BagBoo: A Scalable Hybrid Bagging-the-Boosting Model

Dmitry Pavlov
Yandex Labs
299 S. California Ave., 200
Palo Alto, CA, USA
dmitry-pavlov@yandex-team.ru

Alexey Gorodilov
Yandex
Ulitsa Lva Tolstogo, 16
Moscow, Russia
agorodilov@yandex-team.ru

Cliff Brunk
Yandex Labs
299 S. California Ave., 200
Palo Alto, CA, USA
cliff@yandex-team.ru

ABSTRACT

In this paper, we introduce a novel machine learning approach for regression based on the idea of combining bagging and boosting that we call *BagBoo*. Our *BagBoo* model borrows its high accuracy potential from Friedman's gradient boosting [2], and high efficiency and scalability through parallelism from Breiman's bagging [1]. We run empirical evaluations on large scale Web ranking data, and demonstrate that *BagBoo* is not only showing superior relevance than standalone bagging or boosting, but also outperforms most previously published results on these data sets. We also emphasize that *BagBoo* is intrinsically scalable and parallelizable, allowing us to train order of half a million trees on 200 nodes in 2 hours CPU time and beat all of the competitors in the Internet Mathematics relevance competition sponsored by Yandex and be one of the top algorithms in both tracks of Yahoo ICML-2010 challenge. We conclude the paper by stating that while impressive experimental evaluation results are presented here in the context of regression trees, the hybrid *BagBoo* model is applicable to other domains, such as classification, and base training models.

Categories and Subject Descriptors

I.5.1 [**Models**]: Statistical

General Terms

Algorithms, Experimentation

1. INTRODUCTION

The goal of our paper is two-fold: introduce a novel highly scalable machine learning algorithm inspired by bagging and gradient boosting ideas and demonstrate its utility for the learning-to-rank task. In particular, we show how *BagBoo* model blends bagging and boosting in a bagged ensemble of boosted trees, and how from that it gains an advantage over both of them: In particular, we conduct experiments on public large-scale data sets from LETOR [4] as well as data from Yandex and Yahoo-sponsored ranking challenges and demonstrate that the hybrid *BagBoo* model is superior to both pure bagging and boosting in ranking quality according to *DCG* and *MAP* metrics, as well as outperforms in many cases the best published results on these data sets. We further show that *BagBoo* is much faster than boosting, primarily because *BagBoo* typically gets its best performance on short boosted sequencies that are wrapped into bagging. Bagging is intrinsically parallelizable and while boosting is not, having a short boosted sequence to learn on a single processor is still a doable task. This finding underscores the offline time efficiency and scalability.

There has been previous work on combining bagging and boosting, mostly dealing either with different settings, or applications, or not making the simultaneous scalability and model accuracy argument for the learning-to-rank problem. Here we emphasize three most notable contributions in the past literature on the topic we are aware of. Firstly, *BagBoo* can be viewed as an extension of bagging [1], in which a short sequence of boosted trees is used as the basic element of the bagged ensemble. Next, our *BagBoo* model is similar in flavor, to the additive groves of regression trees [5], where the authors grow and back fit sequences of boosted trees, before voting them as a bag. Finally, in the recent KDD Cup'09 competition [7] that dealt with the prediction of customer churn, the combination of bagging and boosting similar to ours has been shown to compete well against other models.

The rest of our paper is organized as follows: in section 2 we give a succinct introduction to the bagging and boosting ideas, and introduce the combined *BagBoo* model and the algorithm for its learning. In section 3, we describe the ranking application and the data used in our experiments, and in section 4 we discuss experimental results that provide empirical verification of *BagBoo*'s advantage over standard bagging, boosting as well as state-of-the-art relevance algorithms. We conclude the paper in section 5.

2. BAGGING AND BOOSTING

The idea of bagging is to create an ensemble of models by sampling the training data without replacement, and voting the resulting models. Breiman [1] calls all methods in which every model in the ensemble is a function of the independent identically distributed random vectors, a random forest. He further outlines that the random forest family has a number of attractive properties: variance reduction, resistance to overfitting and effective parallelizable computation.

Gradient boosting invented by Friedman [2] has a flavor in many respects similar to that of bagging: it creates an en-

Table 1: Characteristics of the large-scale relevance data sets used for experiments and algorithm comparison.

Data Set	Queries	N-Rows	N-Features	N-Labels	Label Distribution
TD2004	75	74,146	64	0/1	1.5% 1's
MQ2007	1,692	69,623	46	0/1/2	20.3% 1's, 5.5% 2's
IMAT09	9,124	97,290	245	multi $[0,4]$	25.9% non-0's
ICML10	19,944	473,174	704	multi $[0,4]$	26% 0's

semble of base learner models and sees significant training speed increase, approximation accuracy improvement and stability to overfitting resulting from the introduction of bagging-like randomness. The main difference, however, is that each subsequent model is trained in a fashion dependent on all previously obtained models. To make things more precise, in boosting, the least squares error between the additive ensemble of N models (of which the first $N-1$ are fixed and the N-th is optimized) and the target value is iteratively optimized. This procedure, while optimizing the concrete target of interest, suffers from being hard to parallelize.

