GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

E016350 - Artificial Intelligence

Lecture 4

Machine learning
Decision Trees and Ensamble Learning

Aleksandra Pizurica

Ghent University
Spring 2025

Outline

1. Decision trees
2. Ensemble Learning
   - Bagging
   - Random forests
3. Boosting

[R&N], Chapter 19

The slides are based on: S. Russel and P. Norvig: *Artificial Intelligence: A Modern Approach*, (Fourth Ed.), http://aima.cs.berkeley.edu/; D. Klein & P. Abbeel: CS188 Artificial Intelligence (Berkeley) and M. Charikar & Koyejo: CS221 Artificial Intelligence: Principles and Techniques (Stanford).

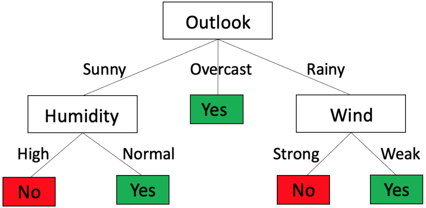
Outline

1. Decision trees
2. Ensemble Learning
   - Bagging
   - Random forests
3. Boosting

Decision trees

- Decision trees are able to learn complex, nonlinear relationships between variables, using a series of simple, **intuitive** decision rules.
- Easy to undersand and interpret. Require little or no data preparation.
- Widely used in today's machine learning approaches.

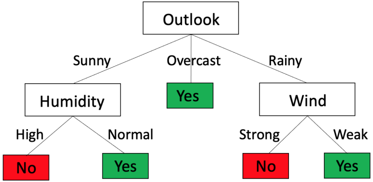
Example: Should I play tennis today?



A simple idea: start with one test, and depending on its outcome decide what the next test will be. Continue until a decision is reached.

Interpretation of a decision tree

Like any supervised ML approach, a decision tree is learned from $(\mathbf{x}, y) \in \mathcal{D}_{train}$, where $\mathbf{x}$ are the values of some features (or attributes) $\mathbf{X}$ and y is the output label.



- Internal nodes test a feature X_i

  In this tree: $X_1 = Outlook$, $X_2 = Humidity$, $X_3 = Wind$
- Branching is determined by the feature value

  E.g. $x_3 = wind \in \{Strong, Weak\}$
- Leaf nodes are outputs (predictions):
  - numerical (regression tree); categorical (classification tree)
  - tuple-valued variable (multi-target trees) or $P(y|\mathbf{x})$ (probability estimation trees)

Case study: "Restaurant domain"

Decide whether to wait for a table in a restaurant depending on the following attributes (R&N):

1. Alternate (Alt): Is there a suitable alternative restaurant nearby?
2. Bar (Bar): Is there a comfortable bar area in the restaurant, where I can wait?
3. Fri/Sat (Fri): True on Fridays/Saturdays
4. Hungry (Hun): Are we hungry?
5. Patrons (Pat): How many people are in the restaurant ($None$, $Some$ or $Full$)
6. Price ($Price$): the restaurant's price range (\$, \$\$, \$\$\$)
7. Raining ($Rain$): Is it raining outside?
8. Reservation (Res): Did we make a reservation?
9. Type ($Type$): the kind of restaurant (French, Italian, Thai or burger)
10. WaitEstimate (Est): the wait time estimated by the host (0-10, 10-30, 30-60, or>60 min)

Decision trees

Examples for the restaurant domain R&N, table 19.2 (adapted notation)

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
1	T	F	F	T	Some	$$$	F	T	French	0–10	T
2	T	F	F	T	Full	$	F	F	Thai	30–60	F
3	F	T	F	F	Some	$	F	F	Burger	0–10	T
4	T	F	T	T	Full	$	F	F	Thai	10–30	T
5	T	F	T	F	Full	$$$	F	T	French	>60	F
6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
7	F	T	F	F	None	$	T	F	Burger	0–10	F
8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
9	F	T	T	F	Full	$	T	F	Burger	>60	F
10	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
11	F	F	F	F	None	$	F	F	Thai	0–10	F
12	T	T	T	T	Full	$	F	F	Burger	30–60	T

Each raw is an example $(\mathbf{x}^{(i)}, y^{(i)})$, where the output $y^{(i)}$ is true (T) or false (F).

