Лекция 24
Обучение ранжированию

Е. А. Соколов
ФКН ВШЭ

17 апреля 2020 г.

В задаче ранжирования требуется построить модель, которая правильным образом задаёт порядок на объектах. Сюда относится, например, задача поискового ранжирования, где документы сортируются по убыванию релевантности (соответствия) запросу. Также к ранжированию может быть сведена задача построения рекомендаций (отсортировать товары для данного пользователя), исправления опечаток (выбрать наиболее подходящие варианты исправления) и т.д. Ниже мы увидим, что задачу ранжирования можно свести к классификации или регрессии — но также рассмотрим и более сложные подходы, которые оптимизируют непосредственно качество сортировки.

1 Постановка задачи

Пусть задано пространство объектов $\mathbb{X}$ (например, $\mathbb{R}^d$) и некоторая обучающая выборка $X = \{x_1, \ldots, x_\ell\} \subset \mathbb{X}$. Также пусть дан некоторый порядок на объектах обучающей выборке — то есть набор таких пар $(i, j) \in R \subset \{1, \ldots, \ell\}^2$, что первый объект из пары должен стоять после сортировки моделью выше второго объекта. Множество пар R заменяет собой целевую переменную.

Например, каждый объект может представлять собой пару (q, d), состоящую из запроса и документа. В этом случае порядок будет задан только на таких парах (q_1, d_1) и (q_2, d_2), которые соответствуют одному запросу ($q_1 = q_2$). Далее мы будем рассуждать именно в терминах такого поискового ранжирования.

Требуется построить такую модель $a : \mathbb{X} \to \mathbb{R}$, что для $(i, j) \in R$ (и только для них) выполнено $a(x_i) < a(x_j)$. Разумеется, на больших выборках вряд ли получится идеально выполнить это требование, поэтому необходимо ввести метрику, определяющую качество решения задачи.

2 Метрики качества ранжирования

Часто в ранжировании ответы задаются не в виде пар, а, для простоты, в виде чисел $y_1, \ldots, y_\ell$. При этом считается, что если $y_i < y_j$, то для модели должно быть выполнено $a(x_i) < a(x_j)$.

Для начала рассмотрим простой случай бинарных ответов $y_i \in \{0, 1\}$ — грубо говоря, каждый документ либо соответствует запросу, либо не соответствует. В этом случае можно применять любые стандартные метрики качества классификации — точность, полноту, F-меру, AUC-ROC и т.д. Как правило, их вычисляют в рамках одного запроса, и затем усредняют по всем запросам из выборки.

Рассмотрим для примера точность (precision) на одном запросе q. Поскольку нас в первую очередь интересует, какие документы оказываются в самом верху поисковой выдачи, логично рассматривать метрику precision@$k(q)$ — точность, вычисленную для документов, которые модель поместила на первые k мест. Данная метрика будет равна единице, если все k документов релевантные, нулю, если они все нерелевантны. При этом она никак не учитывает порядок внутри первых k позиций — релевантный документ и на первой, и на k-й позиции имеет одинаковый вклад. Чуть более сложной метрикой является AP(average precision):

$$\text{AP@}k(q) = \sum_{i=1}^{k} \frac{y_{(i)}}{\sum_{j=1}^{k} y_{(j)}} \text{precision@}i(q),$$

где $y_{(i)}$ — релевантность документа на i-й позиции. В ней уже учитывается порядок, и документ на первой позиции имеет больший вес. Значение AP, усреднённое по всем запросам, называется MAP (mean average precision).

Если ответы являются вещественными (например, при наличии нескольких уровней релевантности), то можно использовать метрику DCG (discounted cumulative gain):

$$\text{DCG@k}(q) = \sum_{i=1}^{k} g(y_{(i)}) d(i).$$

Примерами конкретных функций могут служить $g(y) = 2^y - 1$ и $d(i) = \frac{1}{\log(i+1)}$. Чтобы значение метрики легче было интерпретировать, её можно поделить на значение DCG при идеальном ранжировании — в этом случае получим метрику nDCG (normalized DCG):

$$\text{nDCG@k}(q) = \frac{\text{DCG@k}(q)}{\max \text{DCG@k}(q)}.$$

Далее значение nDCG можно усреднить по всем запросам.

