IE 506: Machine Learning - Principles and Techniques **Jan-May 2024**

Project Mid-term Report : Learning To Rank Using Gradient Descent

Team Name: Macrohard *Team Members: 210110063*

Abstract

In the paper, the efficacy of gradient descent methods for training ranking functions is explored, along with a proposal of a simple probabilistic cost function specifically tailored for this purpose. Additionally, RankNet, a neural network implementation aimed at modeling the underlying ranking function, is introduced. By utilizing neural networks, scores for documents relevant to specific queries are generated with the intention of enhancing search engine performance. Thorough testing on data from a commercial internet search engine is conducted to evaluate the effectiveness of the approach. The research contributes valuable insights and methodologies to the field of learning to rank, offering practical solutions for optimizing ranking functions using neural network-based techniques like RankNet.

1 Introduction

Any system that arranges outcomes for a user based on a utility function of user interest operates through a ranking function. An illustrative scenario is the arrangement of search outcomes, such as those from the internet or an intranet, which is the focus of this paper. In this context, the dataset comprises sets of queries and their corresponding retrieved documents. During the training phase, certain query-document pairs are annotated for relevance, typically categorized as "excellent match" or "good match," among others. Notably, only documents retrieved for a particular query are compared against one another for ranking. Thus, the dataset is structured by query, unlike a conventional dataset where objects are ranked uniformly against each other.

In this paper, a novel approach is proposed to address this ranking problem. A method is adopted that trains on pairs of examples to learn a ranking function, mapping to real values. However, unlike prior approaches, which treat the ranking problem as an ordinal regression task, this approach dispenses with the need for explicit rank boundaries. Instead, the focus is on the relative ordering of items in the output list. For the application, if item A appears above item B in the output, the user infers that A is ranked higher or equal to B. Consequently, the complexities associated with ordinal regression are avoided, streamlining the problem-solving process.

Additionally, a natural probabilistic cost function for pairs of examples is introduced. This approach is agnostic to the specific learning algorithm used. The exploration of these concepts is conducted using neural networks due to their versatility and computational efficiency. It is demonstrated that backpropagation can be readily extended to handle ordered pairs, resulting in the algorithm when combined with the proposed probabilistic cost function.

The experimentation involves testing on data collected from a commercial internet search engine. Specifically, the dataset comprises 325 queries, each potentially yielding up to 40 retrieved documents. These documents represent the top outcomes generated by a basic ranker. Consequently, each query generates a set of up to 46 feature vectors. Notation-wise, the number of relevance levels or ranks is denoted by N, the training sample size by m, and the dimensionality of the data by d.

2 Other Ranking Models

RankProp is a neural network-based ranking model, introduced in 1996. It operates through two phases: first, an MSE regression is performed on the current target values, followed by an adjustment of these target values to align with the ranking provided by the neural network. This process yields a mapping of the data to numerous targets that represent the desired ranking, outperforming simple regression toward the original scaled rank values. RankProp offers the advantage of being trained on individual patterns rather than pairs. However, its convergence under various conditions remains unknown, and it does not yield a probabilistic model.

Approaches framed the task of learning to rank as ordinal regression. This involves mapping input vectors to members of an ordered set of numerical ranks, treating ranks as intervals on the real line. They consider loss functions relying on pairs of examples and their target ranks. The placement of rank boundaries is crucial for determining the final ranking function. A ranking model based on the perceptron algorithm maps feature vectors to real numbers using learned weights and thresholds. It learns one example at a time, which is advantageous compared to pair-based methods requiring learning with a higher number of pairs. However, in certain applications, pair-based methods may be more feasible due to a substantially smaller number of pairs compared to the square of the number of examples. PRank's performance has been compared to batch ranking algorithms, with a quadratic kernel version showing superiority over all such algorithms.

An extension to PRank aims to approximate the Bayes point by averaging over PRank models. The paper intends to compare RankNet with various models, including PRank, kernel PRank, large margin PRank, and RankProp.

A comprehensive framework for ranking tasks utilizing directed graphs accommodates arbitrary ranking functions. They introduce a probabilistic model assigning posterior probabilities to each training pair (A ¿ B). This probabilistic aspect is crucial as ranking algorithms often model preferences, which are inherently subjective. They suggest determining target probabilities through multiple human preferences for each pair and propose cost functions based on the difference between the system's outputs for each pair of examples.

