Chapter 8 Tree-based Methods

– Math 313 Statistics for Data Science

Guangliang Chen

Associate Professor

cheng@hope.edu

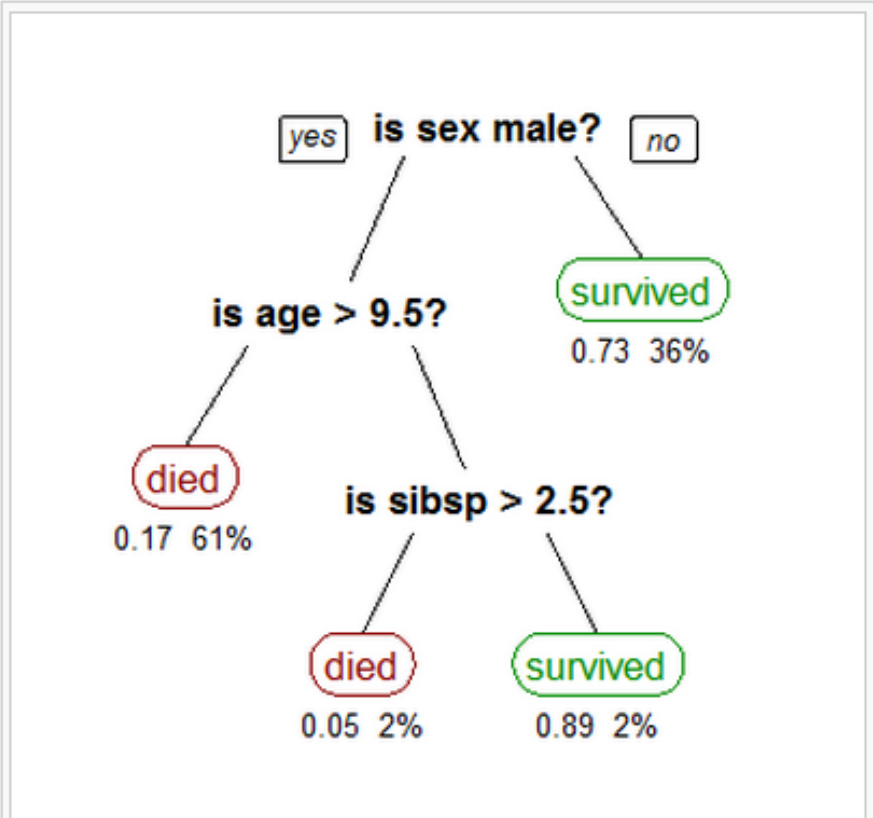
Hope College, Fall 2023

Presentation Overview

1. Classification trees and ensemble learning
2. Regression trees (to be discussed in class)

What is a classification tree?

- A series of binary splits based on training data.
- Each internal node represents a query on one of the variables.
- The terminal/leaf nodes are the decision nodes, typically dominated by one of the classes.
- New observations are classified in the respective terminal nodes through majority vote.



A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf.

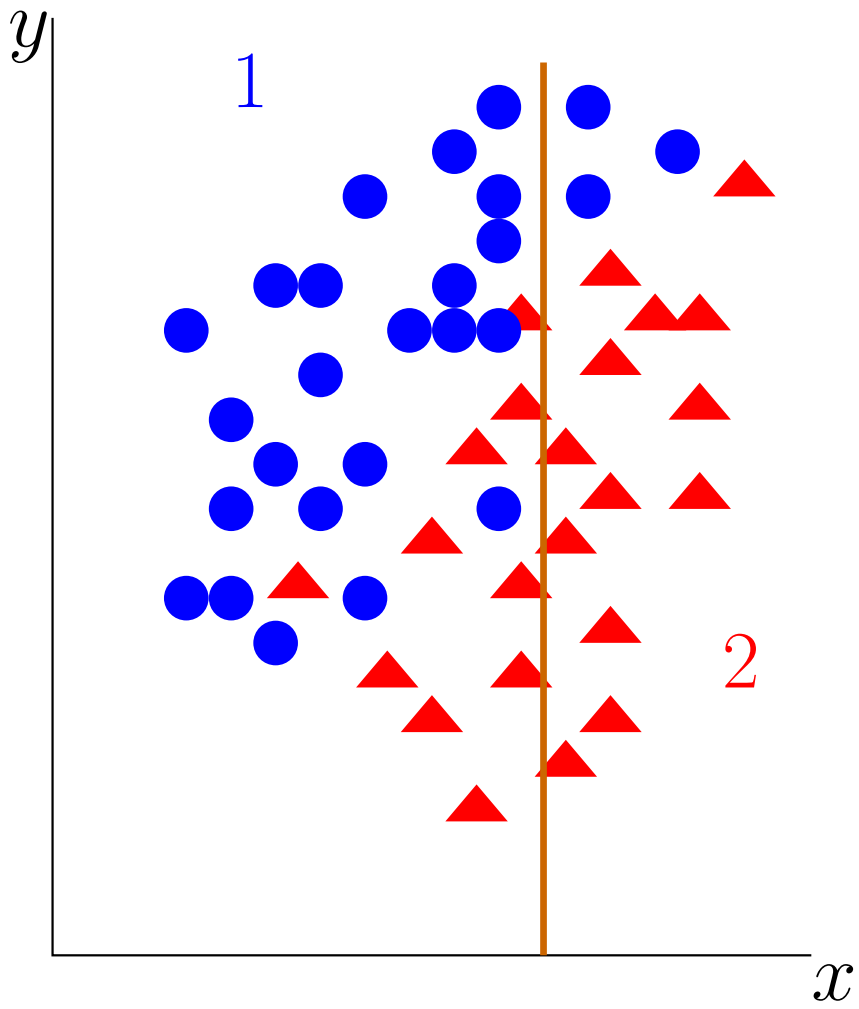
Splitting criterion: Gini impurity score

Optimal splitting in each node is found by comparing all variable + cutoff pairs in terms of **pureness** in each child node.

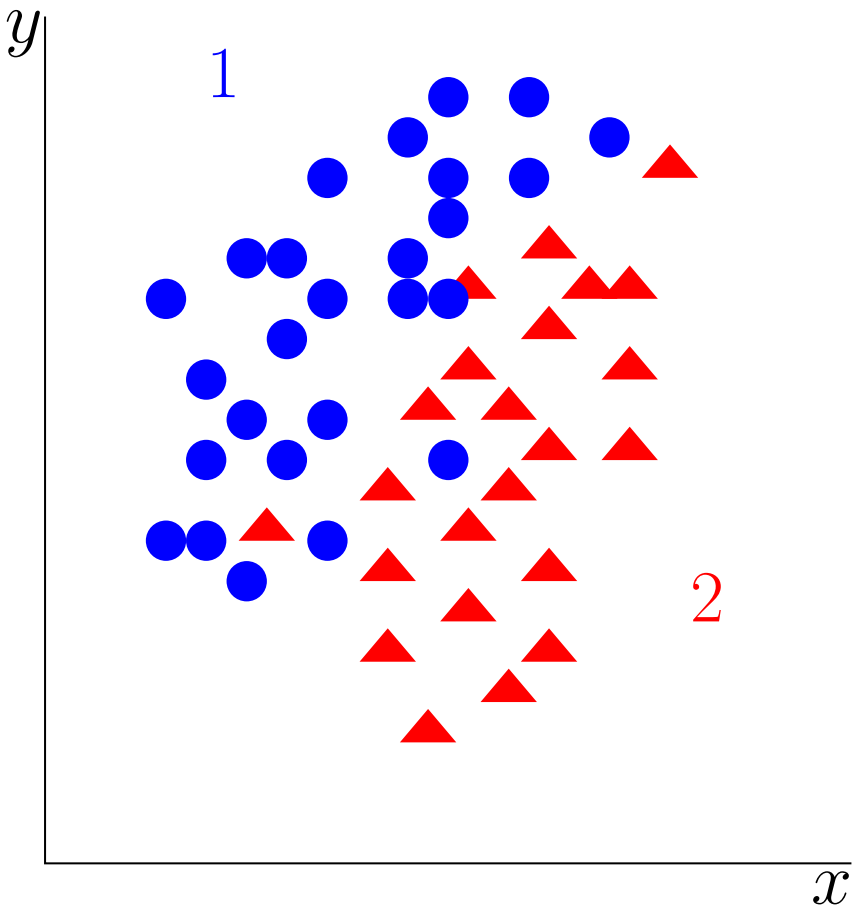
Mathematically, this is done by minimizing the **Gini impurity index** over all possible splits:

