

Bagging, boosting and random forests

Geir Storvik and Aliaksandr Hubin

May 12, 2022

- Assume $Y \in \{0, 1, \dots, K - 1\}$
- Following ideas from **regression trees**

$$\hat{p}_k(\mathbf{x}) = \sum_{j=1}^J P_{j,k} I(\mathbf{x} \in R_j)$$

where $P_{j,k}$ is the (estimated) probability for class k in region j .

- Given regions R_j :

$$\hat{P}_{j,k} = \frac{1}{n_j} \sum_{i: \mathbf{x}_i \in R_j} I(y_i = k)$$

with $n_j = \sum_{i=1}^n I(\mathbf{x}_i \in R_j)$

- Also need to define R_j based on some **fitness** criterion
- Problem: Trees are **very** variable!
- Illustration: `phoneme_tree.R`

- Many models can fit data equally well
- Also a specific model can give variable fits based on different data sets
- Idea: **Combine** different predictors
 - **Bagging**: Combine fit from many different data sets
 - **Boosting**: Similar, but sequential updates and simpler individual models
 - **Random forest**: Combine different models based on subsets of covariates

- Assume training set $\mathbf{Z} = \{(\tilde{\mathbf{x}}_1, y_1), \dots, (\tilde{\mathbf{x}}_n, y_n)\}$
- $C(\mathbf{x})$ classifier based on $\mathbf{Z}$

- Assume training set $\mathbf{Z} = \{(\tilde{\mathbf{x}}_1, y_1), \dots, (\tilde{\mathbf{x}}_n, y_n)\}$
- $C(\mathbf{x})$ classifier based on $\mathbf{Z}$
- Bootstrap sample $\mathbf{Z}_1^* = \{(\tilde{\mathbf{x}}_1^*, y_1^*), \dots, (\tilde{\mathbf{x}}_n^*, y_n^*)\}$
- $C_1^*(\mathbf{x})$ classifier based on $\mathbf{Z}_1^*$
- Repeat, obtain $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$

- Assume training set $\mathbf{Z} = \{(\tilde{\mathbf{x}}_1, y_1), \dots, (\tilde{\mathbf{x}}_n, y_n)\}$
- $C(\mathbf{x})$ classifier based on $\mathbf{Z}$
- Bootstrap sample $\mathbf{Z}_1^* = \{(\tilde{\mathbf{x}}_1^*, y_1^*), \dots, (\tilde{\mathbf{x}}_n^*, y_n^*)\}$
- $C_1^*(\mathbf{x})$ classifier based on $\mathbf{Z}_1^*$
- Repeat, obtain $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$
- **Combine** (binary outcome)

$$C_{bag}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B C_b^*(\mathbf{x})$$

Classification: $\hat{y} = 1$ if $C_{bag}(\mathbf{x}) > 0.5$.

- Assume training set $Z = \{(\tilde{\mathbf{x}}_1, y_1), \dots, (\tilde{\mathbf{x}}_n, y_n)\}$
- $C(\mathbf{x})$ classifier based on Z
- Bootstrap sample $Z_1^* = \{(\tilde{\mathbf{x}}_1^*, y_1^*), \dots, (\tilde{\mathbf{x}}_n^*, y_n^*)\}$
- $C_b^*(\mathbf{x})$ classifier based on Z_1^*
- Repeat, obtain $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$

- Assume training set $Z = \{(\tilde{\mathbf{x}}_1, y_1), \dots, (\tilde{\mathbf{x}}_n, y_n)\}$
- $C(\mathbf{x})$ classifier based on Z
- Bootstrap sample $Z_1^* = \{(\tilde{\mathbf{x}}_1^*, y_1^*), \dots, (\tilde{\mathbf{x}}_n^*, y_n^*)\}$
- $C_b^*(\mathbf{x})$ classifier based on Z_1^*
- Repeat, obtain $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$
- **Combine** through **majority vote**
 $C_{bag}(\mathbf{x})$ is class with most "votes"