Decision trees

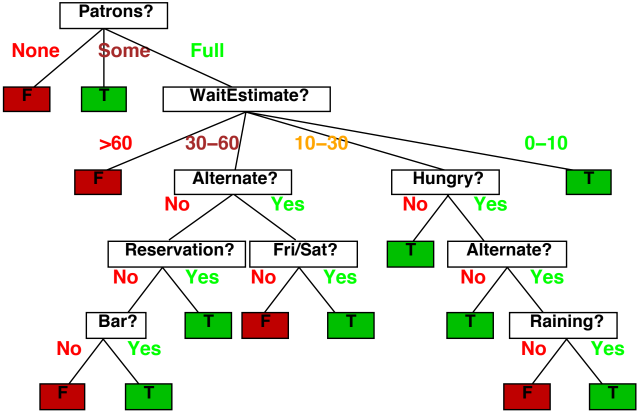
Examples for the restaurant domain R&N, table 19.2 (adapted notation)

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
1	T	F	F	T	Some	$$$	F	T	French	0–10	T
2	T	F	F	T	Full	$	F	F	Thai	30–60	F
3	F	T	F	F	Some	$	F	F	Burger	0–10	T
4	T	F	T	T	Full	$	F	F	Thai	10–30	T
$(\mathbf{x}^{(5)}, y^{(5)})$	T	F	T	F	Full	$$$	F	T	French	>60	F
6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
7	F	T	F	F	None	$	T	F	Burger	0–10	F
8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
9	F	T	T	F	Full	$	T	F	Burger	>60	F
10	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
11	F	F	F	F	None	$	F	F	Thai	0–10	F
12	T	T	T	T	Full	$	F	F	Burger	30–60	T

Each raw is an example $(\mathbf{x}^{(i)}, y^{(i)})$, where the output $y^{(i)}$ is true (T) or false (F).

Decision trees

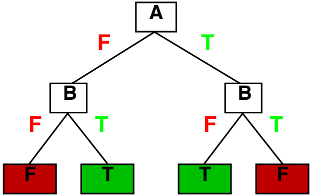
One possible representation for hypotheses
E.g., here is the “true” tree for deciding whether to wait:



Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Trivially, there is a consistent decision tree for any training set
with one path to leaf for each example (unless f nondeterministic in $\mathbf{x}$)
but it probably won't generalize to new examples

We prefer to find more **compact** decision trees

Expressiveness cont'd

How many distinct decision trees with n Boolean attributes??

$=$ number of Boolean functions

$=$ number of distinct truth tables with 2^n rows $= 2^{2^n}$

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 ($\approx 10^{19}$) trees

With 10 Boolean attributes there are about 10^{308} trees

More expressive hypothesis space

– increases chance that target function can be expressed ☺

– increases number of hypotheses consistent w/ training set

$\Longrightarrow$ may get worse predictions ☹

Decision tree learning: Idea

Aim: find a small tree consistent with the training examples

Idea: (recursively) choose "most significant" attribute as root of (sub)tree:

- Start with the whole training set and an empty decision tree
- Pick a feature that gives the best split
- Split on that feature and recurse on sub-partitions

Decision tree learning algorithm

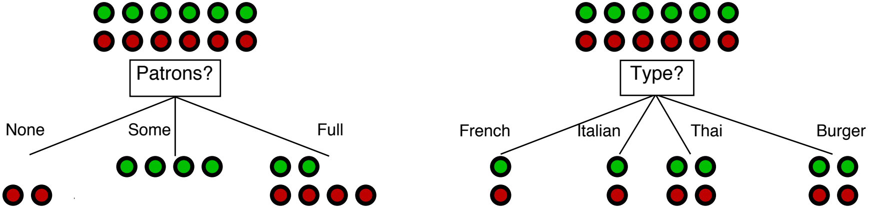
```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree

  if examples is empty then return PLURALITY-VALUE(parent_examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PLURALITY-VALUE(examples)
  else
     A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
     tree ← a new decision tree with root test A
     for each value v of A do
        exs ← {e : e ∈ examples and e.A = v}
        subtree ← LEARN-DECISION-TREE(exs, attributes − A, examples)
        add a branch to tree with label (A = v) and subtree subtree
     return tree
```

