InfiniteBoost: building infinite ensembles with gradient descent

Alex Rogozhnikov*
Faculty of Mathematics
National Research University
Higher School of Economics
Moscow, Russia
alex.rogozhnikov@yandex.ru

Tatiana Likhomanenko
National Research Centre
"Kurchatov Institute"
Moscow, Russia
tata.antares@yandex.ru

Abstract

In machine learning ensemble methods have demonstrated high accuracy for the variety of problems in different areas. The most known algorithms intensively used in practice are random forests and gradient boosting. In this paper we present InfiniteBoost — a novel algorithm, which combines the best properties of these two approaches. The algorithm constructs the ensemble of trees for which two properties hold: trees of the ensemble incorporate the mistakes done by others; at the same time the ensemble could contain the infinite number of trees without the over-fitting effect. The proposed algorithm is evaluated on the regression, classification, and ranking tasks using large scale, publicly available datasets.

1 Introduction

Nowadays ensemble methods of machine learning are one of the widespread approaches used in many applications in industry and science: search engines, recommendations, store sales prediction, customer behavior prediction, web text classification, high energy physics, astronomy, computational biology and chemistry, etc. They give state-of-the-art results not only on many standard benchmarks [5, 9] but also in real-world tasks, like movements of individual body parts prediction [19], click through rate prediction [15], ranking relevance prediction in search engines [20]. Wide applicability of ensembles is confirmed by data challenges: 60% of winning solutions of challenges on Kaggle used XGBoost [7], one of the popular gradient boosting implementations.

In ensemble methods multiple individual predictors are trained for the same problem and after that are combined in some manner. The predictors in the ensemble could be constructed independently, like in bagging [1] and, in particular, random forests [2], or depending upon the performance of the previous models, like in gradient boosting [12].

Random forests suggested by L. Breiman [2] are a combination of tree predictors such that each tree is constructed using the random subset of features and bagging [1] providing bootstrap replicas of the training set. The aggregation of the tree predictors is averaging. It is proved that the generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. Thus, arbitrarily many trees in the ensemble can be used in practice providing not decreasing quality on the unseen data. Also L. Breiman shows that predictors in the ensemble should be accurate but uncorrelated to achieve the better quality than the individual predictors. In practice the accuracy of predictors is provided by very deep trees and low correlation is achieved by bagging and random features subsampling.

In contrast to random forests, J. Friedman develops a general gradient descent boosting paradigm [12, 13], an ensemble method of boosted regression trees. It does the greedy function approximation with gradient descent method in the space of functions. On each iteration of boosting a new tree is constructed to approximate the gradient of the loss function with respect to the decision function. This provides accounting the performance and mistakes of previously constructed trees in the ensemble. Gradient boosting models providing the best quality in the applications often contain thousands of trees, consequently building very large effective ensembles is of interest. However, building arbitrarily large boosting models drives to over-fitting, that is, decrease of quality on the unseen data [10, 16].

As soon as random forest and gradient boosting are widely used in practice and have own advantages and disadvantages we present a novel algorithm, called InfiniteBoost, the goal of which is to combine the best properties of both: the construction of an infinite (arbitrarily large) ensemble and at the same time accounting the mistakes of previously constructed trees in the ensemble with a gradient descent method.

The remainder of the paper is organized as follows: in Section 2 the motivation of InfiniteBoost algorithm is described; in Section 3 the description of InfiniteBoost algorithm is given and also the convergence theorem is proven. The comparison of the proposed algorithm with gradient boosting and random forests is given in Section 4: the experimental results are listed for regression, classification, and ranking problems.

*Blog: http://arogozhnikov.github.io

Finally, the overview of other existing boosting-bagging hybrid algorithms (term introduced by J. Friedman [13]) is provided in Section 5.

