Application of Neural Networks in Ranking

Wanshan Li
Department of Statistics and Data Sciences
wanshanl@andrew.cmu.edu

Weichen Wu
Department of Statistics and Data Sciences
wwu3@andrew.cmu.edu

Abstract

Our project intends to explore the application of neural networks in ranking problems for feature-rich data. We examine the performance of ranking models based on neural networks and some commonly-used traditional methods on a web-query data set and a face-age data set.

1 Introduction

Ranking is a classical problem in statistics and machine learning [2, 17]. Traditionally, most statistical works in this field have been focused on ranking a fixed set objects based on feature-free pairwise comparisons among themselves. However, in most real-world applications, people are usually faced with feature-rich data and problems that require ranking of new objects that do not exist in the training set. Such tasks are usually referred to as *learning to rank*. In recent years, some new ranking methods for such feature-rich data have been proposed, like [3], [4], [5], [8], [10], [19].

An enormous number of research works and industrial projects have demonstrated the success of neural networks in the study of ultra-high-dimensional structured data like texts, audio, images, and videos. Such types of data also occur in some ranking problems. Therefore, our goal in this project is to investigate the application of neural networks in such rich-feature ranking problems, and try to propose new structures that perform better than existing models.

2 Background and Literature

RankNet [3] is the first model that incorporate neural networks into a probabilistic ranking framework. The basic idea of RankNet is to endow each object with a score s, and set the probability that one objects beats another to be an incremental function of the difference between their scores. The model then uses a neural network to generate the scores from the features of the objects.

Following their idea, different structures of neural networks have been developed for ranking various types of objects like query results, documents, and images. These models includes, but are not limited to [1], [6], [11], [9], [12], [13], [15], [18], [20], [21]. Recently, [9] proposed a new deep learning structure for ranking, and showed the advantage of their method by both theoretical analysis and numerical results. [13], [18], [20] apply the idea of RankNet to image data by incorporating Convolutional Neural Networks (CNNs) into the network structure.

Although numerous models have been proposed using neural networks for ranking, we have noticed some limitations of previous works. For example, in tasks like query ranking, the labels in the data sets are not the results of pairwise comparisons. Instead, each document is given a relevance score to the corresponding query. For example, in the LETOR data set[14], these scores are integers ranging from 0 to 4. In practice, we generate pairs of documents and let the ones with higher scores "win" the comparisons. However, it has not been shown empirically that such procedure is necessary in yielding better model performance than simply forming the problem as a regression (or even classification, if the relevance scores are set to binary) task.

Preprint. Under review.

Therefore, in this project, we hope to compare the performance of the ranking models with regression (or classification) models, as well as compare the performance of neural networks with that of "traditional" models.

3 Models and Methods

Ranking problems are usually studied in a pairwise comparison framework. Let i_1 and i_2 be two items being compared, we use $i_1 \succ i_2$ to denote the result that i_1 beats i_2 in the comparison. Furthermore, let x_{i_1} and x_{i_2} represent the features of i_1 and i_2 respectively, and y_i represent the outcome of the comparison between i_1 and i_2. Let $y_i = 1$ if $i_1 \succ i_2$ and $y_i = -1$ if $i_1 \prec i_2$. The goal of ranking algorithms is to rank items by their ability of beating others in pairwise comparisons.

However, in a number of ranking tasks, the data are not organized as pairwise comparisons. For example, in query ranking, for each query q, we retrieve a number of documents $d_1(q), \cdots, d_{n(q)}(q)$ and their scores or relevance $r_1(q), \cdots, r_{n(q)}(q)$. The score or relevance label measures the quality of the item for the query. Therefore, when training ranking models on query data, the standard way is to traverse i, j from 1 to $n(q)$ for every query q and get records for pairwise comparisons $(x_i, x_j, 1)$ if $i \succ j$ (i.e.$r_i > r_j$) and $(x_i, x_j, -1)$ otherwise.

