CS 337, Fall 2023
Bagging, Random Forests

Scribes: Jennisha Agarwal*, Varada Mahanth Naidu*, Veeresh B Patil*,
Pulkit Goyal, Anshika Raman, Injarapu Nikhil
Edited by: Saurabh Kumar

October 17, 2023

1 Recall

Let's recall from our earlier classes, the Bias/Variance decomposition for regression:

$$\begin{aligned} E[(y - h(x))^2] &= (\hbar(x) - f(x))^2 + E[(\hbar(x) - h(x))^2] + \sigma^2 \\ &= (BIAS)^2 + VARIANCE + NOISE \end{aligned} \tag{1}$$

where,
x : test point,
y : label,
h() : predictor function,
$\bar{h}$: mean predictor,
f() : true predictor

2 Bagging

2.1 The Definition

Bagging is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also *reduces variance* and helps to *avoid overfitting*. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

Well, this is how the internet defines it, but let's try to understand it in a better way:
Consider constructing **m** independent datasets $D_1, D_2, ..., D_m$, all sampled uniformly from the same underlying distribution $\boldsymbol{P}$. We can train a predictor for each of the **m** datasets, let's say $h(x; D_i)$.

*Larger credit to these scribes for the final notes.

Given the predictors for each of the **m** independent datasets, we can derive an average predictor given by:

$$Avg.Predictor = \frac{1}{m}\sum_{i=1}^{m} h(x; D_i)$$

Yuppp... that's right.. exactly what the name says!
Q. But, what happens to the bias/variance when using an average predictor ?
(Since ultimately all our goal boils down to this simple idea)

- *BIAS:* remains unchanged
  taking the expectation (average) over multiple independent datasets does not affect the expected value
  $E_{D_1,\ldots,D_m}[\frac{1}{m}\sum_i h(x; D_i)] = \frac{1}{m}\sum_i E_{D_i}[h(x; D_i)] = E_D[h(x; D)]$
- *VARIANCE:* **Reduces !**
  since, $Var_{D_1,\ldots,D_m}[\frac{1}{m}\sum_i h(x; D_i)] = \frac{1}{m^2}\sum_i Var_{D_i}[h(x; D_i)]$

$$= \frac{1}{m} Var_D[h(x; D)]$$

So... there you go, a better solution indeed with a reduced variance !!
BUT..... (always a but isn't it)

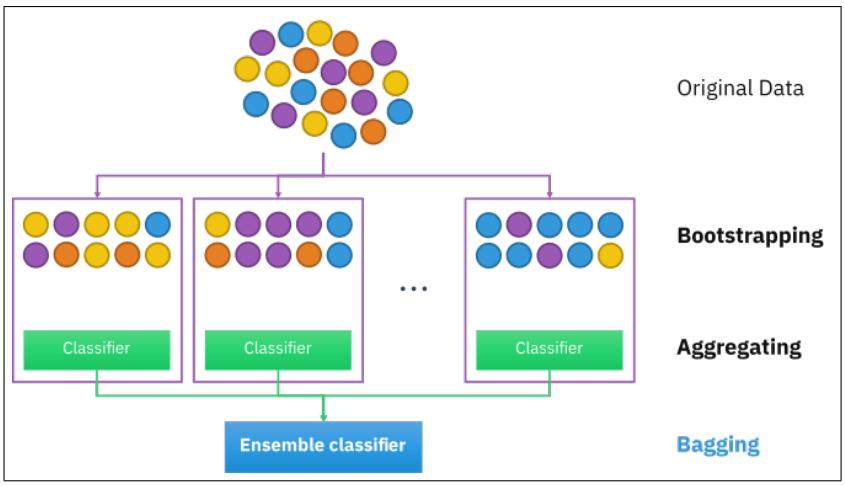
2.2 The Catch and The Solution

Catch: Under normal circumstances of training a standard model, we do not have **m** datasets required to generate the supposedly better average predictor. But instead, we only have 1 (let's call it **D**).

Solution: BOOTSTRAP !
(Creating multiple datasets from the single D, using *sampling with replacement*)

Thus, here we find the motivation for our bagging technique:
***Bootstrap AGGregatING* = "BAGGING"**

Figure 1: A Breakdown of Bagging



2.3 Motivation for Bagging

Bagging, a short form for Bootstrap Aggregation, is motivated by the need to enhance single-model performance using just one dataset. It achieves this by:

1. **Bootstrap Sampling:** Creating multiple datasets by repeatedly sampling from the original dataset with replacement.
2. **Model Diversity:** Training separate models on each of these bootstrap datasets to capture diverse data patterns.
3. **Ensemble Averaging:** Aggregating predictions from these models, reducing variance and improving overall model stability.

Bagging thus addresses the bias-variance trade-off, making models more robust and less prone to overfitting when only one dataset is available. Figure 1 shows the pipeline used in bagging.