2 Motivation

Here and further let's consider the ensembles over decision trees. Random forests construct trees in the ensemble independently using randomness in the feature selection and bagging to provide own training set for each predictor. This independence provides the possibility of infinite ensemble building: quality on the unseen data converges to some constant as the number of trees in the ensemble becomes large.

In contrast to random forests, a gradient boosting procedure allows next tree to incorporate the mistakes done by the previous trees in the ensemble. However, it is known that gradient boosting could over-fit. In practice this can be detected with the learning curve when quality on the holdout increases depending on the iteration of boosting and then decreases starting from some iteration. In this paper we refer to this situation as "over-fitting". Thus, for gradient boosting the arbitrarily large number of predictors in the ensemble is not an option. This over-fitting issue is considered as overspecialization in [16] and described as the following: wherein trees added at later iterations tend to impact the prediction of only a few samples, and make negligible contribution towards the remaining samples. To avoid over-fitting a shrinkage parameter is used during boosting procedure: a new tree is added to the ensemble with a coefficient, called shrinkage (or learning rate). Shrinkage is an important hyperparameter of gradient boosting which should be tuned in practice.

InfiniteBoost is an algorithm which aims to build an infinite ensemble (without over-fitting effect with increasing the number of predictors) such that each new predictor incorporates the errors made by the previous predictors in the ensemble (boosting procedure).

3 InfiniteBoost

We start our study from analyzing the desired properties of an algorithm: a) build an infinite converging ensemble b) encounter the errors made by trees using gradient boosting approach. First, let's remind the gradient boosting algorithm (Algorithm 1), where the ensemble prediction $F(x)$ accumulates contribution of the trees $F(x) = \eta \sum_{m=1}^{M} \text{tree}_m(x)$.

To achieve a stationary state predictions in InfiniteBoost are averaged with weights α_m (that can be taken uniform $\alpha_m = 1$) and scaled by an additional constant c, called ensemble capacity or simply capacity: $F(x) = c \times \dfrac{\sum_m \alpha_m \text{tree}_m(x)}{\sum_m \alpha_m}$. The contribution of each individual tree in this model converges to zero. The model of interest should after sufficiently large amount of iterations converge to a stationary process (we expect the process of tree building to be randomized, which is crucial in building powerful ensembles of trees), and the distribution of newly-built trees should coincide with the distribution of trees already in the ensemble

$$F(x) = c \times \mathbb{E}_{\text{trees at } F(x)} \text{tree}(x)$$

with average taken over the trees generated by usual training procedure

$$\text{trees at } F(x) = \\ = \left\{ \text{tree} \,\middle|\, \text{tree} \leftarrow \text{learn} \left(\left\{ x_i, -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right\}_{i=1}^{n} \right) \right\}.$$

The building process is determined by a vector z of the current ensemble predictions on the training set: $z = (F(x_1), \ldots, F(x_n))$. Let's introduce the average prediction of an ensemble on the training set $\overline{T}(z)_i = \mathbb{E}_{\text{tree at } z} \text{tree}(x_i)$ to see that the stationary point of the process is a solution of an equation

$$z = c \times \overline{T}(z) \tag{1}$$

Theorem 1. *Equation (1) has a solution if $\overline{T}(z)$ is bounded ($||\overline{T}(z)|| < const$) and continuous.*

This follows directly from the Brouwer fixed-point theorem. Boundedness holds if the gradient is bounded (as in logistic loss), for other losses this can be achieved by scaling the gradient after some (large) threshold. As for continuity of $\overline{T}(z)$, this property can be enforced e.g. by adding Gaussian noise to z before building a tree.

