Ensemble Methods

Nandita Bhaskhar

content adapted from

(a) Hastie, Tibshirani & Friedman and
(b) Protopapas, Rader & Pan
(c) Jason Brownlee

Nov 12th, 2021

Outline

- Decision Trees Recap
- Ensemble Methods: Intro
- Model Averaging
- Bagging
- Random Forests
- Boosting
- Gradient boosting

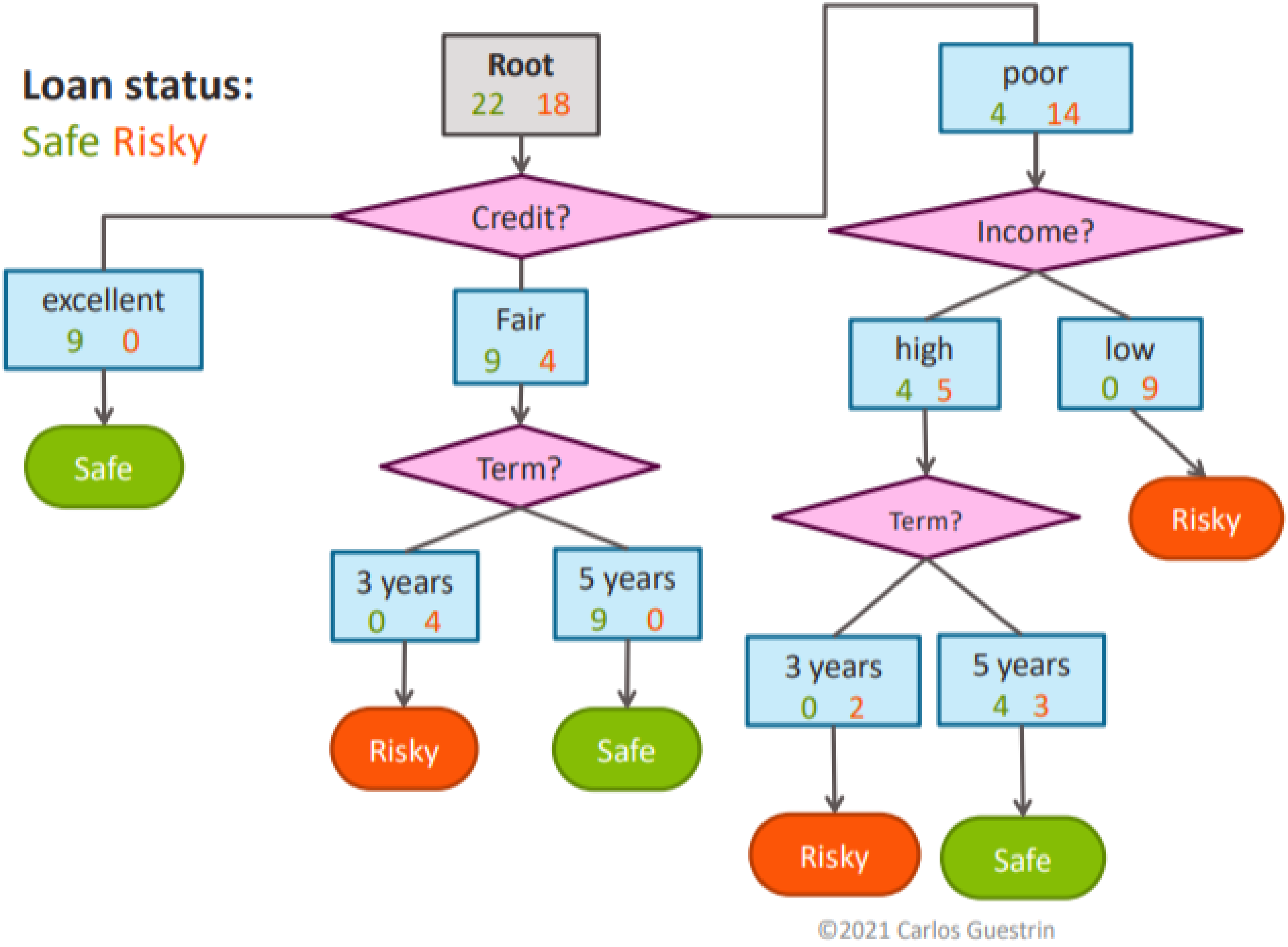
Decision Trees Recap

- Represented by a series of binary splits
- Internal node is a value query on a feature or attribute
  - Is $x_i > 0.5$? If yes, go right. Else, go left
- Terminal nodes are the decision nodes
  - Classification → class label
  - Regression → output score

