Машинное обучение, осень 2016. Никита Волков

Ранжирование

Лекция

1. Задача ранжирования

Пусть D — некоторая коллекция текстовых документов и Q — множество запросов. Обозначим D_q неупорядоченный набор документов, потенциально релевантных запросу $q \in Q$. Основная задача ранжирования — упорядочить документы внутри D_q по убыванию степени их релевантности запросу, то есть более релевантные документы должны иметь более высокий ранг.

Пример. Задача ранжирования — основа поисковых систем. В наших обозначениях множество D — база документов, проиндексированных поисковой системой. Пользователи задают поисковой системе запросы из множества Q. По запросу $q \in Q$ поисковая система в своей базе D должна найти все документы D_q, которые соответствуют заросу q. После того, как документы D_q найдены, поисковой системе нужно упорядочить их по убыванию степени их релевантности запросу. Важность последнего этапа поиска состоит в том, что часто на запросы находятся сотни тысяч различных документов, и ни один пользователь не будет просматривать их всех чтобы найти нужный. Скорее всего пользователь просмотрит не более 10 первых документов. Как следствие, поисковая система должна качественно делать ранжирование документов по степени их релевантности запросу.

Отношение порядка на парах запрос-документ. Если документ d_1 релевантнее документа d_2 по запросу $q \in Q$ (то есть $d_1, d_2 \in D_q$), то будем писать $(q, d_1) \triangleright (q, d_2)$. Таким образом определяется порядок на парах запрос-документ. Отметим, что этот порядок определен только внутри одного конкретного запроса.

Специальные люди, называемые асессорами, для некоторых запросов размечают пары запрос-документ, определяя для них *оценку* релевантности $y(q, d)$. Обозначим X — множество пар (d, q), для которых определена оценка релевантности $y(q, d)$, и назовем их асессорской выборкой. Стоит отметить, что в настоящее время процедура оценивания релевантности ассессорами достаточно хорошо формализована в виде списка многочисленных правил.

Задача. Чтобы упорядочить пары запрос-документ, на основе асессорской выборки будем искать *функцию релевантности* $\alpha(q, d)$, которая "как можно лучше" удовлетворяет условию

$$(q, d_1) \triangleright (q, d_2) \quad \Leftrightarrow \quad \alpha(q, d_1) > \alpha(q, d_2).$$

2. Метрики качества

Различные ранжирования пар запрос-документ сравниваются при помощи метрик качества ранжирования. Для измерения качества ранжирования традиционно (традиция задана конференцией TREC) используются меры точности. Приведем описание некоторых из них.

Mean Average Precision

Метрика Mean Average Precision (MAP) предполагает, что документы по запросу могут иметь только две оценки — релевантный и нерелевантный. Пусть $y(q, d)$ — индикатор

того, что документ d релевантен запросу q, т.е. $y(q,d) = 1$ тогда и только тогда когда ассессоры считают, что документ d релевантен запросу q (в противном случае $y(q,d) = 0$).

Зафиксируем запрос q и упорядочим все документы по этому запросу в соответствии со значением $\alpha(q,d)$, получив набор $\{d_q^{(i)}\}_i$. Метрика Precision (точность) определяется как доля релевантных документов среди первых n:

$$P_n(q) = \frac{1}{n}\sum_{i=1}^{n} y\left(q, d_q^{(i)}\right).$$

Метрика Average Precision (средняя точность) определяется как средняя Precision по позициям релевантных документов

$$AP(q) = \frac{\sum\limits_n P_n(q)\ y\left(q, d_q^{(n)}\right)}{\sum\limits_n y\left(q, d_q^{(n)}\right)}.$$

Усреднение Average Precision по всем запросам дает Mean Average Precision

$$MAP = \frac{1}{|Q|}\sum_{q\in Q} AP(q).$$

NDCG

Предыдущая метрика плоха тем, что она не учитывает степень релевантности документа. Исправить ситуацию помогает метрика $NDCG$. В отличии от метрики MAP, метрика NDCG не накладывает огрничений на возможные значения оценки релевантности $y(q,d)$.

Зафиксируем запрос q и упорядочим все документы по этому запросу в соответствии со значением $\alpha(q,d)$, получив набор $\{d_q^{(i)}\}_i$. Метрика DCG (Discounted Cumulative Gain, взвешенная сумма выигрышей) определяется как

$$DCG_n(q) = \sum_{i=1}^{n} G_q\left(d_q^{(i)}\right)\ D(i),$$

где величина $G_q(d) = 2^{y(q,d)} - 1$ придает больший вес релевантным документам, а величина $D(i) = 1/\log_2(i+1)$ придает больший вес документам, которые функция ранжирования $\alpha(q,d)$ посчитала более релевантными.