2.4 The Algorithm

1. *Given:* Consider a dataset D with n training samples.
2. *Bootstrap:* Create **m** different datasets, each of size **n**, by sampling from D with replacement. (an element may be present multiple times or 0 times in a given dataset)
3. *Aggregating:* Average the predictors (for regression), and consider majority vote across predictors (for classification), from models trained individually on each of the m datasets.

> ***NOTE:*** Since the **m** datasets derived by the *Bootstrap* method are not really independent, we do not get the **1/m** variance reduction as derived before. But, we can show that if the sample predictions have a variance of σ^2 and correlation of ρ, the variance of the averaged predictor works out to be: $Var[\frac{1}{m}\sum_i h(x; D_i)] = \frac{1}{m}(1-\rho)\sigma^2 + \rho\sigma^2$

2.5 OOB samples

Each of the m datasets derived via the Bootstrap method is referred to as **Bootstrap Samples** (obviously...)

A single Bootstrap sample contains roughly *63.2%* of the observed training points in D. (not obviously...)

> ***Why?:*** If we sample from a uniform distribution, the probability of a single training example, in a dataset of size n, not being drawn as a Bootstrap sample datapoint is -
> $\mathbf{P}$(not drawn) $= (1 - \frac{1}{n})^n$ ... (n replaced trials to pick n bootstrap sample elements; element to be chosen in none of the trials!)
> as and when $n \to \infty$, the P(not drawn) $\to \frac{1}{e} \approx 0.368$
>
> P(drawn) = 0.632

The remaining instances in D that do not appear in a given Bootstrap sample are called as Out-of-Bag (OOB) samples.

OOB samples can be used to derive an easy / bagged estimate of test error:

- ✓ for each i^{th} training example, predict the labels using all models for whose bootstrap samples, the i^{th} training example is OOB.
- ✓ aggregate the predicted values and compute the error for the i^{th} training example.
- ✓ average the errors across all n training examples to obtain the estimated test error.

This process can thus be used for testing predictions without using an exquisite validation set.

3 Random Forests **(also introduced earlier)**

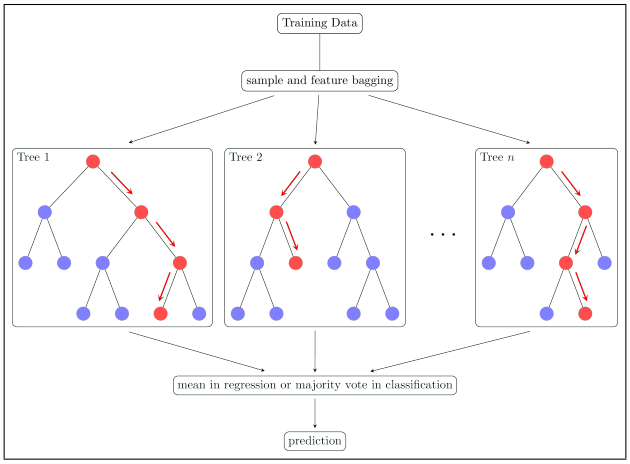
Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random Forest's ability to create diverse trees through bootstrapping and feature subset tweaking, and its robustness against overfitting, makes it a powerful algorithm for various machine learning tasks. It is known for its excellent performance, adaptability, and robustness across a wide range of applications.

To put it in short:
Random Forest = Decision Trees + Bagging + (an additional tweak to further de-correlate the trees)

How it works?

- Create Bootstrap samples
- Train a decision tree for each bootstrap sample
- *Tweak:* for **every** split in **every** Decision Tree, only consider a subset of features - if there are d features, then we use a **random** subset of size k (where typically $k \approx \sqrt{d}$) at each split for every Decision tree.

Figure 2: A Random Forest



4 Bagging vs Boosting

4.1 Similarities Between Bagging and Boosting

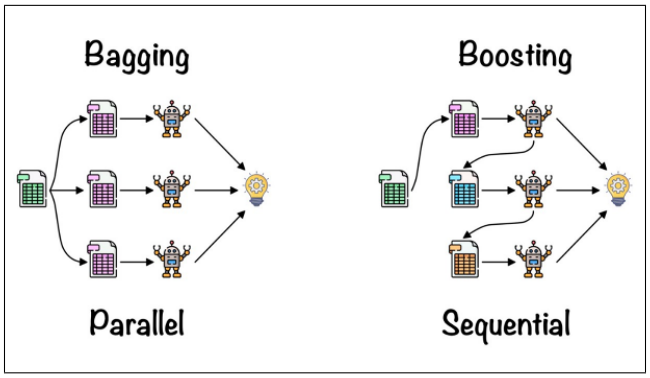
Bagging and Boosting, both being the commonly used methods, have a universal similarity of being classified as ensemble methods. Here we will explain the similarities between them.