Decision Trees Recap

- The tree is **grown** using training data using recursive splitting
- It is also **pruned** to an optimal size (often evaluated using cross-validation)
- **Inference**: pass their features down the tree till it reaches a terminal node to get the decision

Decision Trees Recap



Decision Trees Recap

Pros

- Can handle large datasets
- Can handle mixed predictors (continuous, discrete, qualitative)
- Can ignore redundant variables
- Can easily handle missing data
- Easy to interpret if small

Cons

- Prediction performance is poor
- Does not generalize well
- Large trees are hard to interpret

Outline

- Decision Trees Recap
- Ensemble Methods: Intro
- Model Averaging
- Bagging
- Random Forests
- Boosting
- Gradient boosting

Ensemble Methods: Intro

- Methods to improve the performance of weak learners
- Weak learners (e.g., classification trees) don't perform that well

- What do we do??
- **Wisdom of the crowds!**

Ensemble Methods: Intro

- Wisdom of the crowds!
- Shift responsibility from 1 weak learner to an “ensemble” of such weak learners
- Set of weak learners are combined to form a strong learner with better performance than any of them individually

Ensemble Methods: Intro

- A single decision tree often produces noisy / weak classfiers
- They DON'T generalize well
- But they are super fast, adaptive and robust!
- Solution: Let's learn multiple trees!
- How to ensure they don't all just learn the same thing??
- ~~TRIVIAL Solution~~

Outline

- Decision Trees Recap
- Ensemble Methods: Intro
- Bagging
- Random Forests
- Boosting
- Gradient boosting

Bagging

- Bagging (Breiman, 1996)
- Bootstrap Aggregating: to ensure lower variance
- Bootstrap sampling: get different splits / subsets of the data
- Aggregating: majority voting or averaging

Bagging

- Averages a given procedure over many samples to reduce its variance
- Multiple realizations of the data (via multiple samples) →
  - calculate predictions multiple times →
  - average the predictions →
  - more certain estimations (lesser variance)

