Bagging and Boosting Decision Trees

Andrew Nobel

November, 2021

Preface: Wisdom of Crowds

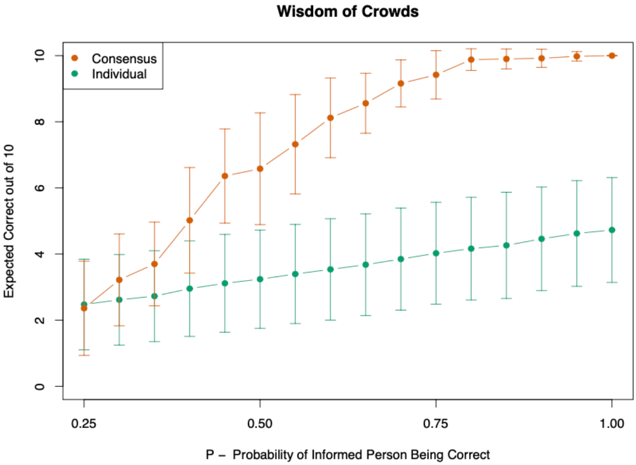
Idea (ESL): The collective knowledge of a diverse and independent body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting.

Example: Multiple choice exam

- 10 questions, 4 possible answers for each question
- 50 students take exam
- For each question random a set of 15 students have probability $p \geq .25$ of selecting the right answer
- Remaining students use random guessing ($p = .25$)

Task: Compare individual scores to the score achieved by majority vote

Multiple Choice Exam: 50 Replicates (ESL)



Bagging $=$ **B**ootstrap **Agg**regation

Bootstrap Resampling

Definition: Let $D_n = z_1, \ldots, z_n$ be a fixed data set. A *bootstrap sample* from D_n is a new, random data set

$$D_n^* = z_1^*, \ldots, z_n^*$$

where each z_i^* is drawn independently at random from $\{z_1, \ldots, z_n\}$

- Bootstrap sample D_n^* has same number of observations as D_n
- In general, D_n^* has repeated values – some z_i's chosen more than once

Another view: Elements $z_1^*, \ldots, z_n^*$ are independent draws from empirical distribution of D_n, which places mass $1/n$ at each original data point z_i

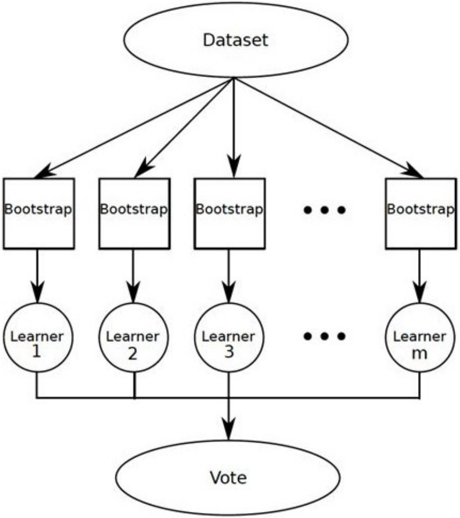
Bagging Regression Trees

Approach: Average regression trees produced from bootstrap samples

- Data set $D_n = (x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathbb{R}$
- Generate B bootstrap samples $D_n^{*(1)}, \ldots, D_n^{*(B)}$ from D_n
- For $b = 1, \ldots, B$ produce a regression tree $\hat{\varphi}^{*(b)}(x)$ from $D_n^{*(b)}$
- Bagged regression estimate is the average of the trees $\hat{\varphi}^{*(b)}(x)$

$$\hat{\varphi}^{\text{bag}}(x) = B^{-1} \sum_{b=1}^{B} \hat{\varphi}^{*(b)}(x)$$

Illustration of Bagging (kdnugets.com)



Bagging Decision Trees

Approach 1: Poll decision trees produced from bootstrap samples

- Data set $D_n = (x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \{0, 1\}$
- Generate B bootstrap samples $D_n^{*(1)}, \ldots, D_n^{*(B)}$ from D_n
- For $b = 1, \ldots, B$ produce a decision tree $\phi^{*(b)}$ from $D_n^{*(b)}$
- Define bagged classification rule by polling the decision trees $\phi^{*(b)}$

$$\hat{\phi}^{\text{bag}}(x) = \text{majority vote}\{\phi^{*(1)}(x), \ldots, \phi^{*(B)}(x)\}$$

Bagging Decision Trees

Approach 2: Average conditional probability estimates

- Data set $D_n = (x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \{0, 1\}$
- Generate B bootstrap samples $D_n^{*(1)}, \ldots, D_n^{*(B)}$ from D_n
- For $b = 1, \ldots, B$ use $D_n^{*(b)}$ to get tree-based estimate $\hat{\eta}^{*(b)}$ of η
- Produce bagged estimate of $\eta(x)$ by averaging estimates $\hat{\eta}^{*(b)}(x)$

$$\hat{\eta}^{\text{bag}}(x) = B^{-1} \sum_{b=1}^{B} \hat{\eta}^{*(b)}(x)$$

- Define bagged classification rule $\hat{\phi}^{\text{bag}}(x) = \mathbb{I}(\hat{\eta}^{\text{bag}}(x) \geq 1/2)$