By combining the good properties of both models we move away from simple tree models that are typically dealt with in random forests and inhale the power of gradient boosted trees in them by making every random forest model be a gradient boosted tree. At the same time, we alleviate the parallelization issues attributable to boosted trees by keeping the boosting sequence short, e.g. 10-20 base trees each, which is much lower than thousands of trees typically used in standard gradient boosting. This way, every boosted tree is quite powerful and can be learned fast on a single processor node, and many such boosted trees are later averaged in a bagged, random forest like, paradigm.

To make things more concrete, lets review the algorithm:

```
1. Input: training data $D$; $NBag$ and $NBoo$ iterations of bagging and boosting respectively
2. Output: Random Forest of $NBag$ x $NBoo$ trees
3. for $i = 1$ to $NBag$ do
4.         $D[i] := SampleData(D)$; # samples both data records and features without replacement
5.         $BT[i] := BoostedTree(D[i], NBoo)$; # NBoo iterations of boosting optimization on D[i]
6. endfor
7. Output additive model $\sum_i BT[i]$;
```

In line 1, we list the inputs to the algorithm which include the training data D to run regression on, and the $NBag$ and $NBoo$ parameters that define the number of bagging and boosting iterations respectively. In line 2, we define the output of $BagBoo$ to be a random forest of $NBag$ by $NBoo$ trees. The trees are learned in the for loop that goes in line 3 from 1 to $NBag$, every time sampling the data records and features (line 4) and learning a boosted sequence $BT[i]$ of trees of length $NBoo$ on the resulting data sample in line 5. After the loop is done, we return the sum of all $BT[i]$ trees.

Thus, the only real and substantial change from bagging is that we bag boosted models, it allows the resulting model to be quite a bit more powerful as we will demonstrate in the experimental section 4. The only big change from boosting is that we reduce the number of boosted iterations while wrapping bagging around it which allows for substantial training time savings. We also employ the idea of sampling features in addition to sampling the data records that works quite well for decision trees allowing them to learn various "aspects" of the data.

Interestingly, wrapping the other way around, i.e. creating the ensemble of boosted bagged models, while possibly resulting in some time saving due to still existing possibility of parallelizing inner bagging iterations, will not be as efficient as $BagBoo$ as the outer boosting process still needs to wait for the previous iteration to complete before starting the new one. Further, bagging of the inner classifiers make them stronger which is not something that is desirable for the external boosting process that typically requires weak classifiers.

In the following sections, we provide empirical comparison between the models.

3. EXPERIMENTAL SETUP

The experimental data was chosen with one main goal in mind: to provide for the most comprehensive comparison of $BagBoo$ model to the main available state-of-the-art methods whose performance may be obtained from the widely accepted published results. In pursuing this goal, we chose to use public standardized data sets: TD2004 from LETOR 3.0 and MQ2007 from LETOR 4.0 [4]. A variety of experimental results with different algorithms were reported on these data sets before, which makes them a widely used and acknowledged test-bed for experimental evaluation of the ranking methods. We also tested $BagBoo$ in recent Yandex and Yahoo-sponsored ranking challenges.

The main parameters of the data sets, including the number of queries, rows, features, labels, and the label distribution are summarized in table 1. The data sets provide nice coverage in terms of main characteristics of interest: the number of queries varies from below 100 in TD2004 to almost $10,000$ in IMAT09, the number of rows in all cases is close to $100,000$, the number of features varies from below 50 in MQ2007 to almost 250 in IMAT09, the label cardinality is binary for TD2004, tertiary in MQ2007 and covering pretty much the continuous range from 0 to 4 for IMAT09, while, finally, the label distribution ranges from being severely skewed towards irrelevant in TD2004 to having over a quarter of relevant judgement in the training data of IMAT09. The biggest advantage of using MQ2007 and TD2004 is represented by having pre-folded data for cross-validation and tools for accuracy assessment, which makes the comparison to other algorithms more reliable [4]. IMAT09 and ICML10 datasets were released by Yandex and

Table 2: Cross-validated results for TD2004 data set in LETOR3.0 collection. Numbers in bold represent the winning method for the metric in a given column. *BagBoo* wins in all metrics. BagBoo fits 1.1 mil. trees, which pure boosting can only afford in 48 days on a single CPU, hence its performance is not quoted.

