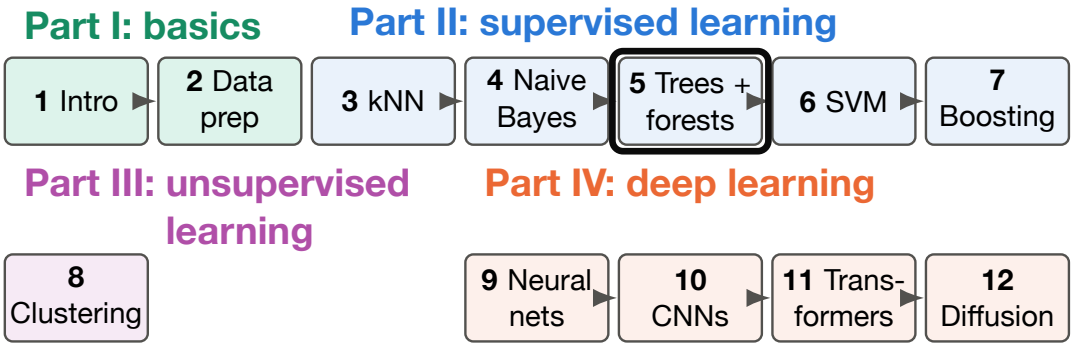
Decision trees and random forests

229352 Lecture 5: decision trees and random forests

Course map

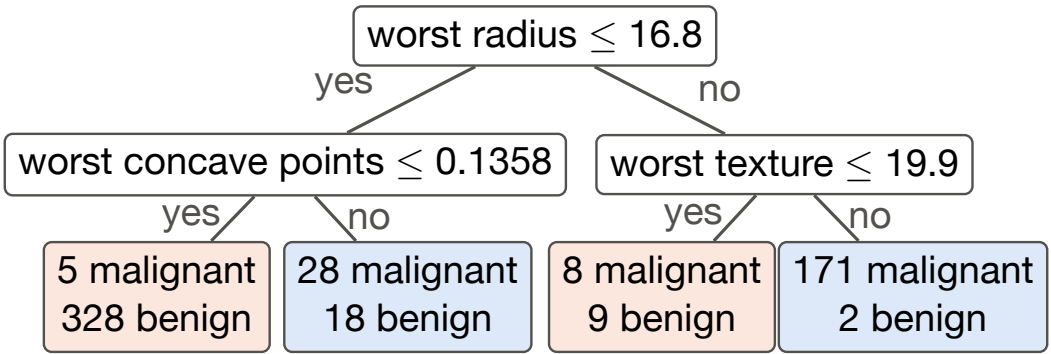

Part I: basics
Part II: supervised learning
1 Intro
2 Data prep
3 kNN
4 Naive Bayes
5 Trees + forests
6 SVM
7 Boosting
Part III: unsupervised learning
Part IV: deep learning
8 Clustering
9 Neural nets
10 CNNs
11 Trans-formers
12 Diffusion

Tumors: malignant or benign?

- The data set has 569 breast tumors. Each tumor has 30 features of the cell nuclei in a microscope image.
- Example: *worst radius*, the mean of the three largest radii of the cell nuclei in the image.
- Class 1 (blue): malignant, 212 tumors;
  class 0 (orange): benign, 357.
- Can a few rules on these features predict the class?

Breast Cancer Wisconsin (Diagnostic) data: W. Wolberg, O. Mangasarian, N. Street, W. Street, UCI Machine Learning Repository, 1993, doi:10.24432/C5DW2B, CC BY 4.0

A decision tree for the tumors



Every path has two rules of the form “feature ≤ value”.
Each bottom box is colored by the most common class of the training tumors in it.

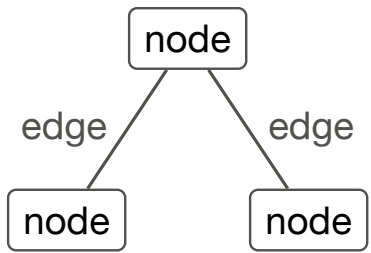
What a decision tree does

- A tree is a sequence of **rules**. A rule is a yes/no question about one feature; it splits the data by that feature's value.
- To predict, start at the top and follow the rules down to a bottom box; predict the most common class of the training points in that box.
- A rule can split a **numeric** feature (worst radius $\leq$ 16.8) or a **categorical** feature (shown later).