Bagging for Trees

Pluses

- Reduces instability of decision and regression trees
- Averaging reduces variance of bagged regression estimates
- Bagging gives smoother estimates than individual trees

Minuses

- Bagged trees are not easily interpretable (bagged trees are not a tree)
- If decision trees are a poor fit, bagging may not help
- Increased computation

Bagging in General

Bootstrap aggregation can be applied to any classification or regression procedure $\varphi_n(x : D_n)$

- Procedure $\varphi_n(x : D_n)$ called “base learner”, usually simple
- Bagging can improve the performance of non-linear base learners

Bagging effectively increases the set of models fit by the base learner, but the increase may be modest

Boosting

Boosting for Classification

Ingredients

- Data set $D_n = (x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \{-1, +1\}$
- Weak learner $L(x : D_n, \text{w})$ that produces a simple classification rule from data D_n *and weights* w(i) *for individual data points*

Weak learner may perform only marginally better than random guessing

Ex: Decision stumps, single split decision tree (root with two children)

- Popular choice of weak learner for boosting
- Assign class labels to two terminal regions using weighted majority vote

Idea Behind Boosting

Question: How to turn a weak learner into a really good classification rule?

Approach: Apply weak learner to the data multiple times, in stages

- At each stage, give more weight to data points where the weak learner made mistakes at previous stages
- Combine rules from different stages via a weighted sum
- Use the sign of the weighted sum to classify new data

Input to Boosting: Data set D_n and weak learner $L(x : D_n, \mathrm{w})$

Overview of AdaBoost (ESL)

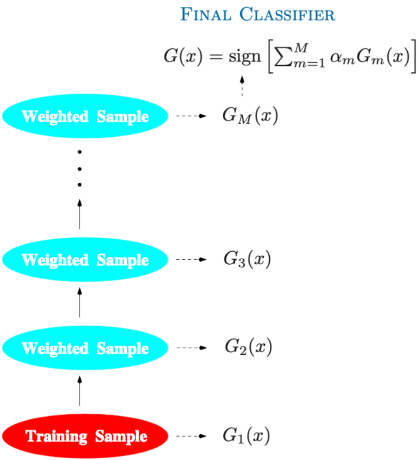


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

AdaBoost Algorithm

1. **Initialize:** Sample weight vector $\mathsf{w}_1(i) = 1/n$ for $i = 1, \ldots, n$

2. **Iterate:** For $m = 1, \ldots, M$ do the following

   a. Fit rule $g_m(x) = L(x : D_n, \mathsf{w}_m)$ to data D_n with sample weights w_m

   b. Assess the weighted empirical risk of the rule g_m

$$r_m \; = \; \sum_{i=1}^{n} \mathsf{w}_m(i) \, \mathbb{I}(g_m(x_i) \neq y_i) \, / \, \sum_{i=1}^{n} \mathsf{w}_m(i)$$

   c. Compute coefficient $\alpha_m = \log[(1 - r_m)/r_m]$ for weak learner g_m

   d. Update weights: $\mathsf{w}_{m+1}(i) = \mathsf{w}_m(i) \exp\{\alpha_m \, \mathbb{I}(g_m(x_i) \neq y_i)\}$

3. **Output:** Aggregate rule $\hat{\phi}^{\text{boost}}(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m g_m(x)\right]$

AdaBoost Algorithm, cont.

Note: The coefficient α_m of the rule g_m is given by

$$\alpha_m = \log \frac{(1 - r_m)}{r_m} = \begin{cases} \text{positive} & \text{if } r_m < 1/2 \\ \text{negative} & \text{if } r_m > 1/2 \end{cases}$$

- If $r_m > 1/2$ then $-g_m$ performs better than $+g_m$ so coefficient $\alpha_m < 0$
- Weights at misclassified points increased if $\alpha_m > 0$
- Weights at misclassified points decreased if $\alpha_m < 0$

Boosting Example: Simulated Data (ESL)

Model: Let $X \sim \mathcal{N}_{10}(0, I)$ and $Y = \text{sign}\left[\, ||X||^2 - \text{med}(||X||^2) \,\right]$

- Prior probabilities for classes $+1, -1$ are each $1/2$
- As $Y = h(X)$, Bayes risk $R^* = 0$ and Bayes rule $\phi^*(x) = h(x)$

Observations: training set D_n with $n =$ 2K; test set D_m with $m =$10K

Methods

- Base learner: decision stump, tree with root and two children
- Standard classification tree
- Boosted base learner (decision stump)

Boosting Example, Results

Test set error rates

- Random guessing: 50%
- Base learner: 46%
- Standard classification tree: 25%
- Boosted base learner ($M = 400$ iterations): 6%