1. Both are ensemble methods to get N learners from 1 learner.
2. Both generate several training data sets by random sampling.
3. Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
4. Both are good at reducing variance and provide higher stability.

4.2 Differences Between Bagging and Boosting

S. No.	**Bagging**	**Boosting**
1	The simplest way of combining predictions that belong to the same type	A way of combining predictions that belong to the different types
2	Aim to decrease variance, not bias	Aim to decrease bias, not variance
3	Each model receives equal weight	Models are weighted according to their performance
4	Each model is built independently	New models are influenced by the performance of previously built models
5	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset	Every new subset contains the elements that were misclassified by previous models
6	Bagging tries to solve the over-fitting problem	Boosting tries to reduce bias
7	If the classifier is unstable (high variance), then apply bagging	If the classifier is stable and simple (high bias) the apply boosting
8	**In this base classifiers are trained parallelly**	**In this base classifiers are trained sequentially**

Figure 3: Bagging vs Boosting



5 k-fold Cross-Validation

Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. We can also say that it is a technique to check how a statistical model generalizes to an independent dataset.

K-fold cross-validation is a technique in machine learning for assessing the performance of a predictive model. It is particularly useful when you have a limited amount of data and want to make the most of it. The main idea behind K-fold cross-validation is to divide your dataset into k- subsets or "k-folds" and systematically train and test your model multiple times.

Note that k-fold cross-validation is to evaluate the model design, not a particular training. Because you re-trained the model of the same design with different training sets.

Figure 4 visualizes this technique for **k=5**.

The general procedure is as follows:

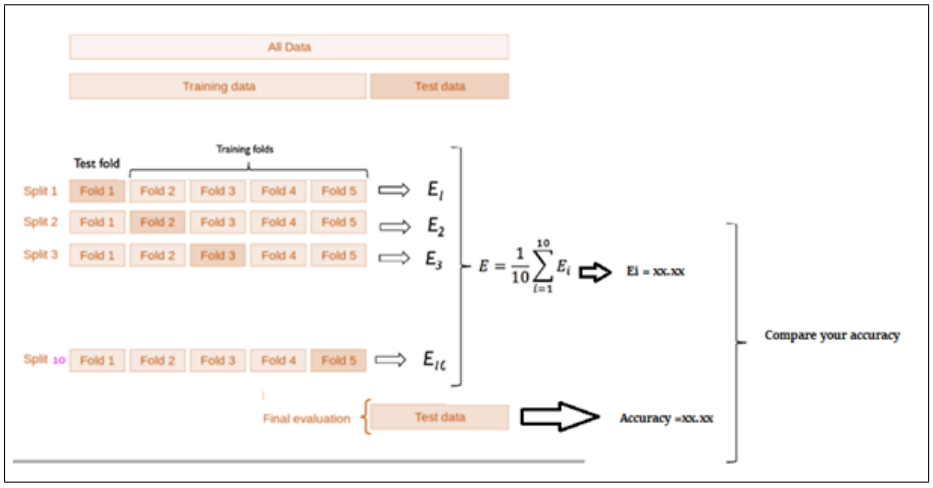
- Shuffle the dataset and split into k number of subsamples
- In the first iteration, the first subset serves as the test dataset, while all the remaining subsets are treated as the training data
- Train the model with the training data and evaluate it using the test subset. Keep the evaluation score or error rate, and get rid of the model
- Now, in the next iteration, select a different subset as the test data set, and make everything else part of the training data
- Re-train the model with the training data and test it using the new test data set, keep the evaluation score and discard the model
- Continue iterating the above k times. Each data subsamples will be used in each iteration until all data is considered. You will end up with a k number of evaluation scores

The total error rate is the average of all these individual evaluation scores. After performing K-fold cross validation, you can choose the best model or set of hyperparameters based on the aggregated performance scores and then retrain the model on the entire dataset for production use.

Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.

Note - If k = n (the data size), it is called **"Leave One Out Cross Validation"**. This is computationally very expensive and can be used for very small datasets only.

Figure 4: k-fold Cross-Validation for k=5



K-fold cross-validation helps in several ways:

- It provides a more robust estimate of model performance because it assesses the model on different subsets of the data
- It helps identify issues such as overfitting. If a model performs very well on the training data but poorly on the test data in multiple iterations, it's a sign of overfitting
- It maximizes the use of available data, which is especially important when you have a limited dataset

5.1 Types of Cross Validation

- K-fold cross-validation
- Hold-out cross-validation
- Stratified k-fold cross-validation
- Leave-p-out cross-validation
- **Leave-one-out cross-validation** (Specific case when k = n for k-fold cross-validation)
- Monte Carlo (shuffle-split)
- Time series (rolling cross-validation)