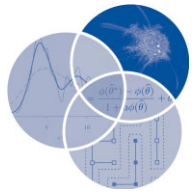
Advanced Review

Boosting



Peter Bühlmann[1] and Bin Yu[2]*

In this contribution, we review boosting, one of the most effective machine learning methods for classification and regression. Most of the article takes the gradient descent point of view, even though we do include the margin point of view as well. In particular, AdaBoost in classification and various versions of L2boosting in regression are covered. Advice on how to choose base (weak) learners and loss functions and pointers to software are also given for practitioners. *WIREs Comp Stat* 2010 2 69–74 DOI: 10.1002/wics.55

Keywords: boosting; gradient descent; AdaBoost; L2Boost; base learner

Boosting is a modern statistical methodology for classification or regression that originated from machine learning in the mid 1990s and has seen much progress in recent years in terms of its understanding and effectiveness in modeling real world data. Its successful applications widely vary and include text mining, image segmentation, language translation, bioinformatics, neuroscience, and finance, just to name a few. Boosting has many distinctive characteristics. First, it is an ensemble method through linear combination of so-called weak learners. Second, it is closely connected to optimization (often convex) gradient descent (iterative) algorithms. The first trait is shared by other statistical machine learning methods such as bagging and random forests, and the second by various programming methods for optimization. The third characteristic of boosting is its early stopping rule to achieve regularization. This trait couples regularization with numerical optimization and makes boosting very special. The fourth characteristic is the re-weighting of data examples (units) based on the outcome of the previous iteration. This trait is a consequence of the form of the gradient of the loss function in the optimization and hence not shared by all variants of boosting as we know now. It is, however, an important feature of the original AdaBoost algorithm.[1]

Kearns and Valiant[2] postulated the boosting conjecture in the framework of probably approximately correct (PAC) learning that a weak classifier (with success probability just a bit over 50%) can be 'boosted' into a strong one in the sense that the training error of the new one would go to zero and with a polynomial-time run time. In his award-wining MIT thesis in Computer Science,[3] answered the question in its affirmative, and Freund and Schapire[1] devised the first practical boosting algorithm, *AdaBoost*, in binary classification.

Breiman[4,5] made the crucial connection of AdaBoost to optimization by re-deriving AdaBoost from a gradient descent point of view on an exponential loss function of the margin (which is the product yf of the true label $y \in \{-1, 1\}$ and the prediction f). This connection was further developed by Friedman et al.[6] and Mason et al.[7] Many variants of boosting followed in the statistics community by generalizations of AdaBoost to different loss functions and through different gradient descent methods. The most prominent ones are LogitBoost[6] in classification and L2Boosting[8] in regression. A separate line of research on AdaBoost has concentrated on the margin view point mainly by the machine learning community[9–13] and recent developments there are emphasizing AdaBoost's ensemble and re-weighting aspects[14] and references therein).

Theoretical analyses of boosting have also been developed by both the theoretical statistics and theoretical machine learning communities. The former community studied, e.g., the Bayes and model selection consistency property and minimax rates of convergence of various boosting procedures (when stopped early)[15–20]; while the latter community studied algorithmic convergence of boosting[10] and upper bounds on the generalization error of AdaBoost via distributions of margins and complexity measures of the model class and the sample size.[9] One insight from these theoretical analyses is that the complexity of the boosting procedure is not simply a linear function of the number of iterations. In a simple regression case studied in Ref 15, the complexity increases with the number of iteration in

*Correspondence to: binyu@stat.berkeley.edu

[1]Seminar für Statistik, ETH Zürich, Zürich, CH–8092, Switzerland

[2]Department of Statistics, University of California, Berkeley, CA 94720-3860, USA

DOI: 10.1002/wics.55

an exponentially diminishing manner—the complexity is upper bounded by the noise variance and the amount of complexity added is exponentially small when the iteration number increases. This partially explains the overfitting resistance of AdaBoost. Eventually, AdaBoost and all other boosting methods will overfit, but it could take a long time especially in classification because of the robustness property of the evaluating 0–1 loss function in addition to the slow increase of the complexity of boosting. Most of the theoretical studies so far have been carried out under the i.i.d. assumption although some generalizations to stationary data sources have been analyzed.[21,22]

In this contribution, we review the general boosting algorithm mainly from the gradient descent point of view. In particular, we cover the AdaBoost, LogitBoost, and L2Boosting algorithms. After a short section on the margin view point, we end with a discussion on the choices of loss functions, base learners, and stopping rules.