RankBoost operates on pairs of examples and directly tackles the preference learning problem. The authors mention successful applications of RankBoost with decision stumps as weak learners, while its cost function considers margins over reweighted examples. Since boosting can be understood as a form of gradient descent, the text ponders the potential of combining RankBoost with the proposed differentiable cost function.

3 Methods and Approaches

Traditional pointwise ranking methods struggle to capture the relative importance or order among items directly. To address this limitation, we propose investigating gradient descent methods for learning ranking functions. We introduce RankNet, a neural network-based approach utilizing a simple probabilistic cost function. Our aim is to enhance the accuracy of ranking algorithms, particularly in internet search engines.

RankNet is a pairwise ranking algorithm that learns to rank documents based on their pairwise preferences over a given query Q.
d_1,d_2,...d_n be the set of documents
S_i be the relevance score of document d_i for query Q
f(Q,(d_1,d_2), (d_1,d_3),... (d_n,d_{n-1})) objective is to learn this function using neural network that scores the documents

Let's understand the RankNet approach: The RankNet function aims to learn a mapping from the query-document pairs to their relevance scores, typically using a neural network. y_i denotes the predicted relevance

score for document d_i given query Q_i.
For a given query, each pair of documents or URLs (U_i, U_j) with different relevance level will be chosen. And let (y_i, y_j) be the computed label from a ranking model. Let $U_i \Rightarrow U_j$ denotes the event that $y_i > y_j$. Then a posterior is defined as:

$$P_{ij} = P(U_i \Rightarrow U_j) = \frac{1}{1 + e^{-(y_i - y_j)}} \tag{1}$$

Then, define the cost function of this pair to be:

$$C_{ij} = -\tilde{P}ij \times \log Pij - (1 - \tilde{P}ij) \times \log(1 - \tilde{P}ij) \tag{2}$$

$\tilde{P}_{ij}$ is the known probability that the ranking of U_i is greater than U_j. Let S_{ij} to be defined to be $\mathbf{1}$, if U_i is labeled more relevant, $\mathbf{0}$ if U_i is not labeled more relevant. Then $\tilde{P}_{ij}$ can be defined as:

$$\tilde{P}ij = \frac{1}{2} \times (1 + Sij) \tag{3}$$

Therefore, the cost function will become:

$$C_{ij} = \frac{1}{2} \times (1 - S_{ij}) \times (y_i - y_j) + \log(1 + e^{-(y_i - y_j)}) \tag{4}$$

It is easy to see that cost function is larger when $U_i \Rightarrow U_j$ while U_j is actually known to be more relevant. And the idea of learning via gradient descent is to update the weights (model parameters) with the gradient of the cost function. Then a neutral network will be used to optimize the ranking model to minimize the cost function with the back-prop equations.

$$W_k \to W_k - \eta \left(\frac{dC_{ij}}{dy_i} \cdot \frac{dy_i}{dW_k} + \frac{dC_{ij}}{dy_j} \cdot \frac{dy_j}{dW_k} \right) \tag{5}$$

Using a neural network we get probabilistic values for each pair of documents for a particular query. We then form a nxn probabilistic matrix for each query, where n represents the number of documents in each query. We then take a summation of all the rows and this sum gives us a value that is proportional to the scores for each document. We then arrange these scores in decreasing order to get a ranking model. To this ranking model we apply NDCG ranking metric to gauge the performance of our ranking model.

4 Dataset Details and Evaluation Metric

The experiment utilized the LETOR 4.0: MQ2007 dataset, structured in text format and divided into five files to facilitate cross-validation. This dataset encompasses approximately 13,000 documents, each meticulously linked to a specific query ID. The division of data into query-document pairs facilitates the evaluation of relevance in information retrieval tasks. On average, each query is associated with around 40 documents, providing a substantial pool for analysis and model training. Each document in the dataset is described by a set of 46 features, derived from both the query and document content, serving as essential descriptors capturing various aspects of the relationship between the query and the document.

By incorporating diverse features, the model can better understand the relevance of documents to specific queries, thus enhancing the effectiveness of information retrieval systems. The relevance of each document

is quantified using a relevance label ranging from 0 to 4, where 0 indicates the most irrelevant and 4 the highest level of relevance. These relevance labels serve as ground truth annotations, enabling the evaluation of the performance of information retrieval models in ranking documents according to their relevance to given queries.