Ещё один пример метрики ранжирования — это pFound, предложенная в компании Яндекс. Пусть ответы лежат на отрезке $[0, 1]$ и отражают вероятность найти ответ в данном документе. Зададим величину p_i, соответствующую вероятности дойти до i-й позиции. Для первой позиции она равна единице, поскольку пользователь точно посмотрит на первый документ: $p_1 = 1$. Вероятность дойти до $(i+1)$-й позиции вычисляется как

$$p_{i+1} = p_i(1 - y_{(i)})(1 - p_{\text{out}}),$$

где p_{out} — вероятность того, что пользователь уйдёт, не узнав ответ на свой запрос. Метрика pFound равна вероятности найти ответ среди первых k документов:

$$\text{pFound@}k(q) = \sum_{i=1}^{k} p_i y_{(i)}.$$

Далее она, как и другие метрики, усредняется по всем запросам. Отметим, что pFound является *каскадной* — она учитывает, что пользователь просматривает поисковую выдачу сверху вниз, и что польза документа зависит от документов выше него.

3 Признаки в моделях ранжирования

В задачах поискового ранжирования выделяют три типа признаков:

- Запросные — зависят только от запроса. Сюда может относиться, например, популярность запроса, его тип (навигационный, товарный и т.д.), число слов в нём.
- Статические — зависят только от документа и могут быть рассчитаны заранее. Сюда могут относиться популярность документа (число ссылок на него в сети), его тематики, распределение слов в нём, средний word2vec-вектор и т.д.
- Динамические — зависят от запроса и документа. Сюда могут относиться, например, различные расстояния между запросом и документом.

Разберём несколько популярных поисковых признаков.

BM25 Документ и запрос можно сравнить, например, путём подсчёта косинусного расстояния между их TF-IDF-представлениями. Более общим способов вычисления близости является функция Okapi BM25. Пусть запрос q состоит из слов $q_1, \dots, q_n$. Тогда его сходство с документов вычисляется как

$$\mathrm{BM25}(q, d) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \frac{\mathrm{tf}(q_i, d)(k_1 + 1)}{\mathrm{tf}(q_i, d) + k_1 \left(1 - b + b \frac{|D|}{\bar{n}_d}\right)},$$

где $\mathrm{tf}(q_i, d)$ — число вхождений слова q_i в документ d, $|D|$ — число документов в выборке, $\bar{n}_d$ — средняя длина документа, а IDF (inverse document frequency) может вычисляться по формуле

$$\mathrm{IDF}(q_i) = \log \frac{|D|}{|\{d \in D \,|\, q_i \in d\}|},$$

т.е. как доля документов, содержащих данное слово. Величины b и k_1 являются параметрами.

Метрика BM25 выводится из определённых вероятностных предположений о релевантности документов запросам, но мы не будем останавливаться на них в данном тексте.

PageRank Алгоритм PageRank позволяет найти для каждого документа величину, характеризующую его «важность». Документы в сети ссылаются друг на друга, образуя граф. Документ считается важным, если на него ссылается много документов,

которые, в свою очередь, мало на кого ссылаются. Формально PageRank для документа d задаётся как

$$\mathrm{PR}(d) = \frac{1-\delta}{|D|} + \delta \sum_{c \in D_d^{\mathrm{in}}} \frac{\mathrm{PR}(c)}{|D_c^{\mathrm{out}}|}, \tag{3.1}$$

где D — множество всех документов, D_d^{in} и D_d^{out} — множества документов, от которых и к которым ведут рёбра из d соответственно.

Данная формула, по сути, отражает вероятность попасть на документ d при случайном блуждании по сети. Согласно ней, пользователь стартует из некоторого документа и либо переходит по одной из ссылок в нём с равными вероятностями, либо с вероятностью δ переходит на случайную страницу из сети.

