ТЕХНОСФЕРА

Learning to Rank 2

Владимир Гулин

21 марта 2018 г.

План лекции

Напоминание

YetiRank

Listwise approach

Learning to rank using clickthrough data

Задача ранжирования

Множество запросов $Q = \{q_1, q_2, \ldots, q_n\}$

Множество документов соответствующих каждому запросу $q \in Q$

$$q \to d_1, d_2, \ldots$$

Для каждой пары (q, d) сопоставляется оценка релевантности $y(q, d)$, чем выше оценка, тем релевантнее документ d по запросу q.

Оценки релевантности сравнимы, только в рамках одного запроса:

$$(q, d_1) \prec (q, d_2) \iff y(q, d_1) < y(q, d_2)$$

Алгоритмы ранжирования

Discounted Cumulative Gain

$$DCG = \sum_{i=1}^{N_q} \frac{2^{rel_i} - 1}{log_2 i + 1}$$

- pointwise (другая целевая функция)
  - Обучение на отдельных примерах запрос-документ
- pairwise (другая целевая функция)
  - Обучение на парах документах в рамках запроса
- listwise
  - Обучение на отранжированных списках

Вопрос:

- Какие есть достоинства и недостатки у pointwise и pairwise подходов?

Yetirank

Дано:

- Pairwise алгоритм
- Хотим оптимизировать

$$L(f(x)) = -\sum_{(i,j)} w_{ij} log \frac{e^{f(x_i)}}{e^{f(x_i)} + e^{f(x_j)}}$$

- (i, j) - пары документов по определенному запросу
- w_{ij} - вес пары (i, j)

Матан

$$L(f(x)) = -\sum_{(i,j)} w_{ij} log \frac{e^{f(x_i)}}{e^{f(x_i)} + e^{f(x_j)}}$$

$$dL(f(x)) = \sum_{(i,j)} w_{ij} \left((df(x_i) - df(x_j)) \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)$$

Обозначим

$$y_t = \sqrt{w_{ij}}(df(x_i) - df(x_j)), \quad a_t = \sqrt{w_{ij}} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}}$$

Будем искать y фиксированной длины

$$\arg\min_{y,|y|=const} \sum_t y_t a_t = \arg\min_{y,|y|=const} \left(1 + 2 \sum_t \frac{y_t a_t}{|a||y|} + 1 \right) =$$

$$= \arg\min_{y,|y|=const} \left(y_t + \frac{|y|}{|a|} a_t \right)^2$$

Матан

Подставим теперь выражения для y_t и a_t и определим $\lambda = |y|/|a|$

$$\arg\min_{\lambda, df} \sum_{(i,j)} w_{ij} \left((df(x_i) - df(x_j)) + \lambda \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2$$

Выберем, к примеру, $\lambda = 1$

$$\arg\min_{df} \sum_{(i,j)} w_{ij} \left((df(x_i) - df(x_j)) + \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2$$

Связь с LambdaRank

В LambdaRank решение задачи упрощается

$$\arg\min_{df} \sum_{(i,j)} w_{ij} \left[\left(df(x_i) + \frac{1}{2} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2 + \left(df(x_j) - \frac{1}{2} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2 \right]$$

Если ввести обозначения

$$Val_i = \sum_j w_{ji} \frac{1}{2} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} - \sum_j w_{ij} \frac{1}{2} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}}$$

$$W_i = \sum_j w_{ij} + \sum_j w_{ji}$$

Тогда

$$\arg\min_{df} \sum_i W_i (df(x_i) - \frac{Val_i}{W_i})^2$$

Снова матан

Если так не делать и вспомнить что у нас специальные деревья, то можно решать задачу оптимизации сразу для пар

$$\arg\min_{df} \sum_{(i,j)} w_{ij} \left(\left(c_{leaf(x_i)} - c_{leaf(x_j)} \right) + \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2$$

$$A = \begin{pmatrix} & & \vdots & & \\ 0 & 1 & -1 & 0 & \dots \\ 1 & 0 & -1 & 0 & \dots \\ & & \vdots & & \end{pmatrix}$$

$$b = \begin{pmatrix} \vdots \\ -\frac{e^{x_3}}{e^{x_2}+e^{x_3}} \\ -\frac{e^{x_3}}{e^{x_1}+e^{x_3}} \\ \vdots \end{pmatrix}$$