Additionally, the evaluation metric utilized in this experiment is the Normalized Discounted Cumulative Gain (NDCG) at rank 15. This metric measures the quality of a ranking by considering the graded relevance of the documents retrieved by the system. For queries with fewer than 15 returned documents, the NDCG is computed for all the returned documents. The inclusion of probability values in the dataset provides insights into the confidence associated with the relevance assessment, offering a nuanced understanding of the uncertainty inherent in relevance judgments. By focusing on relevant features and labels, the experiment aims to develop robust models for information retrieval that can effectively prioritize documents based on their relevance to user queries.

5 Experiment

Neural Network Architecture:

The neural network architecture consists of several key components, starting with the input layer, which is structured with the same number of neurons as features present in the training dataset. This ensures that the network can effectively process the input data. Following the input layer, two hidden layers constitute the neural network. The initial hidden layer comprises 64 neurons utilizing the Rectified Linear Unit (ReLU) activation function, while the subsequent hidden layer comprises 32 neurons, also utilizing the ReLU activation function. These hidden layers play a crucial role in capturing complex patterns and relationships within the data. Moving on to the output layer, it is composed of a single neuron with the Sigmoid activation function. This choice is appropriate for the binary classification task at hand, where the network outputs the probability of a document being assigned a particular relevance label. By utilizing the Sigmoid activation function, the output is constrained between 0 and 1, representing the probability of the document belonging to the positive class.

For training and optimization, the model is compiled with the binary cross-entropy loss function, which is well-suited for binary classification tasks. This loss function quantifies the difference between the predicted probabilities and the actual labels. To optimize the model's parameters, the Adam optimizer is employed. The Adam optimizer is known for its effectiveness in handling large datasets and diverse model architectures, making it a popular choice for deep learning tasks.

Training the model involves utilizing the entire dataset (Xtrain, ytrain) over 10 epochs, with a batch size set to 32. This approach ensures comprehensive learning over multiple iterations, enhancing the model's ability to generalize to unseen data instances. By training over multiple epochs and adjusting the model's parameters iteratively, the neural network learns to make accurate predictions on new data.

Regarding hardware specifications, the system utilizes a Tesla K80 GPU with a compute capability of 3.7, equipped with 2496 CUDA cores and 12GB GDDR5 VRAM. This GPU facilitates accelerated model training and inference tasks, significantly reducing the computational time required for complex deep learning operations. Additionally, the system features a single-core hyper-threaded Xeon processor operating at 2.3GHz, providing supplementary computational support. Approximately 12.6 GB of available RAM ensures ample memory resources for data processing, while around 33 GB of available disk space accommodates storage needs for datasets and model-related files. Leveraging Google Colab as the computational platform enables efficient model training and experimentation, thanks to its cloud-based infrastructure and readily available computing resources.

The experiment was conducted in two parts. Initially, the model was trained without considering L2 regularization, and subsequently, another iteration was performed by incorporating L2 regularization to prevent overfitting. A regularization constant of 0.01 was applied for L2 regularization. This two-part approach helps evaluate the impact of regularization on the model's performance and generalization capabilities.

6 Results

The NDCG score obtained without L2 regularization is 0.5619. While when we used L2 regularization the NDCG score on training data is 0.5214. Optimization was ahieved by checking variation of NDCG with parameters such as the number of nodes, the number of hidden layers, and the batch size/epochs during training.Fine-tuning these parameters can potentially lead to improved ranking performance beyond the initial baseline.

7 Work Done After Midterm

7.1 Introduction

In our study, we've employed the gradient descent approach to rank documents for given queries. However, a fundamental issue arises with this method, as with similar ranking approaches like RankNet. These models aim to enhance accuracy by minimizing a loss function, typically represented by the disparity between the true value and the predicted score for a given document. For instance, RankNet utilizes binary cross-entropy as its loss function and adjusts network weights to minimize it. While effective in theory, this approach lacks direct optimization of the parameter by which the algorithm's efficacy is judged, in our case, NDCG (Normalized Discounted Cumulative Gain).