THE GRADIENT DESCENT POINT OF VIEW

We describe here the general boosting algorithm from the gradient descent point of view.[4–8] In particular, we review AdaBoost and LogitBoost for classification (corresponding to different loss functions L) and various versions of L2Boosting for regression (corresponding to the L2-loss function). Finally, we mention briefly the generalization of boosting from empirical loss to penalized empirical loss such as Lasso[23] in regression.

Suppose that we observe

$$(X_1, Y_1), \ldots, (X_n, Y_n), \tag{1}$$

where $X_i \in \mathbb{R}^p$ denotes a p-dimensional predictor variable and Y_i a univariate response, for example taking values in $\mathbb{R}$ as in regression or in $\{-1, +1\}$ as in binary classification. In the sequel, we denote by $X^{(j)}$ the jth component of a vector $X \in \mathbb{R}^p$. We usually assume that the pairs (X_i, Y_i) are i.i.d. or from a stationary process. The goal is to estimate the regression function $F(x) = \mathbb{E}[Y|X = x]$ or to find a classifier $\text{sign}(F(x))$ where $F(x) : \mathbb{R}^p \to \mathbb{R}$.

The estimation performance is evaluated via a real-valued loss function in the sense that we want to minimize the expected loss or risk:

$$\mathbb{E}L(Y, F(X)), \tag{2}$$

based on data $(X_i, Y_i)(i = 1, \ldots, n)$. The loss function L is assumed to be smooth and convex in the second argument so that the gradient method can be applied. Boosting algorithms are obtainable by minimizing the empirical loss function

$$n^{-1} \sum_{i=1}^{n} L(Y_i, F(X_i)), \tag{3}$$

over an additive expansion of base learners for F via functional gradient descent. The base learners take the form of $h(x, \hat{\theta})$ $(x \in \mathbb{R}^p)$ where $\hat{\theta}$ is an estimated parameter of finite or infinite dimension. For example, the base learners could be stumps with $\hat{\theta}$ describing the axis to split, the split points and the fitted values for the two terminal nodes. Fitting the base learner based on data is part of the base learner.

The boosting methodology in general builds on a user-determined base procedure or weak learner and uses it repeatedly on modified data which are typically outputs from the previous iterations. The final boosted procedure takes the form of linear combinations of the base procedures. Precisely, given a base learner $h(x, \theta)$, boosting is derivable as functional gradient descent on the loss function L.

Boosting (gradient descent view)

1. Start with $F_0(x) = 0$.
2. Given $F_{m-1}(x)$, let

$$(\beta_m, h(x, \hat{\theta}_m))) = \text{argmin}_{\beta \in \mathbb{R}, \theta} \sum_{i=1}^{n} L(Y_i, F_{m-1}(X_i) + \beta h(x, \theta)). \tag{4}$$

3. Set

$$F_m(x) = F_{m-1}(x) + \beta_m h(x, \hat{\theta}_m). \tag{5}$$

4. Stop when $m = M$.

The AdaBoost classifier is $\text{sign}(F_M(x))$.

Classification

In binary classification, $y \in \{-1, +1\}$ and the most commonly used loss is the 0-1 loss. That is, for a classifier $\text{sign}(F(x)) \in \{-1, +1\}$ if the label of x is $y \in \{-1, +1\}$, the 0-1 loss can be written as a function of the margin $yF(x)$:

$$L_{01}(y, F(x)) = I\{yF(x) < 0\}. \tag{6}$$

It is easy to see that the exponential loss function

$$L_{exp}(y, F(x)) = \exp(-yF(x)) \tag{7}$$

is an upper bound on L_{01} and its population minimizer is half of the log odds ratio

$$F(x) = \frac{1}{2}\log\frac{\mathbb{P}(Y=1|X=x)}{\mathbb{P}(Y=-1|X=x)}. \quad (8)$$

For example, AdaBoost, the weak or base learner is a procedure that maps from a given training data set to a classifier with a training error better than a random guess, or less than 50%. Often used are tree-based classifiers. AdaBoost improves upon the current fit in an additive way to minimize the empirical exponential loss function over the base learner (acting on a modified data set) and a multiplier.