Отсюда методом наименьших квадратов можно найти выражение сразу для листьев

$$c = (A^T w A)^{-1} A^T w b$$

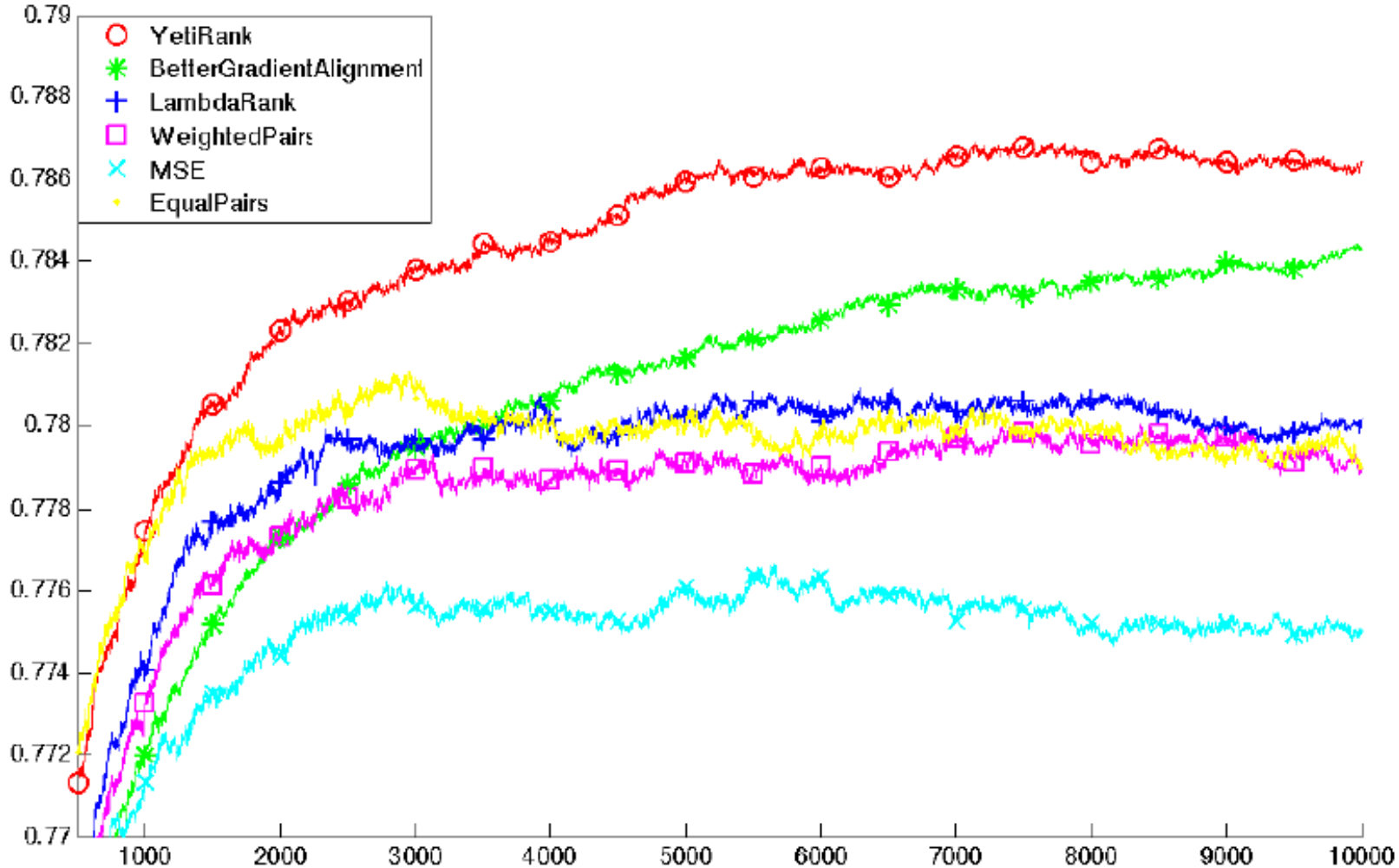
Добавим информацию об ошибках ассесоров в модель