The function IMPORTANCE measures the importance of attributes (as explained next). The PLURALITY-VALUE function selects the most common output value among a set of examples, breaking ties randomly.

Choosing attribute tests

Idea: a good (=**important**) attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice – gives **information** about the classification

Information gain

- Information answers questions
- The more clueless we are about the answer initially, the more information is contained in the answer
- 1 bit = answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$
- Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} -P_i \log_2 P_i$$

(also called entropy of the prior)

Information gain, cont'd

Suppose we have p positive and n negative examples at the root
$\implies H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example
E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification

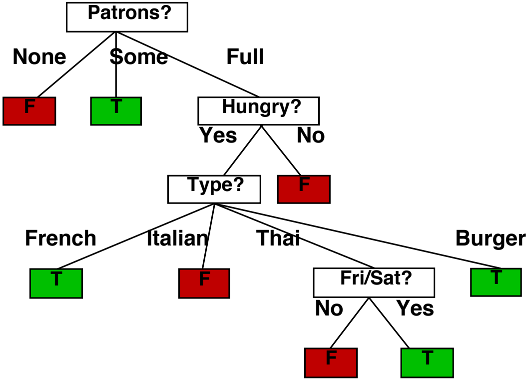
Let E_i have p_i positive and n_i negative examples
$\implies H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example
$\implies$ **expected** number of bits per example over all branches is

$$\sum_i \frac{p_i + n_i}{p+n} H\left(\left\langle \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i} \right\rangle\right)$$

For $Patrons?$, this is 0.459 bits, for $Type$ this is (still) 1 bit
$\implies$ choose the attribute that minimizes the remaining information needed

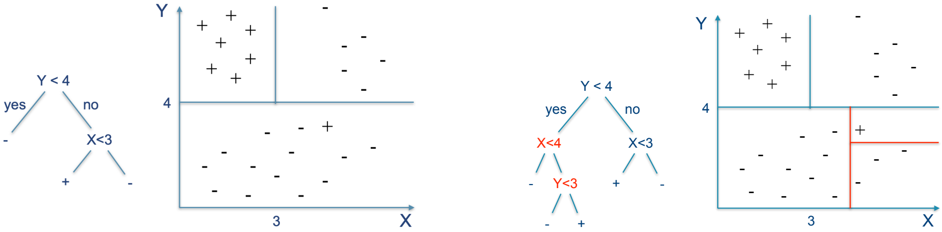
Information gain cont'd

Decision tree learned from the 12 examples:



Substantially simpler than "true" tree — a more complex hypothesis isn't justified by small amount of data

Some considerations



Left: a small tree fits the training data almost perfectly. It can be grown to fit perfectly (right), but a relatively large area to the right will then be predicted positive, while the data contains very little evidence for this.

Outline

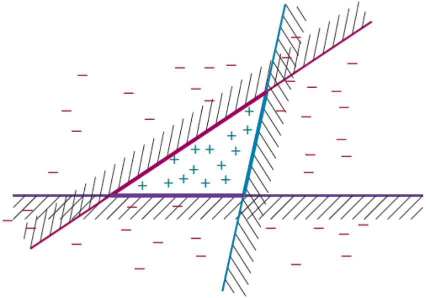
Ensemble Learning

Idea: select a collection, or ensemble, of hypotheses, $h_1, h_2, \ldots, h_n$, and combine their predictions by averaging, voting, or another level of machine learning.