AdaBoost

1. Start with $F_0(x) = 0$;
2. Given $F_{m-1}(x)$, let

$$w_i^{(m)} = \exp(-Y_i F_{m-1}(X_i)), \quad (9)$$

$$h(x,\hat{\theta}_m)) = argmin_\theta \sum_{i=1}^n w_i^{(m)} I(Y_i \neq h(X_i,\theta)), \quad (10)$$

and denote $h(\cdot,\hat{\theta}_m)$'s associated error by

$$err_m = \frac{\sum_{i=1}^n w_i^{(m)} I(Y_i \neq h(X_i,\hat{\theta}_m))}{\sum_{i=1}^n w_i^{(m)}}. \quad (11)$$

Furthermore let

$$\beta_m = \frac{1}{2}\log\frac{1-err_m}{err_m}. \quad (12)$$

3. Set

$$F_m(x) = F_{m-1}(x) + \beta_m h(x,\hat{\theta}_m). \quad (13)$$

4. Stop when $m = M$.
5. The AdaBoost classifier is $y = \text{sign}(F_M(x))$.

Clearly, two key inputs to the AdaBoost algorithm are the choices of base learner $h(\cdot,\theta)$ and the stopping rule for M. Relatively simple tree classifiers such as stumps or eight-node trees have been used effectively as the base learners in many empirical studies. The stopping iteration M acts as a regularization parameter. A test data set, when the data set is large, or cross validation can be used to find such an M. When M tends to infinity, the AdaBoost estimator has been shown to converge to the minimizer of the empirical exponential loss over the set of linear combinations of base learners([18,20,24]).

We defer further discussions on the choices of the base procedure and some stopping rules for M to Section Practical Issues in Applying Boosting.

LogitBoost

If the loss function is the negative log-likelihood function from a logistic model with a logit link function, we get the loss function

$$\log(1+\exp^{-yF}), \quad (14)$$

or equivalently

$$L_{\text{logit}}(y,F) = \log_2(1+\exp^{-yF}), \quad (15)$$

which is an upper bound on the 0–1 loss function L_{01} as a function of the margin yF. Moreover, the expected loss of L_{logit} is minimized by the same function as in Eq. (8), as for the exponential loss function.

In multi-class situations, a log-likelihood function in multinomial models can be used to apply the gradient descent algorithm to arrive at a multi-class boosting algorithm. However, often in practice, the one-against-all approach is used to turn a multi-class problem into many binary classification problem such that AdaBoost or LogitBoost can be applied.

Regression: Boosting With the Squared Error Loss

In regression, a natural loss function is the squared error loss. With this loss function, we get $L2$Boosting.[8] When applying the gradient descent boosting algorithm with the squared loss, we end up with a repeated fitting of residuals with the base learner. Analogously as before, $L2$Boosting is a 'constrained' minimization of the empirical squared error risk $n^{-1}\sum_{i=1}^n (Y_i - F(X_i))^2$ (with respect to $F(\cdot)$) which yields an estimator $\hat{F}(\cdot)$. The regularization of the empirical risk minimization comes in again implicitly by the choice of a base procedure and by algorithmical constraints such as early stopping or some penalty barriers.

***L2*Boosting (with base procedure $h(\cdot,\theta)$)**

1. Start with $F_0 = 0$.
2. Given $F_{m-1}(x)$, Compute residuals $U_i = Y_i - F_{m-1}(X_i)$ $(i = 1,\ldots,n)$. Fit the base procedure to the current residuals:

$$h(x,\hat{\theta}_m) = argmin_\theta \sum_{i=1}^n (U_i - h(X_i,\theta))^2, \quad (16)$$

3.

$$F_m(x) = F_{m-1}(x) + \beta_m h(x, \hat{\theta}_m), \qquad (17)$$

where the line search turns out to give $\beta_m \equiv 1$.