To address this limitation, the authors propose strategies to optimize the loss function by penalizing incorrect ranking sequences. One key adaptation involves optimizing a variant of NDCG known as NDCGβ. Unlike the classical NDCG, NDCGβ is a linear function of the ranking position and tends to assign higher scores to nearly ideal ranking sequences. Additionally, the research introduces RankBoostndcg and RankSVMndcg algorithms, designed specifically to optimize the upper bound of the pair-wise loss function.

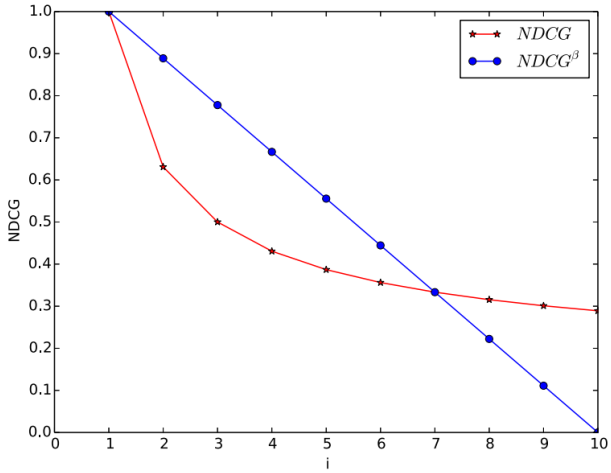

Figure 1: Comparisons of NDCG and NDCGβ measures for a sequence with ten irrelevant instances and one relevant instance, when the relevant one appears in different locations.

7.2 Ranking with Pairwise loss

The instance space is represented by $X \subseteq \mathbb{R}^d$, where d is the number of features. Suppose we are given a set of K queries, denoted by $Q = \{q_k\}_{k=1}^K$. For each query q_k, a collection of instances $S_k \subseteq X$ is related to it. We represent this set of related instances as $S = \{S_k\}_{k=1}^K$.

In a multipartite, or more specifically, an L-partite ranking problem, the set of instances S_k from the query q_k is partitioned into L disjoint subsets $\{S_{kl}\}$ for $0 \leq l \leq L-1$ according to the rating l of the instance, where $S_k = \cup_{l=0}^{L-1} S_{kl}$.

The purpose of a ranking algorithm with pairwise loss is to develop a ranking function $f : X \to \mathbb{R}$. Let x_{ka_i} be the ith instance from S_{ka} and x_{kb_j} be the jth one from S_{kb}. It is desirable to have $f(x_{ka_i}) < f(x_{kb_j})$ for any pair x_{ka_i} and x_{kb_j} such that $a < b$, where (x_{ka_i}, x_{kb_j}) is called a crucial pair [13]. However, it is impossible to have $f(x_{ka_i}) < f(x_{kb_j})$ for all crucial pairs in practical problems. A ranking error occurs when $f(x_{ka_i}) \geq f(x_{kb_j})$. The ranking algorithm then determines a ranking function f to minimize the expected empirical error represented by $R_I(f)$:

$$R_I(f) = \sum_{k=1}^{K} \sum_{0 \leq a < b < L} \sum_{i=1}^{|S_{ka}|} \sum_{j=1}^{|S_{kb}|} I[f(x_{kb_j}) \leq f(x_{ka_i})] \tag{6}$$

where $|S_{ka}|$ and $|S_{kb}|$ denote the number of elements in sets S_{ka} and S_{kb} respectively, and $I[\pi]$ is defined as 1 when the predicate π holds and 0 otherwise.

7.3 Evaluation Measure

In information retrieval, NDCG and MAP are often used to measure the effectiveness of web search engine algorithms.

The DCG measures the gain of an instance (document) based on its position in the result list. The gain is accumulated from the top of the result list to the bottom, and the gain of each result at a lower rank is discounted. Given any permutation g_k of the set S_k and its ratings set $\{y_{ki}\}_{i=1}^{|g_k|}$, DCG is defined as follows:

$$DCG(g_k) = \sum_{i=1}^{|g_k|} \frac{2^{y_{ki}} - 1}{\log_2(i+1)}, \tag{7}$$

where $y_{ki} \in \{0, 1, \ldots, L-1\}$. The quantity DCG varies in length depending on the query and requires normalization, which is achieved by dividing it by the maximum DCG corresponding to the ideal ranking sequence π_k produced by sorting all instances in the sequence g_k by their ratings:

$$NDCG(g_k) = \frac{DCG(g_k)}{DCG(\pi_k)}. \tag{8}$$

For a query-dependent ranking, the NDCG should be computed as the average NDCG of K ranking sequences:

$$NDCG(G) = \frac{1}{K} \sum_{g_k \in G} NDCG(g_k), \tag{9}$$

where $G = \{g_k\}_{k=1}^K$, and g_k is related to the query q_k.