Оценка	Нерелев.	Малополез.	Полез.	Точ. ответ	Обяз. страница
Нерелевантна	0.75	0.22	0.02	0	0
Малополезная	0.34	0.54	0.11	0.01	0
Полезная	0.07	0.13	0.73	0.06	0.01
Точный ответ	0.04	0.04	0.52	0.32	0.08
Обяз. страница	0.03	0.02	0.05	0.08	0.83

$$w_{ij} = c(l_i, l_j), \quad c(l_i, l_j) = \sum_u \sum_v I[u > v] p(u|l_i) p(v|l_j),$$

$$u, v \in 1, 2, 3, 4, 5$$

YetiRank


YetiRank
BetterGradientAlignment
LambdaRank
WeightedPairs
MSE
EqualPairs
0.79
0.788
0.786
0.784
0.782
0.78
0.778
0.776
0.774
0.772
0.77
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

Listwise approach

Что все таки мешает оптимизировать NDCG напрямую?

$$NDCG = \frac{1}{NDCG_{max}} \sum_{q \in Q} \sum_{i=1}^{N_q} \frac{2^{rel_i}}{log_2(i+1)}$$

Listwise approach

Цель:

Оптимизация целевого функционала качества ранжирования

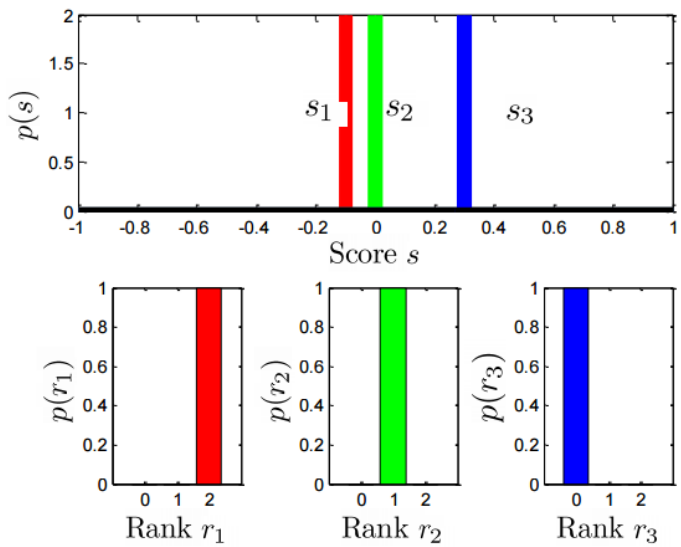
- SoftRank - сглаживание функции метрики
- AdaRank - методы бустинга
- ListNet - оптимизация гладкой функции, похожей на целевую

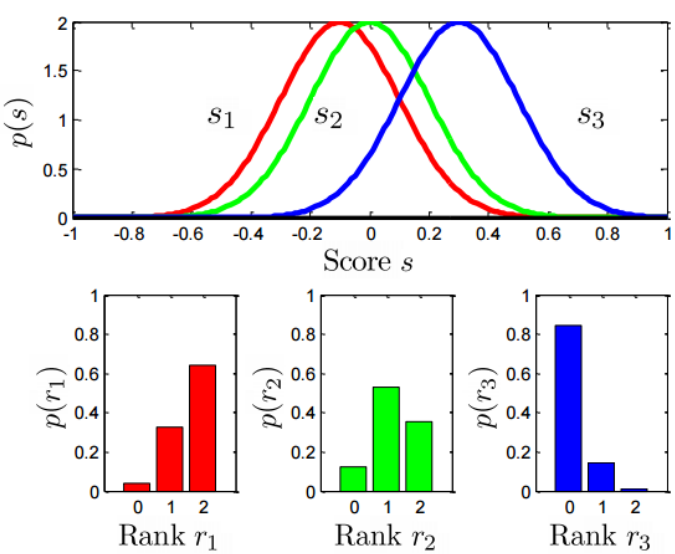
SoftRank

Идея:

Рассматриваем ранк каждого документа как случайную величину, распределенную по нормальному закону

$$p(s_j) = \mathcal{N}(s_j|\bar{s}_j, \sigma_s^2) = \mathcal{N}(s_j|f(\mathbf{w}, \mathbf{x_j}), \sigma_s^2)$$





Вероятностное распределение на позициях документа

Вероятность того, что $i-$ый документ окажется выше, чем документ j:

$$\pi_{ij} \equiv Pr(s_i - s_j > 0) = \int_0^{\infty} \mathcal{N}(s|\bar{s}_j - \bar{s}_j, 2\sigma_s^2) ds$$

Цель:

Хочется получить вероятность того, что конкретный документ находится на позиции r

SoftRank

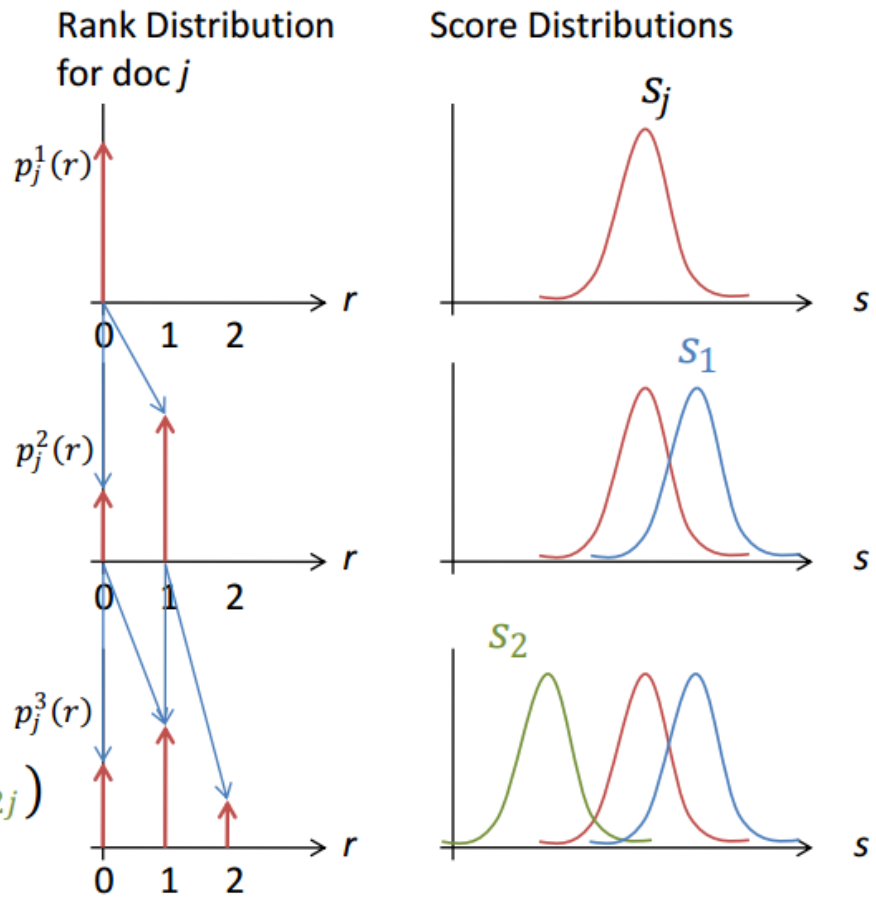
Рекурсивное вычисление вероятностей

$$\pi_{ij} \equiv \Pr(s_i > s_j)$$

$$p_j^1(r) = \delta(r)$$

$$p_j^2(0) = 1 - \pi_{1j}$$
$$p_j^2(1) = \pi_{1j}$$

$$p_j^3(1) = p_j^2(0)\pi_{2j} + p_j^{i-1}(1)(1 - \pi_{2j})$$



SoftRank

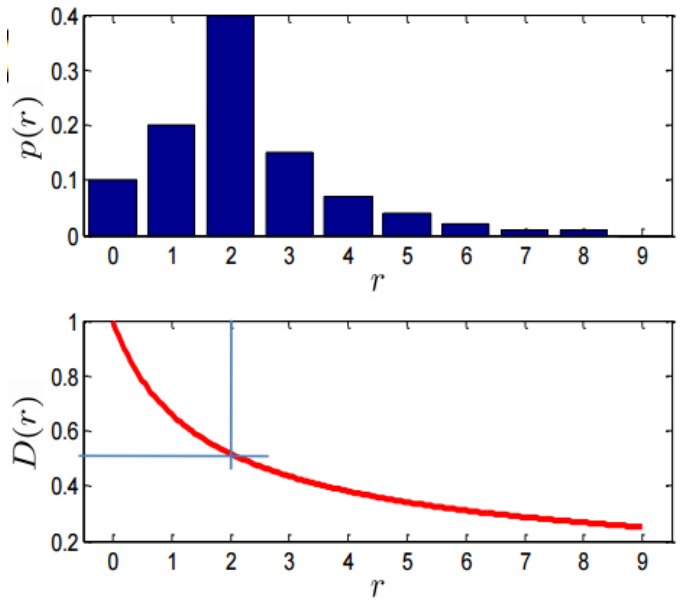
- NDCG:

$$G = G_{max}^{-1} \sum_{j=1}^{N} g(l_j) D(r_j)$$

- SoftNDCG:

$$\mathcal{G} \equiv G_{max}^{-1} \sum_{j=1}^{N} g(l_j) E[D(r_j)]$$

$$\mathcal{G} \equiv G_{max}^{-1} \sum_{j=1}^{N} g(l_j) \sum_{r=0}^{N-1} D(r_j) p_j(r)$$



- Градиент:

$$\frac{\partial \mathcal{G}}{\partial \mathbf{w}} = \frac{\partial \mathcal{G}}{\partial \bar{\mathbf{s}}} \frac{\partial \bar{\mathbf{s}}}{\partial \mathbf{w}}$$

SoftRank

$$\frac{\partial \mathcal{G}}{\partial \bar{s}_m} = G_{max}^{-1} \sum_{j=1}^{N} g(l_j) \sum_{r=0}^{N-1} D(r_j) \frac{\partial p_j(r)}{\partial \bar{s}_m}$$