Ordinal model is the most widely used type of model in pairwise comparison data analysis. An ordinal model assumes that the probability that i_1 beats i_2 is a function of the difference between their competitive powers:

$$\mathbb{P}(Y_i = y_i | x_{i_1}, x_{i_2}; w) = F(y_i[s(x_{i_1}; w) - s(x_{i_2}; w)]) \tag{1}$$

where $s(\cdot; w)$ is an embedding function with parameter w that maps the feature of an item to its competitive power ([16]). It is usually assumed that the function F satisfies the following properties ([9]):

- **Reflexivity**: that $P(Y_1 = 1|x, x) = P(Y_1 = -1|x, x) = F(0) = 0.5$.
- **Anti-symmetry**: that $P(Y = 1|x_1, x_2) + P(Y = 1|x_2, x_1) = F(s(x_1; w) - s(x_2; w)) + F(s(x_2; w) - s(x_1; w)) = 1$. This is frequently guaranteed when $F(x) + F(-x) = 1$.
- **Transitivity**: that if $P(Y = 1|x_1, x_2) > 0.5$ and $P(Y = 1|x_2, x_3) > 0.5$, we have $P(Y = 1|x_1, x_3) > 0.5$.

It should be noticed that these three properties significantly restricts the kind of neural networks that can be applied to construct the function F. A typical Multi-Layer Perceptron (MLP), even when using appropriate activation functions, will not satisfy transitivity.

To learn parameters, people usually minimize the cross-entropy loss

$$\min_w \sum_{i=1}^{n} \left\{ \frac{1 + y_i}{2} \log F(s(x_{i,1}) - s(x_{i,2})) + \frac{1 - y_i}{2} \log[1 - F(s(x_{i,1}) - s(x_{i,2}))] \right\}. \tag{2}$$

Because of the symmetry of the ordinal model, sample $(x_i, x_j, 1)$ and $(x_j, x_i, -1)$ contains identical information and it suffices to fit the model on all transformed records $\{(x_i, x_j, 1)\}_{i \succ j}$.

3.1 Ranking Models based on neural networks

RankNet Let $N(\cdot) : \mathbb{R}^p \to \mathbb{R}$ be a neural network that maps the features of an object to its score, and $F(t) = \frac{1}{1+e^{-t}}$ be the sigmoid function. Thus RankNet is just an ordinal model that uses a neural network as its embedding function, i.e., $s(\cdot; w) = N(\cdot)$ in Eq. 1.

DirectRanker DirectRanker [9] generalizes the idea of RankNet in that it use another neural network to replace function $F(t)$. Specifically, it considers two neural networks: $N_1(\cdot) : \mathbb{R}^p \to \mathbb{R}^{p_1}$, and $N_2(\cdot) : \mathbb{R}^{p_1} \to \mathbb{R}$. The DirectRanker is trained by minimizing the empirical loss

$$\sum_{i=1}^{n} \left[\frac{1 + y_i}{2} \log N_2(N_1(x_{i,1}) - N_1(x_{i,2})) + \frac{1 - y_i}{2} \log[1 - N_2(N_1(x_{i,1}) - N_1(x_{i,2}))] \right].$$

Recall that a well-defined ranking model should have antisymmery and transitivity. It can be easily shown that for a DirectRanker model to have those properties, the second neural network N_2 cannot be more complex than a linear layer followed by a antisymmetric monotone increasing activation function $a(\cdot)$ like `Tanh()`. But a linear layer is additive, so the linear layer in N_2 can be absorbed into N_1. Thus, DirectRanker generalizes RankNet only in the sense that it puts a nonlinear antisymmetric function $a(\cdot)$ on the score difference $s(x_{i,1}) - s(x_{i,2})$. Such similarity is confirmed in numerical experiments.

3.2 Other methods for comparison

In order to evaluate the performance of neural networks in ranking tasks, we firstly implement LambdaMart [19], a ranking algorithm that applies Adaptive Boosting to pairwise comparison data analysis. Furthermore, we also compare the pairwise comparison models with classification models that directly predict the binary relevance scores from the features of the retrieved documents.