4. Stop when $m = M$ and $F_M(x)$ is the final estimator of the regression function.

As before, the number of iterations M is the main tuning parameter for $L2$Boosting in addition to the base learner B. The number of boosting iterations may be estimated by cross validation. As an alternative, one may use some model selection criteria to bypass cross validation and bring computation savings.

When the base procedure selects just one predictor (feature) among all of the p variables, gradient descent becomes coordinatewise gradient descent. For example, for the component-wise smoothing spline base procedure which fits a smoothing spline to the predictor variable reducing residual sum of squares most, the boosting estimator yields an additive model fit.[15] Another widely used procedure is component-wise linear least squares yielding a linear model fit.

L2Boosting: coordinatewise descent for linear models

1. Start with $F_0 = 0$.
2. Given $F_{m-1}(x)$, Compute residuals $U_i = Y_i - F_{m-1}(X_i)$ $(i = 1, \ldots, n)$. Let $X_i^{(j)}$ be the jth component of $X_i \in \mathbb{R}^p$,

$$\hat{j}_m = argmin_{j=1,\ldots,p} \sum_{i=1}^{n} (U_i - \hat{\beta}_m X_i^{(j)}))^2, \qquad (18)$$

$$\hat{\beta}_m = argmin_{\beta} \sum_{i=1}^{n} (U_i - \beta X_i^{(j_m)})^2, \qquad (19)$$

3.

$$F_m(x) = F_{m-1}(x) + \hat{\beta}_m \, x^{(\hat{j}_m)}, \qquad (20)$$

4. Stop when $m = M$ and $F_M(x)$ is the final estimator of the linear regression function.

Friedman[8] introduced shrinkage to L2Boosting by shrinking the base learner using $\nu \cdot h(\cdot, \theta)$ with $0 < \nu \leq 1$. Empirical evidence suggests that the choice for the step-size ν is not crucial as long as ν is small; we usually use $\nu = 0.1$. A related version of L2Boosting has been termed e-L2Boosting in the literature([25,26]):

e-L2Boosting Normalize all predictors to the same scale and let $\hat{\beta}_m$ have a fixed "step-size" $\nu > 0$, but with a sign depending on the correlation between the chosen $X^{(\hat{j}m)}$ and the current residual vector U. That is,

$$\hat{\beta}_m \equiv \nu \, \text{sign}(corr(U, X^{(\hat{j}m)})). \qquad (21)$$

With $m = 2$ and $\nu = 1$, $L2$Boosting has already been proposed by Tukey[27] under the name 'twicing'. e-L2Boosting is also called forward stagewise fitting in Ref 26. It is shown there to be connected to the L1-penalized least squares method Lasso[23] and often shares the sparsity property of Lasso. In fact, Lasso and e-L2Boosting are connected through the Blasso algorithm[28] that consists of e-L2boosting steps (forward steps) and appropriately chosen backward steps where predictors could be removed from the fitted function.

Instead of L1-penalized least squares one can take L0-penalized least squares, although the latter is computationally awkward and infeasible to solve. This is related to SparseBoosting[29] which employs some information criteria such as AIC, BIC, or gMDL, while gMDL was recommended by Ref 29 for overall performance of prediction and sparsity. Recently, Zhang[30] combines forward and backward steps to minimize L0-penalized LS and Friedman and Popescu[31] devise a gradient direction regularization algorithm which does not necessarily come from an explicit penalty function.

THE MARGIN POINT OF VIEW IN CLASSIFICATION

Schapire et al.[9] suggest a new explanation for the effectiveness of AdaBoost via boosting the margin distribution. Note that the margin is a key concept in the methodology and theory of support vector machines. This work started a separate trend of generalizations of AdaBoost in the machine learning community through maximizing various versions of the margin, e.g., leading to LPBoost,[32] SoftBoost,[33] and entropy regularized LPBoost.[14] Among these margin-based algorithms, some are seen to be only 'corrective' in the sense that they base the re-weighting only on the outcomes of the previous iteration, and some are 'totally corrective' because the re-weighting takes into account the outcomes of all previous iterations. Some studies (with C4.5 or radial functions as base learners) indicate that these margin-based methods have similar or slightly better classification accuracy than AdaBoost (and possibly than LogitBoost as well). In a computational speed

comparison study in Ref 14, LPBoost is found to be the fastest, entropy regularized LPBoost slightly slower while SoftBoost is much slower. These algorithms rely on primal-dual formulations in convex optimization from which some convergence rate analyses are known for SoftBoost and entropy regularized LPBoost.