$\frac{\partial p_j(r)}{\partial \bar{s}_m}$ вычисляем через $\frac{\partial \pi_{ij}}{\partial \bar{s}_m}$

$$\frac{\partial \pi_{ij}}{\partial \bar{s}_m} = \begin{cases} \mathcal{N}(0|\bar{s}_m - \bar{s}_j, 2\sigma_s^2) & m = i, m \neq j \\ -\mathcal{N}(0|\bar{s}_i - \bar{s}_m, 2\sigma_s^2) & m \neq i, m = j \\ 0 & m \neq i, m \neq j \end{cases}$$

AdaRank

Ключевые идеи:

- Аналог AdaBoost для задачи ранжирования
- Минимизирует экспоненциальную аппроксимацию функции потерь
- В качестве базовых ранкеров использует упорядочивание по значениям одного признака

AdaRank

Обозначения

Table 1: Notations and explanations.

Notations	Explanations
$q_i \in Q$	i^{th} query
$\mathbf{d}_i = \{d_{i1}, d_{i2}, \cdots, d_{i,n(q_i)}\}$	List of documents for q_i
$y_{ij} \in \{r_1, r_2, \cdots, r_\ell\}$	Rank of d_{ij} w.r.t. q_i
$\mathbf{y}_i = \{y_{i1}, y_{i2}, \cdots, y_{i,n(q_i)}\}$	List of ranks for q_i
$S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$	Training set
$\vec{x}_{ij} = \Psi(q_i, d_{ij}) \in \mathcal{X}$	Feature vector for (q_i, d_{ij})
$f(\vec{x}_{ij}) \in \Re$	Ranking model
$\pi(q_i, \mathbf{d}_i, f)$	Permutation for q_i, $\mathbf{d}_i$, and f
$h_t(\vec{x}_{ij}) \in \Re$	t^{th} weak ranker
$E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i) \in [-1, +1]$	Performance measure function

AdaRank

$$\max_{f \in \mathcal{F}} \sum_{i=1}^{m} E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i)$$

