WIKIPEDIA

Gradient boosting

Gradient boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.[1][2] When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest.[1][2][3] It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Contents

History

The idea of gradient boosting originated in the observation by Leo Breiman that boosting can be interpreted as an optimization algorithm on a suitable cost function.[4] Explicit regression gradient boosting algorithms were subsequently developed by Jerome H. Friedman,[5][6] simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett and Marcus Frean.[7][8] The latter two papers introduced the view of boosting algorithms as iterative *functional gradient descent* algorithms. That is, algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

Informal introduction

(This section follows the exposition of gradient boosting by Li.[9])

Like other boosting methods, gradient boosting combines weak "learners" into a single strong learner in an iterative fashion. It is easiest to explain in the least-squares regression setting, where the goal is to "teach" a model F to predict values of the form $\hat{y} = F(x)$ by minimizing the mean squared error $\frac{1}{n}\sum_i(\hat{y}_i - y_i)^2$, where i indexes over some training set of size n of actual values of the output variable y:

- $\hat{y}_i =$ the predicted value $F(x)$
- $y_i =$ the observed value
- n the number of samples in y

Now, let us consider a gradient boosting algorithm with M stages. At each stage m ($1 \leq m \leq M$) of gradient boosting, suppose some imperfect model F_m (for low m, this model may simply return $\hat{y}_i = \bar{y}$, where the RHS is the mean of y). In order to improve F_m, our algorithm should add some new estimator, $h_m(x)$. Thus,

$$F_{m+1}(x) = F_m(x) + h_m(x) = y$$

or, equivalently,

$$h_m(x) = y - F_m(x).$$

Therefore, gradient boosting will fit h to the *residual* $y - F_m(x)$. As in other boosting variants, each F_{m+1} attempts to correct the errors of its predecessor F_m. A generalization of this idea to loss functions other than squared error, and to classification and ranking problems, follows from the observation that residuals $h_m(x)$ for a given model are proportional equivalent to the negative gradients of the mean squared error (MSE) loss function (with respect to $F(x)$):

$$L_{\rm MSE} = \frac{1}{n}(y - F(x))^2$$

$$-\frac{\partial L_{\rm MSE}}{\partial F} = \frac{2}{n}(y - F(x)) = \frac{2}{n}h_m(x).$$

So, gradient boosting could be specialized to a gradient descent algorithm, and generalizing it entails "plugging in" a different loss and its gradient.

Algorithm

In many supervised learning problems there is an output variable *y* and a vector of input variables *x*, related to each other with some probabilistic distribution. The goal is to find some function $\hat{F}(x)$ that best approximates the output variable from the values of input variables. This is formalized by introducing some loss function $L(y, F(x))$ and minimizing it:

$$\hat{F} = \arg\min_F \mathbb{E}_{x,y}[L(y, F(x))].$$

The gradient boosting method assumes a real-valued y and seeks an approximation $\hat{F}(x)$ in the form of a weighted sum of functions $h_i(x)$ from some class $\mathcal{H}$, called base (or weak) learners:

$$\hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + \text{const}.$$

We are usually given a training set $\{(x_1, y_1), \dots, (x_n, y_n)\}$ of known sample values of x and corresponding values of y. In accordance with the empirical risk minimization principle, the method tries to find an approximation $\hat{F}(x)$ that minimizes the average value of the loss function on the training set, i.e., minimizes the empirical risk. It does so by starting with a model, consisting of a constant function $F_0(x)$, and incrementally expands it in a greedy fashion:

$$F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, \gamma),$$
$$F_m(x) = F_{m-1}(x) + \underset{h_m \in \mathcal{H}}{\arg\min} \left[\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i)) \right],$$

where $h_m \in \mathcal{H}$ is a base learner function.

Unfortunately, choosing the best function h at each step for an arbitrary loss function L is a computationally infeasible optimization problem in general. Therefore, we restrict our approach to a simplified version of the problem.