PRACTICAL ISSUES IN APPLYING BOOSTING

The boosting methodology as we know now has three ingredients: loss function, base learner, and stopping rule. To apply boosting in practice, choices have to be made for these three components. Similar to applying any other methodology in practice, these choices are often subjective, depending on the familiarity of the practitioner, the availability of software or the time for computational implementation. Nevertheless, we give here some rule of thumb advice on these choices based on our and others experience of using boosting in various problems.

Loss function

In classification, AdaBoost and LogitBoost are the most popular choices of loss functions with LogitBoost's performance slightly better than that of AdaBoost. Among margin-based generalizations of AdaBoost, LPBoost seems to be a good choice.[14]

In regression, L2Boosting and often also its shrunken version e-L2Boosting is used. Arguably, more robust loss functions and base procedures should be used,[34] but in practice, they are not in common use perhaps because of available software.

Base learner

At a high level, the conventional wisdom on the choice of base learner is that one should used a 'weak' learner or a simple procedure in terms of complexity. This allows the boosting methodology to adaptively build up the complexity of the boosting procedure tailored to the particular problem through the iterative fitting recipe. For example, in L2Boosting, if one starts with a strong learner such as projection pursuit, things can get worse even at the second step because boosting cannot correct the earlier iterations of overfitting.

The most used base learner with boosting (AdaBoost or its generalizations) in classification is CART or C4.5 (i.e., tree-based classifiers). Stumps have been used in many problems and slightly stronger learners are trees with a moderate number of nodes, say eight-node trees.

In regression, the coordinatewise descent version of L2Boosting with small steps and e-L2boosting have become popular for high-dimensional linear and additive modeling.

Early stopping rule

Performance on a test set or cross validation is usually the choice of early stopping if the goal is prediction. If the sample size is large (relative to the complexity of the fitting), a single test set is often sufficient. Alternatively, if the sample size is small (relative to the complexity of the fitting), the single test-set idea is not accurate and cross validation should be used. However, cross-validated prediction errors incur large variances and hence may not be reliable. A few alternatives exist based on model selection criteria such as AIC (or AIC_c), BIC or, gMDL. The idea is to use a possibly biased estimated prediction error with a smaller variance than the cross-validation error. Moreover, the computation cost may be many fold reduced in comparison to cross validation and thus, for very large data sets, these model-based prediction error estimates are also desirable due to computational savings.

Software in R

The R-package `mboost` provides many specific versions of the gradient descent boosting algorithm, including choices for stopping rules. In addition, `mboost` allows to fit boosting with a user-specific loss function L. A review of boosting including examples using `mboost` is given in Ref 35.

REFERENCES

1. Freund Y, Schapire R. Experiments with a new boosting algorithm. In: *Proceedings of the Thirteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann Publishers Inc; 1996, 148–156.
2. Kearns M, Valiant LG. Cryptographic limitations on learning Boolean formulae and finite automata. *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. New York: ACM Press; 1989, 433–444.
3. Schapire R. The strength of weak learnability. *Mach Learn* 1990, 5:197–227.
4. Breiman L. Arcing classifiers (with discussion). *Ann Stat* 1998, 26:801–849.

