Application of Additive Groves to the Learning to Rank Challenge

Daria Sorokina
daria-sorokina@yandex-team.ru
Yandex Labs, Palo Alto, CA, USA

Editor: Olivier Chapelle, Yi Chang, Tie-Yan Liu

Abstract

This paper describes a submission of team AG to the Yahoo! Learning to Rank Challenge held in 2010. This solution has scored 4th place in the main track. The primary algorithm used is Additive Groves of regression trees.

1. Competition and Data

Yahoo! Labs organized the first Learning to Rank Challenge in spring 2010. The challenge ran from March 1 to May 31 and received 4,736 submissions from 1,055 teams. The competition included two tracks: main track was dedicated to the problem of learning to rank on large data sets itself, and the second track dealt with the transfer learning on smaller data sets. The author participated in the main track, so the rest of the paper will consider the setting of the first problem only.

The training data for the first track of the challenge consisted of 473,134 query-url pairs and 519 features defined for those pairs. The relevance labels took on the values from 0 to 4. As suggested in (Chapelle et al. (2009)), we converted these values to probability of relevance using the formula:

$$R(g) = \frac{2^g}{16}, \tag{1}$$

where g is the original label and $R(g)$ is the new label, probability of relevance.

Competition data also included validation set (71,083 data points) and test set (165,660 data points). Labels for these data sets were not provided.

2. Additive Groves

We refer the reader to the paper where Additive Groves were introduced (Sorokina et al. (2007)) for the detailed description of the algorithm, but here it is in a nutshell:

Additive Groves ensemble consists of bagged additive models, where every element of an additive model is a tree. A single grove is trained similar to an additive model: each tree is trained on the residuals of the sum of the predictions of the other trees. Such training is performed in the backfitting style: trees are discarded and retrained in turn until the overall predictions converge to a stable function. In addition, each grove of a non-trivial size is iteratively built from smaller models: either a tree is added to a grove with fewer trees or the size of trees is increased in a grove with smaller trees; which of the two steps happen

in each particular case is defined by comparing models on out-of-bag data. A single grove consisting of large trees can and will overfit heavily to the training set, therefore bagging is applied on top in order to reduce variance.

In this challenge, we were using the original regression version of the algorithm optimizing least squares loss. Apart from the initial label transfromation, no attempts were made on optimizing for any special ranking metrics, RMSE was used as a performance metric at all times.

3. Experiments

First, the original training data was split into training and validation sets: 75% for training, 25% for validation. The data points corresponding to the same query were always placed into the same set. All parameter tuning was performed on thus created hold-out validation set, the official validation set was not used for training.

Then, an attempt at feature selection was made. We used bagging with multiple counts (Caruana et al. (2006)) to rank the features, as this method proved to be useful as a preprocessing method for Additive Groves in the past (Sorokina (2009)). First, we built several ensembles of 200 bagged trees of different sizes and determined that the ensemble consisting of full trees results in the best performance on the validation set. Then, the features were ranked based on how they were used in that ensemble. In particular, features got scored based on how large are the tree nodes splitted by those features.

Retraining the ensemble using only 200 top features from the resulting list did not change the performance of bagged trees. Using only 100 top features resulted in a seemingly insignificant decrease in the performance. Based on that, it was decided to use 100 top features for parameter tuning stage of training Additive Groves and 200 features for the final model.

On the parameter tuning stage a grid of Additive Groves models was built on the training set and estimated on the hold-out validation set. The grid was built using 181 iterations of bagging. Two main parameters of an Additive Groves model are N — the number of trees in a single grove and α — maximum proportion of the training set in the leaf node (the smaller the value of α, the larger the tree). The grid consisted of models corresponding to all possible pairs of (N, α), where N took values from 1 to 9 and α from the sequence $\{0.5, 0.2, 0.1, 0.05, 0.02, 0.01, ..., 0.0001\}$. The best performance on the validation set was produced by the model with the parameters $N = 9$ and $\alpha = 0.0005$. These values were chosen to train the final model.

