CS761 Final Project: Overview and Exploration of Statistical Convergence Property of LambdaMART

Saikat R. Gomes saikat @cs.wisc.edu
University of Wisconsin, Madison

Abstract

Rank aggregation is a classical problem that has been studies in several contexts such as social choice theory as early as the 18th century and more recently in computer science, statistics, linear algebra and optimization with a variety of applications. Different input rankings and desired aggregated rankings have been considered but in this paper we will look at the rank aggregation of pairwise data that has gained interest in recent years. Specifically we will consider the statistical assumption under which LambdaMART converges to an 'optimal' ranking. A general overview of the LambdaMART algorithm (and its predecessors) is given and then the optimal convergence property of the algorithm is considered under a 'time-reversibility' or Bradly-Terry-Luce (BTL) condition on the data distribution.

1. Introduction

Learning to rank, a sub-domain of machine learning; for Information Retrieval (IR) is a task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance (Liu, 2009). Informally, the rank aggregation problem is to combine many different rank orderings on the same set of candidates, or alternatives, in order to obtain a better ordering (Dwork et al., 2001). Learning to rank algorithms can be categorized into three approaches: point-wise, pair-wise and list-wise. In the point-wise model (also known as the 'score-wise' model) rankings are created by computing a real-valued score for each example independently and then sorting by score. The pair-wise model on the other hand learns to decide which of the given two examples is ranked higher relatively. This model relies on the observation that the relative positions of the examples are important rather than the exact distances between the example scores. The goal is to aggregate these pairwise comparisons into a global ranking over items (Rajkumar & Agarwal, 2014). Whereas a list-wise models are trained by optimizing a loss function that measures predicted ranks compared to actual ranks for a given query's entire candidate set.

In the context of pair-wise rank aggregation algorithm, an 'optimal' ranking is the one which minimizes the probability of disagreement with a random pair-wise comparisons drawn i.i.d from some fixed but unknown probability distribution. In this paper we will only consider the statistical convergence properties of the rank aggregation algorithm LambdaMART under a Bradly-Terry-Luce (BTL) condition on the data distribution. LambdaMART is the boosted tree version of the LambdaRank, which is based on RankNet (Burges, 2010).

In section 2 a brief overview is provided of the LambdaMART algorithm and in section 3 it will be shown that LambdaMART converges to an optimal rank under the BTL condition in probability.

Proceedings of the CS761 Advanced Machine Leaning, University of Wisconsin, Madison.

2. A Brief Overview

In this section an brief overview of RankNet (Burges et al., 2005), LambdaRank (Burges et al., 2006), MART (Friedman, 2001) and eventually LambdaMART (Burges, 2010) is provided. This overview helps in understanding the rank aggregation algorithm LambdaMART as it uses the underlying concepts of MART and LambdaRank (which is it self derived from RankNet).

2.1. RankNet

The underlying model for RankNet (Burges et al., 2005) can be any model who's output is a differentiable fuction of the model parameters. Typically RankNet were implemented with neural nets, but boosted trees have also been used for its implementation (Burges, 2010).
Basically the training data is partitioned by query and at any given point during the training the input feature vector $x \quad \mathbf{R}^n$ is mapped to a number $f(x)$. For a given query, each pair of data-points T_i and T_j with feature vectors x_i and x_j is chosen such that their labels are different and presented to the model. The scores $s_i = f(x_i)$ and $s_j = f(x_j)$ is consequently calculated by the model. The event that T_i should be ranked higher than T_j is denoted by $T_i \quad T_j$; then the learned probability via sigmoid function denoting $T_i \quad T_j$ is given by: $P_{ij} \quad P(T_i \quad T_j) \quad \frac{1}{1+e^{- \ (s_i-s_j)}}$
To penalize the deviation of the model output probabilities from the desired probabilities cross entropy function is applied. Therefore if $\bar{P_{ij}}$ is the known probability that $T_i \quad T_j$, then the cost function is $C = -\bar{P_{ij}} logP_{ij} - (1 - \bar{P_{ij}}) log(1 - p_{ij})$
For the given query, let $S_{ij} \quad \{0, \pm 1\}$; such that $S_{ij} = 1$ if T_i is ranked higher than T_j, $S_{ij} = 0$ if both are labeled the same (ie same rank) and $S_{ij} = -1$ else. With the assumption that the desired ranking is deterministically known such that $\bar{P_{ij}} = \frac{1}{2}(1 + S_{ij})$ On combining we get $C = \frac{1}{2}(1 - S_{ij}) \quad (s_i - s_j) + log(1 + e^{- \ (s_i-s_j)})$, hence
$C = log(1 + e^{- \ (s_i-s_j)})$; when $S_{ij} = 1$
$C = log(1 + e^{- \ (s_j-s_i)})$; when $S_{ij} = -1$
$C = log2$; when $S_{ij} = 0$
This leads to:
$\frac{C}{s_i} = \quad \frac{1}{2}(1 - S_{ij}) - \frac{1}{1+e^{- \ (s_i-s_j)}} \quad = - \frac{C}{s_j}$
Using stochastic gradient descent (as the learning rate), the gradient is used to update the weights $w_k \quad \mathbf{R}$ (model parameters)