- Individual h_i: **base models**; their combination: **ensamble model**

Motivation:

- Reduce **bias**: an ensamble can be more expressive than a single base model



- Reduce **variance**, e.g., majority voting counteracts individual classifier errors

Ensemble Learning: Idealized Example

Consider an ensemble of $K = 5$ binary classifiers combined by majority voting
$\rightarrow$ To missclassify an example at lest 3 classifiers have missclassify it

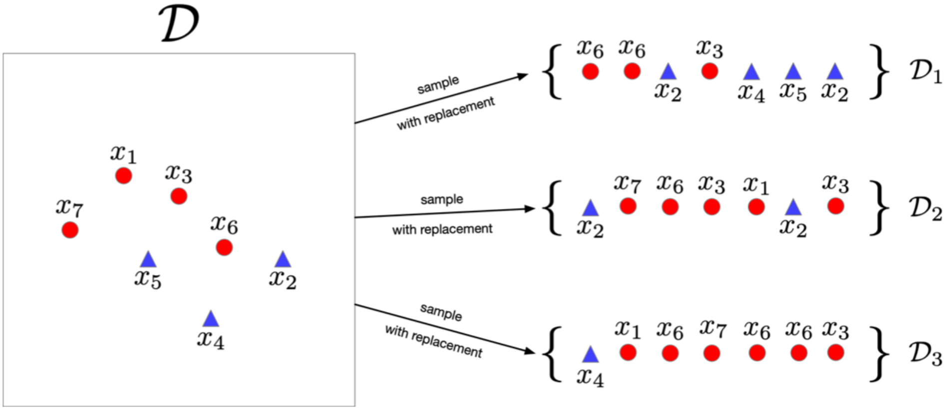
Suppose

- A single classifier trained on $\mathcal{D}_{train}$ is correct in 80% of cases
- We create an ensemble of 5 classifiers
  - individual classifiers trained on different subsets of $\mathcal{D}_{train}$ are **independent**
  - accuracy of each individual classifier is only 75%
- Then the ensemble's majority vote is correct in nearly 90%
- For $K = 17$, and the same accuracies of the base models, this would be 99%

In practice, the independence assumption is unreasonable. **Why?**
But if base classifiers are not strongly correlated
ensemble learning will make fewer miss-classifications.

Outline

Bagging (Bootstrap aggregating)

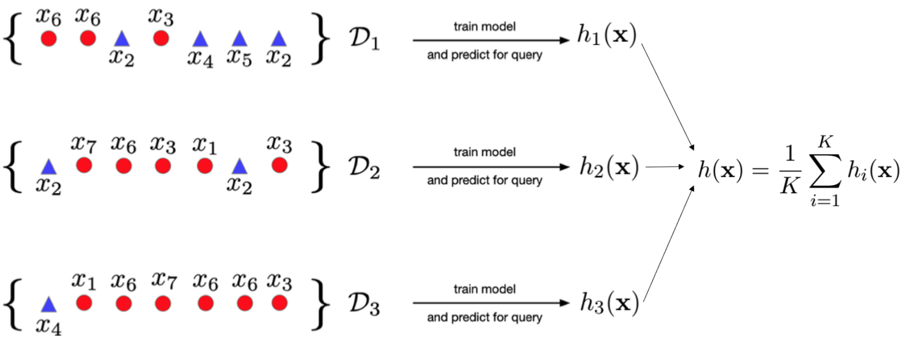


Generate K distinct training sets by sampling with replacement from $\mathcal{D}$.

i.e., randomly pick N examples from the training set, but each of those picks might be an example we picked before.

