Information Retrieval 1 **Homework 3 (260 points)**
2019-2020 Due Date: 22 March, 2020

In the last 2 assignments, you experimented with retrieval with different ranking functions and document representations. This assignment deals directly with learning to rank. You will learn how to implement methods from the three approaches associated with learning to rank: pointwise, pairwise and listwise.

The submission for this homework is a single zip file. The name of this zip file should be your student ids separated by an underscore. You are required to submit the following:

- Your report, written in LaTeX, in a single PDF file. You can use this template. The report should contain implementation details as well as answers to both theory questions and analysis questions. The page limit for the report is 8 pages (including images, excluding references). Answers to the theory questions should be put into a separate section in the report.
- Your python files and/or notebooks with instructions on how to execute them in a README.md file.
- Your final grade in this assignment will be based both on your implementation and the report (theory questions are not graded).
- This assignment counts for 20% of your grade.

This assignment requires knowledge of PyTorch. If you are unfamiliar with PyTorch, you can go over these series of tutorials. You also need to have a system with minimum of 8 GB RAM and a Linux operating system (or a Linux VM) in your team.

Recommended Reading:

- RankNet [1]
- LambdaRank [2]
- From RankNet to LambdaRank to LambdaMART: An overview (Sections 1, 2, 4) [3]

1 Offline Learning to Rank

A typical setup of learning to rank involves a feature vector constructed using a query-document pair, and a set of relevance judgements.
You are given a set of triples (query, document, relevance grade); where relevance grade is an ordinal variable with 5 grades, for example:{`perfect,excellent,good,fair,bad`}, typically labeled by human annotators. The task then involves learning a function from this training data, and applying it to test data and evaluating the performance.

Theory Questions

TQ1.1 In this assignment you are already given the feature vector x for a given document and query pair. Assuming you have control over the design over this feature vector, how would you design it for the e-commerce domain?

TQ1.2 Briefly explain the main limitations of Offline LTR.

TQ1.3 Contrast the problem of learning to rank with ordinal regression/classification.

TQ1.4 Is it possible to *directly* optimize a given metric? For instance, can we use ERR as a loss function? Explain your answer.

Execute `dataset.py`. This downloads and processes the dataset. It also contains code that you need to use throughout the assignment. In `example.py` you can observe how the data is loaded. Furthermore, this script contains code which uses a random model to make predictions, which are then evaluated. This example also uses code from `evaluate.py` and `ranking.py`, which will be used throughout the assignment.

2 Pointwise LTR (50 Points)

Let $x \in \mathbb{R}^d$ be an input feature vector, containing features (in this assignment) for a query-document pair. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a function that maps this feature vector to a number $f(x)$. The pointwise LTR approach treats the problem as a regression (or classification) problem. This means that the data $\{x\}$ are treated as feature vectors and the relevance judgements are treated as the target which we want to predict.

In this section, you will implement a simple Pointwise model using either a regression or classification loss, and use the train set to train this model to predict (or classify) the relevance score.

Implement a simple pointwise LTR model:

- Use a shallow neural network to learn a Pointwise model using either a regression or a classification loss, using the relevance grades as the label. Use NDCG on the validation set to pick optimal hyperparameters (like the learning rate or number of layers) and for early stopping. Report the optimal hyperparameters.
- Evaluate the model on the test set. Report these numbers.

Theory Questions

TQ2.1 Briefly explain the primary limitation of pointwise LTR with an example.

Analysis Questions

AQ2.1 (10 points) Train a new model with the best performing model's hyperparameters. Every few batches (depending on your available compute power), compute NDCG over the validation set. Plot the loss and NDCG together. Comment on what you observe.

AQ2.2 (10 points) Compute a distribution of the scores (if you're using a classification loss, use the argmax) output by your model on the validation and test sets. Compare this with the distribution of the actual grades. If your distributions don't match, reflect on how you can fix this and if your solution is sufficient for LTR.