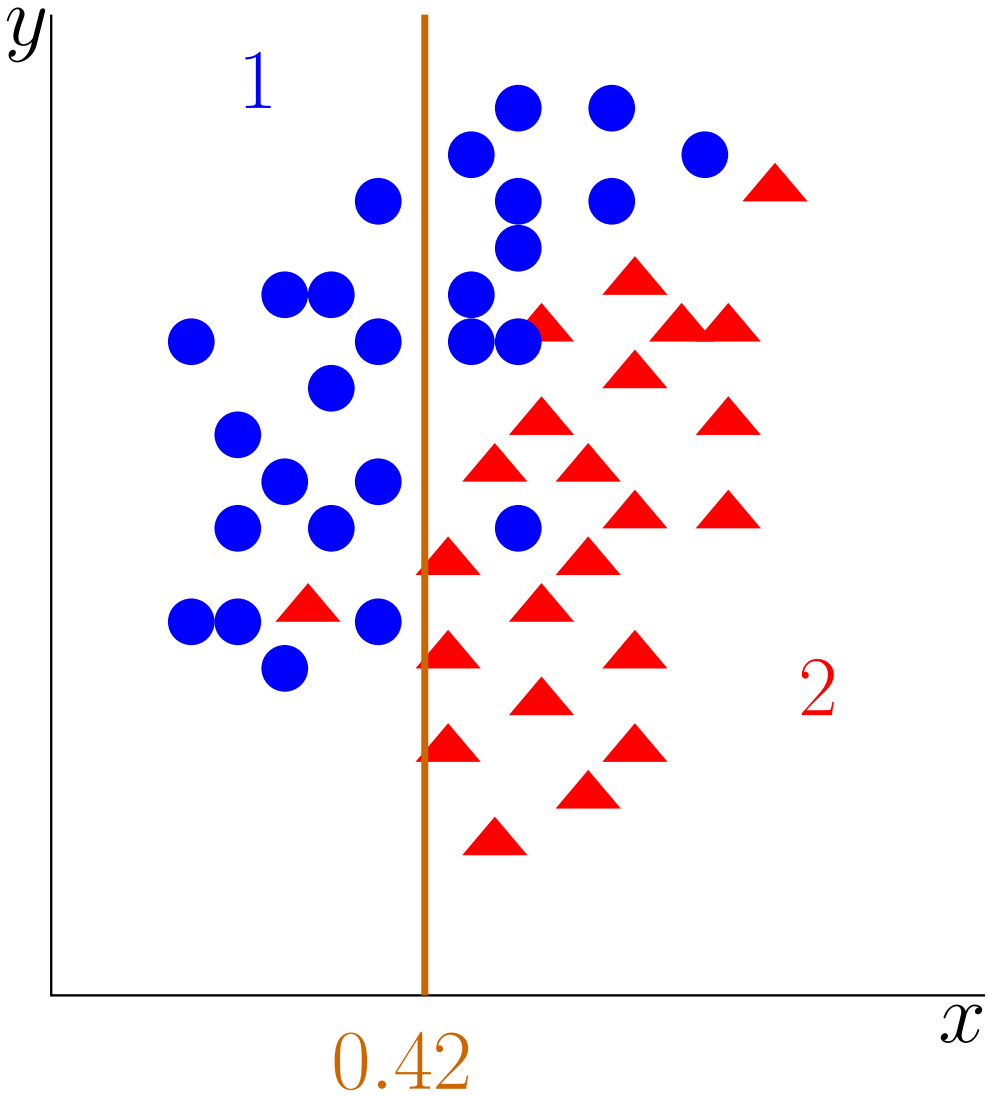
$$n^{(\mathrm{L})} \sum_{j=1}^{c} p_j^{(\mathrm{L})}(1-p_j^{(\mathrm{L})}) + n^{(\mathrm{R})} \sum_{j=1}^{c} p_j^{(\mathrm{R})}(1-p_j^{(\mathrm{R})})$$

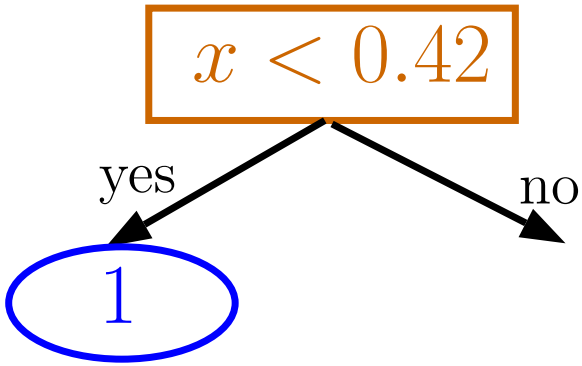
- $n^{(\mathrm{L})}$ ($n^{(\mathrm{R})}$): #training examples in left (right) node
- $p_j^{(\mathrm{L})}$ ($p_j^{(\mathrm{R})}$): proportion of class j in left (right) node

