"Глиняные методы ранжирования"

Автор: Егор Ткаченко

June 2022

1 Pairwise ranking

В прошлой лекции мы остановились на поточечном (Pointwise) методе ранжирования.
Вида $X = (q_i, d_i, y_i)_i^l$. - запрос, документ и релевантность. Все просто - предсказываем $y_i = a(q_i, d_i)$.
Как обсуждалось ранее, этот метод имеет ощутимый минус - мы предсказываем меру релевантности, в то время, как нам нужен всего лишь порядок. Возникает потребность в более специфичных методах, которые будут более точными и конкретными.

2 RankNet

Pairwise метод: рассмотрим множество объектов $R = (i_k, j_k)_k^l$. Имеем некое количество таких пар. Если такая пара входит в R, то для $(i,j) \in R$, справедливо $a(x_i) < a(x_j)$. Т.е j-ый объект имеет более высокую позицию в нашем ранжировании относительно i-го объекта.
Запишем следующий функционал:

$$\mathbb{L} = \sum_{(i,j)\in R} [a(x_j) - a(x-i) < 0]$$

. Тоесть мы штрафуем, если j-ый объект получил скор меньше, чем i-ый объект.
Мы не можем минимизировать это явным образом, т.к индикатор, но мы умеем работать с такими вещами. Мы заменим сумму индикаторов на верхнюю оценку, как и раньше.

$$\sum_{(i,j)\in R} [a(x_j) - a(x-i) < 0] \leqslant \sum_{(i,j)\in R} \tilde{\mathcal{L}}(a(x_j) - a(x_i)) \to \min_a$$

.
Где оценка сверху - $\tilde{L}(z) = \log(1 + e^{-\sigma z}), \sigma \in \mathbb{R}_+$

В этом методе мы берем $a(x) = \langle w, x\rangle, \tilde{\mathcal{L}}$, обучаем нашу модель с помощью SGD. В итоге получаем следующую формулу обновления весов:

$$w := w + \eta \frac{\sigma(x_j - x_i)}{1 + exp(\sigma\langle x_j - x_i, w\rangle)}$$

Дальше была придумана одна очень хорошая эмпирическая штука. Она позволила перейти от решения задачи про долю дефектных парк, к задаче оптимизации некоторой специфичной метрики качества ранжирования.

3 LambdaRank

Что если хотим оптимизировать DCG ?
Отличие DCG от нашей попарной оптимизации из начала конспекта в том, что в DCG нам важны только пары, которые должны стоять высоко. Нижние объекты нас слабо интересуют. Если для

двух объектов мы знаем, что они обы должны стоять где-то нгде-то внизу - нам не важен их порядок относительно друг друга с точки зрения DCG. Чтобы это учесть предлагается добавить в нашу формулу обновления весов множитель $| \triangle F_{ij}|$.
Этот множитель показывает насколько изменится целевая метрика, если мы поменяем x_i и x_j местами. Если нам попадается эта пара, то от того, что мы поменяем их местами в ранжировании наша метрика не изменится. Поэтому мы просто домножим шаг на 0, и скажем модели не подгонять модель под эту пару (x_i, x_j).

4 DSSM

Пусть на входе имеется запрос q и документ d. Далее поверх них мы наворачиваем какую-то нейросеть. Например Embedding-слой, Encoder, RNN и так далее. Что угодно, что на выходе дает нам вектор, который является представлением запроса или документа.
На выходе имеем два вектора: v_q и v_d. Мы считаем между ними косинусное расстояние:

$$\frac{\langle v_q, v_d \rangle}{\|v_q\| \|v_d\|} = a(q, d)$$

Функция потерь для этого метода:
Обозначим вероятность того, что документ d релевантен для запроса q:

$$p(d|q) = \frac{\exp(\sigma \cdot a(q, d))}{\sum\limits_{d' \in \mathcal{D}} \exp(\sigma \cdot a(q, d'))}, \sigma \in \mathbb{R}_+$$

$$L : -\log \prod_{(q,d) \in R} p(d|q) \to min, \text{ где R - множество кликов.}$$

Заметим, что как правило у нас очень большое множество документов, так что напрямую посчитать $\sum\limits_{d' \in \mathcal{D}} \exp(\sigma \cdot a(q, d'))$ не выйдет. Как нам с этим справиться? Изменим формулу:

$$p(d|q) = \frac{\exp(\sigma \cdot a(q, d))}{\exp(\sigma \cdot a(q, d)) + \sum\limits_{q, d_-} \exp(\sigma \cdot a(q, d_-))}$$

, где d_- - подмножество документов, из множества некликнутых документов. Чтобы не насемплировать ну совсем нерелевантные документы в D_-, мы берем документ с вероятностью, пропорциональной a(q, d).