Parts of a tree: node and edge

Node: a box of the tree. It holds a rule, or a decision (a class).

Edge: the connection between two nodes.

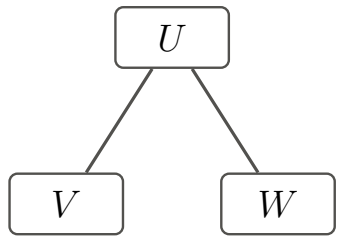


Parts of a tree: parent and child

Child of a node: a node directly below it, joined to it by an edge.

Parent of a node: the node directly above it, joined to it by an edge.

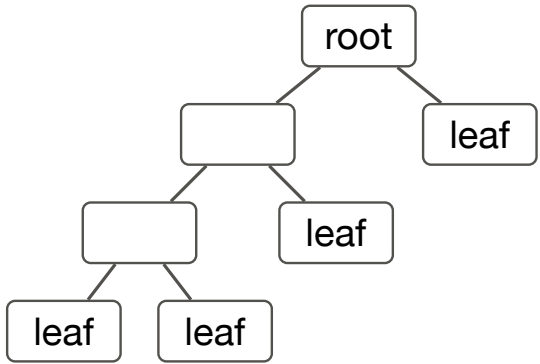
Example: V and W are children of U; U is the parent of V and W.



Parts of a tree: root and leaf

Root: the top node of the tree.

Leaf: a node with no edge below it. A leaf gives the predicted class.

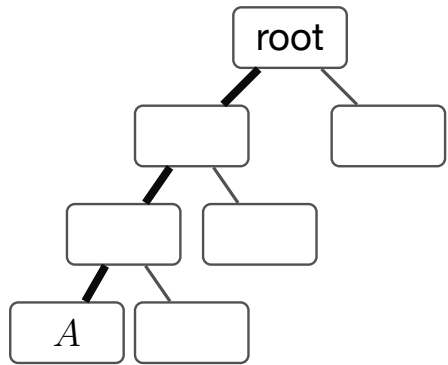


Parts of a tree: depth

Depth of a node: the number of edges from the root to that node.

Example: the depth of A is 3.

Maximum depth: the largest depth in the tree (here 3).

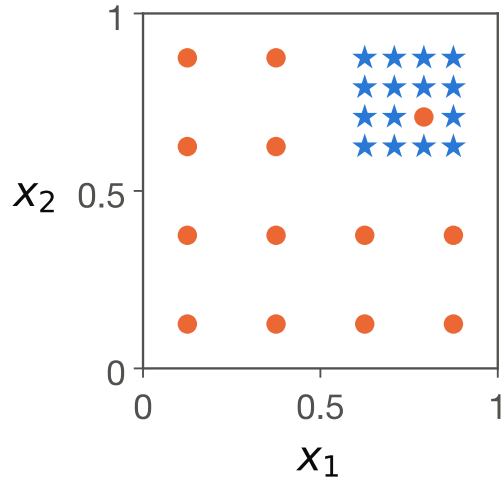


Toy data with two features

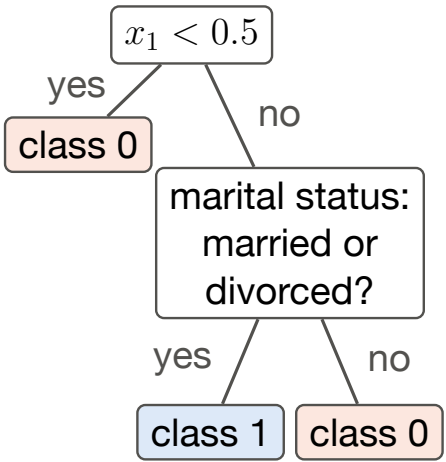
Two features: x_1 and x_2
(x_j: feature j).

stars: class 1 (15 points)
dots: class 0 (13 points)

We try maximum depth 2: at most two splits from the root to a leaf.