4 Experiments on Real Data

We examine the performance of methods in Section 3 on a web-query data set and a face-age data set. Numerical results are shown, compared, and briefly discussed. We make further discussion and conclusion in Section 5.

4.1 Web query ranking

We explore the Microsoft Learning To Rank dataset (MSLR-WEB10k) [14], which contains 10,000 queries from Bing, the Microsoft searching engine. Following the practice of [9], we binarize relevance labels $r_i \in \{0, 1, 2, 3, 4\}$ by making $r_i \leq 1$ as $r_i = 0$ and $r_i \geq 2$ as $r_i = 1$. As has been mentioned in the previous section, we compare the following models:

- Two ranking models based on neural networks: RankNet and DirectRanker;
- LambdaMart, a pairwise comparison model based on Adaptive Boosting;
- Three classification models, Class NN(a neural network with the same structure as the score network N in RankNet), Class LR (logistic regression), Class RF (random forest) and Class Boost (boosting tree).

We use the Normalized Discounted Cumulative Gain (NDCG) to evaluate the performance of these models, see the appendix for the definition of NDCG. To put it briefly, NDCG@k is a number in $[0, 1]$ that measures how well an algorithm recovers the top-k items: 1 corresponds to perfect recovery [9]. In our experiments, we use $k = 10$.

Table 1 shows that classification-based methods perform pretty well. Our implementations of DirectRanker and LambdaMart do not perform as well as those in [9]. Our RankNet, DirectRanker (with an extra `Tanh` before the last sigmoid function, compared with RankNet), and Class NN contain 2 hidden layers with 100 and 30 units each, and we use SGD to optimize (Adam does not fit well). We set learning rate to be 0.001 and batch size to be 100.

Rank-Model	RankNet	DirectRanker	LambdaMart	Rank LR
NDCG@10	0.1923	0.1745	0.2215	0.1718
Class-Model	Class NN	Class LR	Class RF	Class Boost
NDCG@10	0.1465	0.3595	0.3733	0.4543

Table 1: Test NDCG@10 of different methods on MSLR-WEB10k.

To check the details of the training procedure, we draw the path of loss function and NDCG@10 at each epoch in Figure 1.

4.2 Age comparison by face

We are interested in a dataset of human faces labeled with true ages [7]. The dataset contains around 13k images with size around 300×300 of people's faces (mostly Asian) aging from 2 to 80. Figure 2

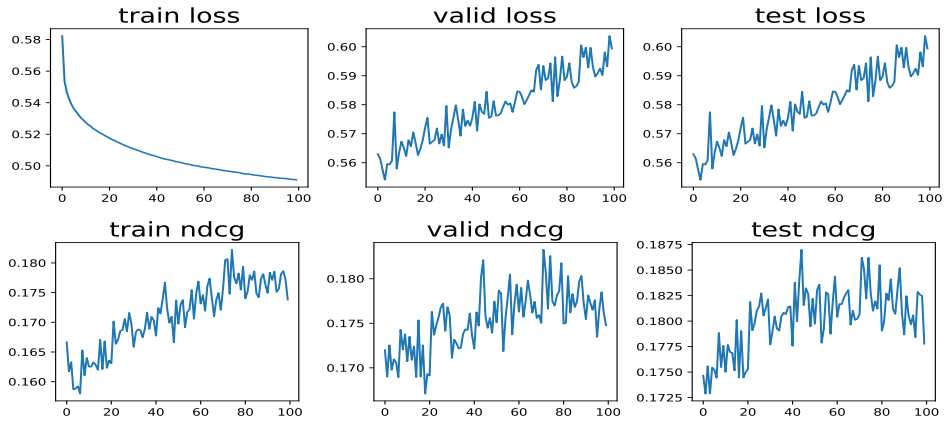

Figure 1: Top: path of (cross-entropy) loss in the training phase on MS10K. Bottom: path of NDCG@10 in the training phase on MS10K. X-axis: number of epochs.

shows 3 pairs of example photos in the data set. A natural ranking problem is to predict which person is older based on the photos of their faces.

