BALANCE LEARNING TO RANK IN BIG DATA

Guanqun Cao, Iftikhar Ahmad, Honglei Zhang, Weiyi Xie, Moncef Gabbouj

Tampere University of Technology, Finland

{name.surname}@tut.fi

ABSTRACT

We propose a distributed learning to rank method, and demonstrate its effectiveness in web-scale image retrieval. With the increasing amount of data, it is not applicable to train a centralized ranking model for any large scale learning problems. In distributed learning, the discrepancy between the training subsets and the whole when building the models are non-trivial but overlooked in the previous work. In this paper, we firstly include a cost factor to boosting algorithms to balance the individual models toward the whole data. Then, we propose to decompose the original algorithm to multiple layers, and their aggregation forms a superior ranker which can be easily *scaled up* to billions of images. The extensive experiments show the proposed method outperforms the straightforward aggregation of boosting algorithms.

Index Terms— distributed learning, learning to rank, Big Data

1. INTRODUCTION

The emergence of Big Data has brought many new challenges to the current multimedia management systems. One of the key questions, is how to *scale up* the capacity of existing machine learning algorithms to understand and organize the continuously increasing images from public websites such as Flickr. Thanks to the recent advances in High Performance Computing, processing the image data of high volume is now made possible in the distributed environment. Powerful supercomputers require parallel techniques to achieve data-intensive tasks. It is infeasible to learn Big Data in a single computer due to memory limits and prolonged training time. Many endeavors are therefore put into reducing serial dependencies over tasks or data of structural models to achieve parallelization.

Web-scale image retrieval is a typical problem from Big Data, which has been extensively studied and successfully commercialized in recent years. It is similar to the conventional search engine while images are retrieved through queries in text or image format. It remains as a challenging issue due to the "semantic gap" between the low level feature description and high level semantics, and the "intent gap" between users' real demands and their representation in the retrieval system. As a result of the growth of the Web and Big Data, several terabytes of queries and web search interaction data are now available per day on a typical commercial search engine [1]. It enlightens a new direction to bridge the semantic and intent gaps by revealing the intention from users to retrieval results using click logs on a real-world search engine. One solution to the problem is to train a web search ranker based on the query text and feature vectors from clicked images, and user clicks are used as class labels [2].

With the above motivations, we propose a distributed learning to rank method to tackle the issue of web-scale image retrieval. An examplar framework of our proposed system is presented in Figure 1. In the online part, users interact with the system with their query text, and further click on the retrieved images they are interested in. Click logs are recorded and transferred to the offline part together with the images. Image features are extracted by the offline system and coupled with query and click logs to form the whole training data. Big training data is partitioned into several data shards, and learning to rank algorithm is applied over distributed subsets. A ranking model is built by aggregating the numerous models from data shards and returned to the online system. It is used to sort new images according to the query.

In this paper, we contribute to scaling up the learning to rank algorithm over Big Data. Previously, a combination of Bagging and Boosting [3, 4] is shown as a successful ranking model, as it maintains the scalability in Bagging and the accuracy in Boosting. However, the correlation between the partial training data against the whole is non-trivial but overlooked in their work. We present a multi-layer cost-sensitive Bagging Boosting method which balances the difference between the training subsets and the whole data, and provide a way to maximize the performance of distributed learning algorithm. Firstly, with careful selection, we include a cost factor to AdaBoost to balance the individual models toward the whole data. Then, we propose to decompose the original algorithm to multiple layers, and their aggregation forms a superior model for distributed learning. We focus our efforts in data parallelism, namely that the process of training is performed by distributing data across computing nodes, which is demonstrated as an effective way to process Big Data concurrently [5].