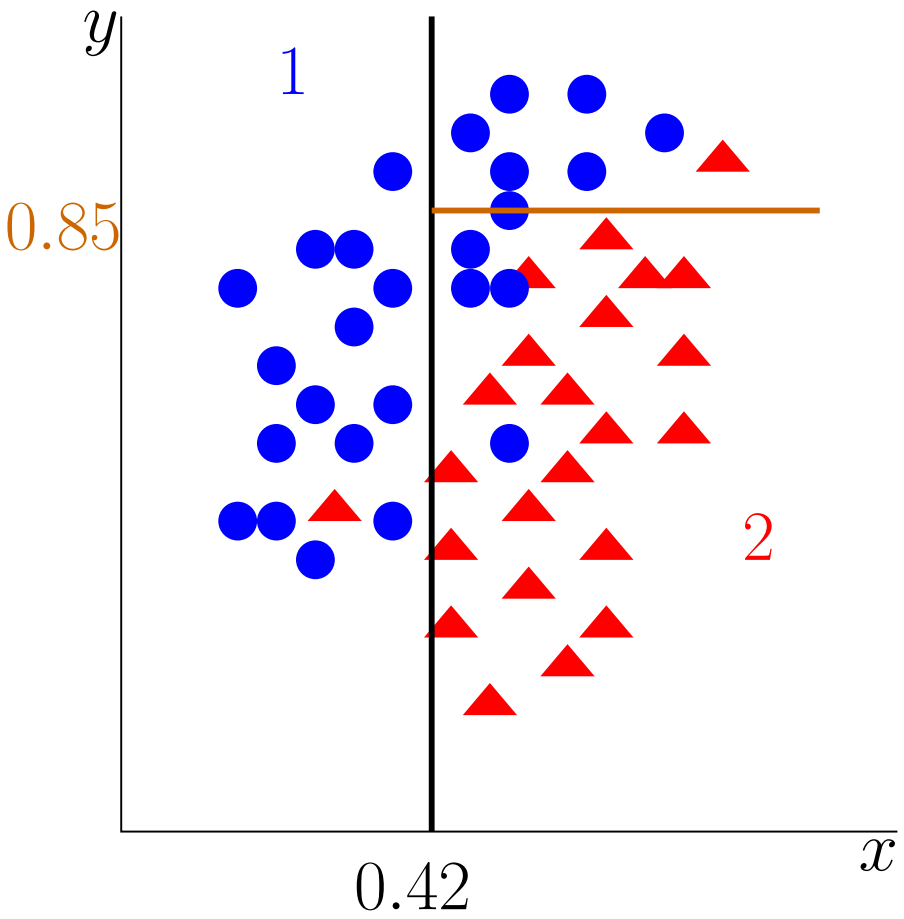


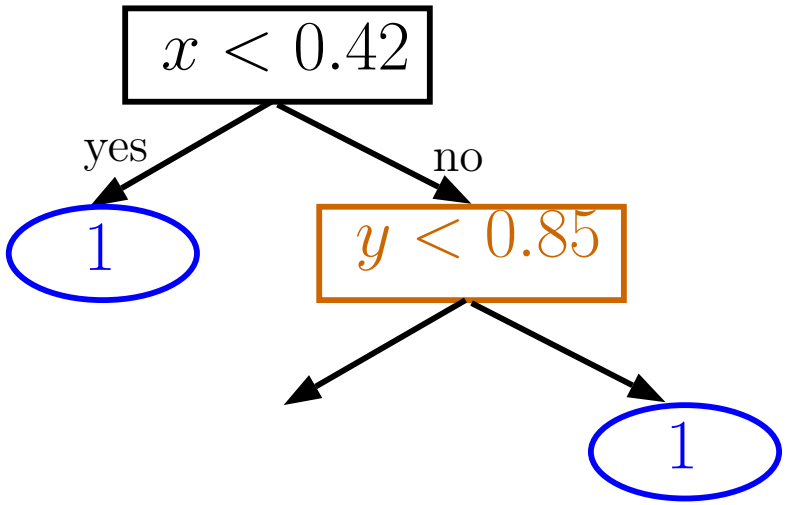
Demo: Building a classification tree

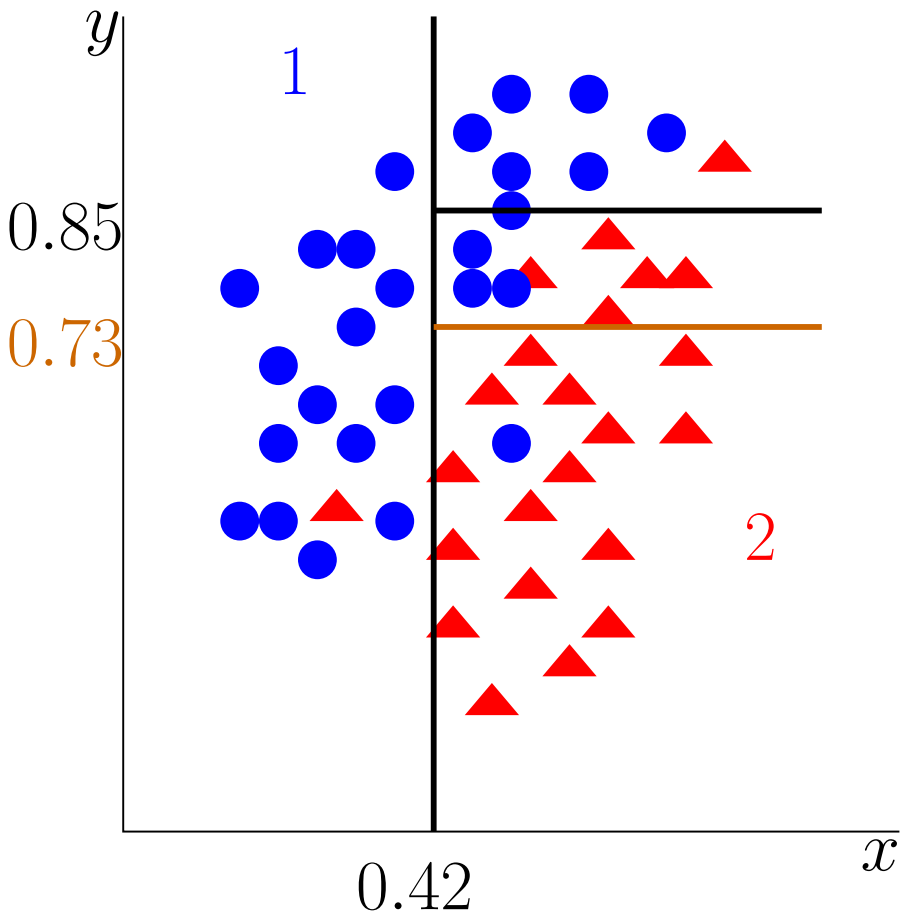


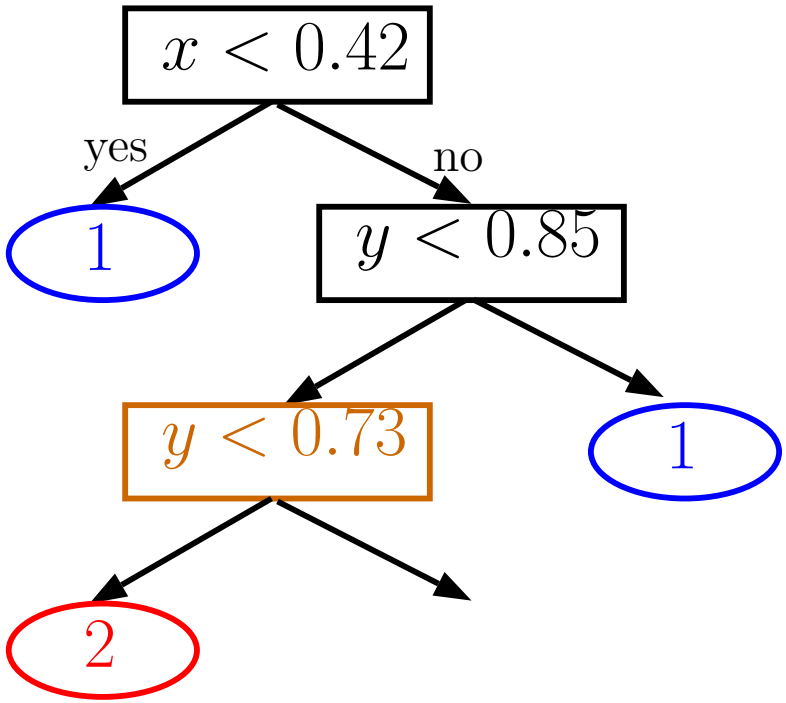

y
1
2
x
0.42

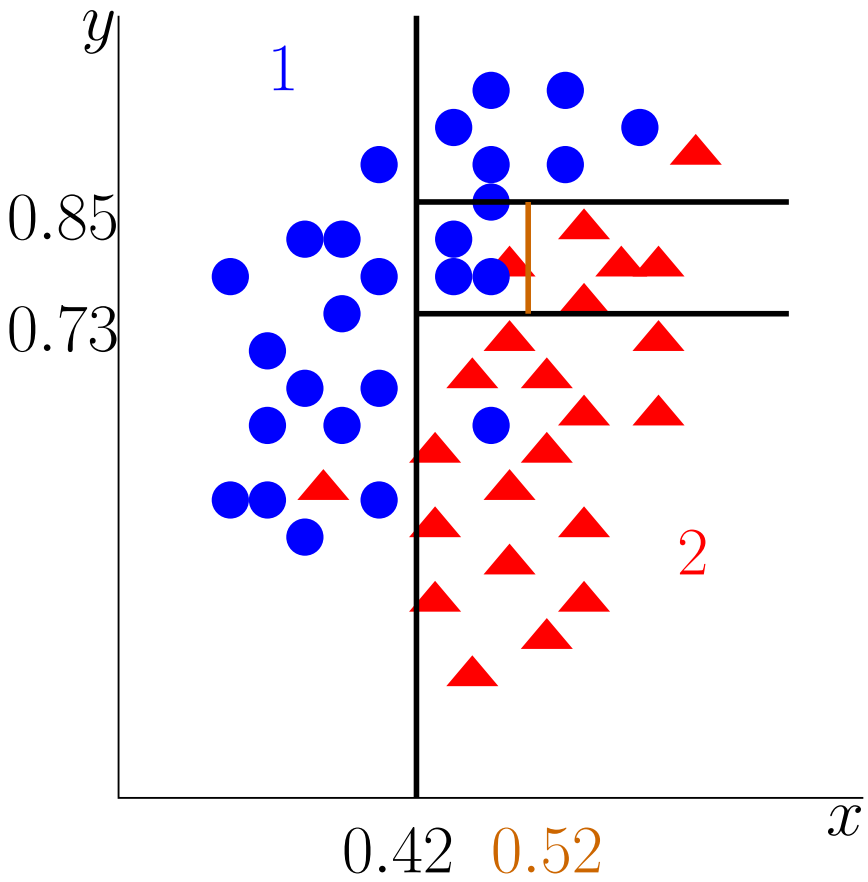

$x < 0.42$
yes
no
1

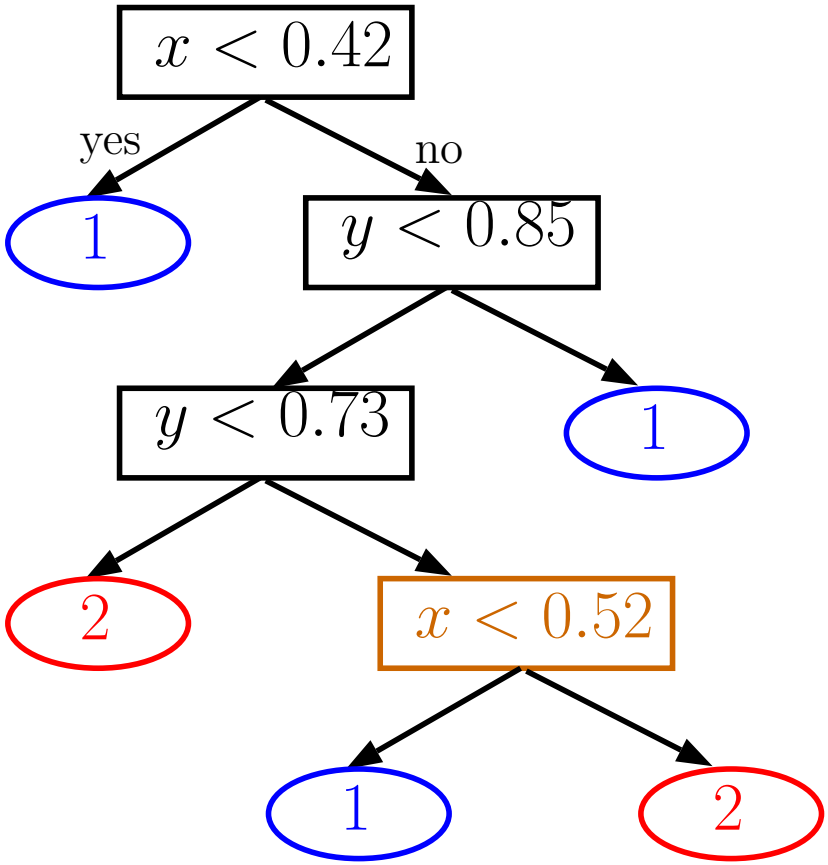

y
1
0.85
2
0.42
x

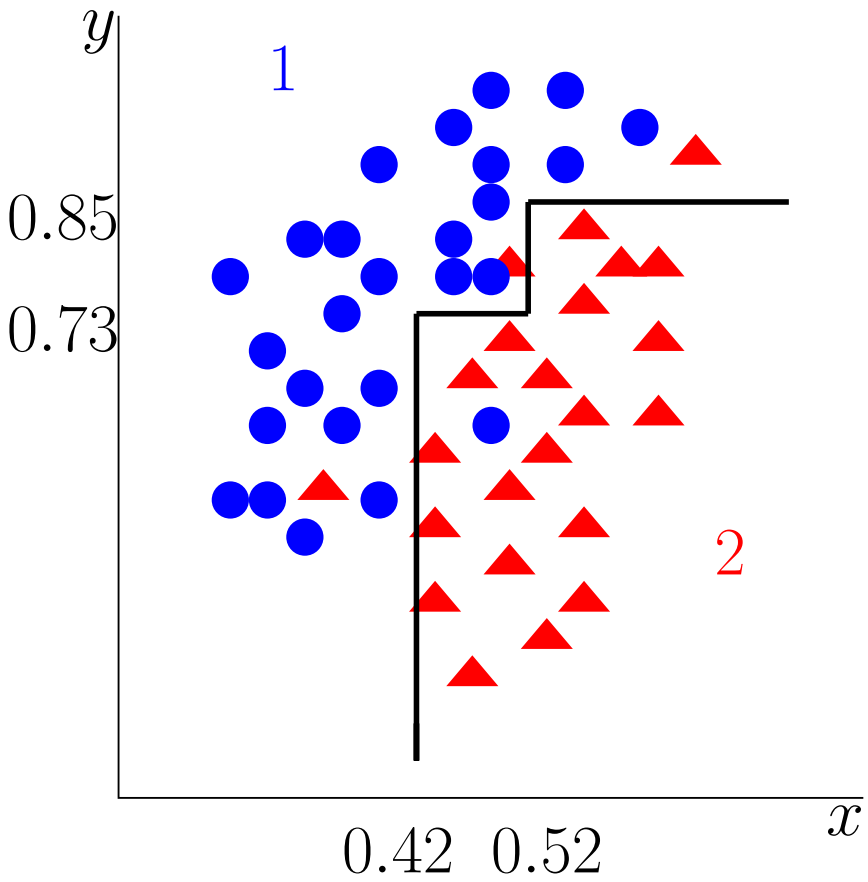

$x < 0.42$
yes
no
1
$y < 0.85$
1

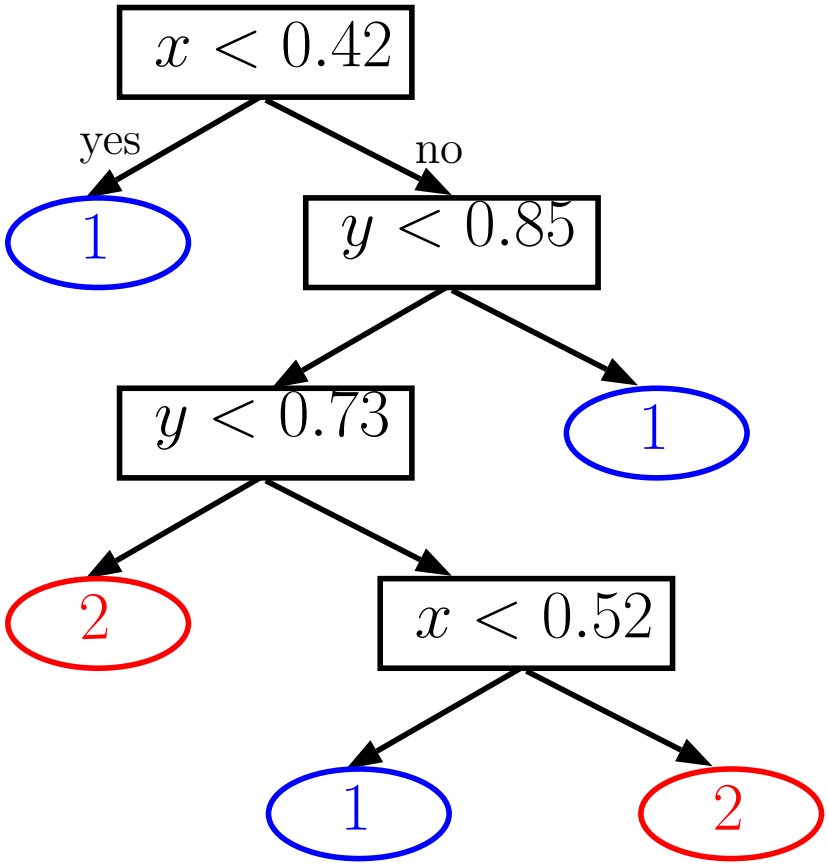

y
1
0.85
0.73
2
0.42
x