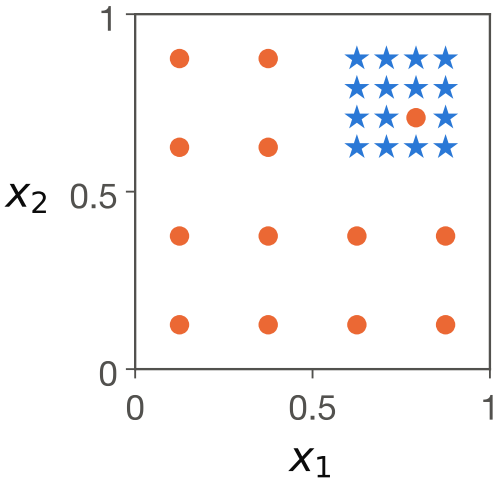


Categorical features



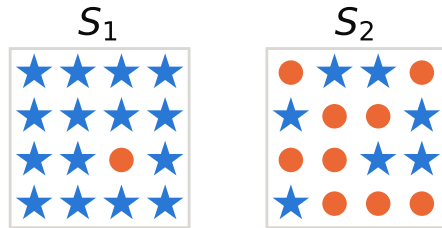
Marital status: single, married, divorced or unspecified.

Choosing a split



How can we choose the best splitting value for a numeric feature?

Two sets of points



Which set is more mixed?

p: the share of class 1 in a set. S_1: $p = 15/16$; S_2: $p = 7/16$

The **impurity** $I(S)$ measures how mixed a set S is.

Three measures of impurity

Two classes; p: the share of class 1 in the set S.

misclassification rate: $I(S) = 1 - \max(p,\, 1-p)$

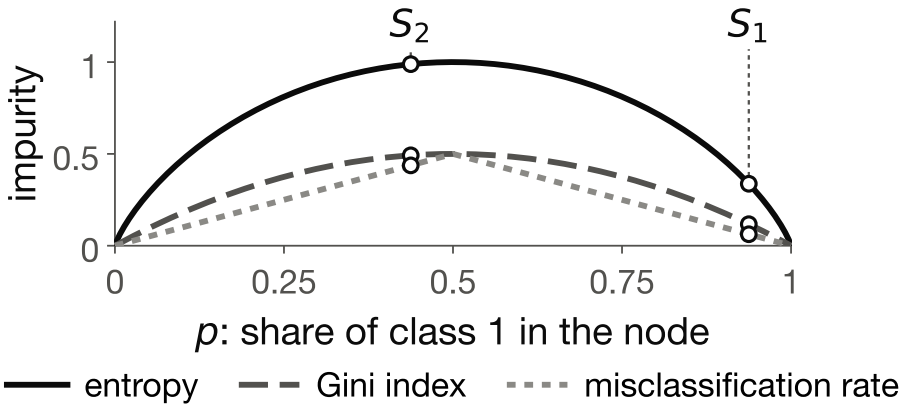
entropy: $I(S) = -p \log_2 p - (1-p) \log_2 (1-p)$

Gini index: $I(S) = 1 - p^2 - (1-p)^2 = 2p(1-p)$

Misclassification rate: the fraction of points not in the most common class.

Entropy uses log base 2, with $0 \log_2 0 = 0$ by convention.

Impurity against the share of class 1



all three are 0 at $p = 0$ and $p = 1$, and largest at $p = 0.5$

Impurity of the two sets

	misclassification	entropy	Gini
$S_1,\ p = 15/16$	0.06	0.34	0.12
$S_2,\ p = 7/16$	0.44	0.99	0.49

By all three measures, S_2 is more mixed than S_1.

Expected information of a split

A rule splits the set S of a node into two children: S_L (rule true) and S_R (rule false).

$|S|$, $|S_L|$, $|S_R|$: the numbers of points in S, S_L, S_R.

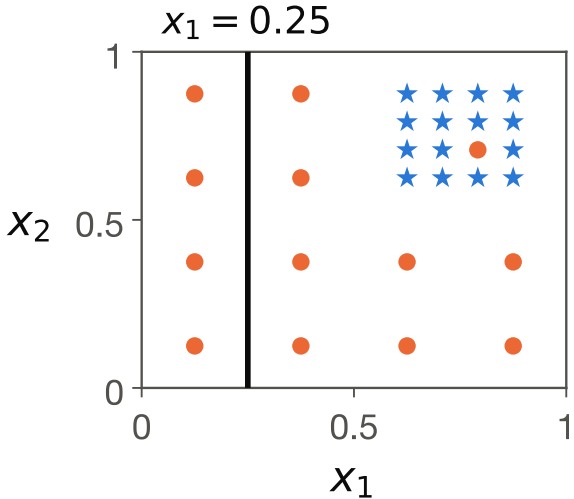
$$\mathrm{EI} = \frac{|S_L|}{|S|}\, I(S_L) + \frac{|S_R|}{|S|}\, I(S_R)$$

I: any of the three impurity measures.