A method we propose to find this stationary state is InfiniteBoost (Algorithm 2). It can be interpreted as a stochastic optimization of the following regularized loss function:

$$\frac{||z||^2}{2} + c \sum_{i=1}^{n} L(y_i, z_i) = \frac{||z||^2}{2} + c\mathcal{L},$$

while in stochastic gradient descent methods an update has the form

$$z \leftarrow z - \varepsilon_m \left(z + c \times \text{grad}_z \mathcal{L} \right) = \\ = (1 - \varepsilon_m) z - \varepsilon_m c \times \text{grad}_z \mathcal{L}.$$

Algorithm 1 Gradient boosting

Input: training set $\{x_i, y_i\}_{i=1}^{n}$; a differentiable loss function $L(y, F(x))$; the number of boosting iterations M; a shrinkage η.

Algorithm ($F(x)$ is output):

Initialize model with a zero value: $F(x) \leftarrow 0$.

for $m = 1, \ldots, M$ **do**

$\text{tree}_m \leftarrow \text{learn} \left(\left\{ x_i, -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right\}_{i=1}^{n} \right)$

$F(x) \leftarrow F(x) + \eta \ \text{tree}_m(x)$

Optional steps like finding initial constant prediction, usage of Newton-Raphson step, or usage of hessian in the tree construction are skipped for brevity.

Algorithm 2 Infinite Boosting (InifiniteBoost)

Input: training set $\{x_i, y_i\}_{i=1}^n$; a differentiable loss function $L(y, F(x))$; the number of boosting iterations M; capacity c.
Algorithm ($F(x)$ is output):

Initialize model with a zero value: $F(x) \leftarrow 0$.
for $m = 1, \ldots, M$ **do**
$\text{tree}_m \leftarrow \text{learn}\left(\left\{x_i, -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right\}_{i=1}^n\right)$
$F(x) \leftarrow \frac{\sum_{k=1}^m \alpha_k \text{tree}_k(x)}{\sum_{k=1}^m \alpha_k} \times c$

Remarks:

- To avoid over-stepping in first iterations capacity c is replaced with a value providing the contribution of a single tree not greater than one.
- To avoid recomputing ensemble ($\varepsilon_m = \frac{\alpha_m}{\sum_{k=1}^m \alpha_k}$):

$$F(x) \leftarrow (1 - \varepsilon_m)F(x) + \varepsilon_m \times c \times \text{tree}_m(x).$$

The gradient step, as usually in gradient boosting, is replaced with building a tree modelling an anti-gradient. Then an ensemble prediction is updated similarly:

$$F(x) \leftarrow (1 - \varepsilon_m)F(x) + \varepsilon_m c \times \text{tree}_m(x).$$

Different choices of ε_m are possible and correspond to different weightings α_m. For example, $\varepsilon_m = 1/m$ correspond to uniform weighting $\alpha_m = 1$, while $\varepsilon_m = 2/(m+1)$ correspond to assigning higher weights for later trees $\alpha_m = m$. In our experiments the second option is used, because it keeps the effective sample size of $\frac{3}{4}m$ (compatible with uniform weighting), but allows ensemble to adapt faster to new z values: to obtain 99% contribution of "new trees" to the ensemble one needs to enlarge size of the ensemble by a factor of 10, whereas with uniform weighting the size should be enlarged by 100 times.

Theorem 2. *InfiniteBoost converges almost surely to the solution of Equation (1), provided that $c \times \overline{T}(z)$ is a contraction mapping and $\varepsilon_m \sim \frac{1}{m}$.*

The contraction requirement may hold only for sufficiently small capacities c, also it implies there is one and only one solution z^* of Equation (1). Theorem follows from an inequality

$$\mathbb{E}||z_{m+1} - z^*||^2 \leq \\ \leq (1 - \text{const}_1\, \varepsilon_m)^2\, \mathbb{E}||z_m - z^*||^2 + \text{const}_2\, \varepsilon_m^2$$

that holds for some positive constants const_1, const_2, consequently, $\lim_{m \to +\infty} \mathbb{E}||z_m - z^*||^2 = 0$.