Bagging

- Let $f(x)$ be the classifier and let b be a sample set from data

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

Or

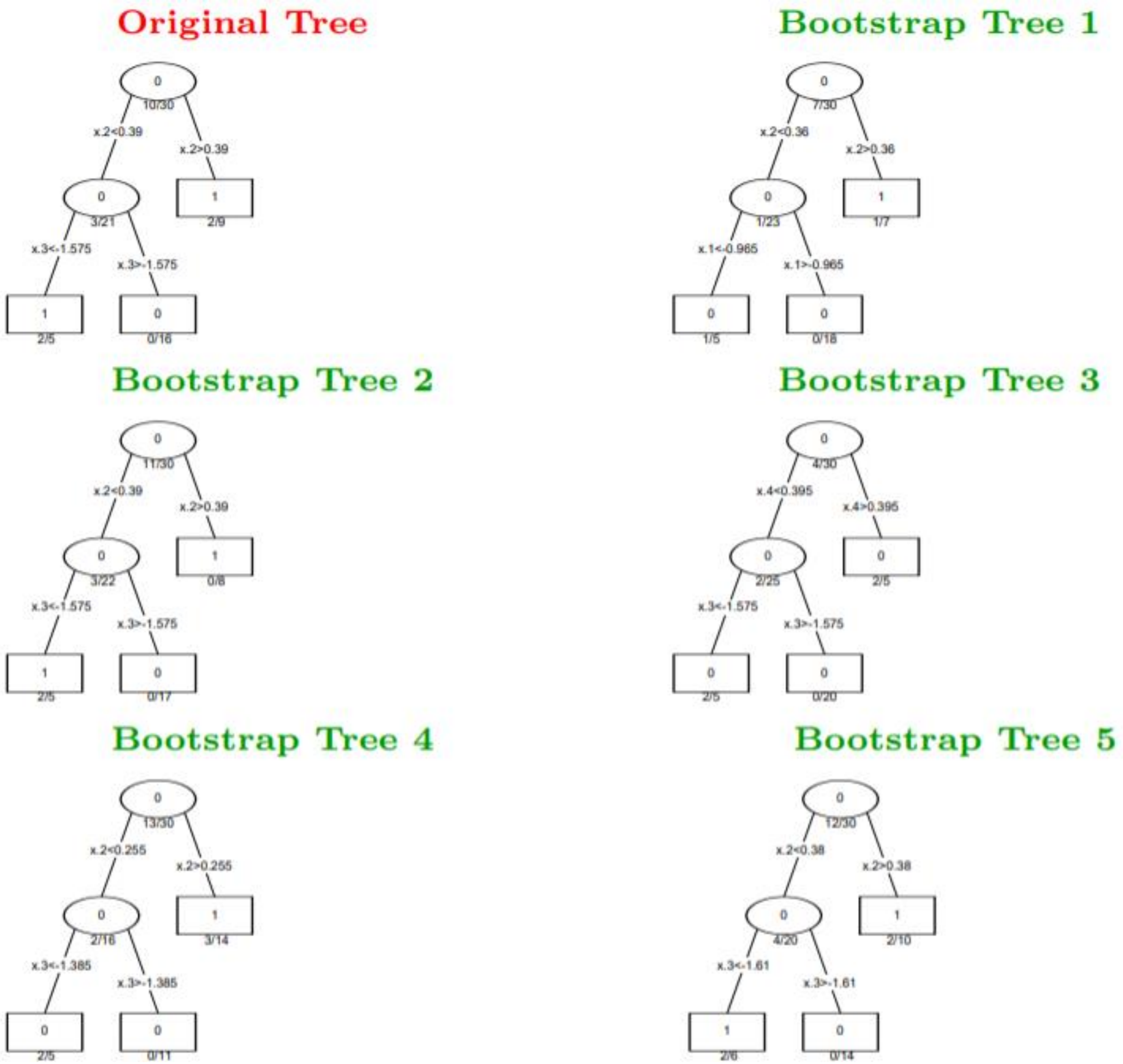
$$\hat{f}_{agg}(x) = \text{Majority Vote}\ \{f_b(x)\}_{b=1}^{B}$$