Method	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5	MAP
BagBoo	**50.67**	**44.00**	**40.80**	**39.86**	**38.98**	**24.99**
Bagging	45.33	44.00	38.88	37.15	34.48	21.73
Boosting	-	-	-	-	-	-
BoltzRank	47.67	41.33	39.02	37.57	36.35	23.90
ListNet	36.00	34.67	35.73	34.69	33.25	22.31
FRank	49.33	40.67	38.75	35.81	36.29	23.88
AdaRank.NDCG	42.67	38.00	36.88	35.24	35.14	19.36
AdaRank.MAP	41.33	39.33	37.57	36.83	36.02	21.89
RankSVM	41.33	34.67	34.67	34.10	32.40	22.37

Yahoo respectively and are available for free download at the time of print.

For model comparison we used *DCG*, *NDCG* and *MAP* metrics that are described in detail in [4] and are omitted here for brevity.

4. EXPERIMENTS

Before embarking on the large scale experiments, we ran an experiment on a small data set called *Concrete* from the UCI repository in order to support our hypothesis that for fixed total number of trees $T = NBag \cdot NBoo$, the best tradeoff between the offline model training time and the accuracy may be achieved when neither of $NBag$ or $NBoo$ equals 1, in other words, doing *BagBoo* with a non-trivial boosting sequence makes sense.

The data set at hand has 1030 records over 7 numeric features, and the regression task for it is to predict the compression strength of the concrete in megapascals. The target values fall in the range $[2.33, 82.6]$ with a mean 35.82 and standard deviation 16.71. For a naive baseline, using the mean as a predictive model results in mean squared error (MSE) of 278.89, while using a single regression tree in 10 fold cross-validation yields an MSE of 39.79.

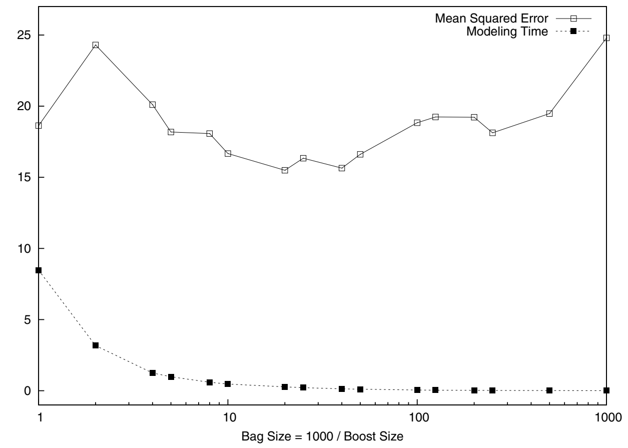

Figure 1: Model accuracy (mean squared error) and offline/modeling time as a function of $NBag$ and $NBoo$ for fixed $T = NBag \cdot NBoo$

We fixed $T = 1000$, and varied $NBoo$ on a logarithmic scale from 1 to 1000, derived $NBag = T/NBoo$ and measured both mean-squared error obtained by each of these models in 10 fold cross-validation as well as the time it took us to obtain the model. All experiments were simulated on a single machine, and appropriately normalized, i.e. we assumed that all bagging runs could be parallelized and would take roughly same time each, and we did not account for network latency and other factors that may have affected the performance should the data be distributed.

The results of this study are presented in Figure 1. The x-axis on this figure is $NBag$, implying the value $NBoo = T/NBag$. Thus, the origin on the x-axis corresponds to pure boosting, while $NBag = 1000$ corresponds to pure bagging, with all x values in between corresponding to some version of *BagBoo*. The y-axis on the figure measures both offline modeling time in seconds and the difference in mean-squared error (MSE).

Note, that boosting takes the longest time, close to 8 seconds on average to obtain the model, while bagging is the fastest (around 1 second), but, to trade this off, there is a significant difference in MSE – boosting performs much better than bagging. The best MSE we see is around 16 which is much better than naive predictions quoted above. *BagBoo* comes in as winner with the modeling time negligibly higher than that of bagging for pretty much any bag size greater than 20, and the best overall MSE performance achieved for $NBag = 50$ and $NBoo = 20$, beating the plain boosting. The results of this experiment lead us to conclude that the model performance close to that of boosting or better can be achieved by the *BagBoo* approach times faster than by plain boosting, and almost as fast as bagging. This effect is largely due to the parallelization abilities inherited from bagging.

Now we turn to the large scale experiments on learning-to-rank data sets and show that *BagBoo* is one of the top performing models among the modern ranking algorithms and across a variety of data sets.