Capacity c of an ensemble can be changed in the process of building, allowing next trees to find another optimal point. Since the loss is differentiable with respect to capacity c, we present an InfiniteBoost modification (Algorithm 3) which adapts c using holdout. In experiments, 5% of the training set is used as a holdout to tune capacity. This way, one may skip the process of tuning capacity (or similar parameter shrinkage of gradient boosting). After properly selecting capacity, holdout can be added to the training set, but this is not done in our experiments.

Algorithm 3 Infinite Boosting with adaptive capacity (InifiniteBoost)

Input: training set $\{x_i, y_i\}_{i=1}^n$; a differentiable loss function $L(y, F(x))$; the number of boosting iterations M.
Algorithm ($F(x)$ is output):

Initialize: model $F(x) \leftarrow 0$; capacity $c = \frac{1}{2}$.
Divide training set into two non-overlapping subsets $\{x_i, y_i\}_{i=1}^n = T \sqcup H$.
for $m = 1, \ldots, M$ **do**
$\text{tree}_m \leftarrow \text{learn}\left(\left\{x_i, -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right\}_{\{x_i, y_i\} \in T}\right)$
$F(x) \leftarrow \frac{\sum_{k=1}^m \alpha_k \text{tree}_k(x)}{\sum_{k=1}^m \alpha_k} \times c$
Use very minor correction of capacity:
$s \leftarrow \text{sgn} \sum_{\{x_i, y_i\} \in H} -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} F(x_i)$
$c \leftarrow c \times \left(\frac{m+1}{m}\right)^s$

4 Experiments

We have evaluated InfiniteBoost for three different tasks: classification, regression, and ranking. Large scale, publicly available datasets are used in comparison. In our evaluation, we compare InfiniteBoost with two extreme cases: random forests and gradient boosting with different shrinkage values. The source code with all experiments is available on the github: `https://github.com/Anonymous`.

4.1 InfiniteBoost and gradient boosting

Datasets used for experiments are listed in Table 1. For Higgs dataset 1 million samples are used for training and 500,000 samples are used for test. For YearPredictionMSD dataset random 75% are taken as training and the remaining samples are taken for test purposes. InifiniteBoost uses adaptive capacity for classification and regression problems. For ranking task the fixed capacity value is used because the loss function is not convex in this case and the adaptation on the holdout significantly underestimates capacity. For all tasks the same hyperparameters are used for gradient boosting and InfiniteBoost: subsample is set to 0.7, max features — 0.7, max depth — 7. For gradient boosting shrinkage is varied to compare with InfiniteBoost. Other hyperparameters settings are tested and the similar behavior is observed. In Figures 1, 3, 4 the learning curves are presented. It can be seen that InifiniteBoost provides similar quality as gradient

Table 1: Datasets description for infinite boosting and gradient boosting comparison

Type	Name	Number of instances	Number of features	Source
classification	Higgs 1M	1,500,000	28	link
regression	YearPredictionMSD	515,345	90	link
ranking	yahoo-letor, set 1	638,794	699	[6]

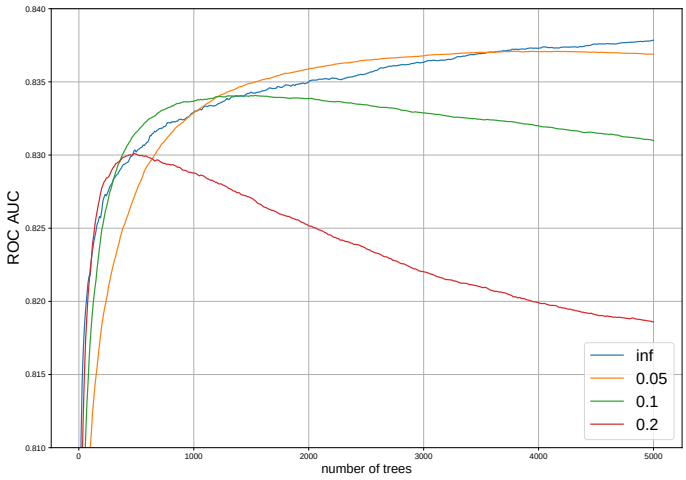

Figure 1: Quality on Higgs dataset for gradient boosting with different shrinkages (0.05, 0.1, 0.2) and infinite boosting with adaptive capacity (inf).

