

# Ensemble methods

Fraida Fund

## Contents

Ensemble methods . . . . .	2
Recap: decision trees . . . . .	2
Ensemble methods - the idea . . . . .	2
Ensemble methods - types (1) . . . . .	2
Ensemble methods - types (2) . . . . .	2
Bagging . . . . .	2
Bagging - background . . . . .	2
Bootstrapping . . . . .	2
Bootstrap aggregation . . . . .	2
Bagging trees . . . . .	2
Correlated trees . . . . .	3
Random forests . . . . .	3
Bagged trees illustration . . . . .	3
A note on computation . . . . .	3
Boosting . . . . .	4
Boosting - training . . . . .	4
AdaBoost (Adaptive Boosting) . . . . .	4
AdaBoost algorithm . . . . .	4
AdaBoost algorithm (inner loop) . . . . .	4
AdaBoost algorithm (final step) . . . . .	4
Gradient Boosting . . . . .	4
Summary of (selected) ensemble methods . . . . .	5

**Math prerequisites for this lecture:** None

## Ensemble methods

### Recap: decision trees

- Let trees grow deep - low bias, high variance
- Don't let trees get deep: low variance, high bias

### Ensemble methods - the idea

Combine multiple **weak learners** - having either high bias or high variance - to create an **ensemble** with better prediction

### Ensemble methods - types (1)

- Combine multiple learners with high **variance** in a way that reduces their variance
- Combine multiple learners with high **bias** in a way that reduces their bias

### Ensemble methods - types (2)

- **Parallel:** build base estimators *independently* and then average their predictions. Combined estimator is usually better than any single base estimator because its *variance* is reduced.
- **Sequential:** (boosting) build base estimators *sequentially* and each one tries to reduce the *bias* of the combined estimator.

## Bagging

### Bagging - background

- Designed for, and most often applied to, decision trees
- Name comes from **bootstrap aggregation**

### Bootstrapping

- Basic idea: Sampling **with replacement**
- Each “bootstrap training set” is *same size* as full training set, and is created by sampling with replacement
- Some samples will appear more than once, some samples not at all

### Bootstrap aggregation

- Create multiple versions  $1, \dots, B$  of training set with bootstrap
- Independently train a model on each bootstrap training set: calculate  $\hat{f}_1(x) \dots, \hat{f}_B(x)$
- Combine output of models by voting (classification) or averaging (regression):

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(x)$$

### Bagging trees

- Construct  $B$  trees using  $B$  bootstrapped training sets.
- Let the trees grow deep, no pruning.
- Each individual tree has low bias, high variance.
- Average the prediction of the trees to reduce variance.

## Correlated trees

Problem: trees produced by bagging are highly correlated.

- Imagine there is one feature that is strong predictor, several moderate predictors
- Most/all trees will split on this feature
- Averaging correlated quantities does not reduce variance as much.

## Random forests

Grow many decorrelated trees:

- **Bootstrap**: grow each tree with bootstrap resampled data set.
- **Split-variable randomization**: Force each split to consider *only* a subset of  $m$  of the  $p$  predictors.

Typically  $m = \frac{p}{3}$  but this should be considered a tuning parameter.