$$w_k \quad w_k - \quad \frac{C}{w_k} = w_k - \quad \frac{C}{s_i}\frac{s_i}{w_k} + \frac{C}{s_j}\frac{s_j}{w_k} \tag{1}$$

Note: summations over repeated indices are assumed.

2.2. Lambda

In ranking, the loss function that we most likely come across optimizing is probably either Normalized Discounted Cumulative Gain (NDCG)(Järvelin & Kekäläinen, 2002), Mean Reciprocal Rank (MRR), Mean Average Precision (MAP) or Expected Reciprocal Rank (ERR) (Chapelle et al., 2009). Unfortunately, these loss functions are not differentiable at all points and hence cannot be used them directly in the Gradient Boosting framework, since its unclear how we can provide the gradients at each training point. This problem is solved with the used of Lambda .
Using the factorization from section 2.1 we have:
$\frac{C}{w_k} = \frac{C}{s_i}\frac{s_i}{w_k} + \frac{C}{s_j}\frac{s_j}{w_k}$
$= \quad \frac{1}{2}(1 - S_{ij}) - \frac{1}{1+e^{- \ (s_i-s_j)}} \quad \frac{s_i}{w_k} + \frac{s_j}{w_k}$
$= \quad {}_{ij} \quad \frac{s_i}{w_k} + \frac{s_j}{w_k}$
Where ${}_{ij}$ is defined as:

$${}_{ij} = \frac{C(s_i - s_j)}{s_i} = \quad \frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{- \ (s_i-s_j)}} \tag{2}$$

For each training point pair i,j, ${}_{i,j}$ is computed that acts as the gradient. Intuitively, ${}_{i,j}$ can be thought of as a force that moves data point up and down the ranked list. For instance, if T_i is ranked lower than T_j, then T_i will receive a push of size $| \ {}_{i,j}|$ downwards in the ranked list, and T_j will be pushed upwards with the same amount.
On summing up all the contributions to the update of weight w_k we get:

$$w_k = - \quad \underset{\{i,j\} \quad I}{} \quad {}_{ij}\frac{s_i}{w_k} - \ {}_{ij}\frac{s_j}{w_k}$$
$$- \quad \underset{i}{} \quad {}_i \frac{s_i}{w_k}$$

Where I is defined as a set of pairs of indicies $\{i, j\}$, for which we want T_i to be ranked differently from T_j (Burges, 2010); also from above

$${}_i = \underset{j:\{i,j\} \quad I}{} \quad {}_{i,j} - \underset{j:\{j,i\} \quad I}{} \quad {}_{i,j} \tag{3}$$

2.3. LambdaRank

The key observation in LambdaRank is that we can simply need the gradient (of the costs with respect to the models scores) to train a model, i.e. we do not need the costs themselves (Burges et al., 2006). The lambda described above are those gradients. Based on experiments (Burges et al., 2006) we modify Eq. (2) to get:

$${}_{ij} = \frac{C(s_i - s_j)}{s_i} = \frac{-}{1 + e^{- \ (s_i-s_j)}} | \quad {}_{NDCG}| \tag{4}$$