In particular, the $DCG(g_k)$ at position p is denoted as $DCG@p(g_k)$:

$$DCG@p(g_k) = \sum_{i=1}^{\min\{p,|g_k|\}} \frac{2^{y_{ki}} - 1}{\log_2(i+1)}. \tag{10}$$

The $NDCG@p(g_k)$ and $NDCG@p(G)$ values can be derived from $DCG@p(g_k)$ as follows:

$$NDCG@p(g_k) = \frac{DCG@p(g_k)}{DCG@p(\pi_k)}, \quad NDCG@p(G) = \frac{1}{K} \sum_{g_k \in G} NDCG@p(g_k), \tag{11}$$

where p is often set to 10 in practice.

7.4 NDCGβ Measure

In this paper, for convenience, we first assume that the set of all instances S with their ratings $\{y_i\}_{i=1}^{|S|}$ are related to a single query q.

The $NDCG_\beta$ on any permutation g of S is calculated as follows:

$$DCG_\beta(g) = \sum_{i=1}^{|g|} y_i(|g| - i), \quad NDCG_\beta(g) = \frac{DCG_\beta(g)}{DCG_\beta(\pi)}, \tag{12}$$

where $y_i \in \{0, 1, \ldots, L-1\}$ and $DCG_\beta(\pi)$ is the normalization factor, which is the DCG value of the ideal permutation π ($DCG_\beta(g) \leq DCG_\beta(\pi)$). We call it the DCG error term

$$\Delta DCG_\beta(\pi, g) = DCG_\beta(\pi) - DCG_\beta(g) \tag{13}$$

Note that $NDCG_\beta$ in (12) is different from the classical NDCG in (7) and (8). Fig. 1 shows a comparison of two $NDCG$ measures for a sequence with ten irrelevant instances and one relevant instance, when the relevant one appears in different locations. Due to its linearity, $NDCG_\beta$ tends to assign a higher score to a nearly ideal ranking sequence than does $NDCG$. In the worst case, $NDCG_\beta = 0$ and $NDCG > 0$. In particular, for $K = 1$, we rewrite (1) as follows (the superscript k is omitted):

$$RI(f) = \sum_{0 \leq a < b < L} \sum_{i=1}^{|S_a|} \sum_{j=1}^{|S_b|} (b - a) I[f(x_{b_j}) \leq f(x_{a_i})]. \tag{14}$$

7.5 Optimizing NDCGβ

Let $\phi(f, x_{ka_i}, x_{kb_j})$ be the function $I[f(x_{kb_j}) \leq f(x_{ka_i})]$. We know that the maximization of $NDCG_\beta$ is equivalent to the minimization of the following loss:

$$1 - NDCG_\beta(G) = \frac{1}{K} \sum_{k=1}^{K} \left(1 - \frac{DCG_\beta(g_k)}{DCG_\beta(\pi_k)}\right) \tag{15}$$

$$= \frac{1}{K} \sum_{k=1}^{K} \frac{\Delta DCG_\beta(\pi_k, g_k)}{DCG_\beta(\pi_k)} \tag{16}$$

$$= \frac{1}{K} \sum_{k=1}^{K} \sum_{0 \leq a < b < L} \sum_{i=1}^{|S_{ka}|} \sum_{j=1}^{|S_{kb}|} \frac{b - a}{K \cdot DCG_\beta(\pi_k)} \phi(f, x_{ka_i}, x_{kb_j}) \tag{17}$$

$$= \mathcal{L}(f), \tag{18}$$

where we denote the right side of the equation as $\mathcal{L}(f)$.