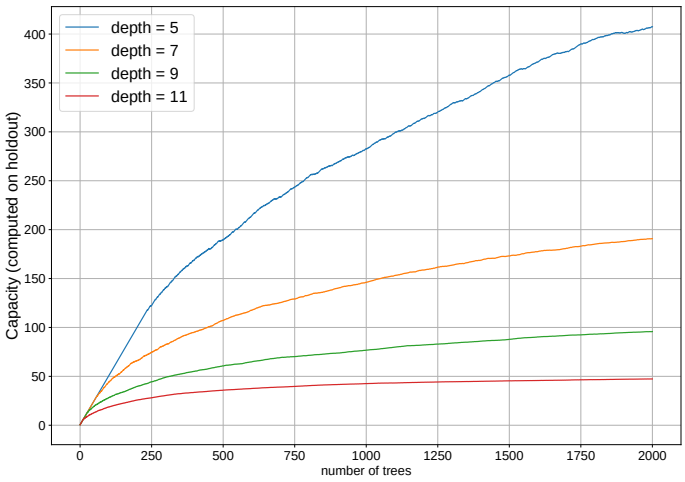

Figure 2: Comparison of infinite boost capacities found by adapting capacity on a holdout for different depths on Higgs dataset.

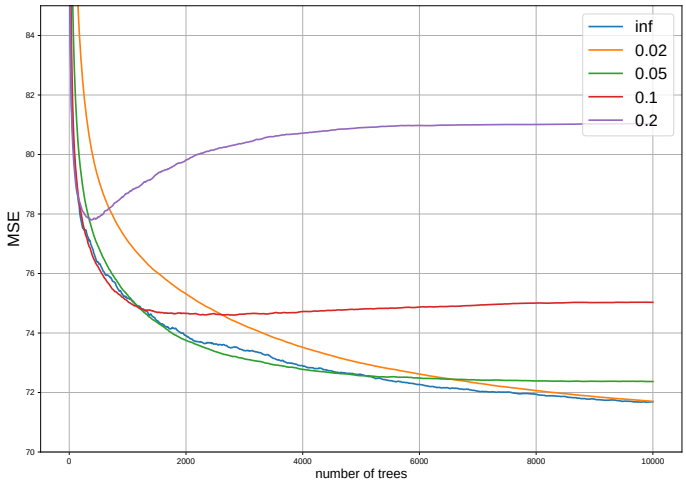

Figure 3: Quality on YearPredictionMSD dataset for gradient boosting with different shrinkages (0.02, 0.05, 0.1, 0.2) and infinite boosting with adaptive capacity (inf).

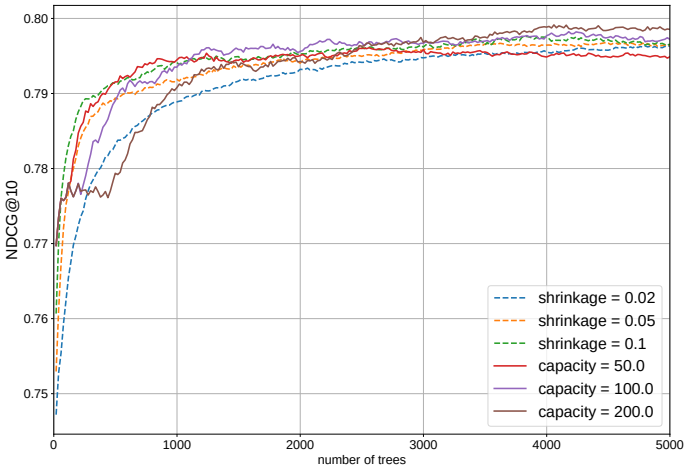

Figure 4: Quality on yahoo-letor dataset for gradient boosting with different shrinkages (0.05, 0.1, 0.2) and infinite boosting with different capacity values (50, 100, 200). Other NDCG@k plots can be found in supplementary material.