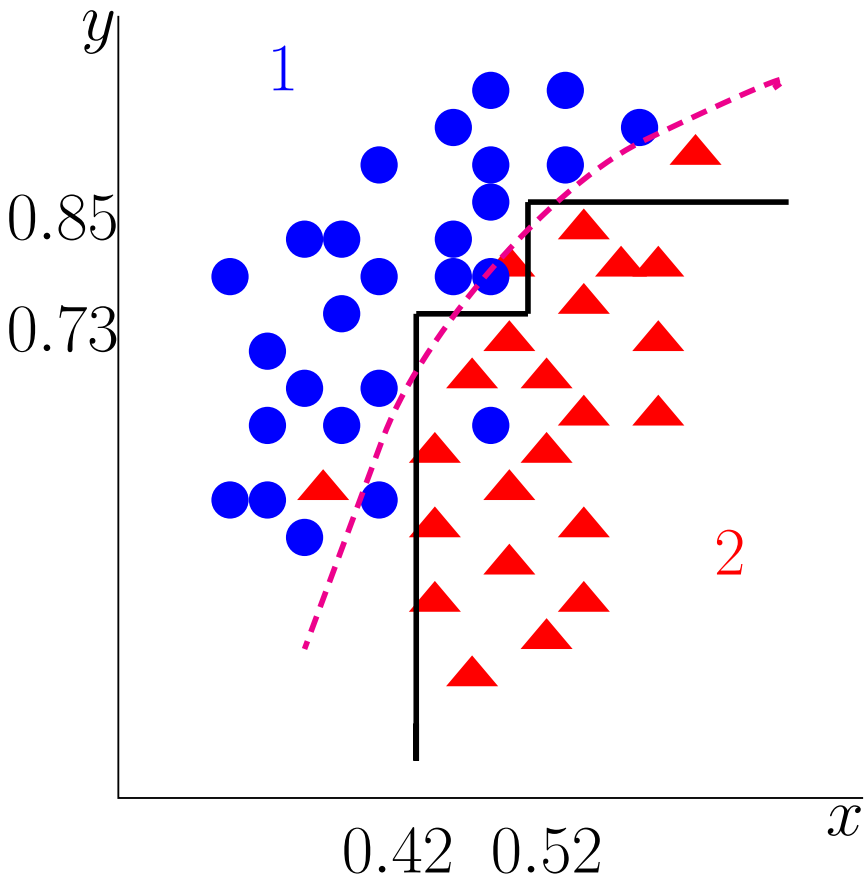

$x < 0.42$
yes
no
1
$y < 0.85$
$y < 0.73$
1
2

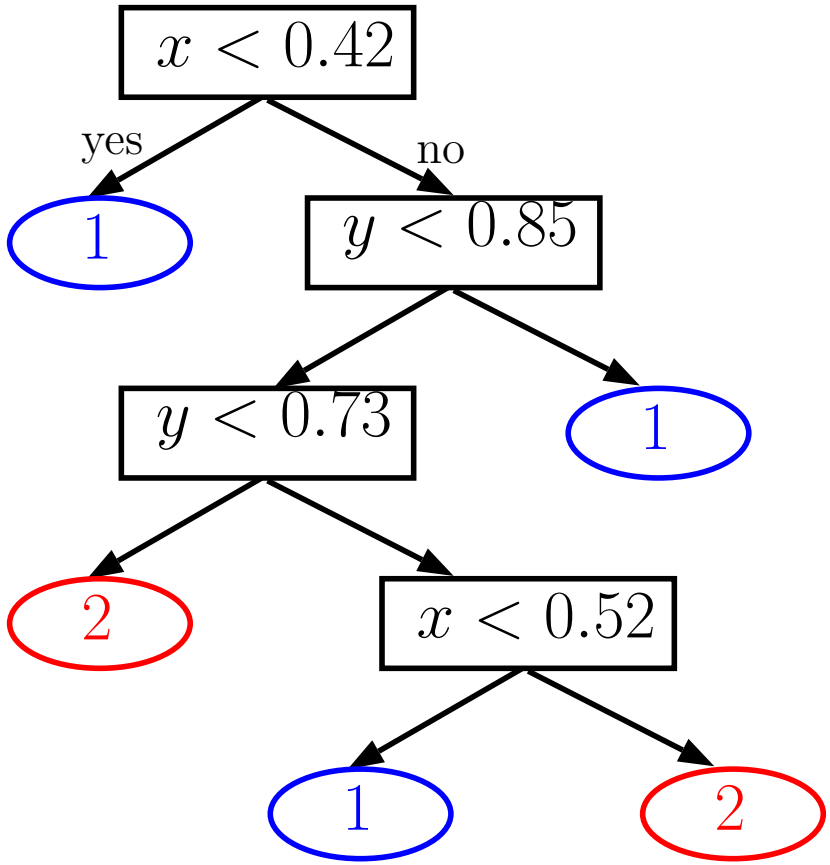

y
1
0.85
0.73
2
x
0.42
0.52


$x < 0.42$
yes
no
1
$y < 0.85$
$y < 0.73$
1
2
$x < 0.52$
1
2


y
1
0.85
0.73
2
0.42
0.52
x


$x < 0.42$
yes
no
1
$y < 0.85$
$y < 0.73$
1
2
$x < 0.52$
1
2


y
1
0.85
0.73
2
0.42
0.52
x


$x < 0.42$
yes
no
1
$y < 0.85$
$y < 0.73$
1
2
$x < 0.52$
1
2

Remark. Classification trees, though easy to build, are considered as weak learners:

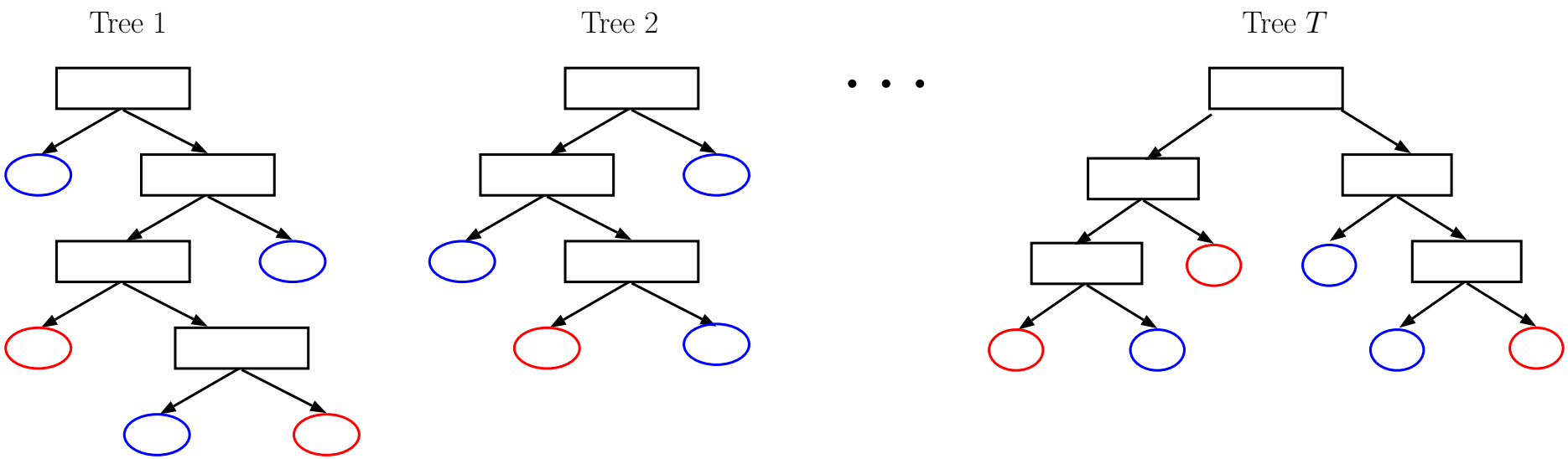
- The decision boundary is piecewise linear
- Unstable (if we change the data a little, the tree may change a lot)
- Prediction performance is often poor (due to high variance)

Advantages of classification trees

- No distribution assumptions (non-parametric)
- Works in the same way in multiclass settings
- Can handle large data sets
- Can handle categorical variables naturally
- Can deal with missing values elegantly

Ensemble methods

Ensemble methods train many weak classifiers (e.g., trees) and combine their predictions to enhance the performance of a single weak learner:

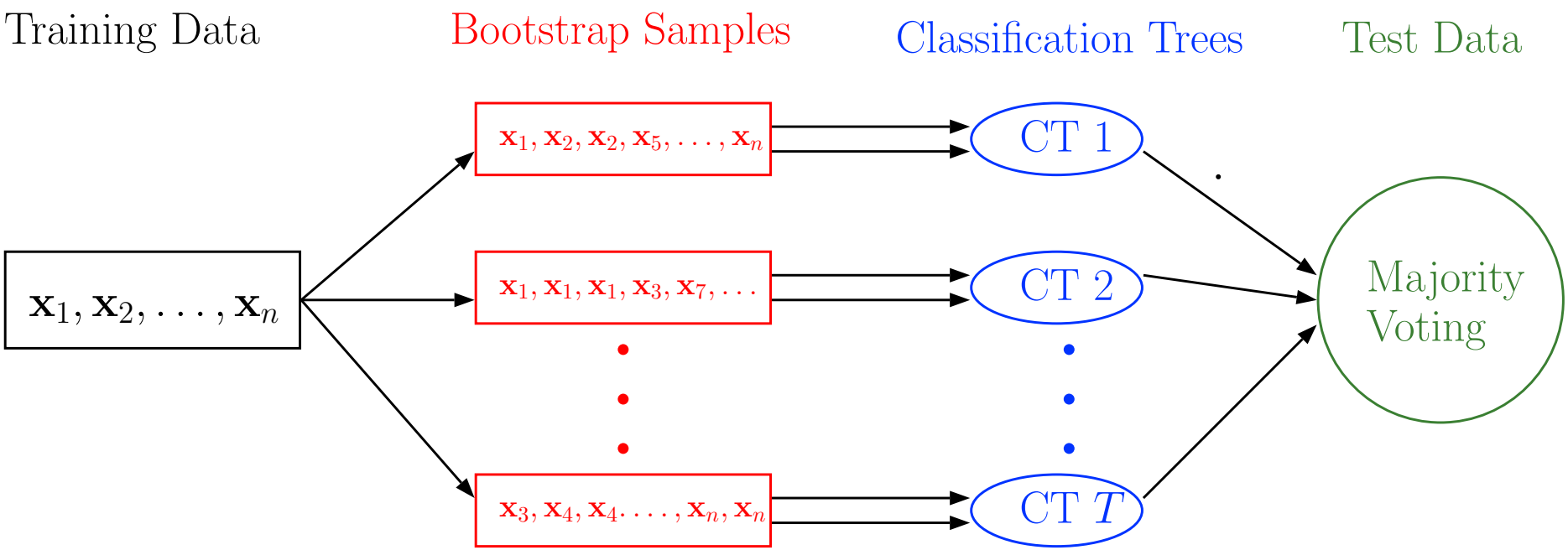


To be covered in this course:

- **Bagging (Bootstrap AGGregatING)**: build many trees independently from different bootstrap samples of the training data and then vote their predictions to get a final prediction
- **Random Forest**: a variant of bagging by allowing to use different subsets of variables at the nodes of any tree in the ensemble
- **Boosting**: build many trees adaptively and then add their predictions

Bagging

To build a classification tree on each separate **bootstrap sample** (random sample with replacement) of the training data and then use majority vote.



Remark.

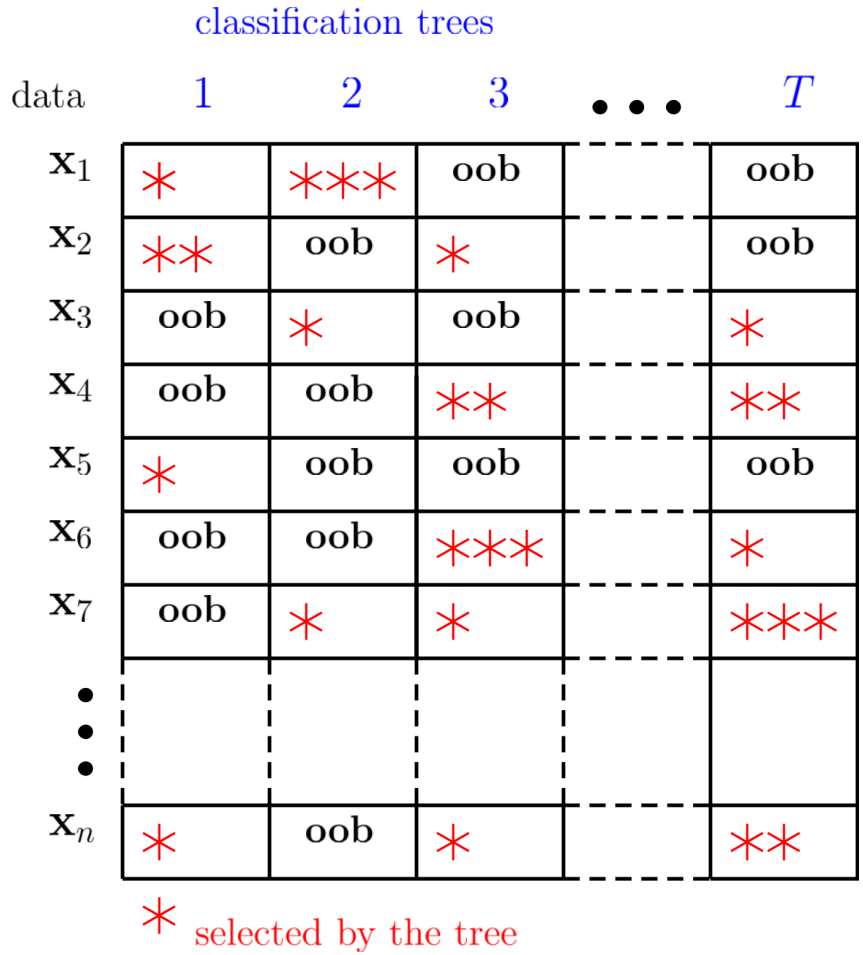
- "Averaging" many trees decreases the variance of the model, without increasing the bias (as long as the trees are not correlated)
- Bootstrap sampling is a way of de-correlating the trees (as simply training many trees on a single training set would give strongly correlated trees).
- The number of bootstrap samples/trees, T, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set.