The dataset itself only contains faces with true ages. To make it suitable for age comparison, we first randomly split faces into train, validation, and test sets. For each set of faces, we randomly select pairs of faces and make a label for each pair to indicate which person is older. To avoid too many ambiguous pairs, we only keep pairs with age difference bigger than 5 years.

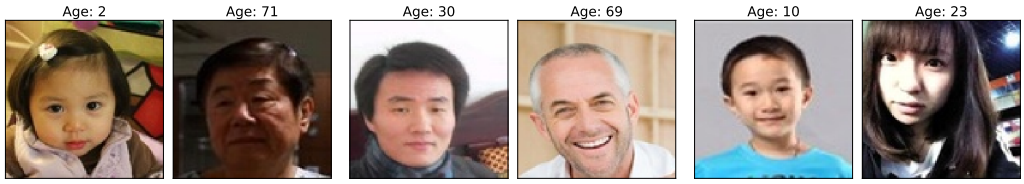

Figure 2: 3 pairs of faces in the face-age data.

In experiment, we use 3 hidden layers of units (1000,500,100) for RankNet and DirectRanker, and SGD to optimize. For RankCNN, we use 2 convolutional layers of shape $(3, 6, 17)$ and $(6, 16, 17)$ (17 is the size of filters), each followed by a `ReLU()` layer and a maxpooling filer $(2, 2)$; 2 hidden fully connected layers of size $(300, 60)$, each followed by a `ReLU()` layer. We set learning rate to be 0.0003 and batch size to be 100.

It should be pointed out that the exceptionally good performance of ranking logistic regression is due to the choice of metric. Recall that for ranking methods, we have $y_i = 1, \forall i$. Now, as we measure the performance by pair-wise accuracy $\frac{1}{n}\sum_{i=1}^{n} 1\{\hat{p}_i > 0.5\}$, the loss for model fitting, cross-entropy $\frac{1}{n}\sum_{i=1}^{n} \log \hat{p}_i$ is a good approximation, and furthermore, maximizing the cross-entropy will lead to an accuracy that's bigger than 0.5. On the other hand, when treating the problem as classification, we will produce many contradicting labels, for instance, a person of age 40 will have both labels 0 and 1 in different comparisons, which can confuses the classifier. We will discuss more of it in Section 5.

Regression-Model	Reg. NN	Reg. LR-ℓ_1	Reg. RF	Reg. Boost
Accuracy	0.5785	0.5720	0.5900	0.5920
Rank-Model	RankNet	DirectRanker	Rank LR	RankCNN
Accuracy	0.6745	0.6754	0.6359	0.6483
Class-Model	Class CNN	Class LR	Class RF	Class Boost
Accuracy	0.3283	0.3165	0.3451	0.3412

Table 2: Prediction accuracy on the face-age dataset.

Another perspective is to check the learned coefficients of the Rank-logistic regression method. We reshape the learned $\hat{\beta}$ into image shape 100×100 and visualize it in Figure 3. It can be seen that the simple logistic regression captures the shape of faces, while in ranking setting, because of contradiction-free labels, it catches a clearer pattern (i.e., the coefficients corresponding to facial information have larger magnitude), which leads to a better performance (0.6359 vs. 0.3165).

In addition, since all images are photos of faces and all faces are put around the center of photos, some classical methods (like MLP and logistic regression) without convolution structures can also perform pretty well. We believe that if images are less regularized, for instance, faces occur at different locations in different images, then CNN will perform better.

