Машинное обучение
Лекция № 13, осень 2022

Деревянное ранжирование

План лекции

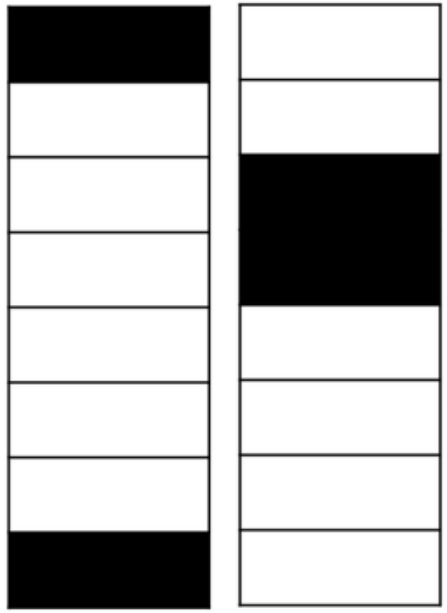
- Недостатки RankNet
- LambdaRank
- MART
- LambdaMART
- YetiRank

Недостатки RankNet

Кратко вспомним, что было в RankNet в качестве функции потерь

Неупорядоченных пар: 6

AP: 5/8 (1/1 + 2/8) / 2 = (10 / 8) / 2
DCG: 1.33



Неупорядоченных пар: 4

AP: 5/12 (1/3 + 2/4) / 2 = (10/12) / 2
DCG: 0.931

Некоторые из документов важнее отранжировать в первую очередь

LamdaRank

Кратко вспомним, что было в RankNet в качестве функции потерь кросс-энтропию C:

s_i, s_j — предсказания релевантности для i-го и j-го документов
$\sigma(x)$ — вообще говоря, любая монотонно возрастающая, положительная функция, а в нашем случае — сигмоида
S_{ij} — новое трансформированное значение целевой переменной (target), получающееся в результате линейного преобразования старых значений (диапазон [0, 1] -> [-1, 1])

$$\frac{\partial C}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) = -\frac{\partial C}{\partial s_j}$$

Learning to Rank with Nonsmooth Cost Functions

LamdaRank

Производная относительно ответа

$$\frac{\partial C}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) = -\frac{\partial C}{\partial s_j}$$

Функция ошибки зависит от i-го и j-го объекта

Производная относительно изменения веса модели

$$\frac{\partial C}{\partial w_k} = \frac{\partial C}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \frac{\partial C}{\partial s_j}\frac{\partial s_j}{\partial w_k} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right)\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right)$$

Получаем 2 подзадачи:

1. Как изменить выход модели

2. Как обновить модель (w), чтобы сдвинуть s_i, s_j

LamdaRank

Производная относительно ответа

$$\frac{\partial C}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) = -\frac{\partial C}{\partial s_j}$$

Функция ошибки зависит от i-го и j-го объекта

Производная относительно изменения веса модели

$$\frac{\partial C}{\partial w_k} = \frac{\partial C}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \frac{\partial C}{\partial s_j}\frac{\partial s_j}{\partial w_k} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right)\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right)$$

$$\lambda_{ij} \equiv \frac{\partial C(s_i - s_j)}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right)$$

$$\sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right)\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right) = \lambda_{ij}\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right)$$

Получаем 2 подзадачи:

1. Как изменить выход модели

2. Как обновить модель (w), чтобы сдвинуть s_i, s_j

LamdaRank

Изменим набор данных:

1. Приведем метки из {-1, 0, 1} в {0, 0.5, 1}
2. Переупорядочим данные так, чтобы i > j и $S_{ij} = 1$ – по нормировке

$$\lambda_{ij} \equiv \frac{\partial C\left(s_i - s_j\right)}{\partial s_i} = \sigma\left(\frac{1}{2}\left(1 - S_{ij}\right) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) \longrightarrow \lambda_{ij} = \frac{\partial C\left(s_i - s_j\right)}{\partial s_i} = -\sigma\left(\frac{1}{1 + e^{\sigma(s_i - s_j)}}\right)$$

Так смогли выделить изменение выхода модели (градиент) – ламбду.
Лямбда может быть сколько угодно сложной фукнцией.

«Вам не нужна функция потерь, все необходимое – это градиенты относительно оценок модели»

$$\delta w_k = -\eta \sum_{\{i,j\} \in I} \left(\lambda_{ij} \frac{\partial s_i}{\partial w_k} - \lambda_{ij} \frac{\partial s_j}{\partial w_k}\right) \equiv -\eta \sum_i \lambda_i \frac{\partial s_i}{\partial w_k} \qquad \lambda_i = \sum_{j:\{i,j\} \in I} \lambda_{ij} - \sum_{j:\{j,i\} \in I} \lambda_{ij}$$

LamdaRank

Градиент зависит от изменения метрики при перестановки i-го и j-го объектов

$$\lambda_{ij} = \frac{\partial C(s_i - s_j)}{\partial s_i} = \frac{-\sigma}{1 + e^{\sigma(s_i - s_j)}} |\Delta_{NDCG}|$$

$$\mathrm{DCG} = \sum_i \frac{2^{\mathrm{rel}_i} - 1}{\log_2(i+1)}$$

$$\mathrm{nDCG} = \frac{\mathrm{DCG}}{\mathrm{IDCG}}$$

$$\lambda_{ij} = N \left(\frac{1}{1 + e^{s_i - s_j}} \right) \left(2^{\mathrm{rel}_i} - 2^{\mathrm{rel}_j} \right) \left(\frac{1}{\log_2(i+1)} - \frac{1}{\log_2(j+1)} \right)$$