Значения получаемой метрики сильно зависят от запроса. Чтобы снизить эту зависимость, значение DCG нормируют, получая NDGC — нормированный DCG

$$NDCG_n(q) = \frac{DCG_n(q)}{\max DCG_n(q)},$$

где максимум берется по всем возможным ранжированиям. Очевидно, что он достигается, если построенная функция релевантности задает тот же порядок, что и асессорские оценки релевантности.

pFound

Все рассмотренные выше метрики построены из некоторых общих соображений. Метрика $pFound$ разработана в Яндексе на базе эмпирических исследований поведения пользователей. Она аппроксимирует вероятность того, что пользователь удовлетворится результатами поиска по запросу. Как и NDCG, метрика $pFound$ учитывает релевантность документа и его позицию в результатах поиска. Кроме того, она учитывает наличие релевантных документов ранее в списке результатов, ведь, найдя первый ответ на свой вопрос, пользователь обычно не просматривает результаты дальше. В данном случае предполагается, что $y(q,d) \in \{0,...,4\}$, причем эти значения имеют определенное название: Vital, Usefull, Relevant+, Relevant-, Not relevant.

Модель поведения пользователя включает следующие предположения

1. с пользователем могут случиться два события при просмотре каждого документа, после которых он не будет продолжать просмотр поисковой выдачи — он может удовлетвориться найденным или, наоборот, он может отчаяться что-либо найти;

2. вероятности этих событий не зависят от количества просмотренных документов и зависят только от релевантности последнего документа;

3. вероятность того, что пользователь отчаялся искать дальше, не зависит от релевантности документов;

4. пользователь не может удовлетвориться нерелевантным документом.

Исходя из этих предположений и на основе экспериментов была получена следующая зависимость от оценки асессоров вероятности $p(q,d)$ того, что пользователь удовлетворится документом d по запросу q

- Vital — 0.61,
- Usefull — 0.41,
- Relevant+ — 0.14,
- Relevant- — 0.07,
- Not relevant — 0.

Вероятность того, что пользователь прекратит поиск получилась равной $P_{break} = 0.15$ на каждом документе.

Пусть набор документов $\{d_q^{(i)}\}_i$ упорядочен в соответствии со значением $\alpha(q,d)$. Тогда вероятность того, что пользователь при просмотре поисковой выдаче дойдет до i-ого документа равна

$$P_i = \left(1 - p\left(d_q^{(i-1)}\right)\right)\left(1 - P_{break}\right) P_{i-1},$$

и $P_1 = 1$. Окончательно,

$$pFound(n) = \sum_{i=1}^{n} P_i \; p\left(d_q^{(i)}\right).$$

3. Методы

RankNet

Пусть $X_{train} = \{X_i, Y_i\}_{i=1}^n$ — обучающая выборка, которая состоит их тех и только тех пар $X_i = (q_i, d_i)$, для которых асессоры определили оценку релевантности $Y_i = y(q_i, d_i)$. Будем искать функцию релевантности в виде функции $f_w(x)$, где $w = (w_k)_k$ — параметры модели, и функция $f_w(x)$ является дифференцируемой по w. В качестве f_w обычно берется нейронная сеть (что и объясняет название метода), но также можно брать композиции деревьев.

Функция $f_w(x)$ порождает оценку вероятностей того, что один документ более релеватный, чем другой по тому же запросу, по следующему правилу

$$P_{ij} = \mathsf{P}(X_i \triangleright X_j) = \frac{1}{1 + e^{-\sigma(s_i - s_j)}},$$

где $s_i = f_w(X_i)$. Пусть так же $Q_{ij} = \mathsf{Q}(X_i \triangleright X_j) = \frac{1}{2}(1 + I\{Y_i > Y_j\} - I\{Y_i < Y_j\})$ — известные вероятности (определяются по оценка асессоров). Отметим так же, что эти формулы справедливы только в том случае, когда $q_i = q_j$. В противном случае порядок между X_i и X_j не определен.

Чтобы подобрать параметры модели w можно минимизировать дивергенцию Кульбака-Лейблера между известным распределением и задаваемым моделью. Пусть p_1 и p_2 — некоторые плотности по одной и той же мере (обе дискретные или обе абсолютно-непрерывные). Дивергенцией между p_1 и p_2 называется величина $KL(p_1||p_2) = \mathsf{E} \log \frac{p_1(\xi)}{p_2(\xi)}$, где ξ — случайная величина, имеющая плотность p_1.