The **expected information** (EI): the children's impurities, weighted by their shares of the points.

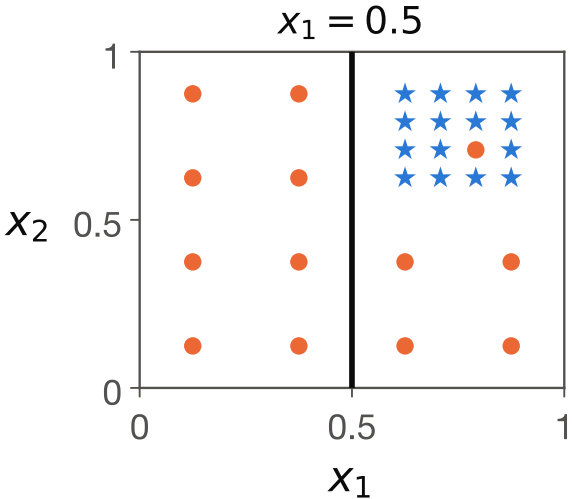
A smaller EI means purer children.

Example: the split $x_1 < 0.25$



$\mathrm{EI}\ (\mathsf{Gini}) =$

Example: the split $x_1 < 0.5$



$\mathrm{EI}\ (\mathsf{Gini}) =$

Tree splitting algorithm

`for` each **leaf** that holds more than one class:

    `for` each **feature** x_j:

        `for` each **splitting value** t:

            compute the EI of the rule $x_j < t$

    split the leaf at the feature and value with the smallest EI

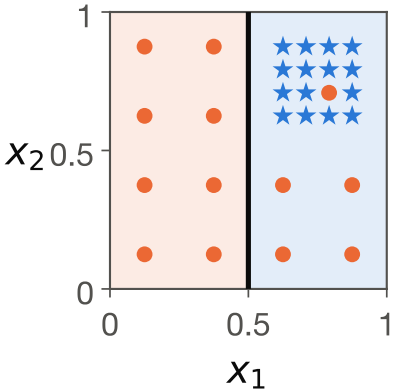
repeat until each leaf holds one class

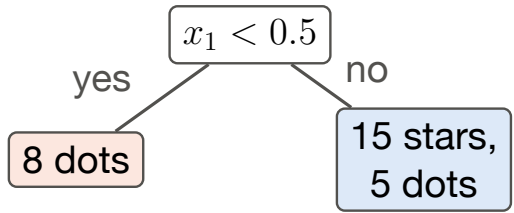
    or has reached the maximum depth

Splitting values: the midpoints between neighboring distinct sorted values of x_j among the points in the leaf.

With midpoints, "$<$" and "$\leq$" split the training points the same way.

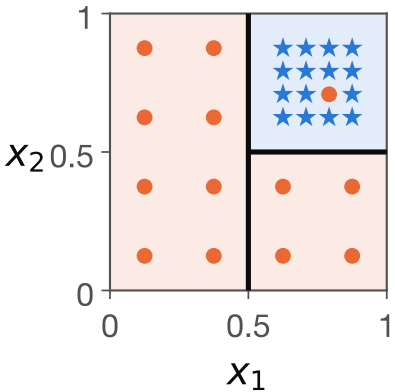
The best first split

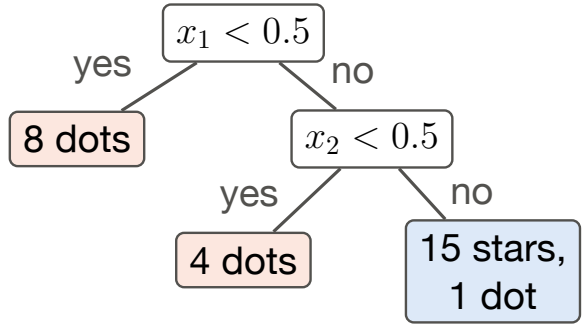




Smallest EI of the 10 candidates:
$x_1 < 0.5$.

The second split: depth 2





Then the rule with the smallest EI splits the right child.

27 of the 28 points are classified correctly (leaf class = true class).

Summary: growing a tree

- The impurity $I(S)$ measures how mixed a set S is: misclassification rate, entropy or Gini index.
- The EI of a split: the children's impurities, weighted by their shares of the points.
- At each leaf, try every feature and every midpoint; split at the smallest EI.
- Stop when each leaf holds one class or has reached the maximum depth.