The final Additive Groves model was trained using 200 features, whole training data set (no more splitting to train and validation), 216 bagging iterations and values of parameters $N = 9$ and $\alpha = 0.0005$ chosen during tuning. Predictions of this model constitute the final submission of team AG. It resulted in the scores $ERR = 0.466157$ and $NDCG = 0.8018$ on the official test set. The best scores achieved on these metrics by other teams, $ERR = 0.468605$ and $NDCG = 0.806$, demonstrate about a 0.5% improvement over AG scores.

4. Parallelization and Running Time

The primary goal of these experiments was to generate the best possible model given the time and resource constraints, and no attempts were made to produce anything resembling precise academic running time measurements. However, to give the reader some idea about the algorithm capabilities, we decided to share some details of that aspect as well.

The whole process took less than 3 weeks: between the moment the author started working in Yandex Labs (May 10th) and the end of the competition (May 31st). All Additive Groves models were build with the help of parallelization. As Additive Groves algorithm is based on bagging, it is trivially parallelizable: each iteration of bagging can run on a separate cluster node. The iterations were trained this way on 220 nodes in parallel; the actual numbers of bagging iterations in the resulting models (181 for parameter tuning, 216 for the final model) correspond to the number of iterations that have actually completed and not crashed for various reasons.

Our logs show that the training of the first Additive Groves model grid started on May 22nd, and was done in four phases, each of which took less than a day. The final model was trained in one phase on May 27-31. It is worth noting that the training took place on a cluster overloaded with other processes during the final days of competition, and most of the time AG processes were getting only half of the CPU time they needed. Some of the iterations (apparently, those that were lucky to get more CPU time) were finished in less than 2 days, others ran for the whole 4 days. Most of the cluster nodes are equipped with 2.33GHz Intel Xeon CPU's.

5. Potential Improvements

A number of choices were made solely due to the time and resource availability constraints of the author. It is possible that without those constraints the performance of Additive Groves on this data set could be further improved.

- Using less than half of the available features did not hamper the performance of bagged trees, but could make a difference in the performance of a more complex model like Additive Groves.
- No tests were performed on whether values of $N > 9$ result in a better model. The difference between $N = 8$ and $N = 9$ was already very small, so no big improvement is likely, but some improvement is still possible.
- The number of bagging iterations used is likely not large enough to achieve complete convergence. Another 100 iterations of bagging could improve the model.

6. Conclusion

Additive Groves method proved to be a very good fit for the learning to rank challenge data. The best model included many additive components ($N = 9$) as well as high non-linear complexity ($\alpha = 0.0005$, which loosely corresponds to depth of trees equal to 12). This means that the ranking function had both strong additive structure and many com-

plex interactions at the same time. Additive Groves consistently outperforms many other learning methods exactly because it is able to fit both of those types of complexity.

7. TreeExtra package

The algorithms used in this submission — Additive Groves and bagging with multiple counts feature evaluation — are available as parts of TreeExtra package. TreeExtra is a set of command line tools written in C++. You can find binaries, code and manuals at

www.cs.cmu.edu/~daria/TreeExtra.htm

8. Acknowledgements

The author would like to acknowledge Rich Caruana, Mirek Riedewald and Art Munson, who helped to develop algorithms used in this submission, as well as Yandex Labs team, in particular Dmitry Pavlov and Cliff Brunk, for support, encouragement and access to hardware resources.

References

Rich Caruana, Mohamed Elhawary, Art Munson, Mirek Riedewald, Daria Sorokina, Daniel Fink, Wesley M. Hochachka, and Steve Kelling. Mining Citizen Science Data to Predict Prevalence of Wild Bird Species. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining (KDD'06)*, 2006.

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 (CIKM'09)*, 2009.

Daria Sorokina. Application of Additive Groves Ensemble with Multiple Counts Feature Evaluation to KDD Cup'09 Small Data Set. In *Proceedings of the KDD Cup workshop*, 2009.

Daria Sorokina, Rich Caruana, and Mirek Riedewald. Additive Groves of Regression Trees. In *Proceedings of European Conference in Machine Learning (ECML'07)*, 2007.