LambdaMART notes

Yafei Zhang

August 12, 2014

Contents

1 Notations

x, x_i, x_j: input feature vector, $x \in \Re^n$.

y, y_i, y_j: input score, measurement of goodness.

U, U_i, U_j: URLs.

$s \equiv F(x)$, $s_i \equiv F(x_i)$, $s_j \equiv F(x_j)$: model learned score, measurement of goodness. F is the learned model.

$o_{ij} \equiv s_i - s_j$ (for a given query).

$\rho_{ij} \equiv \frac{1}{1+e^{o_{ij}}}$ (for a given query).

P_{ij}: a learned probability that U_i should be ranked higher than U_j (for a given query).

$\overline{P}_{ij}$: the known probability that U_i should be ranked higher than U_j (for a given query).

S_{ij}: it is defined as (for a given query)

$$S_{ij} \equiv \begin{cases} 0 & \text{for} \quad y_i = y_j \\ 1 & \text{for} \quad y_i > y_j \\ -1 & \text{for} \quad y_i < y_j \end{cases} \tag{1}$$

C: target function we want to maximize.

C_{ij}: the amount that U_i and U_j contribute to C (for a given query).

λ_{ij}: lambda gradient of U_i and U_j (for a given query).

λ_i: sums of lambda gradient of U_i and all other URLs.

$I \equiv \{(i, j) | S_{ij} = 1\}$: the set of pairs of indices (i, j), for which we desire U_i to be ranked higher than U_j (for a given query).

Z: a measure of L2R such as NDCG, MAP, MRR. It is NDCG in LambdaMART.

ΔZ_{ij} is the change of Z by swapping the rank positions of U_i and U_j (for a given query) (after sorting all documents by their current scores s).

2 Definitions

$$P_{ij} \equiv P(U_i \rhd U_j) \equiv \frac{1}{1+e^{-(s_i-s_j)}} = \frac{1}{1+e^{-o_{ij}}} \tag{2}$$

$$\overline{P}_{ij} \equiv \frac{(1+S_{ij})}{2} \tag{3}$$

C_{ij} is chosen to be:

$$\begin{aligned} C_{ij} &\equiv |\Delta Z_{ij}|\left[-\overline{P}_{ij}\log(P_{ij}) - (1-\overline{P}_{ij})\log(1-P_{ij})\right] && (4)\\ &= |\Delta Z_{ij}|\left[\frac{1}{2}(1-S_{ij})o_{ij} + \log(1+e^{-o_{ij}})\right] && (5)\\ &= \begin{cases} |\Delta Z_{ij}|\log(1+e^{-o_{ij}}) & \text{for} \quad S_{ij}=1 \\ |\Delta Z_{ij}|\log(1+e^{-o_{ji}}) & \text{for} \quad S_{ij}=-1 \end{cases} && (6) \end{aligned}$$

From now on, we assume all $(i,j) \in I$ except extra explanations.

$$C_{ij} \equiv |\Delta Z_{ij}|\log(1+e^{-o_{ij}}) \tag{7}$$

Gradient(treat ΔZ_{ij} a constant):

$$\frac{\partial C_{ij}}{\partial s_i} = \frac{\partial C_{ij}}{\partial o_{ij}} = -\frac{|\Delta Z_{ij}|}{1+e^{o_{ij}}} = -|\Delta Z_{ij}|\rho_{ij} = -\frac{\partial C_{ij}}{\partial o_{ji}} = -\frac{\partial C_{ij}}{\partial s_j} \tag{8}$$

λ-gradient:

$$\lambda_{ij} \equiv \left|\frac{\partial C_{ij}}{\partial o_{ij}}\right| = \frac{|\Delta Z_{ij}|}{1+e^{o_{ij}}} = |\Delta Z_{ij}|\rho_{ij} = -\lambda_{ji} \tag{9}$$

So

$$\frac{\partial C_{ij}}{\partial s_i} = -\lambda_{ij} \tag{10}$$

$$\frac{\partial^2 C_{ij}}{\partial s_i^2} = -\frac{\partial \lambda_{ij}}{\partial s_i} = \frac{|\Delta Z|e^{o_{ij}}}{(1+e^{o_{ij}})^2} = |\Delta Z|\rho_{ij}(1-\rho_{ij}) \tag{11}$$