In this scenario we want to maximize C, therefore instead of Eq. (1) we have: $w_k \quad w_k + \quad \frac{C}{w_k}$; hence

$$C = \frac{C}{w_k} \quad w_k = \quad \frac{C}{w_k} \ ^{2} > 0 \tag{5}$$

It had been shown that LambdaRank optimized NDCG directly (Yue & Burges, 2007; Donmez et al., 2009)

2.4. MART

Multiple Additive Regression Trees (MART) (Friedman, 2001) is a boosted tree model in which the output of a models is a linear combination of the outputs of a set of regression trees using gradient descent in the function space. The final model maps an input feature vector x $\mathbf{R}^d$ to a score $F(x)$ $\mathbf{R}$. MART is a class of algorithms because it can be trained to minimize general cost. It is important to note that the Least Square Regression Tree is the underlying model upon which MART is built. This fact will be later exploited in section 3 to explore the statistical convergence property of LambdaMART. Therefore MART's output $F(x)$ can be further written as

$$F_N(x) = \overset{N}{\underset{i=1}{}} {}_i f_i(x) \qquad (6)$$

where $f_i(x)$ $\mathbf{R}$ is a function modeled by a single regression tree, ${}_i$ $\mathbf{R}$ is the weight associated with the i^{th} regression tree. Given that n trees have already been trained, MART uses gradient descent to decrease the loss (Burges, 2010). Concretely, the next regression three models the m derivatives of the cost with respect to the current model score evaluated at each training point: $\frac{C}{F_N}(x_i), i = 1, ..., m$. Therefore:

$$C \qquad \frac{C(F_N)}{F_N} \; F \qquad (7)$$

If $F = -\;\frac{C}{F_N}$; $C < 0$. Each tree models the gradient of the cost with respect to the model score and the new tree is added to the ensemble with a step size ${}_{kn}$ (Burges, 2010).

2.5. LambdaMART

MART models derivatives and LambdaRank works by specifying the derivatives at any point during the training. LambdaMART is the result of combining these two algorithms (Wu et al., 2010). LambdaMART is simply MART with specific gradient and the Newton step. Just as MART, least squares is used to compute splits and the gradients are ${}_i$. Each tree models the ${}_i$ for the entire dataset and the Newton step is computed by collecting some results from before: for a given data-point pair T_i and T_j such that T_i T_j, the data-points are sorted by score and the ${}_{ij}$ defined as in Eq. (4):

$${}_{ij} = \frac{-\;|\;Z_{ij}|}{1+e^{-\;(s_i-s_j)}} \qquad (8)$$

Z can be NDCG, but more generally it is defined as the utility difference generated by swapping the rank positions of T_i and T_j as Z_{ij} (Burges, 2010). We can simplify the Eq. (3) and denote the sum operation as:

${}_{\{i,j\} \quad I \quad ij} \qquad {}_{j:\{i,j\} \quad I \quad ij} - \qquad {}_{j:\{j,i\} \quad I \quad ij}$

Therefore the utility function for which ${}_i$ is the utility, for an given state of the model or particular set of scores and for any particular data point T_i is given by:

$$C = \underset{\{i,j\} \quad I}{} \;|\; Z_{ij}|log(1+e^{=\;(s_i-s_j)}) \qquad (9)$$

such that:

$$\frac{C}{s_i} = \underset{\{i,j\} \quad I}{} \frac{-\;|\;Z_{ij}|}{1+e^{\;(s_i-s_j)}} \qquad \underset{\{i,j\} \quad I}{} -\;|\;Z_{ij}|\;{}_{ij} \qquad (10)$$

where ${}_{ij}$ $\frac{1}{1+e^{\;(s_i-s_j)}} = \frac{-\;{}_{ij}}{|Z_{ij}|}$; hence

$$\frac{{}^2C}{s_i^2} = \underset{\{i,j\} \quad I}{} -\;{}^2|\;Z_{ij}|\;{}_{ij}(1-\;{}_{ij}) \qquad (11)$$