Out-of-bag (oob)

It turns out that drawing n out of n observations with replacement omits on average **36.8%** of observations for each tree:

$$\left(1-\frac{1}{n}\right)^n \approx \frac{1}{e} = 0.3678794$$

We say that those samples left out by a tree are **out-of-bag (oob)** with respect to that tree.



Out-of-bag error

For each training example $(\mathbf{x}_i, y_i)$, we may vote the predictions of all the oob trees to obtain an estimate of y_i, denoted as $\hat{y}_i^{(\text{oob})}$.

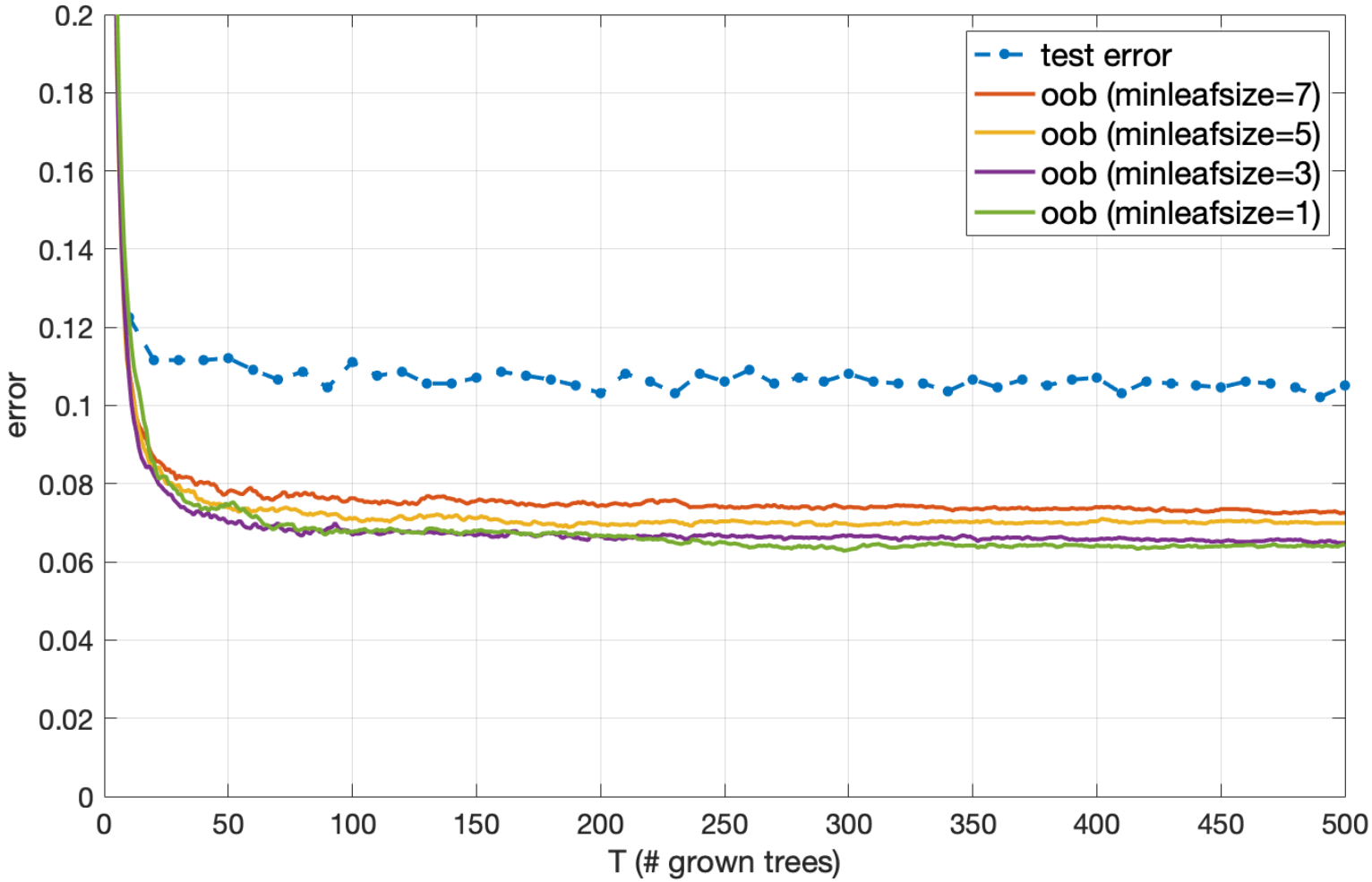
The overall oob error is computed as follows:

$$e^{(\text{oob})}(T) = \sum_{i=1}^{n} 1_{y_i \neq \hat{y}_i^{(\text{oob})}}$$

Such measure is an unbiased estimator of the true ensemble error and it does not require an independent validation dataset for evaluating the predictive power of the model.

An optimal value of T can be found by observing the out-of-bag error.