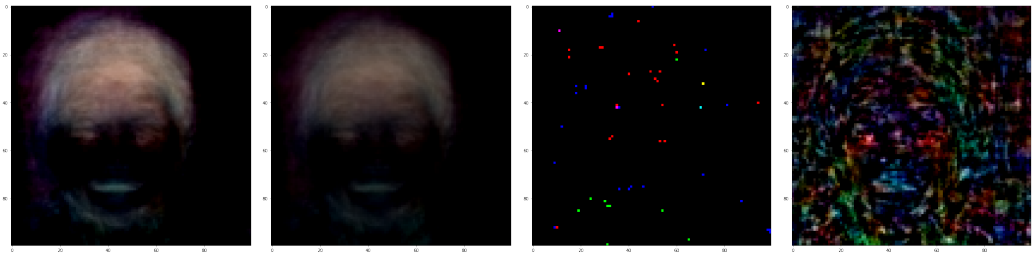

Figure 3: Visualization of the learned coefficients. Left: ranking logistic regression (Rank-LR); mid-left: classification logistic regression (Class-LR); mid-right: ℓ_1-regularized linear regression (Reg.-LR-ℓ_1); right: ℓ_2-regularized linear regression (Reg.-LR-ℓ_2). For better visualization, we multiply the coefficients of logistic regression by 2000.

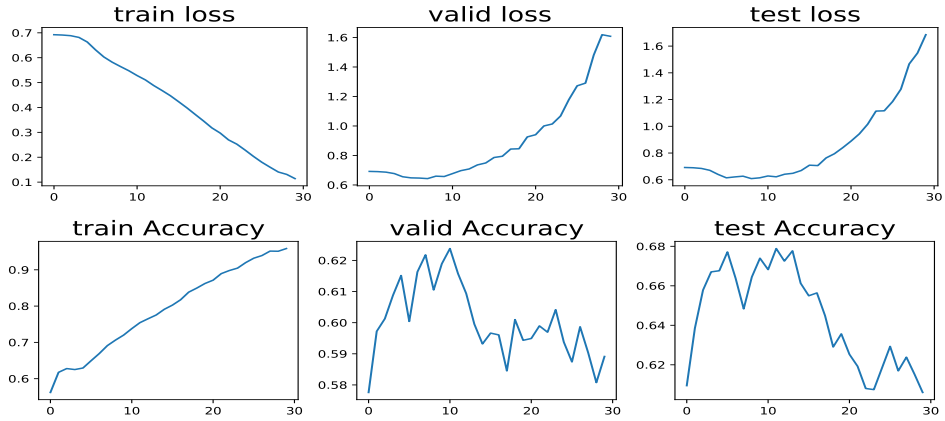

Figure 4: Top: path of (cross-entropy) loss in the training phase of RankCNN on face-age data. Bottom: path of pairwise accuracy in the training phase of RankCNN on face-age data. X-axis: number of epochs.

Incorporate pretrained models We incorporate the pretrained VGG16 as the embedding layer to investigate if the performance of can be improved.

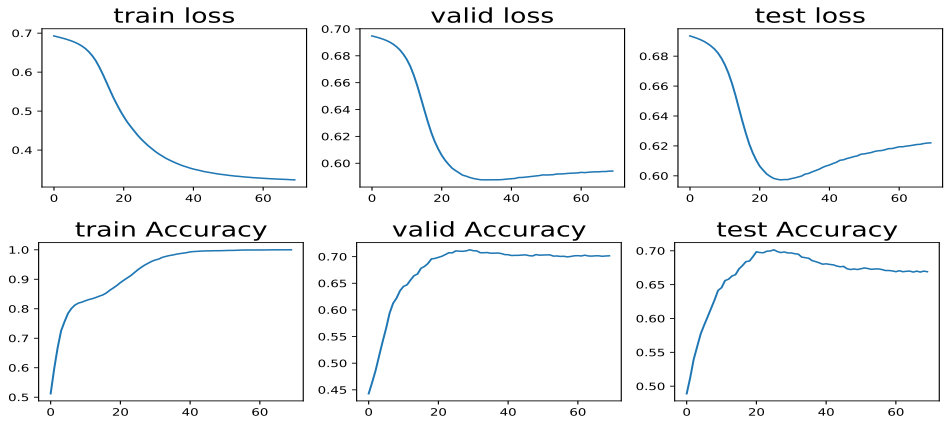

Figure 5: Top: path of (cross-entropy) loss in the training phase of RankNet on face-age data. Bottom: path of pairwise accuracy in the training phase of RankNet on face-age data. The RankNet is fitted on 4096 features from a pretrained VGG.