Boosting on Simulated Data (ESL)

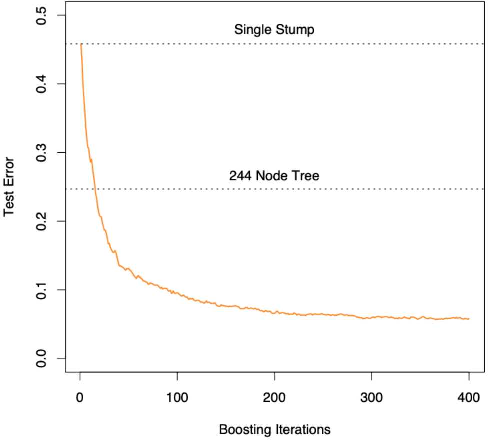


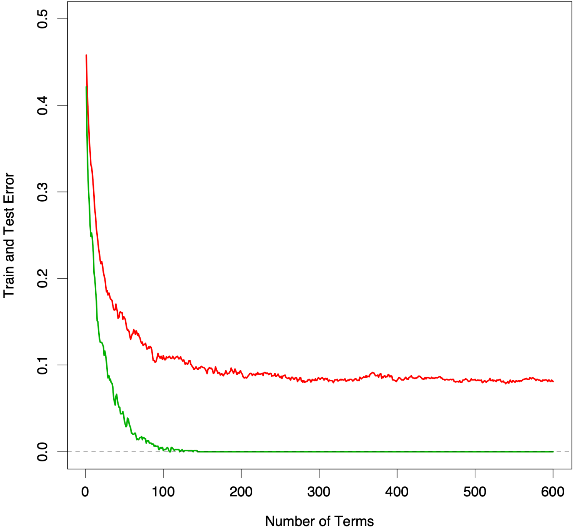
FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

Boosting in Practice

In many cases, as number of iterations M increases

- Training error of $\hat{\phi}^{\text{boost}}$ goes to zero
- Test error of $\hat{\phi}^{\text{boost}}$ decreases, then flattens out

Boosting: Training and Test Error on Simulated Data (Hastie slides)



Additive Models for Classification

Building Additive Models for Classification

Given

- Family $\mathcal{G}$ of simple classification rules $g : \mathcal{X} \to \{\pm 1\}$
- Data $D_n = (x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \{\pm 1\}$

Task: Construct a classification rule of the form $\hat{\phi}(x) = \text{sign}(\hat{f}(x))$ where

$$\hat{f}(x) = \sum_{m=1}^{M} \beta_m g_m(x) \text{ with } g_1, \ldots, g_m \in \mathcal{G}$$

Idea: Construct $\hat{f}$ from D_n in a greedy fashion, one term at a time, using the *exponential loss function*

$$\ell(y, h(x)) = \exp(-y\, h(x))$$

Forward Stagewise Additive Modeling

1. **Initialize:** $f_0(x) = 0$

2. **Iterate:** For $m = 1, \ldots, M$ do the following

   a. Find weight $\beta_m \in \mathbb{R}$ and rule $g_m \in \mathcal{G}$ yielding best single term improvement of current additive expansion

   $$(\beta_m, g_m) \;=\; \underset{\beta, g}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(y_i, f_{m-1}(x_i) + \beta g(x_i))$$

   b. Update additive expansion $f_m(x) = f_{m-1}(x) + \beta_m g_m(x)$

3. **Output:** Final expansion $f_M(x) = \sum_{j=1}^{M} \beta_j g_j(x)$

Boosting vs. Forward Stagewise Additive Modeling (FSAM)

Fact: AdaBoost is a version of FSAM with exponential loss

- Family $\mathcal{G}$ is set of decision stumps
- Step 2a of FSAM corresponds to fitting decision stumps to weighted data
- AdaBoost driven by minimization of exponential loss

AdaBoost Misclassification and Exponential Loss (ESL)

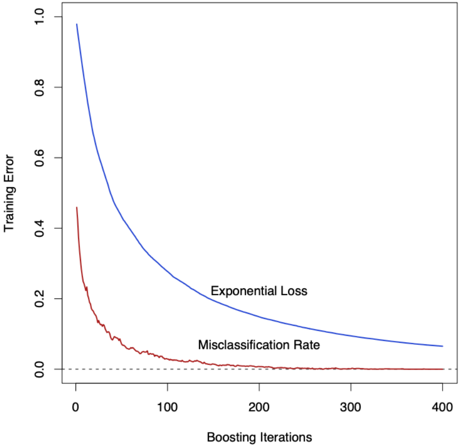


FIGURE 10.3. *Simulated data, boosting with stumps: misclassification error rate on the training set, and average exponential loss:* $(1/N)\sum_{i=1}^{N}\exp(-y_i f(x_i))$*. After about* 250 *iterations, the misclassification error is zero, while the exponential loss continues to decrease.*