
- С градиентами все просто: $\overline{y_i} = \lambda_i$
- Как и в MART, для вычисления сплитов используется метод наименьших квадратов.
- В LambdaMART каждое дерево моделирует λ_i для всего датасета, а не для отдельных запросов.
- Это означает, в частности, что LambdaMART может выбирать сплиты и значения в листах, которые могут снизить полезность для некоторых запросов, однако в общем справляться с задачей лучше.

Материалы

1. *From RankNet to LambdaRank to LambdaMART: An Overview*
2. *Winning The Transfer Learning Track of Yahoo!’s Learning To Rank Challenge with YetiRank*
3. *Which Tricks Are Important for Learning to Rank?*

ДЗ

В этом ДЗ вам предстоит:

- Обучить модель, которая побьет бейзлайн в контесте на платформе Kaggle
- Предоставить код, который подтверждает выбитые скоры

Ссылка на контест

Инвайт линк

Спасибо
за внимание!