The rest of the paper is organized as follows. Section

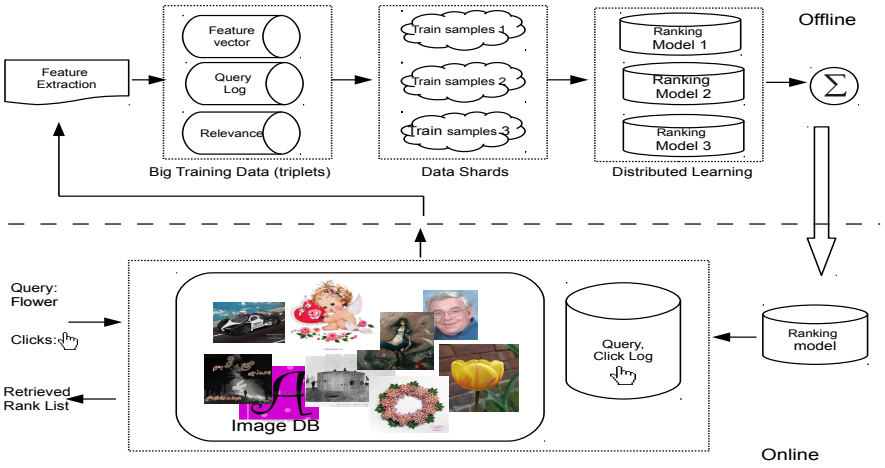

Fig. 1: An illustration of the proposed framework for web scale image retrieval.

2 gives an overview of distributed learning, and learning to rank. In Section 3, we describe the proposed distributed learning to rank method, including multi-layer Bagging Boosting and cost-sensitive boosting. Section 4 presents the performance of proposed method over two ranking dataset: MSR LETOR dataset [6] and MSR-Bing challenge dataset [1], and Section 5 concludes the paper.

2. RELATED WORK

2.1. Distributed machine learning

In general, there are two ways in distributed learning, which includes data sampling and software parallelism [7]. The major problem of distributed learning is the probability distribution of the individual training data is different from that of the whole data. Sampling becomes an effective method, which aims to construct a balanced distribution similar to the whole data by certain mechanisms. On the other hand, cost-sensitive learning provides an alternative way of using costs from different cost matrices for misclassifying any data samples. A cost matrix is considered as a numerical representation to penalize the misclassified samples between classes [8]. The manipulation of values in cost matrices allows to balance the distribution from a particular dataset to Big Data.

While there are algorithms where the output is equivalent to the distributed ones by using the MapReduce model [9], many distributed algorithms build models differing from the ones when training on the whole data. A distributed AdaBoost by Fan et al. [10] is an example, where a classifier is built on selected training data either by randomly sampling from the training set or from a disjoint partition from the whole set. The weights of all training samples are updated after each iteration of boosting through communication across distributed subsets based on a global weight vector. Dean et al. [5] developed a distributed network for Deep Learning, which has also received substantial attention recently.

2.2. Learning to rank

The general idea behind ranking models is to produce a retrieved rank list which sorts new documents or images according to their degrees of relevance to the query. In comparison to hand-crafted rankers, learning to rank aims to train such a single model across query-document pairs, and most algorithms learn the optimal combination of feature vectors from query-document pairs through discriminative learning [2]. The feature vectors can be the number of occurrences of the query terms in the documents, the result from the PageRank model, or image features from the relevant images. Discriminative learning contributes to combining the feature vectors to produce the relevance score in the output. It maps the input objects to the output space based on the hypothesis, and measures the difference between the prediction and expected output and adjust its weights accordingly. The method is applied to multimedia retrieval problems to re-rank the retrieved images [11].

The algorithms of learning to rank mainly follow a common framework, which consists of learning a model over the training data, measuring the quality of retrieval results by an evaluation metric as the loss function and organizing the training samples in three types of data structure. A typical entry in training samples is a triplet of (Query, Feature Vectors, Relevance). The *Discount Cumulative Gain* (DCG) and its normalized version are widely used to evaluate the quality of a retrieval system. The structure of the training set can be pointwise, pairwise or listwise.

3. PROPOSED DISTRIBUTED LEARNING TO RANK

As aforementioned, we distribute the computation and data across supercomputer nodes to achieve data parallelism of ranking. In Section 3.1, we describe the method that we insert a cost factor to the weighting procedure in boosting algorithms to impose a bias towards data of different importance. Then, rather than giving the straight output ranker from boosting, we introduce a new multi-layer boosting method in Section 3.2 to take more intermediate results and generate multiple models as outputs. We aggregate all results from individual models to build up the final ranker for the system in Figure 1.