3 Sum them together

$$C = \sum_i \sum_{j:(i,j)\in I} C_{ij} + \sum_i \sum_{j:(j,i)\in I} C_{ji} \tag{12}$$

$$= \sum_i \sum_{j:(i,j)\in I} |\Delta Z_{ij}| \log(1 + e^{-o_{ij}}) + \sum_i \sum_{j:(j,i)\in I} |\Delta Z_{ji}| \log(1 + e^{-o_{ji}}) \tag{13}$$

$$\frac{\partial C}{\partial s_i} = \sum_{j:(i,j)\in I} \frac{\partial C_{ij}}{\partial s_i} + \sum_{j:(j,i)\in I} \frac{\partial C_{ji}}{\partial s_i} \tag{14}$$

$$= \sum_{j:(i,j)\in I} (-|\Delta Z_{ij}|\rho_{ij}) + \sum_{j:(j,i)\in I} |\Delta Z_{ji}|\rho_{ji} \tag{15}$$

$$= \sum_{j:(i,j)\in I} (-\lambda_{ij}) + \sum_{j:(j,i)\in I} \lambda_{ji} \tag{16}$$

$$\lambda_i \equiv \sum_{j:(i,j)\in I} \lambda_{ij} + \sum_{j:(j,i)\in I} \lambda_{ij} \tag{17}$$

$$= \sum_{j:(i,j)\in I} \lambda_{ij} - \sum_{j:(j,i)\in I} \lambda_{ji} \tag{18}$$

So

$$\frac{\partial C}{\partial s_i} = -\lambda_i \tag{19}$$

$$\frac{\partial^2 C}{\partial s_i^2} = \sum_{j:(i,j)\in I} \frac{\partial^2 C_{ij}}{\partial s_i^2} + \sum_{j:(j,i)\in I} \frac{\partial^2 C_{ji}}{\partial s_i^2} \tag{20}$$

$$= \sum_{j:(i,j)\in I} |\Delta Z_{ij}|\rho_{ij}(1-\rho_{ij}) + \sum_{j:(j,i)\in I} |\Delta Z_{ji}|\rho_{ji}(1-\rho_{ji}) \tag{21}$$

$$= \sum_{j:(i,j)\in I} (-\frac{\partial \lambda_{ij}}{\partial s_i}) + \sum_{j:(j,i)\in I} (-\frac{\partial \lambda_{ji}}{\partial s_j}) \tag{22}$$

$$= \sum_{j:(i,j)\in I} (-\frac{\partial \lambda_{ij}}{\partial s_i}) + \sum_{j:(j,i)\in I} \frac{\partial \lambda_{ji}}{\partial s_i} \tag{23}$$

So

$$\frac{\partial^2 C}{\partial s_i^2} = -\frac{\partial \lambda_i}{\partial s_i} \tag{24}$$

4 Gradient Boosting

In this and next section, we will take a digression to recall some knowledge of GBRT(Gradient Boosted Regression Trees).

We consider L a function of input score y and model function $F = F(x)$: $L = L(y, F)$. Our goal is

$$\min_F L(y, F) \tag{25}$$

As in ordinary gradient descent, F is iteratively updated in functional space.

$$F_{n+1} = F_n - \rho \left.\frac{\partial L(y, F)}{\partial F}\right|_{F=F_n} \tag{26}$$

ρ is a step length, chosen

$$\rho = argmin_\rho C\left(y, F_n - \rho \left.\frac{\partial L(y, F)}{\partial F}\right|_{F=F_n}\right) \tag{27}$$

26 and 27 can be re-interpreted as

$$F_{n+1} = F_n + \rho f_n(x) \tag{28}$$

where $f_n(x)$ is a model that fits pseudo-response $\{\widetilde{y}\}_i$.

$$\widetilde{y}_i = -\left.\frac{\partial L(y_i, F)}{\partial F}\right|_{F=F_n} \tag{29}$$

Then ρ is chosen

$$\rho = argmin_\rho L(y, F_n + \rho f_n(x)) \tag{30}$$

5 Gradient Boosted Regression Trees

Particularly, if we choose $f(x)$ as a regression tree, this results in a GBRT algorithm.

$$f(x) = f(x, \{\gamma_j, R_j\}_i^J) = \sum_{j=1}^{J} \gamma_j 1(x \in R_j) \tag{31}$$

J is the number of leaves. $\{R_j\}_i^J$ are disjoint regions that cover all feature space, each of them is covered in one leaf. $\{\gamma_j\}_i^J$ are the values in the corresponding leaf.