Rubric

- (10 points) Training a Pointwise model
- (5 points) Function to evaluate the Pointwise model
- (5 points) Hyperparameter search and early stopping using the validation set
- (5 points) (Report) Optimal hyperparameters for Pointwise model
- (5 points) (Report) Performance on test set

3 Pairwise LTR (110 Points)

In this section, you will learn and implement RankNet [2], a simple pairwise learning to rank algorithm. For a given query, consider two documents D_i and D_j with two different ground truth relevance labels, with feature vectors x_i and x_j respectively. The RankNet model, just like the pointwise model, uses f to predict scores i.e $s_i = f(x_i)$ and $s_j = f(x_j)$, but uses a different loss during training. $D_i \triangleright D_j$ denotes the event that D_i should be ranked higher than D_j. The two outputs s_i and s_j are mapped to a learned probability that $D_i \triangleright D_j$:

$$P_{ij} = \frac{1}{1 + e^{-\sigma(s_i - s_j)}} \tag{1}$$

where σ is a parameter that determins the shape of the sigmoid. The loss of the RankNet model is the cross entropy cost function:

$$C = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij}) \tag{2}$$

As the name suggests, in the pairwise approach to LTR, we optimize a loss l over pairs of documents. Let $S_{ij} \in \{0, \pm 1\}$ be equal to 1 if the relevance of document i is greater than document j; -1 if document j is more relevant than document i; and 0 if they have the same relevance. This gives us $\bar{P}_{ij} = \frac{1}{2}(1 + S_{ij})$ so that $\bar{P}_{ij} = 1$ if $D_i \triangleright D_j$; $\bar{P}_{ij} = 0$ if $D_j \triangleright D_i$; and finally $\bar{P}_{ij} = \frac{1}{2}$ if the relevance is identical. This gives us:

$$C = \frac{1}{2}(1 - S_{ij})\sigma(s_i - s_j) + \log(1 + e^{\sigma(s_i - s_j)}) \tag{3}$$

Now, consider a single query for which n documents have been returned. Let the output scores of the ranker be s_j ; $j = \{1, \ldots, n\}$, the model parameters be $w_k \in \mathbb{R}^W$, and let the set of pairs of document indices used for training be $\mathcal{P}$. Then, the total cost is $C_T = \sum_{i,j \in \mathcal{P}} C(s_i; s_j)$.

In this assignment, we will implement 2 versions of RankNet.

3.a RankNet

- Implement RankNet [1]. This version of RankNet should construct training samples by creating all possible pairs of documents for a given query and optimizing the loss in Equation (3). Again, use a shallow network. Use NDCG on the validation set to pick optimal hyperparameters (like the learning rate or number of layers) and for early stopping. Report the optimal hyperparameters.
- Evaluate the model on the test set. Report NDCG and ERR.

Rubric

- (30 points) RankNet module class with hyperparameters
- (10 points) Function to train RankNet
- (5 points) Function to evaluate RankNet
- (5 points) Hyperparameter search and early stopping using the validation set
- (5 points) (Report) Optimal hyperparameters for RankNet
- (5 points) (Report) Performance on test set

Theory Questions

TQ3.1 Comment on the complexity of training the RankNet (hint: In terms of n, the number of documents for a given query).

3.b RankNet: Sped up

The derivative of the total cost C_T with respect to the model parameters w_k is:

$$\frac{\partial C_T}{\partial w_k} = \sum_{(i,j)\in\mathcal{P}} \frac{\partial C(s_i, s_j)}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \frac{\partial C(s_i, s_j)}{\partial s_j}\frac{\partial s_j}{\partial w_k} \tag{4}$$

We can rewrite this sum by considering the set of indices j , for which $\{i, j\}$ is a valid pair, denoted by $\mathcal{P}_i$, and the set of document indices $\mathcal{D}$:

$$\frac{\partial C_T}{\partial w_k} = \sum_{i\in\mathcal{D}} \frac{\partial s_i}{\partial w_k} \sum_{j\in\mathcal{P}_i} \frac{\partial C(s_i, s_j)}{\partial s_i} \tag{5}$$

This sped of version of the algorithm first computes scores s_i for all the documents. Then for each $i = 1, \ldots, n$, compute:

$$\lambda_{ij} = \frac{\partial C(s_i, s_j)}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j))}}\right) \tag{6}$$

$$\lambda_i = \sum_{j\in\mathcal{P}_i} \frac{\partial C(s_i, s_j)}{\partial s_i} = \sum_{j\in\mathcal{P}_i} \lambda_{ij} \tag{7}$$

Then, to compute $\frac{\partial s_i}{\partial w_k}$, n forward props are performed, followed finally, by the backprops.