boosting (with appropriate shrinkage) and at the same time it is free from the over-fitting effect. As the number of trees

Table 2: Datasets description for infinite boosting and random forest comparison

Type	Name	Number of instances	Number of features	Source
classification	covertype	581,012	54	link
classification	real-sim	72,309	20,958	link
classification	citeseer	181,395	105,354	link

Table 3: ROC AUC qualities provided by random forest and infinite boosting with different capacities c for classification tasks: ensembles contain 100 trees; provided values are the mean and the error computed with k-folding ($k = 4$)

	covertype	real-sim	citeseer
Random Forest	0.9933 ± 0.0001	0.9907 ± 0.0005	0.8831 ± 0.0086
InfiniteBoost, $c = 1$	0.9937 ± 0.0000	0.9914 ± 0.0005	0.8763 ± 0.0132
InfiniteBoost, $c = 2$	0.9940 ± 0.0001	0.9918 ± 0.0006	0.8797 ± 0.0160
InfiniteBoost, $c = 4$	0.9945 ± 0.0001	0.9931 ± 0.0004	0.8764 ± 0.0172

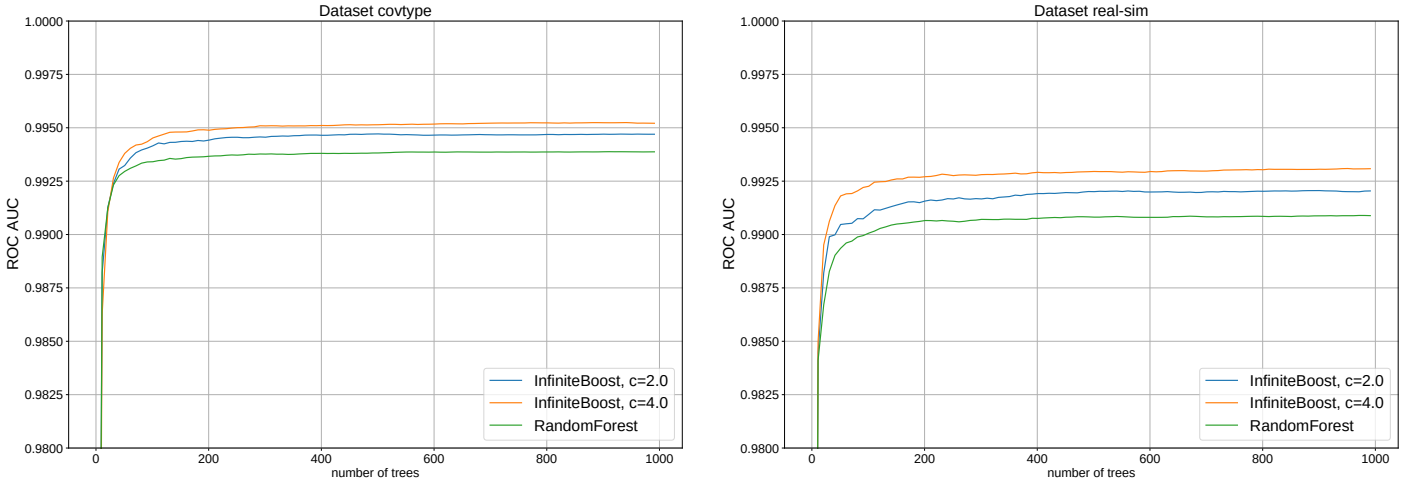

Figure 5: ROC AUC quality on covertype (left) and real-sim (right) datasets for random forest and infinite boosting with capacities $c = 2$ and $c = 4$.

increases the quality of InfiniteBoost tends to some constant. Notably, found optimal capacities differ significantly for trees of different complexity (Figure 2).