Given that the instance set with K queries contains V crucial pairs, we can reformulate (18) by numbering all the crucial pairs:

$$\mathcal{L}(f) = \sum_{i=1}^{V} \tau_i \phi(f, x_i^-, x_i^+), \tag{19}$$

where $(x_i^-, x_i^+) \in \{(x_i^-, x_i^+) | x_i^- \in S_{ka}, x_i^+ \in S_{kb}, a < b, k = 1, 2, \ldots, K\}$ is the ith crucial pair, and τ_i is a coefficient related to the pair (x_i^-, x_i^+). Note that τ_i will be $\frac{b-a}{K \cdot DCG_\beta(\pi_k)}$ when x_i^- and x_i^+ are taken from sets S_{ka} and S_{kb} separately.

In general, optimizing based on the loss $\phi(\cdot) = I(\cdot)$ is computationally intractable due to its discontinuity. Thus, it is commonly replaced with the convex surrogate loss, which can be regarded as the convex upper bound of $I(\cdot)$. In the classical pair-wise approaches, RankSVM uses the hinge loss $\phi(f, x_i^-, x_i^+) = (1 - [f(x_i^+) - f(x_i^-)])_+$, and RankBoost uses the exponential loss $\phi(f, x_i^-, x_i^+) = \exp(-[f(x_i^+) - f(x_i^-)])$.

In the framework of RankSVM, we add the regularization term $\mathcal{R}(w)$ into (19), letting $f(x) = w^T x + b$, as follows:

$$\min_w \mathcal{R}(w) = \sum_{i=1}^{V} \tau_i (1 - w^T (x_i^+ - x_i^-))_+ + \lambda \mathcal{L}(w), \tag{20}$$

where $\mathcal{R}(w)$ may be of the form $\|\boldsymbol{w}\|_2^2$ or $\|\boldsymbol{w}\|_1$ and λ is a regularization factor.

For RankSVMndcg, we use two common loss functions, $[(1 - w^T (x_i^+ - x_i^-))_+]^2$ and $(1 - w^T (x_i^+ - x_i^-))_+$. The former is under the framework of L2-SVM while the latter is under that of L1-SVM. L1 regularization with $\mathcal{R}(w) = \lambda \|\boldsymbol{w}\|_1$ generates a sparse solution called L1-regularized SVM. Note that (20) is related to L2-regularized L1-loss support vector classification. Finally, we can solve for both the primal and dual forms of the L2 regularization loss. A regularization term in the form $\lambda \|\boldsymbol{w}\|_1$ or $\lambda \|\boldsymbol{w}\|_2$ is added to avoid overfitting.

7.6 Result

The NDCG score achieves 0.5214 with L2 regularization without specifically optimizing for it. However, with L2 regularization and optimization using the proposed RankSVMndcg approach, the NDCG score increases to 0.5619. This significant improvement highlights the effectiveness of employing the weighted pair-wise error as the loss function, along with optimization via RankSVMndcg, in learning ranking models

8 Conclusion

We introduced a novel probabilistic cost function that utilizes RankNet, a simple and effective neural network architecture, for training ranking systems with pair-wise examples. This approach is adaptable to various differentiable functions and offers the potential for further improvement through parameter optimization. Our work addresses the limitation of pair-wise ranking algorithms in directly optimizing NDCG. By establishing a connection between NDCGβ (a linear function of ranking position) and the ranking error, we propose RankSVMndcg, an algorithm that optimizes the upper bound of the new pair-wise loss. Experiments on the LETOR4.0 MQ2007 dataset demonstrate that RankSVMndcg outperforms the traditional pair-wise ranking model, RankNet.

References

1. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning* (pp. 89–96). Association for Computing Machinery. doi: 10.1145/1102351.1102363

2. Jin, X.-B., Geng, G.-G., Xie, G.-S., Huang, K. (2018). Approximately optimizing NDCG using pair-wise loss. *Information Sciences*, 455, 170-183. https://doi.org/10.1016/j.ins.2018.04.033

3. Watson, S. (2019, July). Session: AI/ML for Systems and User Analysis. Presented at DevConf.cz 2019, Brno, Czech Republic. Retrieved from `https://devconfcz2019.sched.com/event/JckY`

4. Microsoft Research. (n.d.). LETOR 4.0. Retrieved from `https://www.microsoft.com/en-us/research/project/letor-learning-rank-informationretrieval/letor-4-0/`