5 ListWise ranking

5.1 ListNet

Допустим у нас есть конкретный запрос q и в ответ на этот запрос мы должны отранжировать какое-то количество документов: $\{d_1, d_2, ..., d_{n_q}\}$. Также мы имеем истинные релевантности для этих документов относительно запроса q: $\{y_1, y_2, ...y_{n_q}\}$. Допустим мы получили оценки релевантности от нашей модели: $\{z_1, z_2, ...z_{n_q}\}$. Нам нужно задать функцию потерь, чтобы обучать параметры такой модели.

Предлагается следующая концепция: когда модель выдает скоры, она выдает конкретную сортировку этих документов. Предположим, что у нас умная модель и вместо одной сортировки она выдает распределение вероятностей на всех перестановках наших документов.

Зададим распределение:

$$P_z(\pi) = \prod_{j=1}^{n} \left(\frac{\phi(z_{\pi_j})}{\sum\limits_{k=j}^{n_q} \phi(z_{\pi_k})} \right), \text{ где } \pi_j \text{ - на какую позицию в } \pi \text{ встает j-ый документ.}$$

А $\phi(z)$ - неубывающая, строго положительная функция. ($\exp(z)$). Это распределение имеет следующие свойства:

1. Это распределение (Т.е сумма вероятностей по нему = 1).
2. $P_z(\pi)$ - действительно отражает выходы нашей модели. Просто она их сглаживает.
   Допустим перестановка π ставит x_i, выше x_j, при этом согласно нашей модели: $z_i > z_j$.
   Тогда если мы поменяем местами i-ый и j-ый объекты, то вероятность такой конкретной перестановки уменьшится.
3. Максимальная $P_z(\pi)$ у перестановки, сортирующей документы в точности по убыванию z_i

Рассмотрим $P_z(\cdot)$ - сглаженный выход модели. Мы можем посчитать такое же распределение для правильных релевантностей - $P_y(\cdot)$.
Очевидно, что мы хотим, чтобы эти распределения были максимально близки друг к другу. Что мы делаем, когда хотим измерить расстояние между распределениями? Правильно - считаем KL дивегренцию.
Формула KL дивергенции: $KL(p||q) = \sum\limits_{c=1}^{M} p_c \log \frac{p_c}{q_c}$

Получим функционал для модели: $KL(P_y||P_z) \to \min$. В чем его проблема? Для его подсчета мы вычисляем сумму по всем перестановкам. Получается, что при n_q документах в запросе у нас получается $n_q!$ слагаемых, что очень много.

Предлагается вместо распределения на всех перестановках рассмотреть вероятность документа попасть на первое место. И ранжировать уже по нему.

$$P_z(j) = \frac{\phi(z_j)}{\sum\limits_{k}^{n_q} \phi(z_k)}$$

Получим итоговый функционал:

$$-\sum_{j}^{n_q} P_y(j) \cdot \log P_z(j) \to min.$$

5.2 SoftRank

В этом методе будем считать что у нас есть какая-то метрика качества ранжирования. Мы хотим опять ввести какую-то вероятность.
Допустим мы имеем скор модели $a(q_i, d_j)$. Давайте скажем что это не число, а нормальное распределение вокруг того, что выдала модель $\mathcal{N}(a(q, d_j), \sigma^2)$. Дальше скажем, что мы, пользуясь распределениями, можем посчитать вероятность того, что i-ый документ получит скор больше
, чем j-ый. $\pi_{ij} = P(s_i > s_j) = \mathbb{P}(s_i - s_j > 0)$, где $(s_i - s_j)$ - случайная величина, распределенная, как $\mathcal{N}(a(q, d_i) - a(q, d_j), 2\sigma_s^2)$

$$\mathbb{P}(s_i - s_j > 0) = \int_0^\infty \mathcal{N}(a(q, d_i) - a(q, d_j), 2\sigma_s^2) ds \text{ - вероятность, что } d_i \text{ окажется выше } d_j.$$