Theory Questions

TQ3.2 Comment on the complexity of training the sped-up version of RankNet and contrast it with the first version you implemented. (Hint: Consider the number of pairs of documents for a given query)

- Implement the sped up version of RankNet (Section 2.1 of [3]). This should only change how you compute the loss - everything else should remain more or less the same. You should follow the same setup i.e use NDCG on the validation set to pick optimal hyperparameters (like the learning rate or number of layers) and for early stopping.
- Evaluate the model on the test set. Report NDCG and ERR.

Rubric

- (20 points) Implementation of Sped-up RankNet
- (5 points) (Report) Optimal hyperparameters for Sped-up RankNet
- (5 points) (Report) Performance on test set

Analysis Questions

AQ3.1 (10 points) Contrast the convergence of the sped up version of RankNet with the previous version.

AQ3.2 (10 points) Train a new model with the best performing model's hyperparameters. Every few batches (depending on your available compute power), compute NDCG *and* ARR (Average Relevant Rank) over the validation set. Plot these numbers. Explain what you observe. In particular, comment on which metric is better optimized and what you can conclude from it.

Theory Questions

TQ3.3 Mention the key issues with pairwise approaches. Illustrate it with an example.

4 Listwise LTR with LambdaRank (100 points)

In this section, you will implement LambdaRank [2], a listwise approach to LTR. Consider the computation of λ in Equation (7). λ here amounts to the 'force' on a document given its neighbours in the ranked list. The design of λ in LambdaRank is similar to RankNet, but is scaled by NDCG gain from swapping the two documents in question. Let's suppose that the corresponding ranks of doucment D_i and D_j are r_i and r_j respectively. Given a ranking measure IRM, such as $NDCG$ or ERR, the lambda function in LambdaRank is defined as:

$$\frac{\partial C}{\partial s_i} = \sum_{j \in D} \lambda_{ij} \cdot |\triangle IRM(i,j)| \tag{8}$$

Where $|\triangle IRM(i,j)|$ is the absolute difference in IRM after swapping the rank positions r_i and r_j while leaving everything else unchanged ($|\cdot|$ denotes the absolute value). Note that we do not backpropogate $|\triangle IRM|$, it is treated as a constant that scales the gradients.

Theory Questions

TQ4.1 What does the quantity λ_{ij} represent in LambdaRank?

TQ4.2 You compute a similar λ_{ij} in RankNet, but the approach is still pairwise. In your opinion, does this mean that LambdaRank is still a pairwise approach? Justify your claim.

- Implement LambdaRank [2]. Your hyperparameters should include the retrieval measure being used - you should at least implement NDCG and ERR.
- Again, use a shallow network. Use NDCG on the validation set to pick optimal hyperparameters (like the learning rate or number of layers) and for early stopping. Report the optimal hyperparameters.
- Evaluate the model on the test set. Report NDCG and ERR.

Rubric

- (10 points) Implementation of △ERR (see [4])
- (10 points) Implementation of △NDCG
- (30 points) Implementation of LambdaRank (IR Measure should be a hyperparameter)
- (10 points) Function to train LambdaRank
- (5 points) Function to evaluate LambaRank
- (5 points) Hyperparameter search and early stopping using the validation set
- (5 points) (Report) Optimal hyperparameters for LambdaRank (using both ERR and NDCG)
- (5 points) (Report) Performance on test set

Analysis Questions

AQ4.1 (10 points) Train a new model with the best performing model's hyperparameters. Every few batches (depending on your available compute power), compute NDCG over the validation set. Plot these results. How does this compare to the previous two plots for RankNet and the Pointwise model?

AQ4.2 (10 points) We have in total 3 models: Pointwise, RankNet, and LambdaRank, and 3 corresponding losses. Compute the losses we've discussed so far against each model (select the best performing model). Report these in a table. Reflect on what you observe.

References

[1] 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*, pages 89–96, 2005.

[2] Christopher J Burges, Robert Ragno, and Quoc V Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.

[3] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

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