Regression-Model	Reg. NN	Reg. LR-ℓ_1	Reg. RF	Reg. Boost
Accuracy	0.5945	0.5752	0.5998	0.6040
Rank-Model	RankNet	DirectRanker	Rank LR	RankCNN
Accuracy	0.7006	0.6950	0.6962	-
Class-Model	Class NN	Class LR	Class RF	Class Boost
Accuracy	0.4320	0.3488	0.3538	0.3551

Table 3: Prediction accuracy on the features extracted by the pretrained VGG from the face-age dataset.

To evaluate the predictions of our models, we can randomly select some pairs of faces from correctly predicted pairs and incorrectly predicted pairs, and check if those pairs are easy or hard to compare by our sense of humans' ages. These examples are shown in figures 6 and 7. In these examples, we can see that our model can correctly rank the pairs of photos in which the difference of ages are visually obvious.

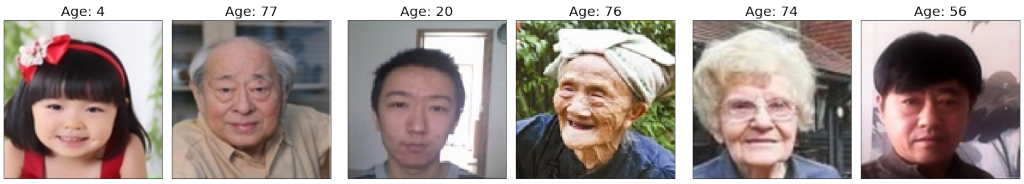

Figure 6: Some pairs that are correctly predicted by Rank LR with VGG-extracted features.

Figure 7: Some pairs that are incorrectly predicted by Rank LR with VGG-extracted features.

5 Discussion and Conclusion

In this section, we will summarize the experimental results and give our understandings of them.

For the query ranking task, we usd **MS10K query dataset**, which contains 10k queries in total, around 50 retrieved documents per query, and 136 extracted features for each query-document pair. We found that

1. Ranking methods perform worse than classification methods;
2. Neural networks perform worse than logistic regression and tree-based methods.

For age ranking task, we used **Face-age dataset**, which contains 2k 300×300 RGB images of human faces with ages ranging from 2 to 80, from which we generated around 12000 evenly selected pairs. In the experiments, we found that

1. Evaluated by ranking accuracy, ranking methods (with binary ordinal labels) perform the best, regression (with true age information) model come second, and classification methods (with binary ordinal labels) perform worst;
2. Incorporating CNN and pre-trained VGG can improve the performance of all methods, though the improvement is not huge.

We will focus our discussion on two perspectives: comparing ranking methods with regression and classification methods, and comparing neural networks with traditional models.

Firstly, we observed in our experiments that ranking models **do not** perform the best in the query ranking task. This is largely due to the fact that in our experiment, the labels in the MS10K dataset are transformed into a nominal variable. As is illustrated in Figure 8, the labels in data sets can be classified into nominal, ordinal, and interval/ratio variables, each with higher level of quantification. It is common practice that we use classification methods for nominal data and regression methods for interval/ratio data. Ranking methods, in the mean time, fits ordinal data the best, when the samples do have an order within themselves, but the difference among them cannot be quantitatively assessed.