4.2 InfiniteBoost and random forest

Random forests are known for their good quality with standard hyperparameters. InfiniteBoost with small capacity almost does not encounter mistakes done at the previous stages, thus, behaves similarly to random forest.

In benchmarks we use the implementation of random forest from scikit-learn [18] with default settings (without limiting depth) and datasets (see Table 2) with known good performance of random forest. To have side-to-side comparison InfiniteBoost uses deep trees with the same parameters, just as random forest. Such deep trees over-fit to the training data, therefore, they are usually considered to be inappropriate for boosting. In the experiments InfiniteBoost is tested with different fixed capacity values and it is observed that adding a bit of boosting behavior to random forest by setting small capacity improves the model for 2 of 3 datasets (see Table 3). New algorithm is capable of using the mistakes done by the previous trees on out-of-bag samples, what makes boosting possible. In experiments when large number of trees was possible, quality of InfiniteBoost converges to some constant showing behavior similar to random forest (Figure 5).

5 Related works

There are different attempts of combining the properties of random forests and gradient boosting. Bagging methods are mainly proposed to effectively reduce the variance of regression predictors, while they leave the bias relatively unchanged. To reduce both bias and variance iterated bagging is developed [3]. It works in stages: on each stage bagging is trained and its outcomes are used to alter the target values; this is repeated until a simple rule stops the process. The idea behind bagging is to train each predictor in the ensemble iteratively on the residual, the difference between the target and prediction of the ensemble constructed by this moment. Thus, iterated bagging includes the property of gradient boosting: each new predictor in the ensemble accounts for the performance of the previous predictors. Bagging procedure and its out-of-bag estimation are used to obtain unbiased estimate of the "true" residual.

Another approach is stochastic gradient boosting [13] that introduces randomness into gradient boosting by providing lower correlation between predictors like in random forests. On each boosting iteration subsample of the training data drawn at random (without replacement) from the full training set is used to fit a new predictor in the ensemble. The lower size of subsample the more the random samples will differ and the more randomness will be introduced into boosting procedure. At the same time lower size of subsample reduces the amount of data used in new predictor fitting. This causes the variance of individual predictors to increase. This version of gradient boosting is widely used in practice and gives good results.

As discussed above, over-fitting is a well-known problem for gradient boosting. The idea of another boosting-bagging hybrid algorithm [16], called DART, is to fight this issue by employing dropouts idea for ensembles of trees: muting complete trees as opposed to muting features in random forests. On each boosting iteration randomly chosen trees form a new ensemble M'. A new regression tree fits the anti-gradient of the loss function with respect to the predictions obtained from the ensemble M'. Adding of a new tree to the initial ensemble is accompanied by a normalization step. Firstly, the new tree is scaled by a factor $1/k$, where k is the number of dropped trees from the initial ensemble, to provide the same order of magnitude for the new tree as the dropped trees. Secondly, the new tree and the dropped trees are scaled by a factor of $k/(k+1)$ and the new tree is added to the ensemble. The last scaling ensures that the combined effect of the dropped trees together with the new tree remains the same as the effect of the dropped trees alone before the introduction of the new tree. InfiniteBoost and DART have similarities in the normalization step procedure, however, DART does not use correction constant (capacity), like InfiniteBoost.

DART implements gradient boosting, if no tree is dropped, and random forests, if all the trees are dropped. With comparison to gradient boosting, this algorithm has no shrinkage hyperparameter, which is replaced by dropout rate, a fraction of trees muted on each iteration. However, the construction of large ensembles is not feasible in this algorithm from the computational point. On each learning iteration it is needed to prepare new target by predicting the training set with a random subset of already constructed trees. This leads to the quadratic (not linear as for gradient boosting) dependence between training time and the number of built trees.[1] Also, the algorithm does not converge to some point for arbitrarily large number of trees because the contribution of newly-built tree does not tend to zero, which causes over-fitting.