Therefore the Newton step for the k^{th} leaf of the m^{th} tree (using $\frac{e^x}{(1+e^x)^2} = \; 1-\frac{1}{1+e^x} \;\; \frac{1}{1+e^x}$) is given by:

$${}_{km} = \frac{\underset{x_i \; R_{km}}{} \frac{C}{s_i}}{\underset{x_i \; R_{km}}{} \frac{{}^2C}{s_i^2}} = \frac{-\;\underset{x_i \; R_{km} \; \{i,j\} \quad I}{} \;|\; Z_{ij}|\;{}_{ij}}{\underset{x_i \; R_{km} \; \{i,j\} \quad I}{} \;|\; Z_{ij}|\;{}_{ij}(1-\;{}_{ij})} \qquad (12)$$

Algorithm 1 schematically summarizes the LambdaMART algorithm (Burges, 2010).

3. Statistical Convergence Perspective

In this section we will start with some preliminaries, notations and some brief background knowledge required to explore the statistical properties. The Time-Reversibility or the Bradly-Terry-Luce and a comparison matrix will also be described. Eventually we will exploit the fact that the underlying model of the LambdaMART is least squares to show that LambdaMART converges to an optimal ranking under the BTL condition.

3.1. Preliminaries, Notation and Background

For the goal of studying convergence we will formally adopt the following conventions:
Let $[n] = \{1, ..., n\}$ denote a set of n item that has to be ranked. And let $\mathcal{X} = \{(i,j) : i, j \quad [n].i < j\}$. The algorithm is given a training sample $S = ((i_1, j_1, y_1), ..., (i_m, j_m, y_m)) \quad (\mathcal{X} \times \{0,1\})^m$, where for each k $[m], (i_k, j)k)$ $\mathcal{X}$ denotes the k^{th} pair of items compared and y_k $\{0,1\}$ denotes the result of the comparison; $y_k = 1$ if j_k is ranked higher than i_k and $y_k = 0$ otherwise. Therefore given S, the goal of

Algorithm 1 LambdaMART

```
Set: $N$=number of trees, $m$=number of training samples, $L$=number of leaves per tree,  =learning rate
for $i = 0$ to $m$ do
  if BaseModel is empty then
    $F_0(x_i) = 0$
  else
    $F_0(x_i) = \text{BaseModel}(x_i)$
  end if
end for
for $k = 1$ to $N$ do
  for $i = 0$ to $m$ do
    $y_i = \ _i$
    $w_i = \frac{y_i}{F_{k-1}(x_i)}$
  end for
  // Create $L$ leaf tree on $\{x_i, y_i\}_{i=1}^m$
  $\{R_{lk}\}_{l=1}^L$
  // Assign Leaf valued based on Newton step, Eq. (11)
  $_{km} = \frac{_{x_i \ R_{km}} \frac{C}{s_i}}{_{x_i \ R_{km}} \frac{2C}{s_i^2}}$
  //With learning rate   take next step
  $F_k(x_i) = F_{k-1}(x_i) + \quad _l \ _{lk} I(x \quad \mathbf{R}_{lk})$
end for
```

the algorithm is to produce a permutation or ranking of the n items, $\quad \mathcal{S}_n$ (Rajkumar & Agarwal, 2014). The item pairs are assumed to be sampled from a probability distribution μ on $\mathcal{X}$. Let μ_{ij} denote the probability of the pair (i, j) such that $i < j$ under μ and $\mu_{ij} = \mu_{ji}$; $\ i < j$. A set of conditional label probabilities from with the labels are drawn are assumed, i.e if i and j items are compared, the probablity that i will be ranked higher than j for each $i < j$ is denoted by $P \quad [0, 1]$. This can be represented as a **pairwise preference matrix P** $\quad [0, 1]^{n \times n}$:

$$P_{ij} = \begin{array}{ccc} P_{ij} & \text{for} & i < j \\ 1 - P_{ji} & \text{for} & i > j \\ 0 & \text{for} & i = j \end{array} \qquad (13)$$

S is assumed to be drawn randomly independently; $S \quad (\mu, \mathbf{P})^m$; and $y_k \quad \text{Bernoulli}(P_{i_k, j_k})$. For the a distribution $(\mu, \mathbf{P})$ the expected pairwise disagreement error of a permutation $\quad \mathcal{S}_n$ is:

$$er^{PD}_{\mu,\mathsf{P}}[\ ] = \ _{i=j} \ \mu_{ij} P_{ij} \mathbf{1}(\ (i) < \ (j)) \qquad (14)$$