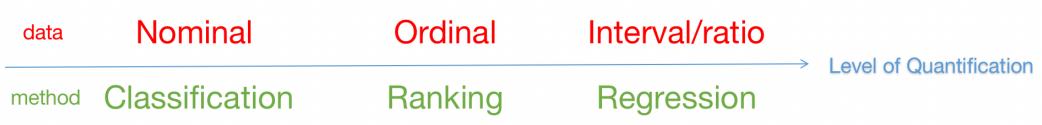

Figure 8: Data types and corresponding methods

For the query ranking task, though relevance score is an ordinal variable, we followed the practice of [9] and transformed the scores into binary labels (scores ≥ 2 were transformed to 1, and otherwise 0). This essentially changes the problem from ranking the relevance of documents to classify the documents into two types: relevant and irrelevant. Furthermore, the criterion for "relevant" seems consistent across different queries, which makes this binary label a fine global metric. Therefore, it comes as no surprise that classification models perform better than ranking models.

As for the age ranking task, although age is an interval variable, comparing different people's ages is apparently easier than predicting the exact age of every person (actually, in our experiments, even the most effective regression model achieves an R^2 of only 28%). In this sense, ranking models are more specific to the task, and therefore get the best performance.

Secondly, we observe that neural networks do not perform better than traditional models in query ranking, give a minor improvement on the face-age data set when using VGG features, and enhance the model performance significantly if we use the original images. For the query-ranking task, this is easy to understand, since the number of features in the MS10K query dataset is relatively small (136), and "traditional" models like random forest and boosting perform sufficiently well. For the age ranking task, the pre-trained VGG network transforms each image into a 4096-dimension feature vector. At this number of features, neural networks begin to show their superiority. When we use the raw images, which contains $3 \times 100 \times 100 = 30000$ features, neural networks performs significantly

better than other models. In summary, our experiment consolidates the notion that neural networks are more powerful and necessary when we deal with high-dimensional data.

6 Work division

Wanshan Li outlined the plan for this project, and implemented the LambdaMart model, classification models, and RankCNN; Weichen Wu implemented RankNet, DirectRanker, Class NN models, and regression models, and build pipelines for incorporating the pretrained VGG model. Both members participated in data preprocessing, numerical experiments, and report writing.

References

[1] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W. Bruce Croft. Unbiased learning to rank with unbiased propensity estimation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 385–394, New York, NY, USA, 2018. Association for Computing Machinery.

[2] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[3] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 89–96, New York, NY, USA, 2005. Association for Computing Machinery.

[4] Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

[5] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking svm to document retrieval. SIGIR '06, page 186–193, New York, NY, USA, 2006. Association for Computing Machinery.

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. Technical Report MSR-TR-2007-40, April 2007.

[7] Jingchun Cheng, Yali Li, Jilong Wang, Le Yu, and Shengjin Wang. Exploiting effective facial patches for robust gender recognition. *Tsinghua Science and Technology*, 24(3):333–345, 2019.

[8] Liangxiao Jiang, Chaoqun Li, and Zhihua Cai. Learning decision tree for ranking. *Knowledge and Information Systems*, 20:123–135, 2009.

[9] Marius Köppel, Alexander Segner, Martin Wagener, Lukas Pensel, Andreas Karwath, and Stefan Kramer. Pairwise learning to rank by neural networks revisited: Reconstruction, theoretical analysis and practical performance. In Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 237–252, Cham, 2020. Springer International Publishing.

[10] Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 897–904, Red Hook, NY, USA, 2007. Curran Associates Inc.

[11] Z. Li, Q. Xu, Y. Jiang, K. Ma, X. Cao, and Q. Huang. Neural collaborative preference learning with pairwise comparisons. *IEEE Transactions on Multimedia*, pages 1–1, 2020.

[12] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. Deeprank: A new deep architecture for relevance ranking in information retrieval. *CIKM*, pages 257–266, 2017.

[13] Zuyao Chen Yangbangyan Jiang Xiaochun Cao Yuan Yao Qianqian Xu, Zhiyong Yang and Qingming Huang. Deep partial rank aggregation for personalized attributes. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages Feb 2–9, 2021.

[14] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *arXiv preprint, arXiv:1306.2597*, 2013.