Checkpoint 1

A leaf holds 2 points of class 1

and 3 points of class 0.

What is its Gini index?

Checkpoint 1

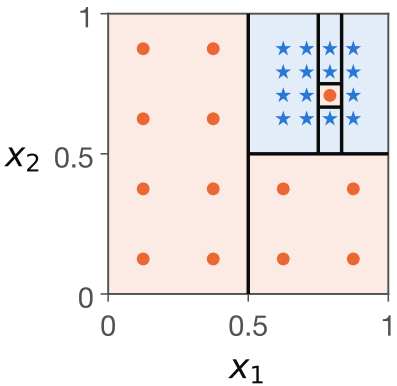
A leaf holds 2 points of class 1

and 3 points of class 0.

What is its Gini index?

$p = 2/5$: $\quad 2p(1-p) = 2 \times 0.4 \times 0.6 = 0.48$

Overfitting



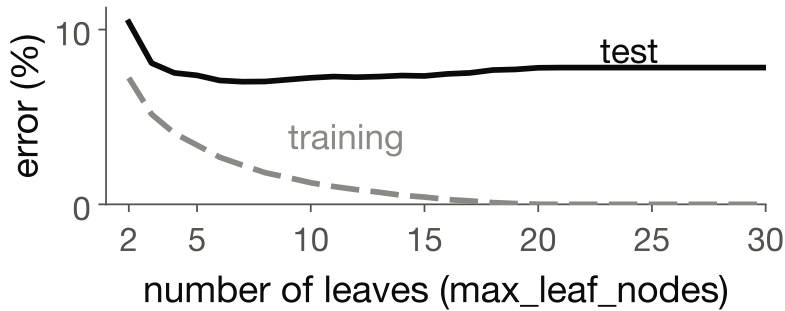
We can keep splitting until every training point is classified correctly (100% training accuracy).

This tree has 7 leaves and depth 6.

One leaf holds only the dot that lies among the stars. We treat that dot as an outlier.

The tree **overfits**: it fits the training data too closely, so its error on new data grows (next slide).

Training and test error as the tree grows



breast cancer data, 50 random splits: 70% training, 30% test

training error falls to 0; test error is lowest near 7 leaves

Settings that stop the growth

Settings of scikit-learn's `DecisionTreeClassifier`:

`max_depth`	the largest depth a leaf may have
`min_samples_split`	the fewest points a node needs to be split
`min_samples_leaf`	the fewest points in a leaf
`max_leaf_nodes`	the most leaves (used for the plot on the previous slide)

Pruning

Grow the full tree; then remove splits from the bottom up.

Cost-complexity pruning: among the pruned trees, keep the one with the smallest

$$\sum_{\text{leaves}} \frac{|S_{\text{leaf}}|}{n} I(S_{\text{leaf}}) \; + \; \alpha \times (\text{number of leaves})$$

S_{leaf}: the training points in a leaf; n: all training points.

I: the tree's criterion.

Choosing the pruning parameter

- $\alpha \geq 0$: a larger α gives the same number of leaves or fewer (scikit-learn: `ccp_alpha`).
- Candidates: the α values where the pruned tree changes.
- Choose one by 5-fold cross-validation on the training set; then refit the tree with that α on the whole training set.
- Breast cancer data: pruned trees have 8 leaves on average over the 50 splits; fully grown trees have 18.

Decision tree: pros and cons

Advantages

- Simple and easy to interpret.
- Numeric and categorical features.
- Any number of classes.
- Flexible: can fit any training set.

Disadvantages

- Fully grown, it overfits the training set.
- Small changes in the data can change the tree a lot (high variance).
- Breast cancer data: less accurate than other models.

Next: how can we improve decision trees?