Simple approach of combining random forests and boosting is applied in [17], called BagBoo. Each predictor in the bootstrap aggregated ensemble is gradient boosting. This approach aims to be well parallelized (each gradient boosting predictor contains 10-20 trees).

For drug discovery data having varying levels of noise composition of another properties of boosting and random forests is proposed. Ensemble Bridge [8] algorithm is capable of transitioning between boosting and random forests using a regularization parameter, which controls the tree depth, feature selection and randomness in the selection of samples for the tree construction.

6 Conclusion

InfiniteBoost is a new hybrid algorithm. From one side, it is random forest with trees in the ensemble incorporating the mistakes done by other trees. From another side, it is a modification of gradient boosting, which has no shrinkage hyperparameter to tune and does not over-fit with increasing the number of boosting iterations. Empirically InfiniteBoost shows the quality not worse than random forests and asymptotically (with increasing the number of trees) not worse than gradient boosting trained with different shrinkage values. This could save the time spent on shrinkage tuning for gradient boosting in applications. Also, experiments demonstrate that learning curve on the unseen data for InfiniteBoost tends to the constant starting from some iteration of boosting (favorable property of random forests).

[1] There is a mode of DART algorithm, called ε-algorithm, with dropping only one tree from the ensemble on each iteration. It is linear in time w.r.t. the number of trees in the ensemble.

References

[1] Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

[2] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[3] Breiman, L. (2001). Using iterated bagging to debias regressions. Machine Learning, 45(3), 261-277.

[4] Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. Learning, 11(23-581), 81.

[5] Caruana, R., & Niculescu-Mizil, A. (2006, June). An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning (pp. 161-168). ACM.

[6] Chapelle, O., & Chang, Y. (2011, January). Yahoo! learning to rank challenge overview. In Yahoo! Learning to Rank Challenge (pp. 1-24).

[7] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.

[8] Culp, M., Johnson, K., & Michailidis, G. (2010). The ensemble bridge algorithm: A new modeling tool for drug discovery problems. Journal of chemical information and modeling, 50(2), 309-316.

[9] Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems. J. Mach. Learn. Res, 15(1), 3133-3181.

[10] Freund, Y. (2001). An adaptive version of the boost by majority algorithm. Machine learning, 43(3), 293-318.

[11] Freund, Y., & Schapire, R. E. (1995, March). A desicion-theoretic generalization of on-line learning and an application to boosting. In European conference on computational learning theory (pp. 23-37). Springer Berlin Heidelberg.

[12] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

[13] Friedman, J. H. (2002). Stochastic gradient boosting. Computational Statistics & Data Analysis, 38(4), 367-378.

[14] Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics, 28(2), 337-407.

[15] He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., ... & Candela, J. Q. (2014, August). Practical lessons from predicting clicks on ads at facebook. In Proceedings of the Eighth International Workshop on Data Mining for Online Advertising (pp. 1-9). ACM.

[16] Korlakai Vinayak, R., & Gilad-Bachrach, R. (2015). DART: Dropouts meet Multiple Additive Regression Trees. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (pp. 489-497).

[17] Pavlov, D. Y., Gorodilov, A., & Brunk, C. A. (2010, October). BagBoo: a scalable hybrid bagging-the-boosting model. In Proceedings of the 19th ACM international conference on Information and knowledge management (pp. 1897-1900). ACM.

[18] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.

[19] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., ... & Moore, R. (2013). Real-time human pose recognition in parts from single depth images. Communications of the ACM, 56(1), 116-124.

[20] Yin, D., Hu, Y., Tang, J., Daly, T., Zhou, M., Ouyang, H., ... & Langlois, J. M. (2016, August). Ranking relevance in yahoo search. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 323-332). ACM.