Уравнения (3.1) можно переписать в векторном виде:

$$R = \frac{1-\delta}{|D|} + \delta A R,$$

где A — модифицированная матрица смежности, где $a_{ij} = \frac{1}{|D_j^{\mathrm{out}}|}$, если j-й документ ссылается на i-й, и $a_{ij} = 0$ в противном случае. Через R здесь обозначен вектор $(\mathrm{PR}(d_1), \ldots, \mathrm{PR}(d_|D|))$. Отсюда получаем, что вектор R можно найти путём обращения регуляризованной матрицы смежности:

$$R = (I - \delta R)^{-1} \frac{1-\delta}{|D|} \vec{1}.$$

Поскольку обращать матрицу смежности может быть слишком сложно, можно искать вектор R итерационно, инициализировав его случайным образом и пересчитывая значения по формулам (3.1). Такой подход является примером использования метода простых итераций.

4 Методы ранжирования

На предыдущей лекции мы договорились, что ответы задаются не в виде пар, а в виде чисел $y_1, \ldots, y_\ell$. При этом считается, что если $y_i < y_j$, то для модели должно быть выполнено $a(x_i) < a(x_j)$.

Также мы обсудили ряд метрик качества ранжирования — среди них MAP, nDCG и pFound. Все они непосредственно учитывают порядок документов, и поэтому зависят от позиций, на которые их поместила модель. Поскольку зависимость позиций документов от параметров модели является кусочно-постоянной (позиции ведь принимают натуральные значения), непосредственно обучаться с данными метриками достаточно сложно. Поэтому многие методы ранжирования пытаются оптимизировать другие метрики, которые являются дифференцируемыми и при этом как-то оценивают качество ранжирования.

§4.1 Поточечные методы

В поточечном (pointwise) подходе предлагается забыть про то, что мы решаем задачу ранжирования, и независимо для каждого объекта x_i предсказывать ответ y_i. В зависимости от типа ответов получим задачу классификации или регрессии.

Если модель $a(x)$, которая получится в результате такого обучения, будет идеально восстанавливать целевую переменную, то и метрика качества ранжирования будет оптимизирована. Если же добиться идеального восстановления целевой переменной нельзя, то могут возникнуть проблемы — ведь поточечная метрика никак не учитывает порядок. С её точки зрения необходимо как можно точнее восстановить ответы, тогда как с точки зрения метрики ранжирования можно пожертвовать точностью предсказания ради корректности порядка (скажем, прогнозы 3 и 4 не будут отличаться от прогнозов 3.9 и 3.95 в плане качества ранжирования друг относительно друга).

§4.2 Попарные методы

Вспомним, что изначально мы определяли задачу ранжирования через пары объектов. Если записывать это формально, то получим функционал ошибки

$$\sum_{(i,j)\in R} [a(x_j) - a(x_i) < 0],$$

где R — множество пар, для которых известен порядок. Этот функционал не является дифференцируемым, но мы уже решали такую проблему в линейной классификации — надо заменить индикатор ошибки $[z < 0]$ на его гладкую верхнюю оценку $L(z)$:

$$\sum_{(i,j)\in R} [a(x_j) - a(x_i) < 0] \leqslant \sum_{(i,j)\in R} L\left(a(x_j) - a(x_i)\right).$$

В качестве оценки $L(z)$ можно брать, например, логистическую функцию $L(x) = \log(1+e^{-\sigma z})$ с параметром $\sigma > 0$ — в этом случае получим метод RankNet. Оптимизировать данный функционал можно обычным стохастическим градиентным спуском. Если использовать линейную модель $a(x) = \langle w, x\rangle$, то один шаг будет иметь вид

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

Существует эмпирическое наблюдение, позволяющее перейти к оптимизации произвольной метрики ранжирования F. Оказывается, для этого надо домножить смещение на изменение метрики ΔF_{ij}, которое произойдёт при перестановке x_i и x_j местами в ранжировании:

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

Данный метод носит название LambdaRank.