OOB and test errors of bagging on the *usps* data set

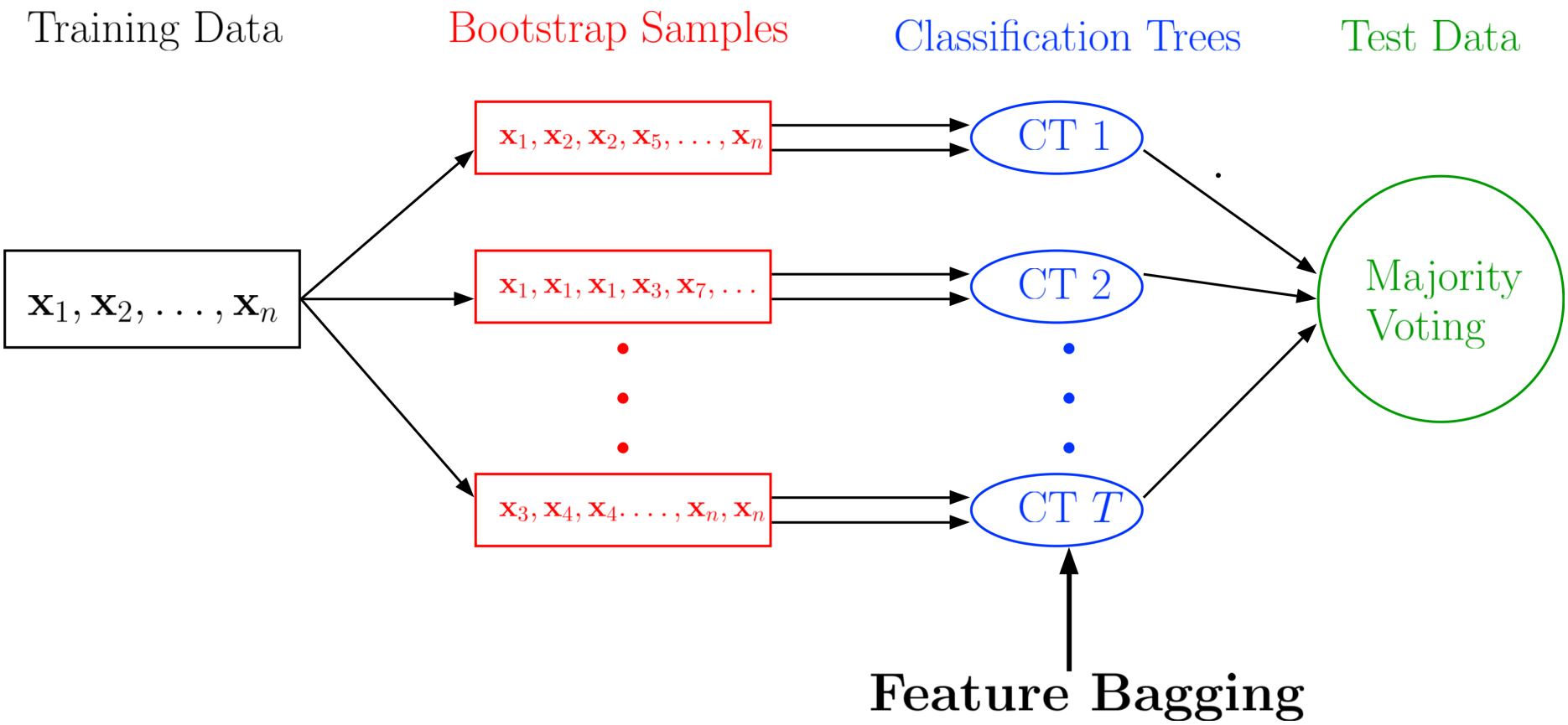


Random forest

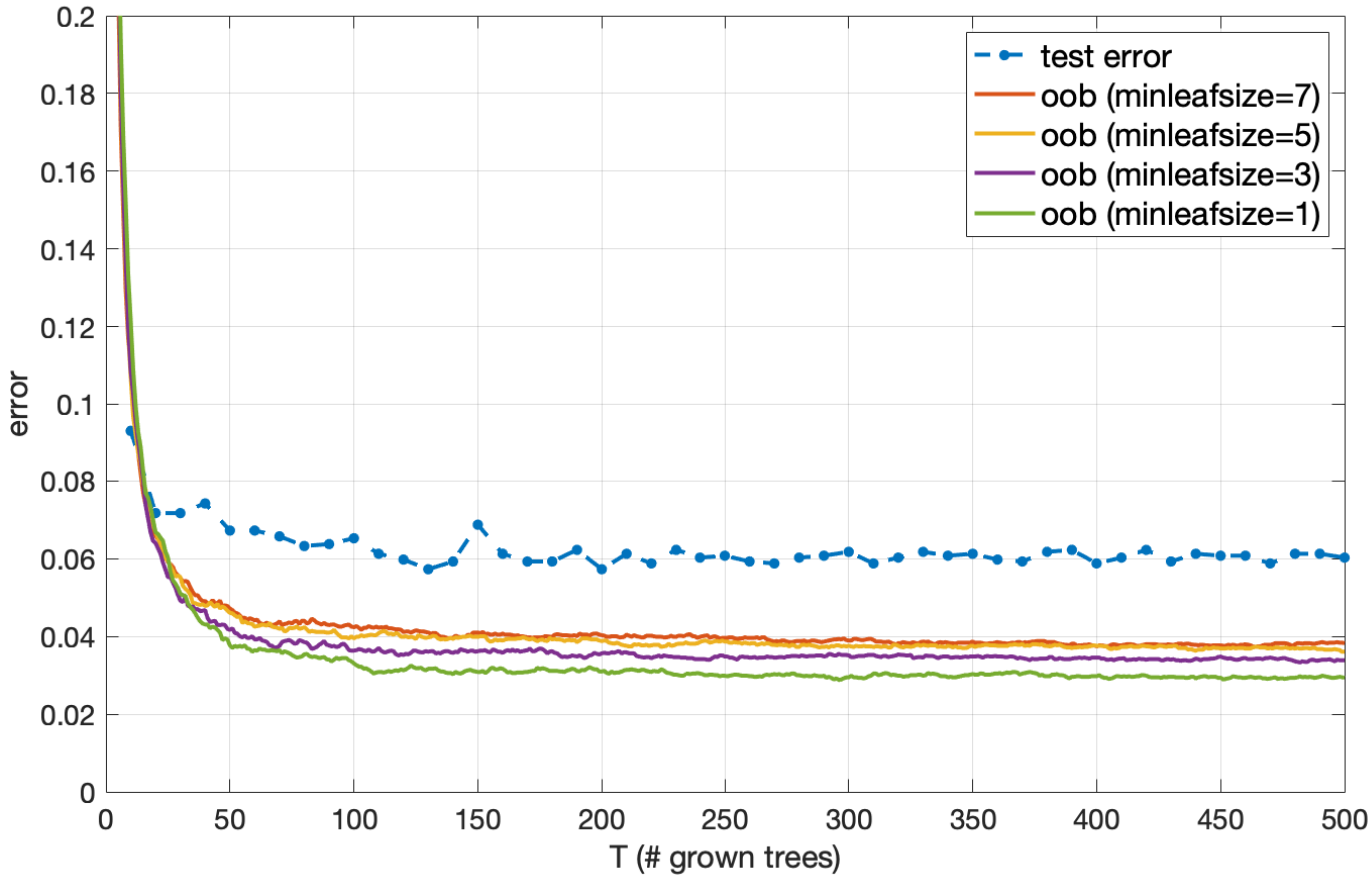
Random forests improve bagging by allowing every tree in the ensemble to randomly select predictors for each of its nodes. This process is sometimes called "**feature bagging**".

The motivation is to further de-correlate the trees in bagging: If one or a few features are very strong predictors for the class label, these features will be selected in many of the trees, causing them to become correlated.

Typically, for a classification problem with d features, $\sqrt{d}$ features (selected at random) are used in each split.

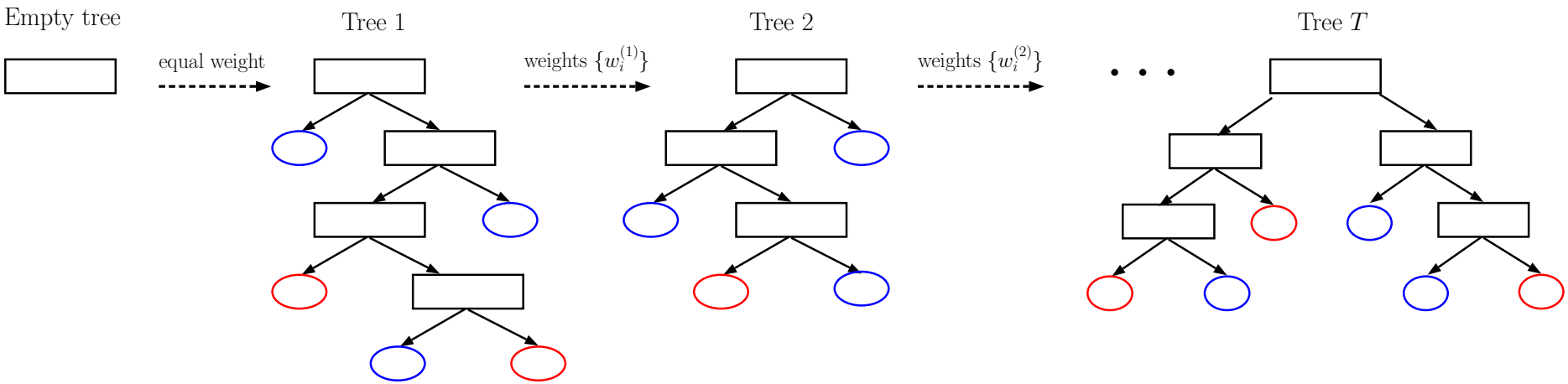

Training Data
Bootstrap Samples
Classification Trees
Test Data
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_2, \mathbf{x}_5, \ldots, \mathbf{x}_n$
$\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_7, \ldots$
$\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_4 \ldots, \mathbf{x}_n, \mathbf{x}_n$
CT 1
CT 2
CT T
Majority Voting
Feature Bagging

OOB and test errors of random forest on the *usps* data set



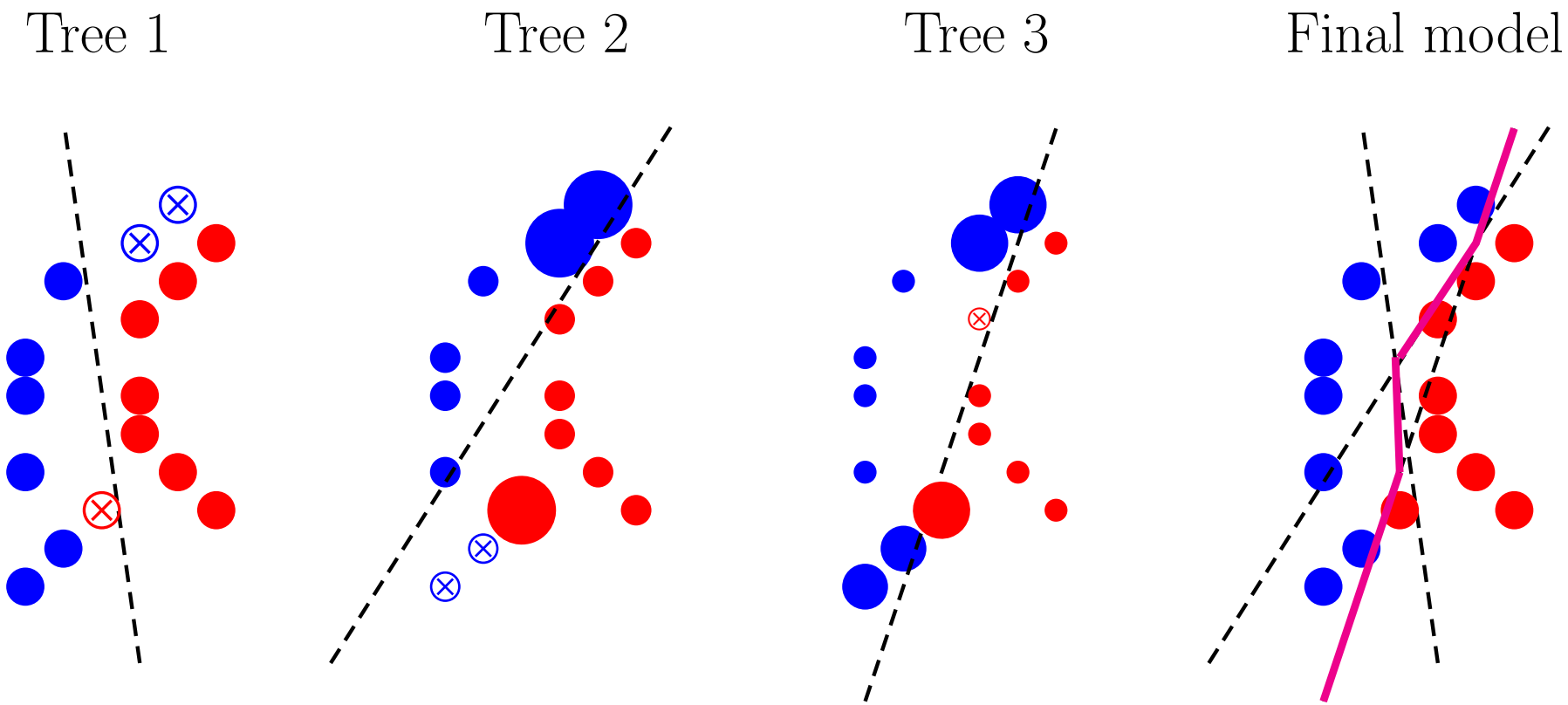
AdaBoost (Adaptive Boosting)

Main idea: Build tree classifiers sequentially by" focusing more attention" on training errors made by the preceding trees.