An 'optimal' permutation $\quad \mathcal{S}_n$ is assumed and is denoted by:

$$argmin \quad _{\mathcal{S}_n} er^{PD}_{\mu,\mathsf{P}}[\ ] \qquad (15)$$

The input is (or can be implicitly thought of as) a **empirical pairwise comparison matrix $\hat{\mathbf{P}}$** $\quad [0, 1]^{n \times n}$ (Rajkumar & Agarwal, 2014); which is constructed from S as:

$$\hat{P_{ij}} = \begin{array}{ccc} \frac{N_{ij}^{(1)}}{N_{ij}} & \text{for} & i < j; N_{ij} > 0 \\ 1 - \frac{N_{ji}^{(1)}}{N_{ji}} & \text{for} & i > j; N_{ij} > 0 \\ 0 & \text{for} & otherwise \end{array} \qquad (16)$$

where $N_{ij} = \ _{k=1}^m \mathbf{1}(i_k = i, j_k = j)$ and $N_{ij}^{(1)} = \ _{k=1}^m \mathbf{1}(i_k = i, j_k = j, y_k = 1)$. From definition $\mathbf{P}$ satisfies $P_{ij} + P_{ji} = 1$ for all $i < j$; but in case of $\hat{\mathbf{P}}$, if a pair $i < j$ is not observed in S we can have $\hat{P_{ij}} = \hat{P_{ji}} = 0$. Elements of $\hat{\mathbf{P}}$ satisfy a bounded difference property with high probability, therefore we can analyze $\hat{\mathbf{P}}$ with using Kutin's extension of McDiarmid's inequality (Kutin, 2002).

3.2. Properties of the Comparison Matrix $\hat{\mathbf{P}}$

Some useful properties of the comparison Matrix (Rajkumar & Agarwal, 2014) with $\mu_{min} = \min_{i<j} \mu_{ij}$ and $B(\mu_{min}) = 3(\frac{12}{\mu_{min}^2} + 3) \ln(\frac{12}{\mu_{min}^2} + 3)$:

- Let $i = j$, if $m \quad \frac{4}{\mu_{min}}$, then using Hoeffding's inequality $\hat{\mathbf{P}_{\mathsf{ij}}}$ is strongly difference-bounded by $(1, \frac{2}{m\mu_{min}}, \exp(-m\mu_{min}^2))$
- Let $i = j$, let $0 < \quad < 2 \quad \overline{2}$. if $m \quad B(\mu_{min})$, then $\mathbf{P}(|\hat{P_{ij}} - \mathbf{E}[\hat{\mathbf{P}_{\mathsf{ij}}}]| \quad) \quad 4\exp(\frac{-m \ ^2\mu_{min}^2}{32})$
- Let $i = j$, let $\quad > 0$, since $\mathbf{E}[\hat{\mathbf{P}_{\mathsf{ij}}}] = p_{ij}(1 - (1 - \mu_{ij})^m)$; if $m \quad \frac{1}{\mu_{min}} \ln(\frac{1}{\ })$, then $|\mathbf{E}[\hat{\mathbf{P}_{\mathsf{ij}}}] - P_{ij}|$
- let $i = j$, let $0 < \quad < 4 \quad \overline{2}$; if m $\max(B(\mu_{min}), \frac{1}{\mu_{min}} \ln(\frac{2}{\ }))$, then $\mathbf{P}(|\hat{P_{ij}} - P_{ij}| \quad) \quad 4\exp(\frac{-m \ ^2\mu_{min}^2}{128})$
- Let $P_{ij} \quad (0, 1) \quad i = j$, and let $P_{min} = \min_{i=j} P_{ij}$, let $\quad (0, 1]$; if $m \quad \frac{1}{\mu_{min} P_{min}} \ln(\frac{n(n-1)}{\ })$, then with probablity at least $1 - \ $, $\hat{P_{ij}} > 0, \quad i = j$.