- Independent of type of classifier

Bagging

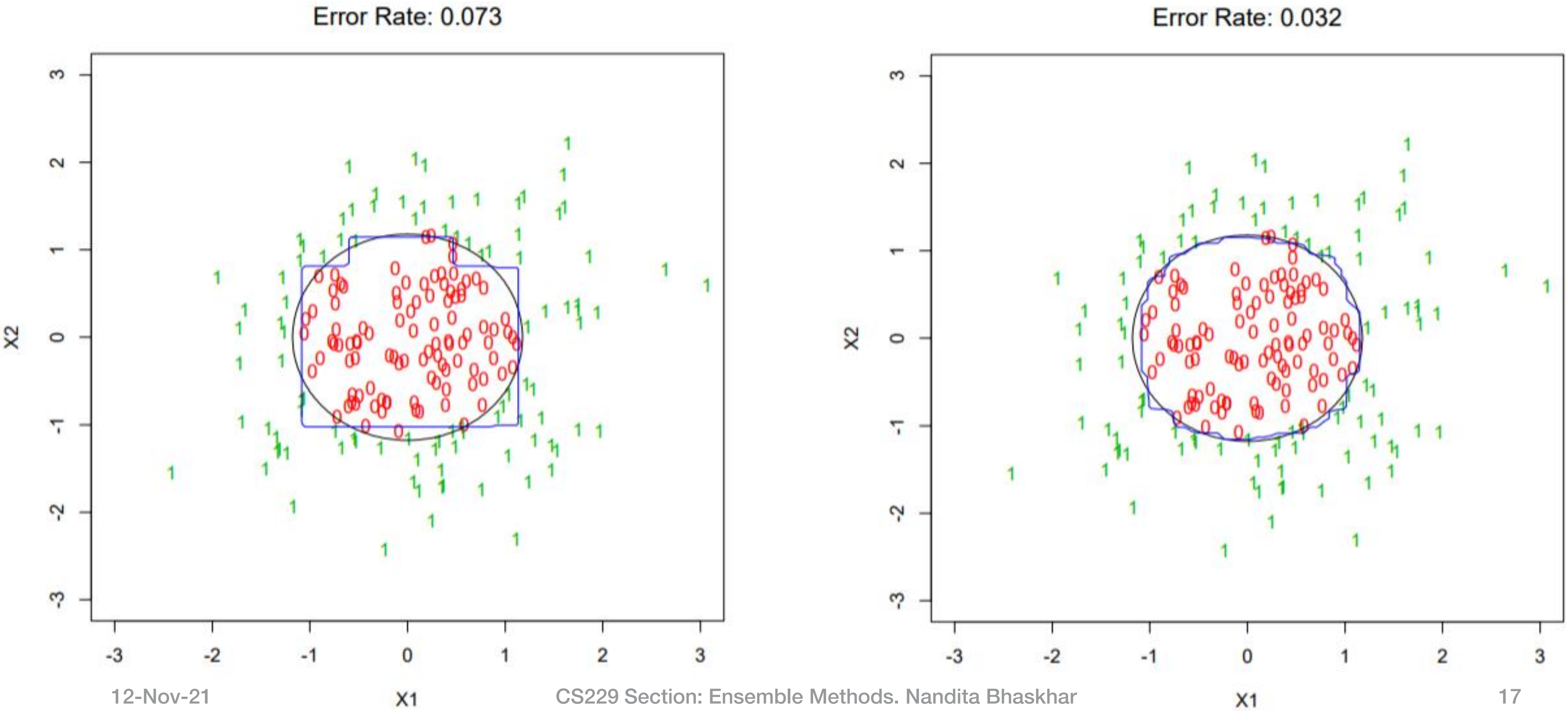
- Bootstrap sampling:
- Collect B $(\cong 100)$ subsets by sampling with replacement from training data
- Construct B trees (one classifier for one subset)
- Aggregate them using aggregator of your choice
- Parallelizable

Bagging



Original Tree
0
10/30
x.2<0.39
x.2>0.39
0
3/21
1
2/9
x.3<-1.575
x.3>-1.575
1
2/5
0
0/16
Bootstrap Tree 1
0
7/30
x.2<0.36
x.2>0.36
0
1/23
1
1/7
x.1<-0.965
x.1>-0.965
0
1/5
0
0/18
Bootstrap Tree 2
0
11/30
x.2<0.39
x.2>0.39
0
3/22
1
0/8
x.3<-1.575
x.3>-1.575
1
2/5
0
0/17
Bootstrap Tree 3
0
4/30
x.4<0.395
x.4>0.395
0
2/25
0
2/5
x.3<-1.575
x.3>-1.575
0
2/5
0
0/20
Bootstrap Tree 4
0
13/30
x.2<0.255
x.2>0.255
0
2/16
1
3/14
x.3<-1.385
x.3>-1.385
0
2/5
0
0/11
Bootstrap Tree 5
0
12/30
x.2<0.38
x.2>0.38
0
4/20
1
2/10
x.3<-1.61
x.3>-1.61
1
2/6
0
0/14

Bagging



Bagging

- What about cross validation?
- Each bootstrap sample set uses only a subset of the data
- Unused samples: out-of-bag samples (OOB)
- Calculate overall error rate on out-of-bag samples for all bootstraps

Bagging

- Reduces overfitting (i.e., variance)
- Can work with any type of classifier (here focus on trees)
- Easy to parallelize
- But loses on interpretability to single decision tree

Outline

- Decision Trees Recap
- Ensemble Methods: Intro
- Bagging
- Random Forests
- Boosting
- Gradient boosting

Random Forests

Issues with Bagging:

- Expectation of bagged trees is equal to expectation of individual trees

$$\mathbf{E}\left[\hat{f}_{agg}(x)\right] = \mathbf{E}\left[f_b(x)\right]$$

- Bias of bagged trees is the same as that of individual trees
- Each tree is identically distributed (i.d. not i.i.d). Bagged trees are correlated!