3.1. Adding cost to a single-layer boosting

AdaBoost and its variant AdaRank are powerful algorithms for learning to rank in multimedia information retrieval [12, 13]. The algorithm provides a weighting procedure where the weights of misclassified samples are strengthened on each iteration, while the correctly ranked samples become weaker for the next base ranker. In this way, the learner will be enhanced by training on misclassification. The weighting procedure can be incorporated with cost factors in dataspace for imbalanced learning [8]. We add such a cost factor C_i in the exponential term of the data weighting scheme, which will interfere with each sample d_i being boosted. It subsequently equalizes the imbalance between data shards by iteratively modifying the boosting factors of samples between the majority and minority classes according to the pre-defined cost metrics. The algorithm of cost-sensitive AdaBoost is shown in Algorithm 1. AdaRank is similar but aims to reduce the measurement error directly. Due to the inclusion of cost, the weight updating parameter α_t on each iteration is determined in Equation 3. We calculate r shown in Equation 1 below, and the derivation of α and r can be found in [8].

$$r = \sum_i D_t(i) y_i h_t(x_i) C_i, \tag{1}$$

where on iteration t, $D_t(i)$ is the distribution, y_t denotes the class labels and $h_t(x_i)$ is the hypothesis made to each sample. The input $S = \{q_i, d_i, y_i\}_{i=1}^m$, where q_i denotes the i^{th} query, $d_i = \{d_{i1}, d_{i2}, ..., d_{i,n(q_i)}\}$ is the retrieved documents or images by q_i, and $y_i = \{y_{i1}, y_{i1}, ..., y_{i,n(q_i)}\}$ is the list of ranks by query q_i.

3.1.1. Selection of cost factor

Though cost-sensitive dataspace weighting is an effective method in imbalanced learning, the cost factor is empirically determined in most cases such as in [8]. To balance the distributed learning models, we formulate a viable approach to determine the cost matrix C. The inclusion of cost takes effect on the importance of learning samples. Namely, the increase of cost on misclassified samples will penalize them heavier in the training process, and the correctly ranked samples will drift further away from future rankers with a stronger C_i. We setup a cost matrix for samples of each class. Moreover, we need to know the class probabilities in the whole training data (p_b) and its data subsets denoted as p_s. In case this is infeasible to compute, we can randomly sample the Big Data several time, and compute the average probability.

Data: $S = \{q_i, d_i, y_i\}_{i=1}^m$,
Initialize: $D_1(i) = 1/m$ for $i = 1, ..., m$.
for *t=1,..,T* **do**
Train the base ranker $h_t \rightarrow \mathcal{Y}$ using distribution D_t;
Aim: select h_t to minimize the weighted error:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) I(y_i \neq h_t(x_i)). \tag{2}$$

Determine weight updating parameter

$$\alpha_t = \frac{1}{2} \ln(\frac{1+r}{1-r}), \tag{3}$$

where r is defined in Section 3.1;
Update and normalize sample weights

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t h_t(x_i) y_i C_i)}{Z_t}, \tag{4}$$

where Z_t is a normalization factor, and *C* is the cost matrix;
end
Output: the final ranker

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)). \tag{5}$$

Algorithm 1: Cost-sensitive AdaBoost algorithm.

We give the cost matrix for binary classifications below, and it can easily be extended to multi-class ranking problems.

$$C = \begin{array}{c} \\ \text{Predict Negative} \\ \text{Predict Positive} \end{array} \begin{array}{c} \begin{array}{cc} \text{Actual Negative} & \text{Actual Positive} \end{array} \\ \begin{pmatrix} 1 & \frac{p_s(positive)}{p_b(positive)} \\ \frac{p_s(negative)}{p_b(negative)} & 1 \end{pmatrix} \end{array} \tag{6}$$

Suppose we have a binary class as positives and negatives, then $p_b(positive)$ denotes the probability of occurrences of positive samples in the whole training data, while $p_s(negative)$ measures the frequency of negatives in one data shard. The rows indicate the predicted class labels, and the columns are the true class labels. Therefore, on each boosting iteration, the misclassified samples will be further boosted if less represented since the probability ratio from its opposite class is high. On the other hand, the prevailing positive samples for example will be suppressed with less probability in the occurrence of negative samples.

3.2. Multi-layer Bagging Boosting models

As shown in Algorithm 1, the idea of (cost-sensitive) AdaBoost is to train a strong model by gradually reducing the

misclassification errors. While the outcome is an exceptionally strong ranker to the local data, there exists a risk of overfitting the subsets resulting in a weak generalization over the entire set. We propose to take into consideration of more intermediate results as learners, so that it not only avoids overfitting but strengthens the stability of the entire model through the bagging step thereafter. We describe a multi-layer boosting model over one data shard in Figure 2. Once the algorithm continues in training base ranker, several intermediate models are taken out and combined. The output from the original model is also included in the proposed algorithm. Numerous intermediate models are trained over data subsets, which form multiple layers of learners. To this end, we construct a new model by merging the numerous learners from the same boosting algorithm after a single training session.