Также по аналогии с методом опорных векторов можно вывести метод RankSVM:

$$\begin{cases} \dfrac{1}{2}\|w\|^2 + C\sum\limits_{(i,j)\in R} \xi_{ij} \to \min\limits_{w,\xi} \\ \langle w, x_j - x_i\rangle \geqslant 1 - \xi_{ij}, \quad (i,j) \in R; \\ \xi_{ij} \geqslant 0, \quad (i,j) \in R. \end{cases}$$

Отметим, что нередко именно попарный подход оказывается лучшим при решении задачи ранжирования.

§4.3 Списочные методы

Мы уже отмечали выше, что непосредственно оптимизировать метрику качества ранжирования вряд ли получится из-за её дискретной структуры. Такая проблема возникала и раньше (например, при обучении метрик или при визуализации), и решали мы её путём введения некоторого вероятностного распределения. Разберём метод ListNet, который позволяет непосредственно учитывать порядок объектов в процессе обучения.

Рассмотрим один запрос q, для которого надо отранжировать n_q документов $(d_1, \dots, d_{n_q})$. Для этих документов известны истинные оценки релевантности $(y_1, \dots, y_{n_q})$, которые определяют истинное ранжирование. Пусть также имеет некоторая модель $a(x)$, которая выдаёт оценки $(z_1, \dots, z_{n_q})$. Метрика качества ранжирования (например, nDCG) измеряет, насколько ранжирования по оценкам модели близко к истинному ранжированию.

В постановке, которую мы только что описали, модель выдаёт одну конкретную перестановку документов для данного запроса, и мы измеряем качество этой перестановки. Сгладим этот процесс: предположим, что на самом деле модель выдаёт распределение на всех возможных перестановках документов, причём вероятность конкретной перестановки π определяется как

$$P_z(\pi) = \prod_{j=1}^{n_q} \frac{\varphi(z_{\pi(j)})}{\sum_{k=j}^{n_q} \varphi(z_{\pi(k)})},$$

где φ — произвольная неубывающая и строго положительная функция. Данные вероятности обладают рядом полезных свойств:

- Вероятности $P_z(\pi)$ задают распределение на множестве всех перестановок n_q элементов.
- Пусть перестановка π ставит объект x_i выше объекта x_j, и $z_i > z_j$. Если поменять эти объекты местами в перестановке (то есть поставить x_j выше, чем x_i), то новая перестановка будет иметь меньшую вероятность, чем старая. Иными словами, чем ближе перестановка к оптимальной, тем выше её вероятность.
- Максимальную вероятность имеет перестановка, которая сортирует объекты по убыванию z_i; минимальную вероятность имеет обратная к ней перестановка.

Таким образом, данные вероятности отдают предпочтение тем перестановкам, которые ближе к сортировке объектов по предсказаниям алгоритма. Значит, мы действительно получим способ «сглаживания» ранжирования документов по прогнозам z_i.

Всего перестановок объектов $n_q!$, и посчитать по ним всем матожидание не представляется возможным. Чтобы упростить подсчёты, рассмотрим вероятность $P_z(j)$ попадания объекта x_j на первое место после перестановки. Можно показать, что они вычисляются по формуле

$$P_z(j) = \frac{\varphi(z_j)}{\sum_{k=1}^{n} \varphi(s_k)}$$

Данные вероятности образуют распределение на всех n_q документах. Также для объектов с $z_i > z_j$ выполнено $P_z(i) > P_z(j)$ — то есть введённые вероятности задают на объектах порядок, совпадающий с ранжированием по оценкам модели.

Теперь мы можем сравнить истинное ранжирование документов и ранжирование по оценкам модели, посчитав кросс-энтропию между соответствующими им распределениями:

$$Q(y, z) = -\sum_{j=1}^{n_q} P_y(j) \log P_z(j).$$

Если взять, например, $\varphi(x) = \exp(x)$, то кросс-энтропия будет дифференцируема по оценкам модели — а значит, можно найти производные и по параметрам модели. Благодаря сглаживанию мы смогли ввести функционал, который отражает требования к перестановке объектов, и при этом позволяет обучение градиентными методами.