Random Forests

Issues with Bagging:

- Averaging B i.i.d. variables scales their variance σ^2 to σ^2/B
- But averaging B i.d. variables with pairwise correlations ρ and variance σ^2 gives their final variance to be

$$\rho\sigma^2 + \frac{1-\rho}{\mathrm{B}}\sigma^2$$

- Only the 2nd term reduces on bagging, but the first term remains

Random Forests

- How to decorrelate the trees generated for bagging?
- We want to generate B i.i.d. trees such that their bias is the same, but variance reduces
- Ideas:
  - We can restrict how many times a feature can be used
  - We only allow a certain number of features
  - Etc..

Random Forests

- Ideas:
  - We can restrict how many times a feature can be used
  - We only allow a certain number of features
  - Etc..
- Bias changes for the above ideas ☹
- Instead, choose only subset of features for each bag
- Decorrelated trees when you randomly select the subset

Random Forests

- As in bagging, choose B bootstrapped splits (or bags)
- For each split in the B trees, consider only k features from the full feature set m
- $k = m$ → same as Bagging
- $k < m$ → Random Forests
- OOB error rate can be used to fit RF in one sequence with cross validation done along the way

Random Forests

- Works great in practice. k to be treated as a hyperparameter

Issues:

- When you have large number of features, yet very small number of relevant features
- Prob(selecting the relevant feature in k) is very small

Outline

- Decision Trees Recap
- Ensemble Methods: Intro
- Bagging
- Random Forests
- Boosting
- Gradient boosting

Boosting

- Boosting does not involve bootstrap sampling
- Trees are grown sequentially: each tree is grown using information from previously grown trees
- Like bagging, boosting involves combining many decision trees, $f_1, \dots, f_B$
- Lecture slides: AdaBoost

Boosting

- AdaBoost:
- Weighted observations
- Put more weight on difficult to classify instances and less on those already handled well
- New weak learners are added sequentially that focus their training on the more difficult patterns

Outline

- Decision Trees Recap
- Ensemble Methods: Intro
- Bagging
- Random Forests
- Boosting
- Gradient boosting

Gradient Boosting

Generalization: AdaBoost, Adaptive Reweighting & Combining

Three elements –

- A loss function to be optimized
- A weak learner to make predictions (decision trees)
- An additive model to add weak learners to minimize the loss function

Gradient Boosting

Loss Function:

- Any differentiable function
- Most standard loss functions
  - L2 loss for regression
  - Log loss for classification

Gradient Boosting

Weak Learners:

- Decision trees (regression trees) learnt greedily
- Constrain the trees to ensure they remain weak
  - Number of layers, leaves, nodes, splits, etc

Gradient Boosting

Additive Model:

- Add trees one at a time (existing trees are not changed)
- Functional gradient descent to minimize loss when adding trees
  - calculate the loss
  - add the tree to the model that reduces the loss (i.e., follow the gradient)
  - parameterize the tree, then modify the parameters of the tree and move in the right direction by reducing the residual loss.

Gradient Boosting

Given the current model,

- We fit a decision tree to the residuals from the model
- Response variable now is the residuals
- We then add this new decision tree into the fitted function in order to update the residuals
- The learning rate must be controlled

Gradient Boosting

Tunable Parameters:

- **Number of trees (B):** Boosting can overfit unlike Bagging / RFs. Use cross-validation!
- **Shrinkage parameter (λ):** small positive number that sets the learning rate
- **Number of splits in each tree (d):** Usually just choose $d = 1$, i.e., tree stumps work well

Gradient Boosting

Variants:

- Varying the tree constraints
- Weighting each tree to the additive sum using a learning rate (shrinkage)
- Sampling strategies: stochastic gradient boosting
- Regularization: L1 / L2
- Successful: XGBoost

Thank you!

General tips for projects:

- Use tree-based methods + ensembling as baselines when dealing with:
  - Categorical data
  - Mixed data types
  - Missing data