The idea is to apply a steepest descent step to this minimization problem (functional gradient descent).

The basic idea behind the steepest descent is to find a local minimum of the loss function by iterating on the $F_m(x)$. In fact, it can be proven that the maximum-descent direction (strongest negative derivative) of the loss function to the local minimum along the $F_m(x)$ is the function itself subtracted by the loss function gradient itself.[10] Hence:

$$F_m(x) = F_{m-1}(x) - \gamma \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))$$

Where $\gamma > 0$. This implies: $L(y_i, F_m(x_i)) \le L(y_i, F_{m-1}(x_i))$.

Furthermore, we can optimize γ by finding the γ value for which the Loss Function has a minimum:

$$\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_m)\right) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))\right),$$

If we considered the continuous case, i.e. where $\mathcal{H}$ is the set of arbitrary differentiable functions on $\mathbb{R}$, we would update the model in accordance with the following equations

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))$$

Where:

$$\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))\right),$$

where the derivatives are taken with respect to the functions F_i for $i \in \{1, .., m\}$, and γ_m is the step length. In the discrete case however, i.e. when the set $\mathcal{H}$ is finite, we choose the candidate function h closest to the gradient of L for which the coefficient γ may then be calculated with the aid of line search on the above equations. Note that this approach is a heuristic and therefore doesn't yield an exact solution to the given problem, but rather an approximation. In pseudocode, the generic gradient boosting method is:[5][2]

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

Algorithm:

1. Initialize model with a constant value:
       $F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, \gamma).$
2. For $m$ = 1 to $M$:
    1. Compute so-called pseudo-residuals:
           $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$   for $i = 1, \ldots, n.$
    2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-
       residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$.
    3. Compute multiplier $\gamma_m$ by solving the following one-dimensional optimization problem:
           $\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$
    4. Update the model:
           $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$
3. Output $F_M(x)$.
```

Gradient tree boosting

Gradient boosting is typically used with decision trees (especially CART trees) of a fixed size as base learners. For this special case, Friedman proposes a modification to gradient boosting method which improves the quality of fit of each base learner.

Generic gradient boosting at the m-th step would fit a decision tree $h_m(x)$ to pseudo-residuals. Let J_m be the number of its leaves. The tree partitions the input space into J_m disjoint regions $R_{1m}, \ldots, R_{J_m m}$ and predicts a constant value in each region. Using the indicator notation, the output of $h_m(x)$ for input x can be written as the sum:

$$h_m(x) = \sum_{j=1}^{J_m} b_{jm} \mathbf{1}_{R_{jm}}(x),$$

where b_{jm} is the value predicted in the region R_{jm}.[11]

Then the coefficients b_{jm} are multiplied by some value γ_m, chosen using line search so as to minimize the loss function, and the model is updated as follows:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \quad \gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

Friedman proposes to modify this algorithm so that it chooses a separate optimal value γ_{jm} for each of the tree's regions, instead of a single γ_m for the whole tree. He calls the modified algorithm "TreeBoost". The coefficients b_{jm} from the tree-fitting procedure can be then simply discarded and the model update rule becomes:

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}_{R_{jm}}(x), \quad \gamma_{jm} = \underset{\gamma}{\arg\min} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma).$$

Size of trees

J, the number of terminal nodes in trees, is the method's parameter which can be adjusted for a data set at hand. It controls the maximum allowed level of interaction between variables in the model. With $J = 2$ (decision stumps), no interaction between variables is allowed. With $J = 3$ the model may include effects of the interaction between up to two variables, and so on.

Hastie et al.[2] comment that typically $4 \leq J \leq 8$ work well for boosting and results are fairly insensitive to the choice of J in this range, $J = 2$ is insufficient for many applications, and $J > 10$ is unlikely to be required.

Regularization

Fitting the training set too closely can lead to degradation of the model's generalization ability. Several so-called regularization techniques reduce this overfitting effect by constraining the fitting procedure.

One natural regularization parameter is the number of gradient boosting iterations M (i.e. the number of trees in the model when the base learner is a decision tree). Increasing M reduces the error on training set, but setting it too high may lead to overfitting. An optimal value of M is often selected by monitoring prediction error on a separate validation data set. Besides controlling M, several other regularization techniques are used.

Another regularization parameter is the depth of the trees. The higher this value the more likely the model will overfit the training data.

Shrinkage

An important part of gradient boosting method is regularization by shrinkage which consists in modifying the update rule as follows:

$$F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x), \quad 0 < \nu \leq 1,$$

where parameter ν is called the "learning rate".

Empirically it has been found that using small learning rates (such as $\nu < 0.1$) yields dramatic improvements in models' generalization ability over gradient boosting without shrinking ($\nu = 1$).[2] However, it comes at the price of increasing computational time both during training and querying: lower learning rate requires more iterations.

Stochastic gradient boosting

Soon after the introduction of gradient boosting, Friedman proposed a minor modification to the algorithm, motivated by Breiman's bootstrap aggregation ("bagging") method.[6] Specifically, he proposed that at each iteration of the algorithm, a base learner should be fit on a subsample of the training set drawn at random without replacement.[12] Friedman observed a substantial improvement in gradient boosting's accuracy with this modification.