One tree against other models

model	test accuracy (%)	sd
one tree, fully grown	92.4	2.3
one tree, pruned	92.7	2.3
random forest, 500 trees	95.5	1.5
k-NN, standardized	96.2	1.6

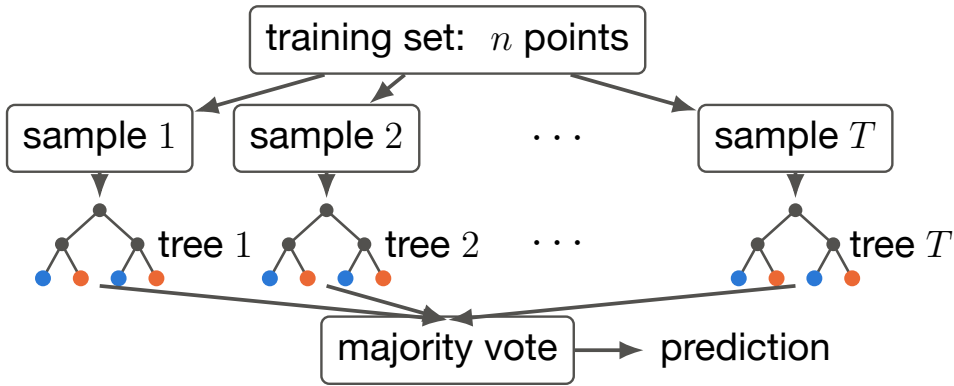
mean and sd (standard deviation) over the same 50 splits

Summary: overfitting

- A tree grown until every leaf holds one class fits the training set exactly, outliers included.
- Stop the growth early, for example with `max_depth`, or prune with `ccp_alpha` chosen by cross-validation.
- One tree is easy to read. On the breast cancer data it is less accurate than a random forest or k-NN.

Bagging

Bagging (bootstrap aggregating): fit one tree on each bootstrap sample, then take the majority vote of the trees.



Bootstrap sample: draw n training points *with replacement*.

A bootstrap sample

Points 1 to 10. Draw 10 times, with replacement:

9, 7, 6, 3, 4, 1, 1, 1, 2, 9

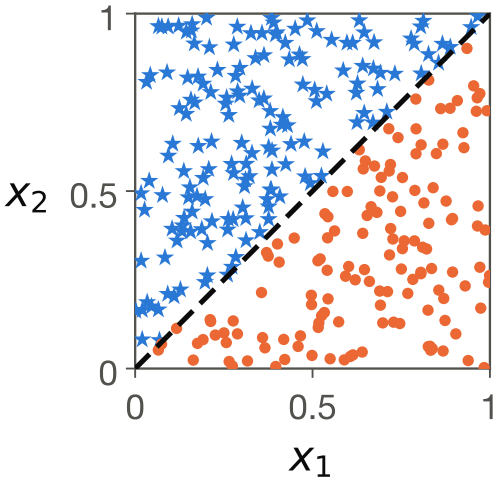
Some points are drawn more than once;
never drawn: 5, 8, 10.

A point is missed by all n draws with probability

$$(1 - 1/n)^n \approx e^{-1} \approx 0.37 \quad \text{for large } n.$$

So, on average, a bootstrap sample contains about 63% of the distinct training points (for large n); the other draws repeat points.

The diagonal problem for a single tree

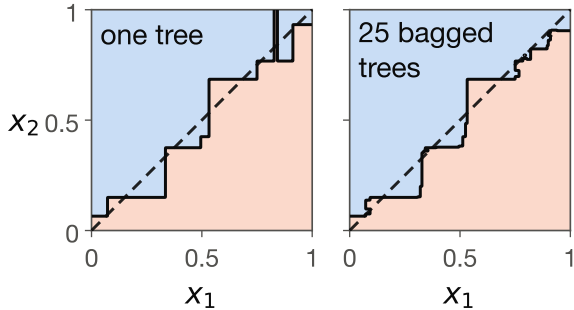


The data are 300 simulated training points: class 1 if $x_2 > x_1$, so the boundary is the dashed diagonal.

Each split of a tree is horizontal or vertical, so a tree's boundary is a staircase.

Test set: 10,000 new points.

One tree and 25 bagged trees



test error: one tree 5.8%, 25 bagged trees 5.1%

Random forest

A **random forest** is bagging of decision trees with one more source of randomness:

- At each split, choose m of the d features at random.
- Find the best split among these m features only.
- scikit-learn's default for classification: $m = \lfloor \sqrt{d} \rfloor$ (`max_features="sqrt"`).
- Breast cancer data: $d = 30$, so $m = 5$.

Hyperparameters of a random forest

A **hyperparameter** is a setting chosen before fitting.