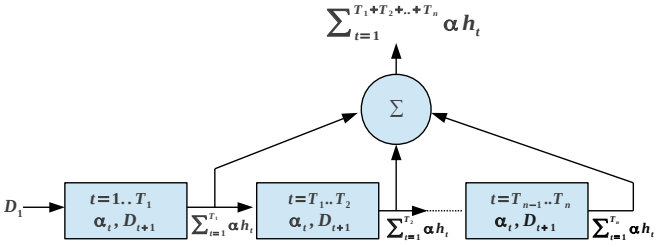

Fig. 2: A block diagram representation of the multi-layer AdaBoost algorithm.

Moreover, we present a pseudo code to show the bagging boosting procedure specifically over the communications between distributed learners in the following algorithm.

Algorithm 2: Generate the ranking model by distributed learning.

1. Distribute the training data and replicas of boosting models to different supercomputer nodes.

2. Wait until data shards are trained by the cost-sensitive boosting algorithm, then recall the models, and build multi-layer rankers.

3. Collect the distributed rankers and make aggregated predictions on new datasets.

4. EXPERIMENTS

In this section, we evaluate the proposed method over two sets of benchmark datasets, which are Micrsoft LETOR4 [6], and a new clickthrough data named MSR-Bing web-scale image retrieval dataset [1].

4.1. Evaluation metrics

In the following experiments, we use Normalized Discounted cumulative gain (NDCG) as an evaluation measure. For a given query q_i, the rank list y_{ij} with respect to its relevance to the query, and a permutation π_i on document d_i, then $\mathrm{NDCG_p}$ is defined as

$$\mathrm{NDCG_p} = \frac{\mathrm{DCG_p}}{\mathrm{IDCG_p}}, \text{where } \mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{y_{ij}} - 1}{\log_2(\pi_i + 1)}, \quad (7)$$

$\mathrm{IDCG_p}$ is the result from the ideal rank list. We may have a fixed length of ranking list, which is denoted as $\mathrm{NDCG@}\mathcal{N}$, where $\mathcal{N}$ denotes the number of items retrieved .

4.2. Experiment setup

We adopt RankBoost (AdaBoost ranking) [14] and AdaRank [13] as two iterative ranking algorithms to evaluate the proposed method. A decision stump performs as the weak learner, and the data is organized listwise in all cases. The algorithms not only are exclusively trained to predict the ranking lists, but they are extended to our proposed method to demonstrate their improvements. The number of training iterations is determined dynamically when the ranker performance starts to decrease over the training set. We use $\mathrm{NDCG@5}$ as the measure in training.

4.3. Experiments on MQ2007

LETOR 4.0 collects two query sets from TREC2007, and the Gov2 web page collection for retrieval. It is a widely used dataset to test ranking algorithms. The data is divided into 5 folds. In distributed learning, we take the first 3 folds to train rankers separately, and aggregate them to produce the ranking results, which is denoted as "MLC_" or "Bag_" in Figure 3. We also train the first 3 data folds together, the other fold is used for testing, shown in "whole". In MQ2007, there are around 1700 queries, and roughly 70,000 query-URL pairs.

The results are shown in Figure 3. The left two bars in every evaluation indicate results by training over the entire 3 folds. The middle four are results from the proposed algorithms with either 1 iteration or 5 iterations as an interval between layers. For example, "MLC_AdaBoost_1" indicates we generate ranking results by the proposed multi-layer cost-sensitive AdaBoost (RankBoost) algorithm, and we take every iteration to build the layers. The last two are generated by bagging the models from straightforward training over each data fold. It shows we receive a performance gain using the proposed "MLC_" algorithm when comparing with the unbalanced bagging results ("Bag_"), especially the cost-sensitive AdaRank with 5 iterations per layer wins the rest corresponding algorithms in all categories. It also indicates training over the whole data does not give superior results, while the distributed rankers can mine the discriminative patterns over the subsets.