One way to realize this idea is to use the output of the current ensemble to **reweight the training data** for the next tree.

Demo: an ensemble of 3 boosted trees



How to choose weights

Initially, all training examples have an equal weight: $w_i^{(0)} = \frac{1}{n}$.

Now, in step 1:

- Fit the first tree classifier $C_1(\mathbf{x})$ on the training data using the above weights
- Compute the weighted error of $C_1(\mathbf{x})$:

$$\epsilon_1 = \sum_{i=1}^{n} \frac{1}{n} I_{y_i \neq C_1(\mathbf{x}_i)}$$

- Modify the weights of the training data points first by

$$w_i^{(1)} = \begin{cases} \frac{1}{n} \cdot \sqrt{\frac{\epsilon_1}{1-\epsilon_1}}, & C_1(\mathbf{x}_i) = y_i \\ \frac{1}{n} \cdot \sqrt{\frac{1-\epsilon_1}{\epsilon_1}}, & C_1(\mathbf{x}_i) \neq y_i \end{cases}$$

  and then normalize the new weights $w_i^{(1)}$ to have a sum of 1 for the next iteration.

For each additional step $t = 2 : T$, do the following:

- Fit a tree classifier $C_t(\mathbf{x})$ to the training data using weights $w_i^{(t-1)}$ from previous iteration
- Compute the weighted error of $C_t(\mathbf{x})$:

$$\epsilon_t = \sum_{i=1}^{n} w_i^{(t-1)} I_{y_i \neq C_t(\mathbf{x}_i)}$$

- Modify the weights of the training data points first by

$$w_i^{(t)} = \begin{cases} w_i^{(t-1)} \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}, & C_t(\mathbf{x}_i) = y_i \\ w_i^{(t-1)} \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}, & C_t(\mathbf{x}_i) \neq y_i \end{cases}$$

  and then normalize the new weights $w_i^{(t)}$ to have a sum of 1 for the next iteration.

How to use weights

Consider a weighted training set $(\mathbf{x}_i, y_i, w_i^{(t)}), 1 \le i \le n, t \ge 0$. In all calculations (wherever used), every training point $\mathbf{x}_i$ will count as "$w_i^{(t)}$ points".

For example, when computing the Gini impurity for a candidate split

$$n^{(\mathrm{L})} \sum_{j=1}^{k} p_j^{(\mathrm{L})}(1 - p_j^{(\mathrm{L})}) + n^{(\mathrm{R})} \sum_{j=1}^{k} p_j^{(\mathrm{R})}(1 - p_j^{(\mathrm{R})})$$

we will use instead

$$n^{(\mathrm{L})} = \sum_{i \in \text{left node}} w_i^{(t)} \qquad \text{and} \quad p_j^{(\mathrm{L})} = \frac{1}{n^{(\mathrm{L})}} \sum_{i \in (\text{class } j \cap \text{left node})} w_i^{(t)}$$

How to combine the trees

Classifying a test point $\mathbf{x}_0$ is through voting with weights

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$

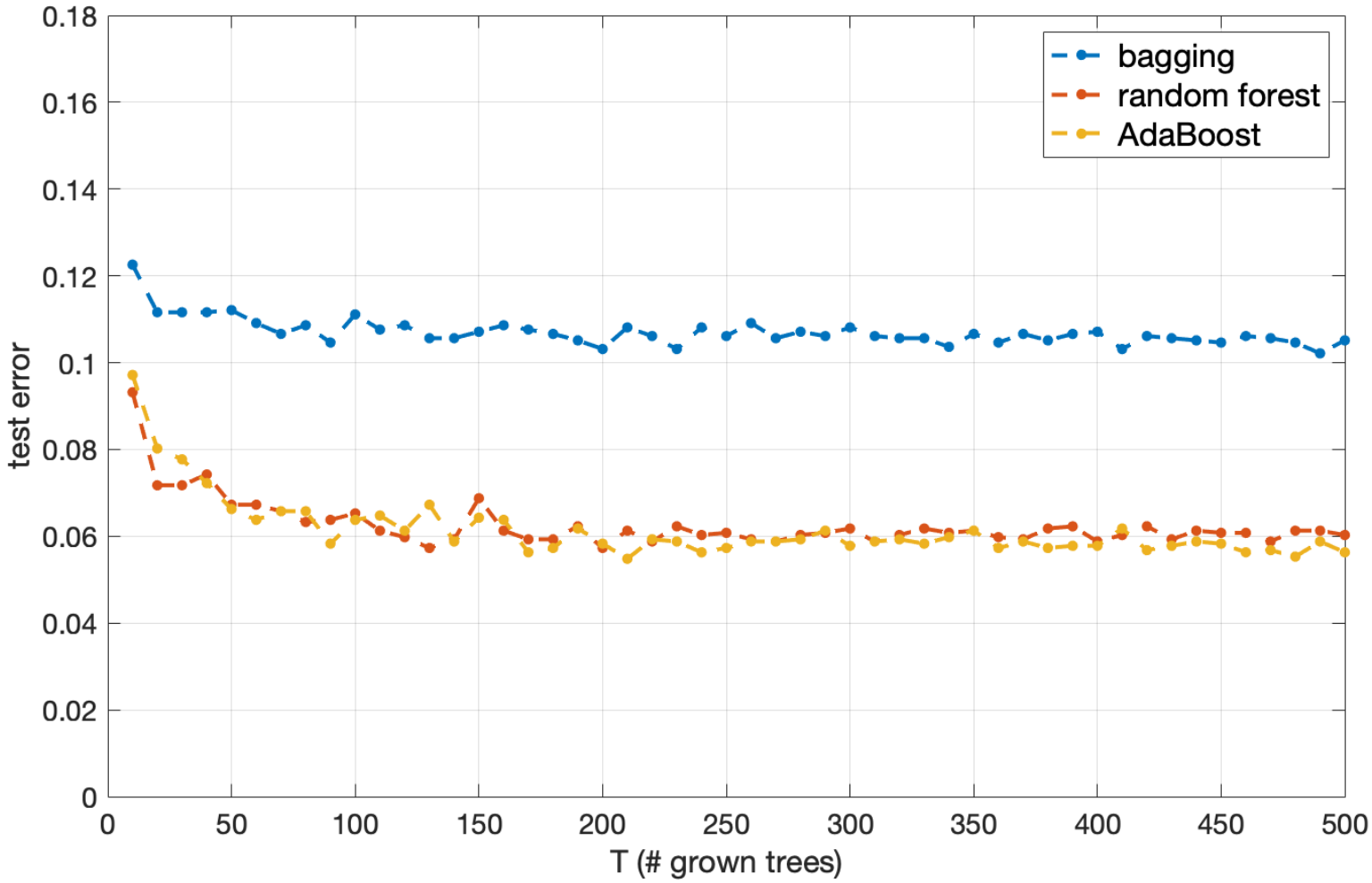
for the different trees:

$$\hat{y}_0 = \text{argmax}_j \sum_{t\,:\,C_t(\mathbf{x}_0)=j} \alpha_t$$

Remark. For binary classification with labels $y_i = \pm 1$, the above decision rule can be written as

$$\hat{y}_0 = \text{sgn}\left(\sum_{t=1}^{T} \alpha_t C_t(\mathbf{x}_0)\right)$$

Test errors of bagging, random forest, and AdaBoost on the *usps* data set



Some comments on AdaBoost

- Given any weak learner better than random guess (i.e., error rate < 0.5), AdaBoost can achieve an arbitrarily high accuracy on the training data.
- Fast convergence
- Excellent generalization performance
- Can handle many features easily
- In general, AdaBoost $\succ$ random forest $\succ$ bagging $\succ$ single tree

Further learning

- "Trees, Bagging, Random Forests and Boosting", lecture slides by Trevor Hastie of Stanford University[1]
- "Trees and Random Forests", lecture slides by Adele Cutler of Utah State University[2]
- "Boosting", lecture notes by Kilian Weinberger of Cornell University[3]

[1] http://jessica2.msri.org/attachments/10778/10778-boost.pdf

[2] http://www.math.usu.edu/adele/randomforests/uofu2013.pdf

[3] https://www.cs.cornell.edu/courses/cs4780/2021fa/lectures/lecturenote19.html

Regression trees (to be discussed in class)

Read the textbook before next Monday. We will discuss the following questions in class:

- How does a regression tree make predictions in the leaf nodes? What kind of model does it fit? And how does it compare with linear regression?
- How does binary splitting work in a regression tree? In particular, what kind of criterion is used?
- How are the different regression trees combined in an ensemble (bagging, random forest and boosting)?