Solving the Ranking Problem Using Continuous Optimization Methods

Author: Yuri Sapronov
Supervisor: A. N. Beznosikov
Scientific Advisor: A. V. Gasnikov

Learning-to-Rank (LTR) Task

- Given:
  - A query $q \in Q$ (set of possible queries).
  - A set of associated documents $D_q = \{d_1, d_2, \dots, d_n\}$.
- Each query-document pair (q, d_i) is represented as a feature vector:
  $$x_i = f(q, d_i), \quad x_i \in \mathbb{R}^k,$$
  where $f(q, d_i)$ is a feature extraction function.
- The task is to learn a scoring function $s(x_i)$ that ranks documents $d_i \in D_q$ by relevance.
- Output: A permutation π of documents such that $s(x_{\pi(1)}) \geq s(x_{\pi(2)}) \geq \dots \geq s(x_{\pi(n)})$.

Evaluating Ranking Algorithms

- Various metrics are used to evaluate the quality of a ranking algorithm.
- One of the most popular metrics is the Normalized Discounted Cumulative Gain (NDCG).

Normalized Discounted Cumulative Gain (NDCG)

- For a ranked list of documents π:

$$\mathrm{DCG}(\pi, p) = \sum_{i=1}^{p} \frac{2^{\mathrm{rel}_{\pi(i)}} - 1}{\log_2(i+1)}$$

- Ideal DCG (IDCG):

$$\mathrm{IDCG}(p) = \mathrm{DCG}(\pi^*, p),$$

  where π^* is the ideal permutation of documents sorted by relevance.
- Normalized DCG:

$$\mathrm{NDCG}(\pi, p) = \frac{\mathrm{DCG}(\pi, p)}{\mathrm{IDCG}(p)}$$

- The metric evaluates how closely the ranked list π matches the ideal ranking π^*.

Two Main Approaches

- Gradient Boosting.
- Neural Networks.

In our work, the primary focus is on using neural networks.

StochasticRank Results

Method	Dataset	NDCG@5	MRR
λ-MART	Yahoo Set 1	74.53	90.21
λ-Loss	Yahoo Set 1	74.73	-
Eλ-MART	Yahoo Set 1	74.57	90.30
Eλ-Loss	Yahoo Set 1	74.57	-
SoftRank	Yahoo Set 1	71.98	90.17
SR-R_1^{soft}	Yahoo Set 1	74.68	**91.07**
SR-R_1	Yahoo Set 1	**74.92**	90.97
λ-MART	Yahoo Set 2	73.87	91.48
λ-Loss	Yahoo Set 2	73.89	-
Eλ-MART	Yahoo Set 2	73.91	91.48
Eλ-Loss	Yahoo Set 2	73.91	-
SoftRank	Yahoo Set 2	73.91	92.16
SR-R_1^{soft}	Yahoo Set 2	73.95	93.16
SR-R_1	Yahoo Set 2	**74.15**	**93.56**
λ-MART	WEB10K	48.22	81.85
λ-Loss	WEB10K	48.33	-
Eλ-MART	WEB10K	48.29	81.72
Eλ-Loss	WEB10K	48.47	-
SoftRank	WEB10K	42.82	81.38
SR-R_1^{soft}	WEB10K	48.19	83.08
SR-R_1	WEB10K	**48.53**	**83.30**
λ-MART	WEB30K	49.55	83.79
λ-Loss	WEB30K	49.45	-
Eλ-MART	WEB30K	49.49	83.79
Eλ-Loss	WEB30K	49.52	-
SoftRank	WEB30K	43.46	82.73
SR-R_1^{soft}	WEB30K	**49.67**	**85.19**
SR-R_1	WEB30K	49.59	85.01

Table: Performance Comparison of StochasticRank and Other Methods

Work Objective

- To surpass the best performance of gradient boosting using a neural architecture.

ListNet Method

- One of the first methods based on neural networks.
- Loss function:

$$\mathcal{L}(f \mid \mathcal{D}_q) = -\frac{1}{|\mathcal{D}_q|} \sum_{(x,y)\in\mathcal{D}_q} \sum_{i=1}^{N_q} P(y_i \mid \mathcal{D}_q) \log P'(x_i \mid f, \mathcal{D}_q)$$

where:

$$P'(x_i \mid f, \mathcal{D}_q) = \frac{\exp(f(x))}{\sum_{x'\in\mathcal{D}_q} \exp(f(x'))}.$$

- Applying softmax to labels:

$$P(y_i \mid \mathcal{D}_q) = \frac{\exp(y_i)}{\sum_{y'\in\mathcal{D}_q} \exp(y')}.$$

Comparison of ListNet and StochasticRank

Method + Architecture	NDCG@5	NDCG@10	NDCG@all
ListNet + MLP	**50.01**	**50.32**	74.25
StochasticRank + GB	48.53	49.71	**78.81**

Table: Performance comparison of different methods and architectures

Alternative Distributions for ListNet

- The target distribution in the loss function can use functions other than softmax.
- We decided to use labels directly as probabilities.

Comparison of ListNet and StochasticRank

Method + Architecture	NDCG@5	NDCG@10	NDCG@all
ListNet + MLP	50.01	50.32	74.25
Modified ListNet + MLP	**51.15**	**51.73**	**75.29**

Table: Comparison of default ListNet and modified one

Table of Other Methods Tried

Method + Architecture	NDCG@5	NDCG@10	NDCG@all
ListNet + MLP	53.5	53.8	77.6
SoftNDCG + TabNet	40.9	45.1	73.2
ListNet + TabNet	47.0	51.01	76.08
SoftNDCG + MLP	24.24	27.21	63.49

Table: Performance comparison of other known methods on the Web10k dataset.

Transformer Architecture

- Idea: Incorporate document relationships not only in the loss function but also in the model architecture.

Comparison of Transformer Configurations

Parameter	Paper Configuration	My Configuration
Number of Blocks	4	2
Number of Heads	4	4
Hidden Size	512	512
Feedforward Size	2048	2048
Dropout Rate	0.3	0.3
Number of Parameters	6.3M	3.1M
Learning Rate	0.001	0.001
Batch Size	240	256

Table: Comparison of transformer configurations: paper vs. custom.

Training Transformer

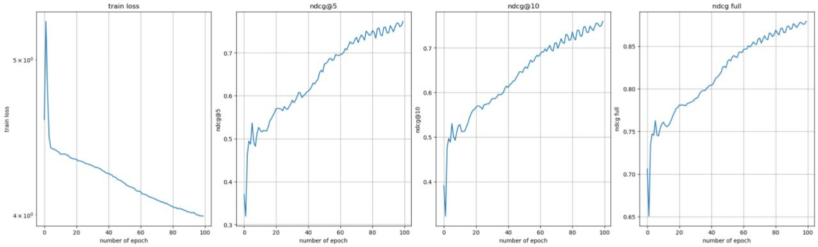

Figure: Loss and metric evaluation across epochs.

Final Comparison

Method + Architecture	NDCG@5	NDCG@10	NDCG@all
StochasticRank + Gradient Boosting	48.3	49.2	78.9
Ordinal Loss + Transformer	53.0	54.9	75.2
ListNet + our Transformer	**79.5**	**79.7**	**91.7**

Table: Performance comparison of the best approaches on the Web10k dataset.

Future Work Plan

- Further optimize the transformer architecture.
- Experiment with additional datasets.
- Attempt to integrate mirror gradients methods into the training logic. Mirror AdamW is of the most interest.