## Bagged trees illustration

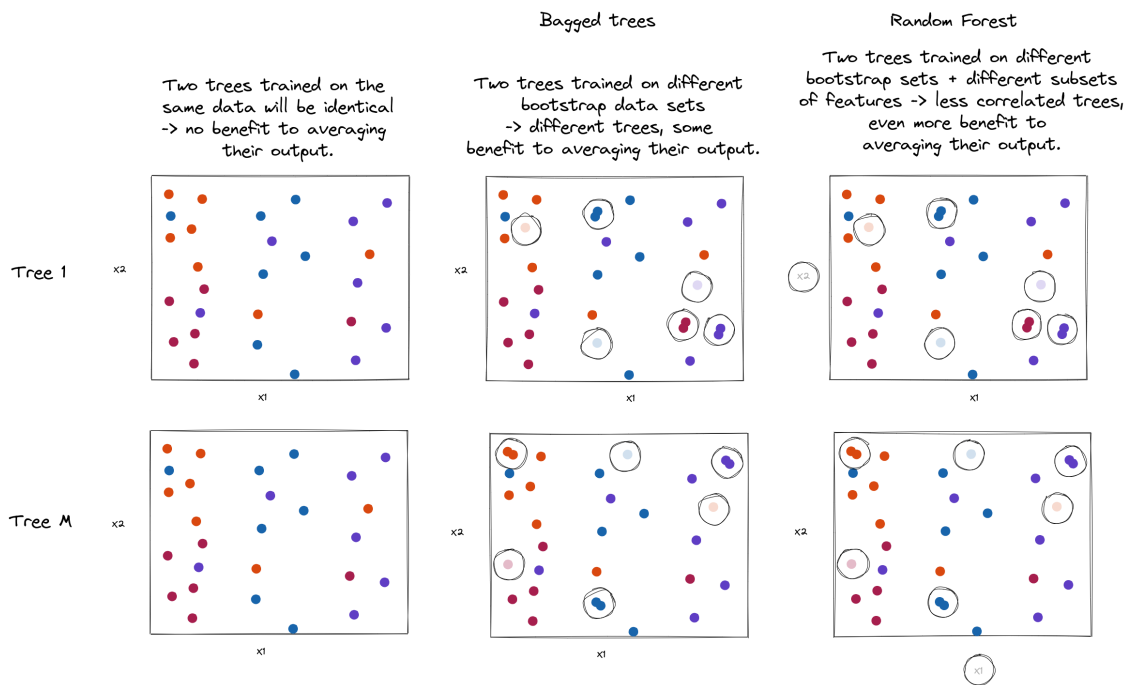


Figure 1: Identical data, bootstrapped data, and bootstrapped data with split variable randomization.

## A note on computation

- Bagged trees and random forests can be fitted in parallel on many cores!
- Each tree is built independently of the others

## Boosting

### Boosting - training

**Iteratively** build a succession of models:

- Train a weak model. Typically a very shallow tree.
- In training set for  $b$ th model, focus on errors made by  $b - 1$ th model.
- Use (weighted) model output
- Reduces bias *and* variance!

### AdaBoost (Adaptive Boosting)

Adjust *weights* so that each successive model focuses on more “difficult” samples.

Consider classification problem, where sign of model output gives estimated class label and magnitude gives confidence in label.

### AdaBoost algorithm

1. Let  $w_i = \frac{1}{N}$  for all  $i$  in training set.
2. For  $m = 1, \dots, M$ , repeat:

#### AdaBoost algorithm (inner loop)

- Fit a tree  $\hat{f}^m$ , compute weighted error  $err_m$  using weights on training samples  $w_i$ :

$$err_m = \frac{\sum_{i=1}^N w_i 1(y_i \neq \hat{f}^m(x_i))}{\sum_{i=1}^N w_i}$$

- Compute coefficient  $\alpha_m = \log\left(\frac{1-err_m}{err_m}\right)$
- Update weights:  $w_i \leftarrow w_i e^{\alpha_m 1(y_i \neq \hat{f}^m(x_i))}$

#### AdaBoost algorithm (final step)

3. Output boosted model:

$$\hat{f}(x) = \text{sign} \left[ \sum_{m=1}^M \alpha_m \hat{f}^m(x) \right]$$

## Gradient Boosting

- General goal of boosting: find the model at each stage that minimizes loss function on ensemble (computationally difficult!)
- AdaBoost interpretation (discovered years later): Gradient descent algorithm that minimizes exponential loss function.
- Gradient boosting: works for any differentiable loss function. At each stage, find the local gradient of loss function, and take steps in direction of steepest descent.

## **Summary of (selected) ensemble methods**

- Can use a single estimator that has poor performance
- Combining the output of multiple estimators into a single prediction: better predictive accuracy, less interpretability