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{m} (1 - E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i))$$

$$e^{-x} \geq 1 - x$$

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i)\}$$

$$f(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

$$\min_{h_t \in \mathcal{H}, \alpha_t \in \Re^+} L(h_t, \alpha_t) = \sum_{i=1}^{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, f_{t-1} + \alpha_t h_t), \mathbf{y}_i)\}$$

AdaRank

Input: $S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$, and parameters E and T
Initialize $P_1(i) = 1/m$.
For $t = 1, \cdots, T$

- Create weak ranker h_t with weighted distribution $\boldsymbol{P}_t$ on training data S.
- Choose α_t

$$\alpha_t = \frac{1}{2} \cdot \ln \frac{\sum_{i=1}^m P_t(i)\{1 + E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\}}{\sum_{i=1}^m P_t(i)\{1 - E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\}}.$$

- Create f_t

$$f_t(\vec{x}) = \sum_{k=1}^{t} \alpha_k h_k(\vec{x}).$$

- Update $\boldsymbol{P}_{t+1}$

$$P_{t+1}(i) = \frac{\exp\{-E(\pi(q_i, \mathbf{d}_i, f_t), \mathbf{y}_i)\}}{\sum_{j=1}^m \exp\{-E(\pi(q_j, \mathbf{d}_j, f_t), \mathbf{y}_j)\}}.$$

End For
Output ranking model: $f(\vec{x}) = f_T(\vec{x})$.

ListNet

- Вместо максимазации NDCG, будем минимизировать “расстояние” между истинным ранжированием и ранжированием, порождаемым ранжирующей функцией
- Метки релевантности или значения ранжирующей функции на документах порождают вероятностное распределение на множестве перестановок этих документов (модель Luce-Plackett)
- Максимизируем близость распределения, порождаемого значениями ранжирующей функции к распределению порождаемому истинными метками релевантности

ListNet

- Probability of permutation π is defined as

$$P_s(\pi) = \prod_{j=1}^{n} \frac{\varphi(s_{\pi(j)})}{\sum_{k=j}^{n} \varphi(s_{\pi(k)})}$$

- Example:

$$P_f(\mathrm{ABC}) = \frac{\varphi(f(\mathrm{A}))}{\varphi(f(\mathrm{A}))+\varphi(f(\mathrm{B}))+\varphi(f(\mathrm{A}))} \cdot \frac{\varphi(f(\mathrm{B}))}{\varphi(f(\mathrm{B}))+\varphi(f(\mathrm{C}))} \cdot \frac{\varphi(f(\mathrm{C}))}{\varphi(f(\mathrm{C}))}$$

P(A ranked No.1)

P(B ranked No.2 | A ranked No.1)
= P(B ranked No.1)/(1- P(A ranked No.1))

P(C ranked No.3 | A ranked No.1, B ranked No.2)

ListNet

- Модель - нейронная сеть
- Обучается градиентным спуском
- Функция потерь - KL дивергенция между распределениями, порожденными истинными метками релевантности и значениями функции ранжирования ($\phi = exp$)

$$L(h) = -\sum_{q \in Q} \sum_{G(j_1,\ldots,j_n)} \left(\prod_{t=1}^{n} \frac{e^{rel_{j_t}}}{\sum_{u=t}^{n} e^{rel_{j_u}}} \right) log \left(\prod_{t=1}^{n} \frac{e^{h(x_{j_t})}}{\sum_{u=t}^{n} e^{h(x_{j_u})}} \right)$$

Вопрос:

- Какая есть проблема с обучением такой модели?

ListNet

- Модель - нейронная сеть
- Обучается градиентным спуском
- Функция потерь - KL дивергенция между распределениями, порожденными истинными метками релевантности и значениями функции ранжирования ($\phi = exp$)

$$L(h) = -\sum_{q \in Q} \sum_{G(j_1,\ldots,j_n)} \left(\prod_{t=1}^{n} \frac{e^{rel_{j_t}}}{\sum_{u=t}^{n} e^{rel_{j_u}}} \right) log \left(\prod_{t=1}^{n} \frac{e^{h(x_{j_t})}}{\sum_{u=t}^{n} e^{h(x_{j_u})}} \right)$$