Для двух пар запрос-документ (X_i, X_j) функцию потерь по правилу $C_{ij} = KL(\mathsf{Q}_{\xi_{ij}}||\mathsf{P}_{\xi_{ij}})$, $\mathsf{Q}_{\xi_{ij}}$ и $\mathsf{P}_{\xi_{ij}}$ — распределения случайной величины $\xi_{ij} = I\{X_i \triangleright X_j\}$ по мерам Q и P соответственно. Распишем эту функцию потерь

$$C_{ij} = KL(\mathsf{Q}_{\xi_{ij}}||\mathsf{P}_{\xi_{ij}}) = Q_{ij} \log \frac{Q_{ij}}{P_{ij}} + (1 - Q_{ij}) \log \frac{1 - Q_{ij}}{1 - P_{ij}} =$$

$$= -H(\mathsf{Q}_{\xi_{ij}}) - Q_{ij} \log P_{ij} - (1 - Q_{ij}) \log(1 - P_{ij}),$$

где $H(\mathsf{Q}_{\xi_{ij}})$ — энтропия, которая не зависит от распределения P, а значит и от модели.

Посмотрим на некоторые особенности такой функции потерь. Пусть $Y_i > Y_j$, тогда $Q_{ij} = 1$ и $C_{ij} = \log\left(1 + e^{-\sigma(s_i - s_j)}\right)$. Если же $Y_i < Y_j$, тогда $Q_{ij} = 0$ и $C_{ij} = \log\left(1 + e^{-\sigma(s_j - s_i)}\right)$. Заметим, что обоих случаях $C_{ij} = \log 2$ при $s_i = s_j$, то есть функция потерь включает в себя отступ. Наконец, если $Y_i = Y_j$ и $s_i = s_j$, то $Q_{ij} = P_{ij} = 1/2$ и $C_{ij} = 0$.

Теперь мы можем определить функционал риска $Q(w) = \sum_{i,j:q_i=q_j} C_{ij}$, который мы будем минимизировать по w при помощи градиентного спуска

$$w_k^{t+1} = w_k^t - \eta \sum_{i,j:q_i=q_j} \frac{\partial C_{ij}}{\partial w_k^t} = w_k^t - \eta \sum_{i,j:q_i=q_j} \left(\frac{\partial C_{ij}}{\partial s_i} \frac{\partial s_i}{\partial w_k^t} + \frac{\partial C_{ij}}{\partial s_j} \frac{\partial s_j}{\partial w_k^t} \right).$$

Можно использовать стохастический градиентный спуск — обновлять веса после рассмотрения каждой пары запрос-документ. Это соответствует pair-wise подходу.

Факторизация. Интересная особенность RankNet — возможность факторизации. Для начала заметим, что если $Y_i > Y_j$, то

$$\frac{\partial C_{ij}}{\partial s_i} = -\frac{\sigma}{1 + e^{\sigma(s_i - s_j)}} = -\frac{\partial C_{ij}}{\partial s_j}.$$

Таким образом,

$$\frac{\partial C_{ij}}{\partial w_k} = \frac{\partial C_{ij}}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \frac{\partial C_{ij}}{\partial s_j}\frac{\partial s_j}{\partial w_k} = -\frac{\sigma}{1+e^{\sigma(s_i-s_j)}}\left(\frac{\partial s_i}{\partial w_k} + \frac{\partial s_j}{\partial w_k}\right) = \lambda_{ij}\left(\frac{\partial s_i}{\partial w_k} + \frac{\partial s_j}{\partial w_k}\right),$$

где $\lambda_{ij} = -\frac{\sigma}{1+e^{\sigma(s_i-s_j)}}$. Аналогично можно расписать для случаев $Y_i < Y_j$ и $Y_i = Y_j$.

Определим $I = \{\{i,j\} | X_i \triangleright X_j \text{ или } X_j \triangleright X_i\}$ — множество пар индексов, для которых определен порядок на соответствующих парах запрос-документ. Поскольку RankNet использует вероятности, допускаются циклические зависимости вида $X_1 \triangleright X_2$, $X_2 \triangleright X_3$, $X_3 \triangleright X_1$.

Введем

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

Тогда

$$\frac{\partial C}{\partial w_k} = \sum_i \lambda_i \frac{\partial s_i}{\partial w_k}.$$

В итоге получаем следующую формулу

$$w_k^{t+1} = w_k^t - \eta \sum_i \lambda_i \frac{\partial s_i}{\partial w_k},$$

где $s_i = f_{w^t}(X_i)$.