On TD2004 and MQ2007 data sets, we ran the model training for each of the 5 folds on the training partition, with a quality control performed on the validation partition and results assessed on the test partition of a given fold. During the model training and quality control steps, we adjusted various parameters of *BagBoo*. In particular, the maximum number of leaves in each individual base tree model was 8, the feature sampling rate was 75% uniform without replacement, the shrinkage factor [2] was equal

Table 3: **Cross-validated results for MQ2007 data set in LETOR4.0 collection. Numbers in bold found represent the winning method for the metric in a given column.** *BagBoo* **wins in all metrics but** *NDCG*1 **where it is in a close 3-rd place after RankBoost and RankSVM. BagBoo fits 1.1 mil. trees, which pure boosting can only afford in 48 days on a single CPU, hence its performance is not quoted.**

Method	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5	MAP
BagBoo	40.71	**41.35**	**41.76**	**41.83**	**42.29**	**46.76**
Bagging	39.24	40.03	39.81	40.28	40.58	46.29
Boosting	-	-	-	-	-	-
RankSVM-Struct	40.96	40.73	40.63	40.84	41.43	46.44
ListNet	40.02	40.63	40.91	41.44	41.70	46.52
AdaRank.NDCG	38.76	39.67	40.44	40.67	41.02	46.02
AdaRank.MAP	38.21	39.00	39.84	40.24	40.70	45.77
RankBoost	**41.34**	40.94	40.72	41.22	41.83	46.62

to 0.1, $NBoo = 250$ and $NBag = 4500$ for the total of $T = NBoo \cdot NBag = 1,125,000$ tree models. The large number of models possible to train in a couple of hours time on 200 nodes is a direct result of *BagBoo*'s high scalability. The quality assessment on the test partition consisted of computing metrics $NDCG@1-5$ as well as MAP. The final numbers presented in tables 2 and 3 represent the averages of these metrics across the folds. Numbers in **bold** font represent the winning method for the metric in a given column.

Tables 2 and 3 show that despite the fact that *BagBoo* is a pointwise method, i.e. it doesn't explicitly optimize the ranking over documents, it outperforms the formidable listwise approaches, such as ListNet [3] and BoltzRank [6]. It is also the best overall model in quality based on the experiments we conducted, where it looses only once to RankBoost and RankSVM on the MQ2007 data set in metric $NDCG@1$. Boosting with complexity equal to that of *BagBoo*, i.e. fitting the same number of trees, could not compete with *BagBoo* in any of the runs as it would take estimated 48 days to get the results for it, which is far out of practicality realm. The ranking performance of pure bagging which is given in the tables was noticeably lower than for any of the winning algorithms.

The *BagBoo* method also won (unofficially as an insider) the Internet Mathematics competition under id *Joker* held by Yandex in 2009. The impressive parallelization capability of *BagBoo* lead us to building 20,000 bags of 20 boosted trees for a whopping total of 400,000 trees in a winning combination. On a 200 node cluster, the whole process took about 2 hours, including the network latency and the map-reduce process to train the models and collect the results.

The *BagBoo* model won the 3 places in both tracks of Yahoo-sponsored ICML-2010 contest, where the reciprocal rank was a final scoring metric, and scored the first and the second with respect to the *nDCG* metric.

5. CONCLUSIONS

In this paper, we presented the method for combining bagging and boosting into a hybrid approach we called *BagBoo*. While similar ideas were discussed in the recent literature, our particular setup and extensive experimental results on the learning-to-rank data sets are novel. The main motivation for this research came from the observation that typically boosting outperforms bagging but suffers from being hard to parallelize. The *BagBoo* model fills this gap nicely: not only that it is comparable to boosting in terms of quality or even sometimes better but it is also straightforward to parallelize given that the main wrapper algorithm is bagging. We use this ability to scale the training to learn nearly half a million trees to dominate in the Internet Mathematics ranking competition. We also use over 1.1 million trees to win in all but one metric on TD2004 and MQ2007 data sets. In both cases, the training took only couple hours on the cluster of 200 machines, while boosting would take orders of magnitude longer provided it were able to fit this many trees in memory at once at all. The experimental results we presented in the paper also confirm that *BagBoo* is one of the top to-date known ranking methods that, being pointwise itself, still outperforms the best known pointwise, pairwise and listwise methods.

It is worth a final note that, while we focus in our experiments on the tree based models, the whole setup is generalizable to non-decision tree base learners.

Overall, *BagBoo* appears to be in an attractive position of being among the top most accurate ranking models, and, at the same time, enjoying the benefits of high scalability which we hope will make it a valuable tool in the IR community.

6. REFERENCES

[1] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[2] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

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

[4] 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 *LR4IR 2007*, 2007.

[5] D. Sorokina, R. Caruana, and M. Riedewald. Additive groves of regression trees. In *ECML'07*, pages 323–334.

[6] M. Volkovs and R. Zemel. BoltzRank: Learning to maximize expected ranking gain. In *ICML'09*, pages 1089–1096.

[7] J. Xie, V. Rojkova, S. Pal, and S. Coggeshall. A Combination of Boosting and Bagging for KDD Cup 2009 - Fast Scoring on a Large Database. In *JMLR*, volume 7, pages 35–43, 2009.