Subsample size is some constant fraction f of the size of the training set. When $f = 1$, the algorithm is deterministic and identical to the one described above. Smaller values of f introduce randomness into the algorithm and help prevent overfitting, acting as a kind of regularization. The algorithm also becomes faster, because regression trees have to be fit to smaller datasets at each iteration. Friedman[6] obtained that $0.5 \leq f \leq 0.8$ leads to good results for small and moderate sized training sets. Therefore, f is typically set to 0.5, meaning that one half of the training set is used to build each base learner.

Also, like in bagging, subsampling allows one to define an out-of-bag error of the prediction performance improvement by evaluating predictions on those observations which were not used in the building of the next base learner. Out-of-bag estimates help avoid the need for an independent validation dataset, but often underestimate actual performance improvement and the optimal number of iterations.[13][14]

Number of observations in leaves

Gradient tree boosting implementations often also use regularization by limiting the minimum number of observations in trees' terminal nodes. It is used in the tree building process by ignoring any splits that lead to nodes containing fewer than this number of training set instances.

Imposing this limit helps to reduce variance in predictions at leaves.

Penalize complexity of tree

Another useful regularization techniques for gradient boosted trees is to penalize model complexity of the learned model.[15] The model complexity can be defined as the proportional number of leaves in the learned trees. The joint optimization of loss and model complexity corresponds to a post-pruning algorithm to remove branches that fail to reduce the loss by a threshold. Other kinds of regularization such as an ℓ_2 penalty on the leaf values can also be added to avoid overfitting.

Usage

Gradient boosting can be used in the field of learning to rank. The commercial web search engines Yahoo[16] and Yandex[17] use variants of gradient boosting in their machine-learned ranking engines. Gradient boosting is also utilized in High Energy Physics in data analysis. At the Large Hadron Collider (LHC), variants of gradient boosting Deep Neural Networks (DNN) were successful in reproducing the results of non-machine learning methods of analysis on datasets used to discover the Higgs boson.[18]

Names

The method goes by a variety of names. Friedman introduced his regression technique as a "Gradient Boosting Machine" (GBM).[5] Mason, Baxter et al. described the generalized abstract class of algorithms as "functional gradient boosting".[7][8] Friedman et al. describe an advancement of gradient boosted models as Multiple Additive Regression Trees (MART);[19] Elith et al. describe that approach as "Boosted Regression Trees" (BRT).[20]

A popular open-source implementation for R calls it a "Generalized Boosting Model",[13] however packages expanding this work use BRT.[21] Yet another name is TreeNet, after an early commercial implementation from Salford System's Dan Steinberg, one of researchers who pioneered the use of tree-based methods.[22] XGBoost is another popular modern implementation of the method with some extensions, like second-order optimization.

Disadvantages

While boosting can increase the accuracy of a base learner, such as a decision tree or linear regression, it sacrifices intelligibility and interpretability.[1][23] Furthermore, its implementation may be more difficult due to the higher computational demand.

See also

- AdaBoost
- Random forest
- Catboost
- LightGBM
- XGBoost
- Decision tree learning

References

1. Piryonesi, S. Madeh; El-Diraby, Tamer E. (2020-03-01). "Data Analytics in Asset Management: Cost-Effective Prediction of the Pavement Condition Index" (https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29IS.1943-555X.0000512). *Journal of Infrastructure Systems*. **26** (1): 04019036. doi:10.1061/(ASCE)IS.1943-555X.0000512 (https://doi.org/10.1061%2F%28ASCE%29IS.1943-555X.0000512). ISSN 1943-555X (https://www.worldcat.org/issn/1943-555X).

2. Hastie, T.; Tibshirani, R.; Friedman, J. H. (2009). "10. Boosting and Additive Trees" (https://web.archive.org/web/20091110212529/http://www-stat.stanford.edu/~tibs/ElemStatLearn/). *The Elements of Statistical Learning* (2nd ed.). New York: Springer. pp. 337–384. ISBN 978-0-387-84857-0. Archived from the original (http://www-stat.stanford.edu/~tibs/ElemStatLearn/) on 2009-11-10.
3. Piryonesi, S. Madeh; El-Diraby, Tamer E. (2021-02-01). "Using Machine Learning to Examine Impact of Type of Performance Indicator on Flexible Pavement Deterioration Modeling" (http://ascelibrary.org/doi/10.1061/%28ASCE%29IS.1943-555X.0000602). *Journal of Infrastructure Systems*. **27** (2): 04021005. doi:10.1061/(ASCE)IS.1943-555X.0000602 (https://doi.org/10.1061%2F%28ASCE%29IS.1943-555X.0000602). ISSN 1076-0342 (https://www.worldcat.org/issn/1076-0342).