[15] Leonardo Rigutini, Tiziano Papini, Marco Maggini, and Monica Bianchini. A neural network approach for learning object ranking. In Véra Kůrková, Roman Neruda, and Jan Koutník, editors, *Artificial Neural Networks - ICANN 2008*, pages 899–908, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[16] Nihar B. Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramch, ran, and Martin J. Wainwright. Estimation from pairwise comparisons: Sharp minimax bounds with topology dependence. *Journal of Machine Learning Research*, 17(58):1–47, 2016.

[17] Louis L Thurstone. A law of comparative judgment. *Psychological Review*, 34(4):273, 1927.

[18] Baiyang Wang and Diego Klabjan. An attention-based deep net for learning to rank. *arXiv preprint, arXiv:1702.06106*, 2017.

[19] Qiang Wu, Chris J.C. Burges, Krysta M. Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13:254–270, June 2010.

[20] Q. Xu, Z. Yang, Y. Jiang, X. Cao, Q. Huang, and Y. Yao. Deep robust subjective visual property prediction in crowdsourcing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8985–8993, 2019.

[21] Yuan Yi, Jian Cao, YuDong Tan, QiangQiang Nie, and XiaoXi Lu. adfr: An attention-based deep learning model for flight ranking. In Zhisheng Huang, Wouter Beek, Hua Wang, Rui Zhou, and Yanchun Zhang, editors, *Web Information Systems Engineering – WISE 2020*, pages 548–562, Cham, 2020. Springer International Publishing.

7 Appendix

NDCG There are two widely used metrics for ranking problems, Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP). Both are defined for a single query and averaged over all queries. Suppose a query retrieved n items $d_1, \cdots, d_n$ with relevance label as $r_1, \cdots, r_n$, and denote the ranking produced by the model as $\pi_m(\cdot) : \text{rank} \to \text{index}$, i.e., $\pi_m(k)$ is the index of item that is ranked as the k-th one by the model, and the ranking by the true relevance of items to the query as π_r. The $\text{NDCG@}k$ is defined as

$$\text{NDCG@}k(\pi_m) = \frac{\text{DCG@}k(\pi_m)}{\text{DCG@}k(\pi_r)} = \left(\sum_{i=1}^{k} \frac{2^{r_{\pi_m(i)}} - 1}{\log_2(i+1)}\right) / \left(\sum_{i=1}^{k} \frac{2^{r_{\pi_r(i)}} - 1}{\log_2(i+1)}\right).$$

Notice that $r_{\pi_r(i)}$ is the i-th biggest relevance label of the n retrieved items, so $\text{DCG@}k(\pi_r)$ is the largest value of $\text{DCG@}k$ among all possible ranking results, and hence $\text{NDCG@}k(\pi_m) \leq 1$ for any π_m produced by the model.

MAP Suppose a query retrieved n items $d_1, \cdots, d_n$ with relevance label as $r_1, \cdots, r_n$, and denote the ranking produced by the model as $\pi_m(\cdot) : \text{rank} \to \text{index}$, i.e., $\pi_m(k)$ is the index of item that is ranked as the k-th one by the model, and the ranking by the true relevance of items to the query as π_r. For MAP, we need to first transform r_i's to make them binary, then define

$$\text{MAP}(\pi_m) = \left(\sum_{k=1}^{n} r_{\pi_m(k)} P(k)\right) / \left(\sum_{k=1}^{n} r_{\pi_m(k)}\right), \quad P(k) := \frac{1}{k}\sum_{i=1}^{k} r_{\pi_m(i)}.$$

It can be seen that $\text{MAP}(\pi_m) \leq 1$ and $\text{MAP}(\pi_r) = 1$.

Since both metrics involve ordering items by their relevance, they are not continuously differentiable w.r.t. the relevance labels, and hence they are not good choices for loss functions. Usually, people train their model by a smooth loss function like entropy, and tune hyperparameters by the performance in NDCG on the validation set.