3.3. Time-Reversibility or Bradly-Terry-Luce (BTL) Condition

The pairwise preference matrix $\mathbf{P} \quad [0, 1]^{n \times n}$ satisfies the time-reversal condition if the Markov chain $\mathbf{Q}$ given by:

$$Q_{ij} = \begin{array}{ccc} \frac{1}{n} P_{ij} & \text{if} & i = j \\ 1 - \frac{1}{n} \quad _{k=i} P_{ik} & \text{if} & i = j \end{array} \qquad (17)$$

is time-reversible, i.e. if **Q** is irreducible and aperiodic and the stationary probability vector of **Q** satisfies $_iQ_{ij} = {}_jQ_{ji}\ \ i, j \quad [n]$.
P satisfies the Bradly-Terry-Luce (BTL) condition if it corresponds to a BTL model, i.e. if **w** $\mathbf{R}^n_+$ with $w_i > 0\ \ i$ such that $P_{ij} = \frac{w_j}{(w_i+w_j)}\ \ i = j$. **P** satisfies time-reversible condition iff it satisfies the BTL condition (Rajkumar & Agarwal, 2014) and $P_{ij} \quad (0,1)\ \ i = j$.

3.4. Convergence of LambdaMART

To analyze the convergence of LambdaMART under previously defined reversibility/BTL condition on the preference matrix **P**, let's recall the fact that the underlying model is MART which uses least squares to compute the splits and the lambdas are nothing but gradients. Therefore converges of LambdaMART is very similar to that of the least squares algorithm (Jiang et al., 2011; Rajkumar & Agarwal, 2014). $\hat{\mathbf{P}}$ is converted to a skew-symmetric matrix $\hat{\mathbf{Y}}$ (Jiang et al., 2011); therefore for a give preference matrix **P**, a skew-symmetric matrix **Y** $\mathbf{R}^{n\times n}$ can be defined as:

$$Y_{ij} = \begin{matrix} \ln(\frac{P_{ij}}{P_{ji}}) & \text{if} & i = j; P_{ij} \quad (0,1) \\ 0 & \text{otherwise} & \end{matrix} \tag{18}$$

Let $E = \{(i,j) \quad : P_{ij} = 0||P_{ji} = 0\}$,
and let **f** $argmin_{\mathbf{f}\ \mathbf{R}^n} \quad {}_{(i,j)\ E}((f_i - f_j) - Y_{ij})^2$
Since by definition $P_{ij} = 1 - P_{ji}\ \ i = j;\ E = \mathcal{X}$
Therefore the solution (minimum norm) to the optimization problem as discussed in (Jiang et al., 2011):

$$f_i = -\frac{1}{n} \quad {}^{n}_{k=1} Y_{ik} \tag{19}$$

Lemma 20: let $(\mu, \mathbf{P})$ be such that **P** satisfies the BTL condition, let **f** $\mathbf{R}^n$ be defined as in Eq. (19) then $argsort(\mathbf{f}\ )\quad argmin \quad {}_{\mathcal{S}_n} er^{PD}_{\mu,\mathbf{P}}[\ ]$.

In other words if **P** satisfies time-reversibility/BTL condition, then ranking the items according to the decreasing order of scores f_i as in Eq. (19) outputs an optimum ranking (Rajkumar & Agarwal, 2014) w.r.t the pairwise disagreement error defined in Eq. (15).

Theorem 21: Let $(\mu, \mathbf{P})$ be such that $\mu_{min} > 0$ and $P_{ij} \quad (0,1)\ \ i = j$. Let **Y** $\mathbf{R}^{n\times n}$ and **f** $\mathbf{R}^n$ be defined as in Eq. (18) and Eq. (19), Let $P_{min} = \min_{i=j} P_{ij}$, let $0 < \quad 1$ and $\quad (0,1]$, if $m \quad \max(\frac{128}{P^2_{min}\mu^2_{min}}(1 + {}^2)^2 \ln(\frac{16n^2}{}), B(\mu_{min}))$, then with probability at least $1 -$ (over the random draw of $S \quad (\mu, \mathbf{P})^m$ from which $\hat{\mathbf{P}}$ is constructed), the score vector $\hat{f}$ produced by the algorithm satisfies $||\hat{\mathbf{f}} - \mathbf{f}\ ||$.

