Постановка задачи

Задача ранжирования естественным образом возникает в разных приложениях, таких как:

- информационный поиск
- рекомендации
- персонализированный feed новостей
- поиск похожих объектов

Первым делом набор данных определяется **множеством запросов:**

$$\mathcal{Q} = \{q^{(1)}, q^{(2)}, \ldots, q^{(m)}\}$$

Стоит понимать, что запрос не обязательно представлен в виде текста. Часто запросом может являться контекст пользователя: его профиль, история взаимодействия. При поиске похожих объектов запросом является сам объект.

Каждому запросу в соответствие ставится некоторое **множество документов**, которые необходимо упорядочить:

$$d^{(i)} = \left(d_1^{(i)}, d_2^{(i)}, \ldots, d_{n^{(i)}}^{(i)}\right)$$

Как правило ранжирующие модели работают поверх простых, но быстрых алгоритмов, отбирающих top кандидатов. В веб поиске это может быть матч по словам запроса на инвертированном индексе, в рекомендациях: быстрое матричное разложение или knn. Хорошими бейзлайнами для ранжирования могут быть:

- tf/idf или bm25 в полнотекстовом поиске
- ALS, lightFM и другие рекомендательные модели
- knn при поиске похожих документов

Известные оценки релевантности документов для запроса $q^{(i)}$:

$$y^{(i)} = \left(y_1^{(i)}, y_2^{(i)}, \ldots, y_{n^{(i)}}^{(i)}\right)$$

Исторически в веб поиске в качестве таргетов используются асессорские оценки релевантности документа запросу. Обычно это небольшой набор лейблов, каждый из которых имеет конкретный смысл, например:

- 3 – vital
- 2 – relevant
- 1 – weakly relevant
- 0 – not relevant

Если $y_j^{(i)} > y_k^{(i)}$, то документ $d_j^{(i)}$ в идеальном ранжировании должен быть показан раньше, чем документ $d_k^{(i)}$, будем обозначать такую ситуацию через:

$$d_j^{(i)} \prec d_k^{(i)}$$

Бывают ситуации, когда нам неизвестны оценки релевантности документов, а известен только частичный порядок на некоторых парах:

$$\{d_j^{(i)} \prec d_k^{(i)}\}_{j,k \subset \{0,\dots,n^{(i)}\} \times \{0,\dots,n^{(i)}\}}$$

В частности такая ситуация возникает, когда у нас нет оценок релевантности, а мы пытаемся достать таргет из лога кликов:

1. Kernel Machines
http://svm.first.gmd.de/
2. Support Vector Machine
http://jbolivar.freeservers.com/
3. SVM-Light Support Vector Machine
http://ais.gmd.de/~thorsten/svm_light/
4. An Introduction to Support Vector Machines
http://www.support-vector.net/
5. Support Vector Machine and Kernel Methods References
http://svm.research.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL.AC.UK
http://www.jiscmail.ac.uk/lists/SUPPORT–VECTOR–MACHINES.html
7. Lucent Technologies: SVM demo applet
http://svm.research.bell-labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
http://svm.dcs.rhbnc.ac.uk/
9. Support Vector Machine - The Software
http://www.support-vector.net/software.html
10. Lagrangian Support Vector Machine Home Page
http://www.cs.wisc.edu/dmi/lsvm

Признаковое описание пары запрос/документ $\left(q^{(i)}, d_j^{(i)}\right)$:

$$x_j^{(i)} = \Psi(q^{(i)}, d_j^{(i)}) \in \mathbb{R}^D$$
$$x^{(i)} = \left(x_1^{(i)}, x_2^{(i)}, \ldots, x_{n^{(i)}}^{(i)}\right)$$

Признаки принято делить на:

- признаки запроса $\psi(q^{(i)})$
- признаки документа $\psi(d_j^{(i)})$
- парные признаки $\psi(q^{(i)}, d_j^{(i)})$

По составу признаки делятся на:

- контентные признаки (tfidf, bm25, etc.)
- поведенческие признаки (ctr, als, etc.)

Обучающая выборка:

$$\mathcal{T} = \left\{ \left(x^{(i)}, y^{(i)}\right) \right\}_{i=1}^{m}$$

Скор ранжирующей функции:

$$s_j^{(i)} = f(x_j^{(i)})$$
$$s^{(i)} = \left(s_1^{(i)}, s_2^{(i)}, \ldots, s_{n^{(i)}}^{(i)}\right)$$

Оптимизируем функцию потерь на запросах:

$$\sum_{i=1}^{m} \mathcal{L}(s^{(i)}, y^{(i)}) \to \min$$

Часто удобно оперировать не значениями скоров и известными оценками релевантности, а порядком, которые они задают:

$$r^{(i)} = \left(r_1^{(i)}, r_2^{(i)}, \ldots, r_{n_{(i)}}^{(i)}\right) = \arg\operatorname{sort} s^{(i)}$$
$$r^{*(i)} = \left(r_1^{*(i)}, r_2^{*(i)}, \ldots, r_{n(i)}^{*(i)}\right) = \arg\operatorname{sort} y^{(i)}$$

Такие, что:

$$\left(s^{(i)}(r_1^{(i)}), s^{(i)}(r_2^{(i)}), \ldots, s^{(i)}(r_{n^{(i)}}^{(i)})\right)$$
$$\left(y^{(i)}(r_1^{*(i)}), y^{(i)}(r_2^{*(i)}), \ldots, y^{(i)}(r_{n^{(i)}}^{*(i)})\right)$$

являются отсортированными по убыванию списками.

Тогда $\left(d^{(i)}(r_1^{*(i)}), d^{(i)}(r_2^{*(i)}), \cdots, d^{(i)}(r_{n^{(i)}}^{*(i)})\right)$ задаёт идеальное ранжирование, а $\left(d^{(i)}(r_1^{(i)}), d^{(i)}(r_2^{(i)}), \cdots, d^{(i)}(r_{n^{(i)}}^{(i)})\right)$– ранжирование скорами из $s^{(i)}$.

По аналогии с $r_j^{(i)}$ определим $t_j^{(i)}$ как номер документа $d_j^{(i)}$ в списке, упорядоченном по скорам модели $s^{(i)}$. По сути $r^{(i)}$ и $t^{(i)}$ задают перестановки на $n^{(i)}$ элементах, такие, что $r^{(i)} = \left(t^{(i)}\right)^{-1}$.

Для примера: пусть $s^{(i)}$ задано как:

$$s^{(i)} = [0.38, 0.77, -0.26, -0.85, 0.97]$$

Тогда:

$$r^{(i)} = [5, 2, 1, 3, 4]$$
$$t^{(i)} = [3, 2, 4, 5, 1]$$