4. Breiman, L. (June 1997). "Arcing The Edge" (https://statistics.berkeley.edu/sites/default/files/tech-reports/486.pdf) (PDF). *Technical Report 486*. Statistics Department, University of California, Berkeley.
5. Friedman, J. H. (February 1999). "Greedy Function Approximation: A Gradient Boosting Machine" (https://statweb.stanford.edu/~jhf/ftp/trebst.pdf) (PDF).
6. Friedman, J. H. (March 1999). "Stochastic Gradient Boosting" (https://statweb.stanford.edu/~jhf/ftp/stobst.pdf) (PDF).
7. Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (1999). "Boosting Algorithms as Gradient Descent" (http://papers.nips.cc/paper/1766-boosting-algorithms-as-gradient-descent.pdf) (PDF). In S.A. Solla and T.K. Leen and K. Müller (ed.). *Advances in Neural Information Processing Systems 12*. MIT Press. pp. 512–518.
8. Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (May 1999). "Boosting Algorithms as Gradient Descent in Function Space" (https://web.archive.org/web/20181222170928/https://www.maths.dur.ac.uk/~dma6kp/pdf/face_recognition/Boosting/Mason99AnyboostLong.pdf) (PDF). Archived from the original (https://www.maths.dur.ac.uk/~dma6kp/pdf/face_recognition/Boosting/Mason99AnyboostLong.pdf) (PDF) on 2018-12-22.
9. Cheng Li. "A Gentle Introduction to Gradient Boosting" (http://www.chengli.io/tutorials/gradient_boosting.pdf) (PDF).
10. Lambers, Jim (Summer Session 2011-12). "The Method of Steepest Descent" (https://www.math.usm.edu/lambers/mat419/lecture10.pdf) (PDF). Check date values in: `|date=` (help)
11. Note: in case of usual CART trees, the trees are fitted using least-squares loss, and so the coefficient b_{jm} for the region R_{jm} is equal to just the value of output variable, averaged over all training instances in R_{jm}.
12. Note that this is different from bagging, which samples with replacement because it uses samples of the same size as the training set.
13. Ridgeway, Greg (2007). Generalized Boosted Models: A guide to the gbm package. (https://cran.r-project.org/web/packages/gbm/gbm.pdf)
14. Learn Gradient Boosting Algorithm for better predictions (with codes in R) (https://www.analyticsvidhya.com/blog/2015/09/complete-guide-boosting-methods/)
15. Tianqi Chen. Introduction to Boosted Trees (http://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf)
16. Cossock, David and Zhang, Tong (2008). Statistical Analysis of Bayes Optimal Subset Ranking (http://www.stat.rutgers.edu/~tzhang/papers/it08-ranking.pdf) Archived (https://web.archive.org/web/20100807162855/http://www.stat.rutgers.edu/~tzhang/papers/it08-ranking.pdf) 2010-08-07 at the Wayback Machine, page 14.
17. Yandex corporate blog entry about new ranking model "Snezhinsk" (http://webmaster.ya.ru/replies.xml?item_no=5707&ncrnd=5118) (in Russian)
18. Lalchand, Vidhi (2020). "Extracting more from boosted decision trees: A high energy physics case study". arXiv:2001.06033 (https://arxiv.org/abs/2001.06033) [stat.ML (https://arxiv.org/archive/stat.ML)].

19. Friedman, Jerome (2003). "Multiple Additive Regression Trees with Application in Epidemiology". *Statistics in Medicine*. **22** (9): 1365–1381. doi:10.1002/sim.1501 (https://doi.org/10.1002%2Fsim.1501). PMID 12704603 (https://pubmed.ncbi.nlm.nih.gov/12704603).
20. Elith, Jane (2008). "A working guide to boosted regression trees" (https://doi.org/10.1111%2Fj.1365-2656.2008.01390.x). *Journal of Animal Ecology*. **77** (4): 802–813. doi:10.1111/j.1365-2656.2008.01390.x (https://doi.org/10.1111%2Fj.1365-2656.2008.01390.x). PMID 18397250 (https://pubmed.ncbi.nlm.nih.gov/18397250).
21. Elith, Jane. "Boosted Regression Trees for ecological modeling" (https://cran.r-project.org/web/packages/dismo/vignettes/brt.pdf) (PDF). *CRAN*. CRAN. Retrieved 31 August 2018.
22. https://www.kdnuggets.com/2013/06/exclusive-interview-dan-steinberg-salford-systems-data-mining-solutions-provider.html
23. Wu, Xindong; Kumar, Vipin; Ross Quinlan, J.; Ghosh, Joydeep; Yang, Qiang; Motoda, Hiroshi; McLachlan, Geoffrey J.; Ng, Angus; Liu, Bing; Yu, Philip S.; Zhou, Zhi-Hua (2008-01-01). "Top 10 algorithms in data mining". *Knowledge and Information Systems*. **14** (1): 1–37. doi:10.1007/s10115-007-0114-2 (https://doi.org/10.1007%2Fs10115-007-0114-2). hdl:10983/15329 (https://hdl.handle.net/10983%2F15329). ISSN 0219-3116 (https://www.worldcat.org/issn/0219-3116). S2CID 2367747 (https://api.semanticscholar.org/CorpusID:2367747).

Further reading

- Boehmke, Bradley; Greenwell, Brandon (2019). "Gradient Boosting". *Hands-On Machine Learning with R*. Chapman & Hall. pp. 221–245. ISBN 978-1-138-49568-5.

External links

- How to explain gradient boosting (http://explained.ai/gradient-boosting/index.html)
- Gradient Boosted Regression Trees (https://blog.datarobot.com/gradient-boosted-regression-trees)
- LightGBM (https://github.com/microsoft/LightGBM)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Gradient_boosting&oldid=1048099522"