Hyperparameters of scikit-learn's `RandomForestClassifier`, with defaults:

`n_estimators`	number of trees T	100
`max_features`	m: features tried at each split	`"sqrt"`
`max_depth`	as for one tree	no limit
`bootstrap`	a bootstrap sample per tree	`True`
`max_samples`	size of each sample	n

Random forest algorithm

`RandomForest(T, m)`, for n training points with d features:

`for` $b = 1$ to T:

draw a bootstrap sample of n points

grow a tree, called tree$_b$, on it, with no depth limit,

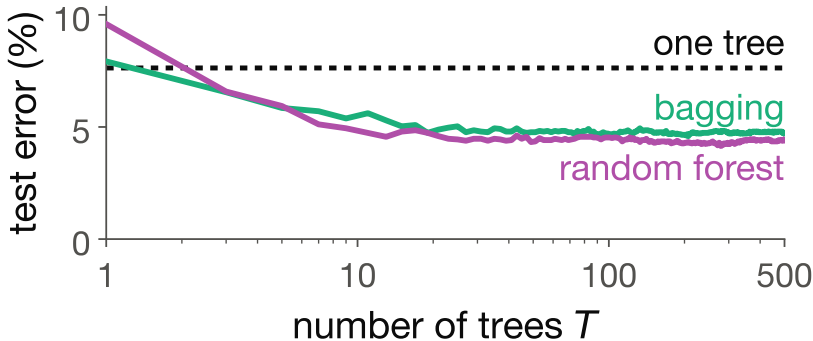
until every leaf holds one class; at each node:

choose m of the d features at random

find the best split among these m features

Prediction: majority vote of tree$_1, \ldots,$ tree$_T$.

Test error against the number of trees



breast cancer data: mean over 20 random splits

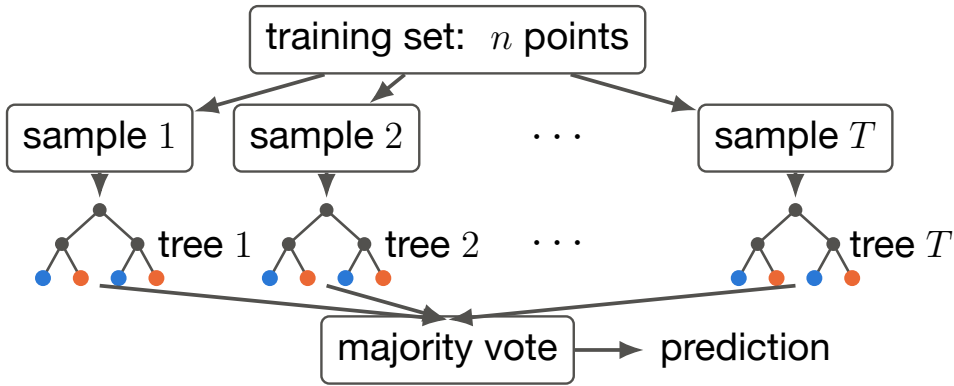
bagging and random forest level off after about 50 trees, about 3 points below one tree

Bagging and random forest at 500 trees

Breast cancer data, mean test error over the same 20 splits:

one tree, fully grown	7.6%
bagging, 500 trees	4.7%
random forest, 500 trees	4.4%

Random forest: the diagram



Each split uses m of the d features, chosen at random.

Summary: random forests

- Bagging: T trees on T bootstrap samples of n points each, then a majority vote.
- Random forest: bagging with m random features at each split (default $m = \lfloor \sqrt{d} \rfloor$).
- On the diagonal problem and the breast cancer data, bagging many trees gave a lower test error than one tree.
- Settings: `n_estimators` (T), `max_features` (m).

Checkpoint 2

In a random forest, set $m = d$:

each split may then use every feature.

Which method is this?

Checkpoint 2

In a random forest, set $m = d$:

each split may then use every feature.

Which method is this?

Bagging

each tree is grown on its own bootstrap sample,
and every split may use every feature

Trees and forests in scikit-learn

```
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
X, y = load_breast_cancer(return_X_y=True)
y = 1 - y  # 1 = malignant (scikit-learn: 0 = malignant)
X_tr, X_te, y_tr, y_te = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=0)
tree = DecisionTreeClassifier(random_state=0).fit(X_tr, y_tr)
forest = RandomForestClassifier(n_estimators=500,
                                random_state=0).fit(X_tr, y_tr)
# score: the share of test tumors classified correctly;
# one split, so these differ from the means over 50 splits
print(round(tree.score(X_te, y_te), 4))  # -> 0.9064
print(round(forest.score(X_te, y_te), 4))  # -> 0.9415
```