- Assume training set $Z = \{(\tilde{\mathbf{x}}_1, y_1), \dots, (\tilde{\mathbf{x}}_n, y_n)\}$
- $p_k(\mathbf{x})$ is estimate of $\Pr(Y = k|\mathbf{x})$ based on Z
- Bootstrap sample $Z_b^* = \{(\tilde{\mathbf{x}}_1^*, y_1^*), \dots, (\tilde{\mathbf{x}}_n^*, y_n^*)\}$
- $p_k^{b*}(\mathbf{x})$ classification probabilities based on Z_b^*
- **Combine** (binary outcome)

$$p_{k,bag}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B p_k^{b*}(\mathbf{x})$$

- Assume training set $Z = \{(\tilde{\mathbf{x}}_1, y_1), \dots, (\tilde{\mathbf{x}}_n, y_n)\}$
- $\mu(\mathbf{x})$ is estimate of $E(Y|\mathbf{x})$ based on Z
- Bootstrap sample $Z_b^* = \{(\tilde{\mathbf{x}}_1^*, y_1^*), \dots, (\tilde{\mathbf{x}}_n^*, y_n^*)\}$
- $\mu^{b*}(\mathbf{x})$ ~~classification probabilities~~ based on Z_b^*
- **Combine** (binary outcome)

$$\mu_{k,bag}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B \mu_k^{b*}(\mathbf{x})$$

estimates of means

- Can in principle combine results from **any** classifier/regression model
 - Highest benefit when original method is highly variable (e.g trees)
- Will often improve predictive ability but have greater difficulty in interpreting the results
- Can also be used for **regression** problems

- Division of data into training, validation, testing
 - Validation: Specify complexity of method/model selection

- Division of data into **training, validation, testing**
 - Validation: Specify **complexity** of method/model selection
- Bagging: Bootstrap samples from training set
- **Out-of-bag**: Those observations **not** in bootstrap sample can be used for validation
- Can also be used as an alternative to cross-validation

- Bagging/boosting: Putting different weights on samples
- **Alternative idea:** Consider different **subsets of covariates**

- Bagging/boosting: Putting different weights on samples
- **Alternative idea:** Consider different **subsets of covariates**
 - When **trees** are used on each subset: **Random forest**
 - At every tree node it selects randomly among a small subset $q \ll p$ of covariates to split
 - Tuning parameters is **size of subset** and **number of trees**

- Bagging/boosting: Putting different weights on samples
- **Alternative idea:** Consider different **subsets of covariates**
 - When **trees** are used on each subset: **Random forest**
 - At every tree node it selects randomly among a small subset $q \ll p$ of covariates to split
 - Tuning parameters is **size of subset** and **number of trees**
- The **global error** converges to a lower bound when B increases
- Increasing B does not cause overfitting
- Alternative definition of **Random forest**
 - Any classifier obtained as a combination of a set of classification trees
 - Includes bagging and boosting
- `phoneme_tree.R`

- When building tree b :
 - A new bootstrap sample is selected
 - A new sample of q covariates are selected
 - Out-of-bag can also be used to specify q
- Much faster because every single tree is based on fewer variables
- Can use parallel computing

- mean decrease of accuracy in predictions on the out of bag samples when a given covariate is excluded from the model, or
- total decrease in node impurity that results from splits over that variable, averaged over all trees

- Bagging: $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **parallel**, all samples made similarly
- **Boosting** : $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **sequence**

- Bagging: $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **parallel**, all samples made similarly
- **Boosting** : $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **sequence**
 - Data to be used for $C_b^*(\mathbf{x})$ depend on $C_1^*(\mathbf{x}), \dots, C_{b-1}^*(\mathbf{x})$

- Bagging: $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **parallel**, all samples made similarly
- **Boosting** : $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **sequence**
 - Data to be used for $C_b^*(\mathbf{x})$ depend on $C_1^*(\mathbf{x}), \dots, C_{b-1}^*(\mathbf{x})$
 - Assign **greater weight** to observations predicted **poorly** in the earlier stages