Зная это мы хотим посчитать распределение для r_j - позиция по котором у встает d_j в ранжировании по модели. На это можно смотреть так: есть j-ый документ. Он соревнуется с каждым другим документом. И то, сколько раз он проиграл - на такую позицию он и встает.
Это называется: Rank-Binomial distribution - количество успехов в $n-1$ соревновании, у каждого из которых своя вероятность. Вероятности вычисляются итерационно:

$\mathbb{P}_j^{(1)}(r) = [r=0]$ - вероятность для d_j стоять на первом месте
$\mathbb{P}_j^{(i)}(r) = \mathbb{P}_j^{(i-1)}(r-1)\pi_{ij} + \mathbb{P}_j^{(i-1)}(r)(1-\pi_{ij})$ - вероятность для d_j стоять на i-ом месте

Рассмотрим эти формулы подробнее:

В каком случае j-ый документ попадет на позицию r? Если у нас j-ый документ стоял на одну позицию выше $(r-1)$, затем появляется i-ый документ и побеждает j-ый документ. Это приводит к тому, что j-ый документ опускается на позицию $r: \mathbb{P}_j^{(i-1)}(r-1)\pi_{ij}$
Либо же он может и до этого стоять на позиции r и победить i-ый документ. В этом случае его позиция не меняется: $\mathbb{P}_j^{(i-1)}(r)(1-\pi_{ij})$

Запишем метрику, которую хотим оптимизировать: DCG

$$DCG@k(q) : \sum_{i=1}^{k} g(y_{(i)})d(i), \text{ где g(y) } = 2^y,\ d(i) = \frac{1}{\log(i+1)}$$

Напомним что эта метрика требует, чтобы у нас на высоких позициях были релевантные документы. Если релевантный документ с большим y оказывается на высокой позиции, то мы его почти не штрафуем. В случае, если он оказывается низко, то штраф получается большой. Значение функции получается меньше, а нам нужно ее максимизировать.

$$DCG@k(q) : \sum_{i=1}^{k} g(y_i)d(r_i)$$

Давайте теперь согласно тому распределению, что у нас есть мы вместо обычного DCG посчитаем его матожидание.

$$\mathbb{E}[DCG@k(q)] = \sum_{i=1}^{k} g(y_i)\mathbb{E}[d(r_i)]$$

А $d_i(r_i)$ - распределение рангом мы знаем. Мы его посчитали. Получаем:

$$\sum_{i=1}^{k} g(y_i) \sum_{r=0}^{n_q} d(r) \cdot p_i(r), \text{ где } p_i(r) \text{ - вероятность, что i-ый документ встанет на позицию r.}$$

6 YetAnotherMethod (Pi-Rank)

$$DCG@k(q) : \sum_{i=1}^{k} \frac{g(y_{(i)})}{\log(1+i)} = \sum_{i=1}^{k} \frac{[P_z(g)]_i}{\log(1+i)}$$

P_z - матрица перестановки соответсвующей сортировки объектов по $z_i = a(q, d_i)$
Идея: сгладить перестановочную матрицу, так как изначально она разреженная и бинаризированная.

Заведем унимодальную матрицу A= $(a_{i,j})_{i,j=1}^n$ такая что: $\sum\limits_{j=1}^{n} a_{i,j} = 1$. И мы потребуем, чтобы максимальный элемент в каждой строке находился в уникальном столбце. Наша идея в том, что выбирая построчные максимумы в такой матрице мы получаем такую же перестановку, как и в оригинальной матрице P_z. Далее на лекции говорилось о существовании формул, которые позволяют выписать такую матрицу A, но они настолько глиномесные, что даже в саму лекцию по глиномесным методам не попали. Поверим наслово. Далее просто берем эту параметризацию и оптимизируем.