Consequently the following complexity bound can be inferred (Rajkumar & Agarwal, 2014):

Corollary 22: Let $(\mu, \mathbf{P})$ be such that $\mu_{min} > 0$ and **P** satisfies the BTL condition, and $(i = j) : P_{ij} = \frac{1}{2}$. Let **f** as Eq. (19) and let $r_{min} = \min_{i,j:f_i = f_j} |f_i - f_j|$. Let $(0,1]$. if $m \quad \max(\frac{128}{P^2_{min}\mu^2_{min}}(1 + \frac{6}{r_{min}})^2 \ln(\frac{16n^2}{}), B(\mu_{min}))$, then with probability at least $1 -$ (over the random draw of $S \quad (\mu, \mathbf{P})^m$ from which $\hat{\mathbf{P}}$ is constructed), the permutation $\hat{}$ output by the algorithm satisfies: $\hat{} \quad argmin \quad {}_{\mathcal{S}_n} er^{PD}_{\mu,\mathbf{P}}[\ ]$

4. Future/Related Work

Currently I am working on an open-source python implementation of LambdaMART. Also I am planning on implementing LambdaMART for the RecSys challenge (Chapelle & Chang; Mobasher et al., 2012) and try improving the results by using more enriched feature vectors. For this reason movie data was collected by crawling www.imdb.com to supplement the the tweeter data.

5. Conclusion

In this paper we looked at the statistical converge perspective of LambdaMART under the Bradly-Terry-Luce (BTL) condition. We have seen that under the BTL condition LambdaMART converges to an optimal ranking or permutation. The problem of rank aggregation from pairwise comparison data has received much interest recently and hence it an interesting machine-learning algorithm to study.

References

Burges, Chris, Shaked, Tal, Renshaw, Erin, Lazier, Ari, Deeds, Matt, Hamilton, Nicole, and Hullender, Greg. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pp. 89–96. ACM, 2005.

Burges, Christopher JC. From ranknet to lambdarank to lambdamart: An overview. 2010.

Burges, Christopher JC, Le, Quoc Viet, and Ragno, Robert. Learning to rank with nonsmooth cost functions. 2006.

Chapelle, Olivier and Chang, Yi. Yahoo! learning to rank challenge overview.

Chapelle, Olivier, Metlzer, Donald, Zhang, Ya, and Grinspan, Pierre. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 621–630. ACM, 2009.

Donmez, Pinar, Svore, Krysta M, and Burges, Christopher JC. On the local optimality of lambdarank. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 460–467. ACM, 2009.

Dwork, Cynthia, Kumar, Ravi, Naor, Moni, and Sivakumar, D. Rank aggregation revisited, 2001.

Friedman, Jerome H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Järvelin, Kalervo and Kekäläinen, Jaana. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

Jiang, Xiaoye, Lim, Lek-Heng, Yao, Yuan, and Ye, Yinyu. Statistical ranking and combinatorial hodge theory. *Mathematical Programming*, 127(1):203–244, 2011.

Kutin, Samuel. Extensions to mcdiarmids inequality when differences are bounded with high probability. *Dept. Comput. Sci., Univ. Chicago, Chicago, IL, Tech. Rep. TR-2002-04*, 2002.

Liu, Tie-Yan. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. ISSN 1554-0669. doi: 10.1561/1500000016. URL http://dx.doi.org/10.1561/1500000016.

Mobasher, Bamshad, Jannach, Dietmar, Geyer, Werner, and Hotho, Andreas. 4th acm recsys workshop on recommender systems and the social web. In *RecSys*, pp. 345–346, 2012.

Rajkumar, Arun and Agarwal, Shivani. A statistical convergence perspective of algorithms for rank aggregation from pairwise data. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 118–126, 2014.

Wu, Qiang, Burges, Christopher JC, Svore, Krysta M, and Gao, Jianfeng. Adapting boosting for information retrieval measures. *Information Retrieval*, 13 (3):254–270, 2010.

Yue, Yisong and Burges, C. On using simultaneous perturbation stochastic approximation for learning to rank, and the empirical optimality of lambdarank. Technical report, Tech. Rep. MSR-TR-2007-115, Microsoft Research, 2007.