Смысл лямбд. Мы можем понимать λ_i как небольшие стрелки, которые прикреплены к каждому урлу в поисковой выдаче. Направление стрелки соответствует тому, куда мы хотим переместить данный урл, а ее длина соответствует тому, насколько сильно мы хотим его переместить.

Так же вместо использования стохастического градиентного спуска по парам запрос-документ, мы можем накапливать лямбды для каждого урла, и только затем выполнять обновление весов. Это приводит к очень существенному ускорение обучения RankNet. (процесс обновления весов стоит дорого, например, для модели нейронной сети с обратным распространением ошибки). На самом деле, время обучения сократилось с близкой к квадратичной по числу урлов для каждого запроса, к близкой к линейной.

LambdaRank

Как же нам соединить обучение методом RankNet и оптимизацию некоторой метрики? Можно, например, вычислять ее на валидационной выборке и использовать как критерий остановки. А можно поступить иначе. Давайте придумаем такую функцию потерь $\overline{C}$, с помощью которой можно максимизировать значение метрики. Однако, наша функция потерь является гладкой, в то время как все метрики дискретны, поскольку они зависят только от порядка. Поэтому такую функцию так просто не придумать.

Ключевое наблюдение — нам не нужна сама функция $\overline{C}$, а нужен только ее градиент. Если мы зададим его так, чтобы семшанные производные совпадали, то такая функция $\overline{C}$ будет существовать (лемма Пуанкаре), но явно ее строить нам не нужно.

Обозначим $Z(q,f)$ значение метрики Z для запроса q при ранжировании с помощью функции релевантности f. Пусть f_{ij} — функция релевантности, определенная по правилу

$$f_{ij}(x) = \begin{cases} f(X_j), & \text{если } x = X_i; \\ f(X_i), & \text{если } x = X_j; \\ f(x), & \text{если } x \notin \{X_i, X_j\}. \end{cases}.$$

Определим приращение метрики при перестановке X_i и X_j как $\Delta Z_{ij} = Z(q, f) - Z(q, f_{ij})$.

Тогда определим

$$\lambda_{ij} = -\frac{\sigma}{1 + e^{\sigma(s_i - s_j)}} \left| \Delta Z_{ij} \right|.$$

Но в таком случае вместо функции потерь мы получим функцию полезности, которую будем максимизировать. Это и основная идея метода LambdaRank.

LambdaMART

Метод MART (Multiple additive regression trees) есть градиентный бустинг на основе регрессионных деревьев (GBRT). Фактически, он является частным случаем RankNet для случая, когда f_w есть композиция деревьев. Его модификацей по принципу LambdaRank есть метод LambdaMART. Для полного вывода метода требуется расписывать большое количество достаточно технических формул, поэтому мы приведем только результат.

Композиция строится по принципу $F_t(x) = F_{t-1}(x) + \gamma$

$$\gamma_{km} = \frac{-\sum\limits_{x_i \in R_{km}} \sum\limits_{(i,j) \rightleftharpoons I} \Delta Z_{ij} \dfrac{1}{1 + e^{\sigma(s_i - s_j)}}}{\sum\limits_{x_i \in R_{km}} \sum\limits_{(i,j) \rightleftharpoons I} \sigma \Delta Z_{ij} \dfrac{1}{1 + e^{\sigma(s_i - s_j)}} \left(1 - \dfrac{1}{1 + e^{\sigma(s_i - s_j)}}\right)},$$

где R_{kt} — подвыборка, которая попала в k-й лист t-го дерева.

Данный метод является примером метода обучения ранжированию на основе бустинга на деревьях. Такие методы в настоящее время широко используются поисковыми системами, например, метод MatrixNet Яндекса.

Литература

[1] К. В. Воронцов — Методы обучения ранжированию (Learning to Rank), лекции ШАД — http://www.machinelearning.ru/wiki/images/8/89/Voron-ML-Ranking-slides.pdf

[2] А. Гулин, П. Карпович, Д. Расковалов, И. Сегалович — Оптимизация алгоритмов ранжирования методами машинного обучения, Яндекс на РОМИП'2009. — http://romip.ru/romip2009/15_yandex.pdf

[3] Christopher J.C. Burges — From RankNet to LambdaRank to LambdaMART: An Overview — http://research.microsoft.com/pubs/132652/msr-tr-2010-82.pdf