5. Breiman L. Prediction games and arcing algorithms. *Neural Comput* 1999, 11:1493–1517.
6. Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting (with discussion). *Ann Stat* 2000, 28:337–407.
7. Mason L, Baxter J, Bartlett P, Frean M. Functional gradient techniques for combining hypotheses. In: Smola A, Bartlett P, Schölkopf B, Schuurmans D, eds. *Advances in Large Margin Classifiers*. Cambridge: MIT Press; 2000.
8. Friedman J. Greedy function approximation: a gradient boosting machine. *Ann Stat* 2001, 29:1189–1232.
9. Schapire R, Freund Y, Bartlett P, Lee W. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann Stat* 1998, 26:1651–1686.
10. Rätsch G, Onoda T, Müller K. Soft margins for AdaBoost. *Mach Learn* 2001, 42:287–320.
11. Rosset S, Zhu J, Hastie T. Boosting as a regularized path to a maximum margin classifier. *J Mach Learn Res* 2004, 5:941–973.
12. Rätsch G, Warmuth M. Efficient margin maximizing with boosting. *J Mach Learn Res* 2005, 6:2131–2152.
13. Rudin X, Schapire RE, Daubechies I. Analysis of boosting algorithms using the smooth margin function. *Ann Stat* 2007, 35:2723–2768.
14. Warmuth M, Glocer K, Vishwanathan S. Entropy regularized LP Boost. In: Freund Y, Györfi L, Turán G, Zeugmann T, eds. *Advanced Lectures on Machine Learning*, *Lecture Notes in Computer Science*. NewYork: Springer; 2008b, 256–271.
15. Bühlmann P, Yu B. Boosting with the L_2 loss: regression and classification. *J Am Stat Assoc* 2003, 98:324–339.
16. Jiang W. Process consistency for AdaBoost (with discussion). *Ann Stat* 2004, 32:13–29 (disc. pp. 85–134).
17. Lugosi G, Vayatis N. On the Bayes-risk consistency of regularized boosting methods (with discussion). *Ann Stat* 2004, 32:30–55 (disc. pp. 85–134).
18. Zhang T, Yu B. Boosting with early stopping: convergence and consistency. *Ann Stat* 2005, 33:1538–1579.
19. Bühlmann P. Boosting for high-dimensional linear models. *Ann Stat* 2006, 34:559–583.
20. Bartlett P, Traskin M. AdaBoost is consistent. *J Mach Learn Res* 2007, 8:2347–2368.
21. Lutz R, Bühlmann P. Boosting for high-multivariate responses in high-dimensional linear regression. *Stat Sin* 2006, 16:471–494.
22. Lozano A, Kulkarni S, Schapire R. Convergence and consistency of regularized boosting algorithms with stationary β-mixing observations. In: Weiss Y, Schölkopf B, Platt J, eds. *Advances in Neural Information Processing Systems*, vol. 18. Boston: MIT Press; 2006, 819–826.
23. Tibshirani R. Regression shrinkage and selection via the Lasso. *J Roy Stat Soc, Ser B* 1996, 58:267–288.
24. Bickel P, Ritov Y, Zakai A. Some theory for generalized boosting algorithms. *J Mach Learn Res* 2006, 7:705–736.
25. Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. New York: Springer; 2001.
26. Efron B, Hastie T, Johnstone I, Tibshirani R. Least angle regression (with discussion). *Ann Stat* 2004, 32:407–451.
27. Tukey J. *Exploratory Data Analysis*. Reading: Addison-Wesley; 1977.
28. Zhao P, Yu B. Stagewise Lasso. *J Mach Learn Res* 2007, 8:2701–2726.
29. Bühlmann P, Yu B. Sparse boosting. *J Mach Learn Res* 2006, 7:1001–1024.
30. Zhang T. *Adaptive forward-backward greedy algorithm for sparse learning with linear models*. Technical report, Rutgers Statistics Department. NIPS, 2008.
31. Friedman JH, Popescu BE. Predictive learning via rule ensembles. *Annal Appl Stat* 2008, 2:916–954. *Gradient directed regularization*. Technical report, Stanford University.
32. Demiriz A, Bennett K, Shawe-Taylor J. Linear programming boosting via column generation. *J Mach Learn Res* 2002, 46:225–254.
33. Warmuth M, Glocer K, Rätsch G. (2008a), Boosting algorithms for maximizing the soft margin. In: Platt J, Koller D, Singer Y, Roweis S, eds. *Advances in Neural Information Processing Systems 20*. Boston: MIT Press; 2000, 1585–1592.
34. Lutz R, Kalisch M, Bühlmann P. Robustified L2 boosting. *Comput Stat Data Anal* 2008, 52:3331–3341.
35. Bühlmann P, Hothorn T. Boosting algorithms: regularization, prediction and model fitting (with discussion). *Stat Sci* 2007, 22:477–505.