4.4. Experiment on MSR-Bing Web-Scale Image Retrieval

MSR-Bing dataset is a newly released large-scale real-world image dataset with click logs. It aims to bridge the semantic

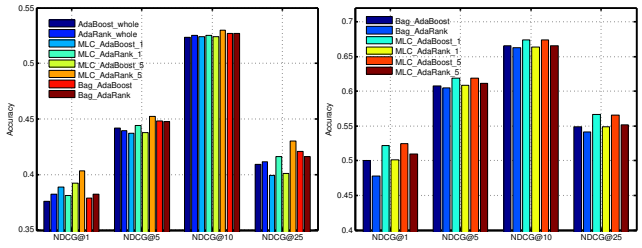

Fig. 3: Ranking accuracy on MQ2007 and MSR-Bing retrieval datasets.

and intent gap in web-scale image retrieval using clicks from Bing search engine. Its current release has 1 million images and 11.7 million unique image-query pairs. We formulate it as the same ranking problem. Distributed rankers are trained over query-image pairs and their relevance, and finally their aggregator is used to rank the entire test dataset. In this experiment, we adopt 280,000 image-query pairs for training, while every 10,000 pairs are separately trained as a distributed data shard, and we use 1,000 most popular queries for testing. This training procedure can easily be *scaled up* to more data using more computations. Similar to the conventional ranking procedure, we need to construct our own training data of triplets of (Query, Feature Vectors, Relevance), given the query-image pairs and number of clicks. We extract three visual features shown in Table 1 to represent each image. And the relevance y_{ij} in each query-image pair is empirically determined according to the click counts n in Equation 8. In Figure 3, the left two bars of each evaluation are the results by the direct aggregation of boosting without cost-balancing, while on the right, it shows the proposed algorithm significantly improves the accuracy in the distributed environment. RankBoost is the most competitive ranker in all cases.

Name	Type	Feature length
Color Structure Descriptor	Color	32
Local Binary Patterns	Texture	57
Canny Edge	Shape	61

Table 1: Image features and their length.

$$y_{ij} = \begin{cases} 3 & \text{if } n > 20, \\ 2 & \text{else if } 10 < n \leq 20, \\ 1 & \text{else if } 0 < n \leq 10, \\ 0 & \text{Otherwise.} \end{cases} \quad (8)$$

5. CONCLUSION

In this paper, we aim to *scale up* the learning to rank method by balancing the distributed ranking models toward Big Data, and subsequently improve its performance. We made two major contributions: we introduce a cost factor to boosting algorithm and give an explicit definition to the cost matrix. Then, we propose a multi-layer boosting algorithms, and summarize the learning procedure for distributed environment. The experiments show the proposed method outperforms the straightforward ranking aggregation when ignoring the data imbalance. The method is also highly parallel and can be *extensible* to billions of images. In the future, we plan to investigate the effect of learning different training subsets, and their weighted combinations in ranking problems. Also, more advanced visual features will be taken into consideration to further boost the system performance.

References

[1] X.-S. Hua, L. Yang, J. Wang, J. Wang, M. Ye, K. Wang, Y. Rui, and J. Li. Clickage: Towards bridging semantic and intent gaps via mining click logs of search engines. In *Proceedings of ACM International Conference on Multimedia*, pages 243–252. ACM, 2013.

[2] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[3] D. Y. Pavlov, A. Gorodilov, and C. A. Brunk. BagBoo: a scalable hybrid bagging-the-boosting model. In *Proceedings of ACM International conference on Information and Knowledge Management*, pages 1897–1900. ACM, 2010.

[4] Y. Ganjisaffar, R. Caruana, and C. V. Lopes. Bagging gradient-boosted trees for high precision, low variance ranking models. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 85–94. ACM, 2011.

[5] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng. Large scale distributed deep networks. In *Neural and Information Processing Systems (NIPS)*, 2012.

[6] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, pages 3–10, 2007.

[7] F. Provost and V. Kolluri. A survey of methods for scaling up inductive algorithms. *Data mining and knowledge discovery*, 3(2):131–169, 1999.

[8] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[9] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[10] W. Fan, S. J. Stolfo, and J. Zhang. The application of adaboost for distributed, scalable and on-line learning. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 362–366. ACM, 1999.

[11] Y.-H. Yang, W. H. Hsu, and H. H. Chen. Online reranking via ordinal informative concepts for context fusion in concept detection and video search. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(12):1880–1890, 2009.

[12] M. Merler, R. Yan, and J. R. Smith. Imbalanced RankBoost for efficiently ranking large-scale image/video collections. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*, pages 2607–2614. IEEE, 2009.

[13] J. Xu and H. Li. AdaRank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[14] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.