- Сумма по всем перестановкам $O(n!)$ в асимптотике

Top-k Probability

- При подходе “в лоб” на практике использование listnet невозможно
- Top-k Probability
  - Определим Top-k подгруппу $G(j_1, \ldots, j_k)$, содержащую все перестановки, у которых top-k документов $j_1, \ldots, j_k$

$$P_s(G(j_1, \ldots, j_k)) = \prod_{t=1}^{k} \frac{e^{h(x_{j_t})}}{\sum_{u=t}^{n} e^{h(x_{j_u})}}$$

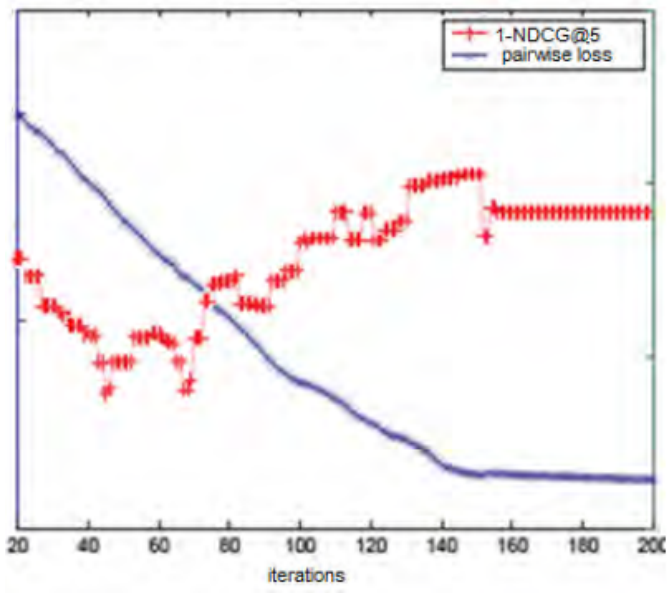
ListNet

- Модель - нейронная сеть
- Обучается градиентным спуском
- Функция потерь - KL дивергенция между Top-k распределениями, порожденными истинными метками релевантности и значениями функции ранжирования ($\phi = exp$)

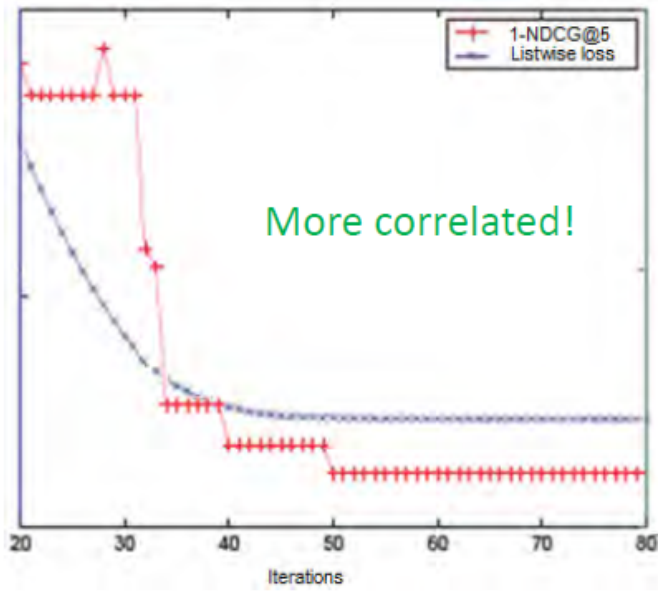
$$L(h) = -\sum_{q \in Q} \sum_{G(j_1,\ldots,j_k)} \left(\prod_{t=1}^{k} \frac{e^{rel_{j_t}}}{\sum_{u=t}^{n} e^{rel_{j_u}}} \right) log \left(\prod_{t=1}^{k} \frac{e^{h(x_{j_t})}}{\sum_{u=t}^{n} e^{h(x_{j_u})}} \right)$$

- Таким образом снижаем сложность с $O(n!)$ до $O(n!/(n-k)!)$

ListNet



Pairwise (RankNet)



Listwise (ListNet)

Training Performance on TREC Dataset

А какой подход лучше?

- Pointwise
- Pairwise
- Listwise

А какой подход лучше?

- Pairwise (информация пропорциональна количеству пар)
- Pointwise (информация поточечная)
- Listwise (информация позапросная)

Вопросы