In this circumstance, we first construct $\{R_j\}_i^J$ to fit $\{x_i, \widetilde{y}\}_i$ by least-squares, then use 26,

$$F_{n+1} = F_n + \sum_{j=1}^{J} \gamma_j 1(x \in R_j) \tag{32}$$

use 27.

$$\gamma_j = argmin_\gamma \sum_{i:x_i \in R_j} L(y_i, F_n(x_i) + \gamma) = argmin_\gamma \sum_{i:x_i \in R_j} g \tag{33}$$

where

$$g = g(y_i, F_n(x_i)) = L(y_i, F_n(x_i) + \gamma) \tag{34}$$

When there is no closed form solution to 33, we can approximate it by a single Newton step. This is(actually $-\gamma$ is the Newton step, Newton update is as $F_{n+1}(x_i) = F_n(x_i) - (-\gamma)$)

$$\gamma_j = \frac{\sum_{i:x_i \in R_j} \frac{\partial g}{\partial F_n}}{\sum_{i:x_i \in R_j} \frac{\partial^2 g}{\partial F_n^2}} \tag{35}$$

GBRT is sometimes called GBDT(Gradient Boosted Decision Trees) or MART(Multi Additive Regression Trees).

Note that GBDT is so-called for its decision purpose, but each tree in it is still a regression tree.

6 LambdaMART

Our goal is

$$\min_F(-C) \tag{36}$$

with $-C$ defined in 13, with the first order derivative $-\frac{\partial C}{\partial s_i}$ defined in 19, with the second order derivative $-\frac{\partial^2 C}{\partial s_i^2}$ defined in 24.

F is MART. Overall training framework is depicted below.

input : Training samples $\{x_i, y_i, U_i\}_1^N$(x_i contains a query), the metric Z, the number of leaves in a tree L, the number of trees M, learning rate ν

output: F, $\{s_i\}_1^N$

data : pseudo responses $\{\widetilde{y}_i\}_1^N$, weights $\{w_i\}_1^N$, model output $\{s_i\}_1^N$;

for $i = 1$ **to** N **do**
    $s_i = F_0(x_i) = BaseModel(x_i)$;
end
for $m = 1$ **to** M **do**
    for $i = 1$ **to** N **do**
        $\widetilde{y}_i = \lambda_i$ defined in 19;
        $w_i = \frac{\partial \lambda_i}{\partial s_i}$ defined in 24;
    end
    Create a tree $\{R_{lm}\}_{l=1}^L$ to fit $\{x_i, \widetilde{y}_i\}_1^N$ by least-squares;
    for $l = 1$ **to** L **do**
        $\gamma_{lm} = \frac{\sum_{i:x_i \in R_{lm}} \widetilde{y}_i}{\sum_{i:x_i \in R_{lm}} w_i}$;
    end
    for $i = 1$ **to** N **do**
        $s_i = F_m(x_i) = F_{m-1}(x_i) + \nu \sum_l \gamma_{lm} 1(x_i \in R_{lm})$;
    end
end
$F = F_M$;

Algorithm 1: The LambdaMART algorithm

References

[1] Qiang Wu, Christopher J.C. Burges, Krysta M. Svore, Jianfeng Gao. Adapting Boosting for Information Retrieval Measures. Learning to Rank for Information Retrieval DOI 10.1007/s10791-009-9112-1, 2009

[2] Christopher J.C. Burges. From RankNet to LambdaRank to LambdaMART: An Overview. Microsoft Research Technical Report MSR-TR-2010-82, 2010.

[3] Jerome H. Friedman. Greedy function Approximation: A Gradient Boosting Machine. Annals of Statistics, Vol. 29 (2001), pp. 1189-1232, 1999.