- Bagging: $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **parallel**, all samples made similarly
- **Boosting** : $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **sequence**
 - Data to be used for $C_b^*(\mathbf{x})$ depend on $C_1^*(\mathbf{x}), \dots, C_{b-1}^*(\mathbf{x})$
 - Assign **greater weight** to observations predicted **poorly** in the earlier stages
 - Each $C_b^*(\mathbf{x})$ are **weak** learners (typically **small trees**)

- Bagging: $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **parallel**, all samples made similarly
- **Boosting** : $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **sequence**
 - Data to be used for $C_b^*(\mathbf{x})$ depend on $C_1^*(\mathbf{x}), \dots, C_{b-1}^*(\mathbf{x})$
 - Assign **greater weight** to observations predicted **poorly** in the earlier stages
 - Each $C_b^*(\mathbf{x})$ are **weak** learners (typically **small trees**)
 - **Slow learning**: Shrinkage parameter λ small

$$C_{boost}^b(\mathbf{x}) = C_{boost}^{b-1}(\mathbf{x}) + \lambda C_b^*(\mathbf{x})$$

- Bagging: $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **parallel**, all samples made similarly
- **Boosting** : $C_1^*(\mathbf{x}), \dots, C_B^*(\mathbf{x})$ made in **sequence**
 - Data to be used for $C_b^*(\mathbf{x})$ depend on $C_1^*(\mathbf{x}), \dots, C_{b-1}^*(\mathbf{x})$
 - Assign **greater weight** to observations predicted **poorly** in the earlier stages
 - Each $C_b^*(\mathbf{x})$ are **weak** learners (typically **small trees**)
 - **Slow learning**: Shrinkage parameter λ small

$$C_{boost}^b(\mathbf{x}) = C_{boost}^{b-1}(\mathbf{x}) + \lambda C_b^*(\mathbf{x})$$

- Number of iterations b is a **tuning** parameter

Boosting on regression trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \dots, B$, repeat:
 - (a) Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r) .
 - (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \quad (8.10)$$

- (c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \quad (8.11)$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x). \quad (8.12)$$

Boosting on classification trees

Algorithm 5.2 Boosting (AdaBoost)

1. Initialize weights $w_i = 1/n$, $i = 1, 2, \dots, n$.
2. Cycle for $b = 1, \dots, B$:
 - a. Fit a classification model $C_b(x)$ to the training set, with target values 0 or 1, by weighting the observations by w_i .
 - b. Obtain:

$$\text{err}_b \leftarrow \frac{\sum_{i=1}^N w_i I(y_i \neq C_b(x_i))}{\sum_{i=1}^N w_i},$$

$$\alpha_b \leftarrow \log \frac{1 - \text{err}_b}{\text{err}_b}.$$

- c. Assign the new weights:

$$w_i \leftarrow w_i \exp\{\alpha_b I(y_i \neq C_b(x_i))\}, \quad i = 1, 2, \dots, n.$$

3. The new classifier is:

$$C(x) = \begin{cases} 1 & \text{if } \frac{\sum_{b=1}^B \alpha_b C_b(x)}{\sum_{b=1}^B \alpha_b} > \frac{1}{2}, \\ 0 & \text{otherwise} \end{cases}$$

Gradient tree boosting

1. Initialize $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$.

2. For $m = 1$ to M :

(a) For $i = 1, 2, \dots, N$ compute

$$r_{im} = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}.$$

(b) Fit a regression tree to the targets r_{im} giving terminal regions

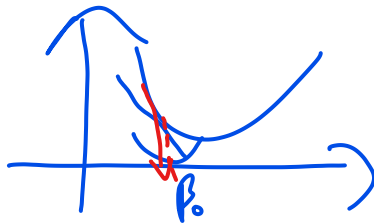
$$R_{jm}, j = 1, 2, \dots, J_m.$$

(c) For $j = 1, 2, \dots, J_m$ compute

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma).$$

(d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.

3. Output $\hat{f}(x) = f_M(x)$.



Possible to use only a sub-sample of the data! Then stochastic gradient boosting is derived!