1. Прогоняем все объекты через модель и получаем s_i
2. Берем все пары и оцениваем изменение метрики при перестановке
3. Для каждого объекта считаем лямбду

$$\lambda_i \equiv \sum_{j \in P_i} \frac{\partial C(s_i, s_j)}{\partial s_i}$$

Learning to Rank with Nonsmooth Cost Functions

MART

MART (Multiple Additive Regression Trees) — алгоритм бустинга регрессионных деревьев.

$$S_j \equiv \sum_{i \in L} (y_i - \mu_L)^2 + \sum_{i \in R} (y_i - \mu_R)^2$$

1. Переберём все значения этого признака и будем строить разбиение объектов простым условием: значение признака j меньше некоторого порогового числа k. Если это условие выполняется, то объект попадает в левое поддерево, если же оно ошибочно — в правое.
2. Затем по всем объектам левого поддерева L и правого R мы рассчитываем среднее значение предсказываемого значения μ.
3. В каждом из двух множеств считаем среднеквадратичное отклонение (СКО). Общее СКО S_j считаем как сумму СКО в левом и правом множествах.
4. Затем среди всех перебранных комбинаций разбиений мы выбираем разбиение с наименьшим значением этого среднеквадратичного отклонения S_j.

Ансамбль N алгоритмов с весами:

$$F_N(x) = \sum_{i=1}^{N} \alpha_i f_i(x) \qquad \bar{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

MART

MART (Multiple Additive Regression Trees) — алгоритм бустинга регрессионных деревьев.

$$S_j \equiv \sum_{i \in L} (y_i - \mu_L)^2 + \sum_{i \in R} (y_i - \mu_R)^2$$

Ансамбль N алгоритмов с весами:

$$F_N(x) = \sum_{i=1}^{N} \alpha_i f_i(x) \qquad \bar{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}$$

То есть каждое следующее дерево говорит о том, как нужно скорректировать текущее состояние

$$\lambda_i \equiv \sum_{j \in P_i} \frac{\partial C(s_i, s_j)}{\partial s_i}$$

LambdaMART

Algorithm 1 The LambdaSMART algorithm.

for $i = 0$ to N **do**
  $F_0(x_i) = BaseModel(x_i)$ \\$BaseModel$ may be empty or set to a sub-model.
end for
for $m = 1$ to M **do**
  for $i = 0$ to N **do**
    $y_i = \lambda_i$
    $w_i = \frac{\partial y_i}{\partial F(x_i)}$
  end for
  $\{R_{lm}\}_{l=1}^{L}$ \\Create L-terminal node tree on $\{y_i, x_i\}_{i=1}^{N}$
  $\gamma_{lm} = \frac{\sum_{x_i \in R_{lm}} y_i}{\sum_{x_i \in R_{lm}} w_i}$ \\Find the leaf values based on approximate Newton step.
  $F_m(x_i) = F_{m-1}(x_i) + v \sum_l \gamma_{lm} 1(x_i \in R_{lm})$
end for

Раньше

$$L(y, f) = (y - f)^2$$

Градиент

$$(y - f)$$

Лямбда градиент

$$\lambda_{ij} = \frac{\partial C\,(s_i - s_j)}{\partial s_i}$$

From RankNet to LambdaRank to LambdaMART: An Overview

YetiRank

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

Pairwise функция потерь, **но с весами**, где опционально можно добавить лямбду

$$w_{ij} = N_{ij} c(l_i, l_j)$$

Насколько пара важна для ранжирования

Насколько уверены в разметке, насколько близки метки

$$\hat{x}_i = x_i + \log \frac{r_i}{1 - r_i} \qquad N_{ij} = \frac{1}{n} \sum_{t=1}^{n} \frac{1}{index_t(\min(i,j))}$$

$$c(l_i, l_j) = \sum_{u,v} 1_{u>v} p(u|l_i)\, p(v|l_j)$$

- Добавляем шум в предсказание
- Реранжируем выборку с шумом
- Берем MRR для каждой последовательной пары
- Повторяем 100 раз

- Заранее берем матрицу оценок релевантности
- Пробегаем по всем оценкам (u,v), где u > v
- Сумма тем больше, чем больше уверенность в оценке

YetiRank

Матрицы перехода оценок релевантности (confusion matrix)

	Bad	Poor	Good	Exc.	Perf.
Bad	0.869	0.103	0.02	0.001	0.007
Poor	0.016	0.878	0.1	0.005	0.002
Good	0.003	0.098	0.85	0.046	0.004
Exc.	0	0.01	0.094	0.896	0
Perf.	0	0	0.019	0.016	0.965

	Bad	Poor	Good	Exc.	Perf.
Bad	0.75	0.22	0.02	0	0
Poor	0.34	0.54	0.11	0.01	0
Good	0.07	0.13	0.73	0.06	0.01
Exc.	0.04	0.04	0.52	0.32	0.08
Perf.	0.03	0.02	0.05	0.08	0.83

	Bad	Poor	Good	Exc.	Perf.
Bad	0.88	0.09	0.02	0	0
Poor	0.26	0.65	0.07	0.01	0
Good	0.05	0.08	0.78	0.07	0.01
Exc.	0.03	0.02	0.24	0.60	0.10
Perf.	0.03	0.02	0.03	0.05	0.86

Как определить матрицу: $b = \{d_i : \forall d_j \in b, |d_i - d_j| < \epsilon\}$