Bagging, cont'd

- For regression problems:



- For classification problems, we take instead of averaging the majority vote
- Bagging reduces variance and is most commonly used with decision trees
  - Appropriate because decision trees are unstable
    (slightly different $\mathcal{D}$ can lead to a quite tree)

Bagging: Effect on variance

Consider a regression problem and let $\hat{y}_i = h_i(\mathbf{x})$ be the prediction of the ith base model. Bagging gives:

$$h(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} h_i(\mathbf{x}) \quad \text{i.e.,} \quad \hat{y} = \frac{1}{K}\sum_{i=1}^{K} \hat{y}_i$$

Does bagging influence bias? And variance?
To simplify, assume that $\hat{y}_i$ are independent. Then

$$\mathbb{E}(\hat{y}) = \mathbb{E}\Big(\frac{1}{K}\sum_{i=1}^{K} \hat{y}_i\Big) = \mathbb{E}(\hat{y}_i) \quad \text{and} \quad Var(\hat{y}) = Var\Big(\frac{1}{K}\sum_{i=1}^{K} \hat{y}_i\Big) = \frac{1}{K} Var(y_i)$$

- Bagging reduces variance and is most commonly used with decision trees
  - Appropriate because decision trees are unstable
    (slightly different $\mathcal{D}$ can lead to a quite different tree)

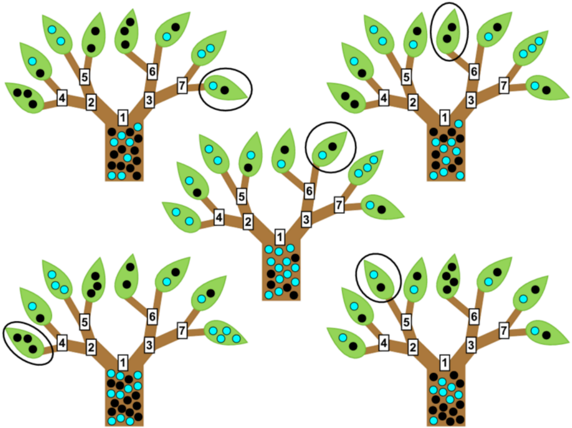
Outline

1. Decision trees
2. Ensemble Learning
   - Bagging
   - Random forests
3. Boosting

Random forests

- Random forests = bagged decision trees, with one extra trick to decorrelate the predictions
- When choosing each node of the decision tree, choose a random set of d input features, and only consider splits on those features
- Random forests are one of the most widely used ML algorithms

Extensions: Random Forest



Paul T Baker *et al*. Multivariate Classification with Random Forests for Gravitational Wave Searches of Black Hole Binary Coalescence. *Phys. Rev. D*, vol. 91 (2015).

Random decision **forest** combines a multitude of decision trees (T.K. Ho, 1995; L. Breiman, 2001; A. Cutler, 2005)

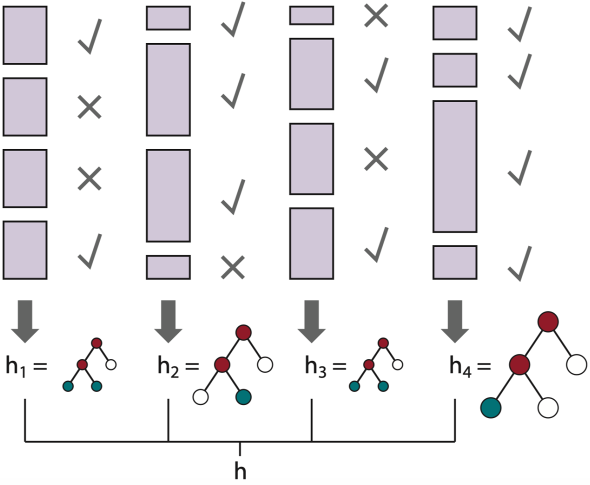
Output:

- The mode of the classes (in classification tasks)
- Mean prediction (in regression tasks)

Bagging (Bootstrap aggregating) — improve the performance by combining classifications on randomly generated training sets. Reduces variance and helps to avoid overfitting

Outline

Boosting



- Samples can have different weights
- Generate new hypotheses by giving more weight to difficult-to-classify samples
- Hypotheses that do better on their respective weighted training sets get more weight finally:

$$h(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} z_i h_i(\mathbf{x})$$

Next lesson

- Perceptron
- Neural networks