MLEA: Machine Learning

Course 3: Decision Tree, Bagging, Boosting

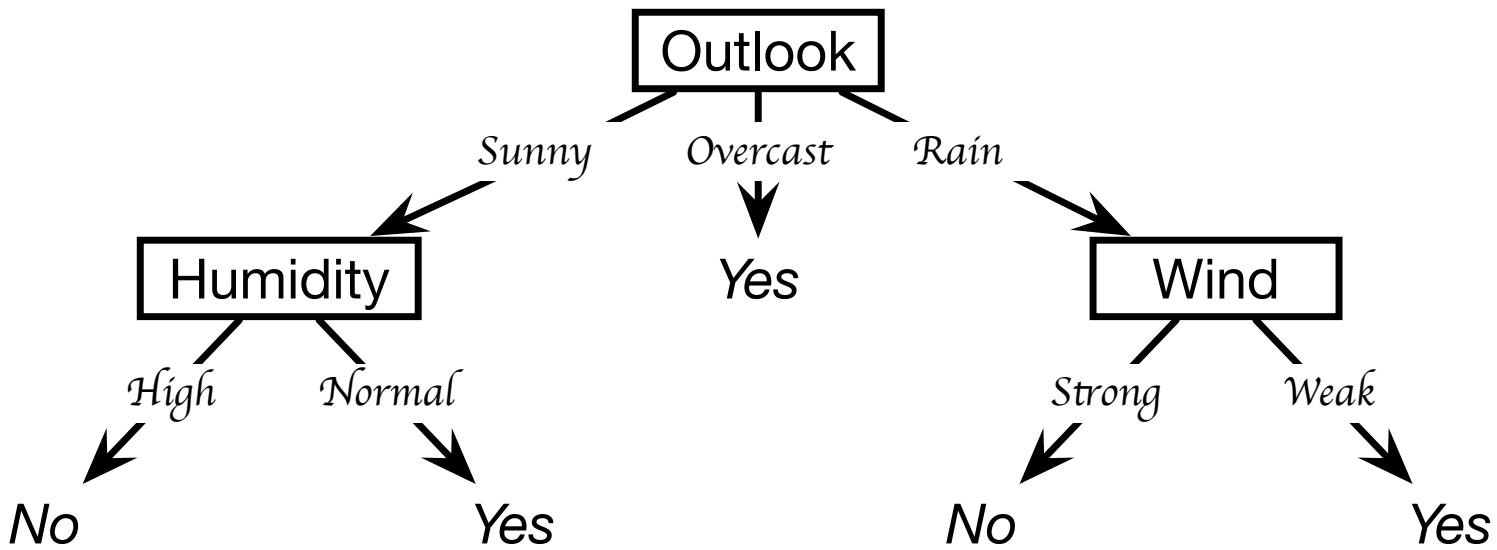
Réda DEHAK
reda.dehak@epita.fr

EPITA

November 17th, 2022

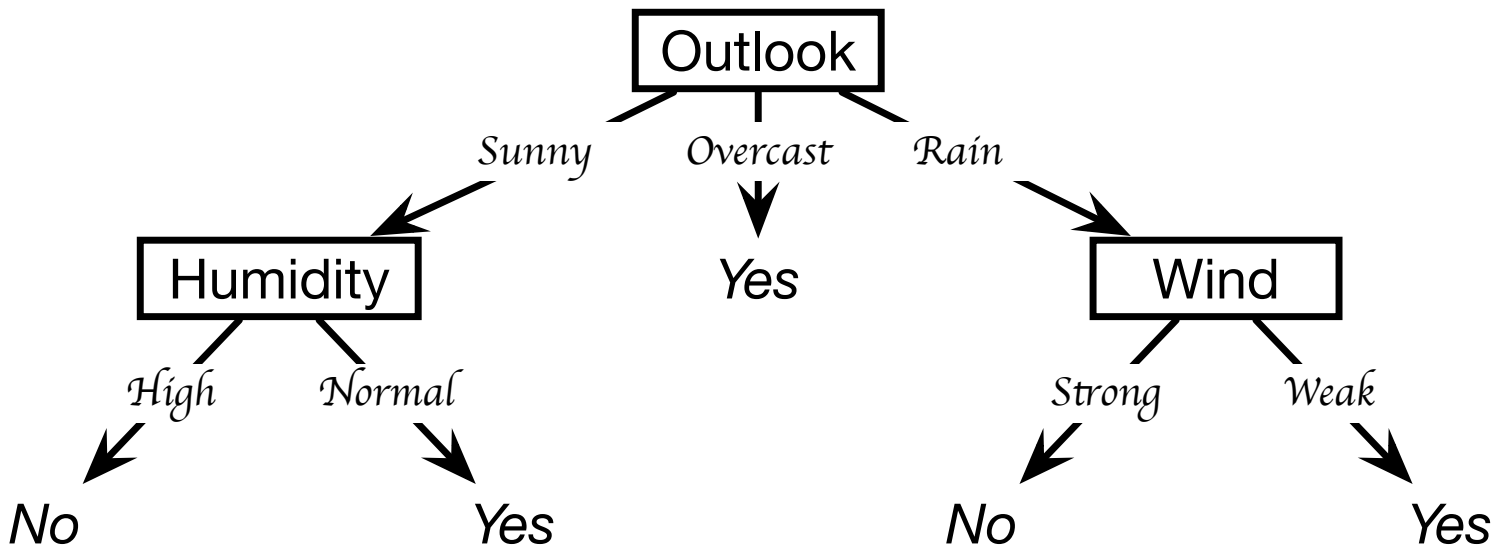
Decision Trees: Use Nodes and Leaves

- Decision Trees are supervised learning methods used for classification and regression.
- Learning simple decision rules inferred from the data features.



Decision Trees: Use Nodes and Leaves

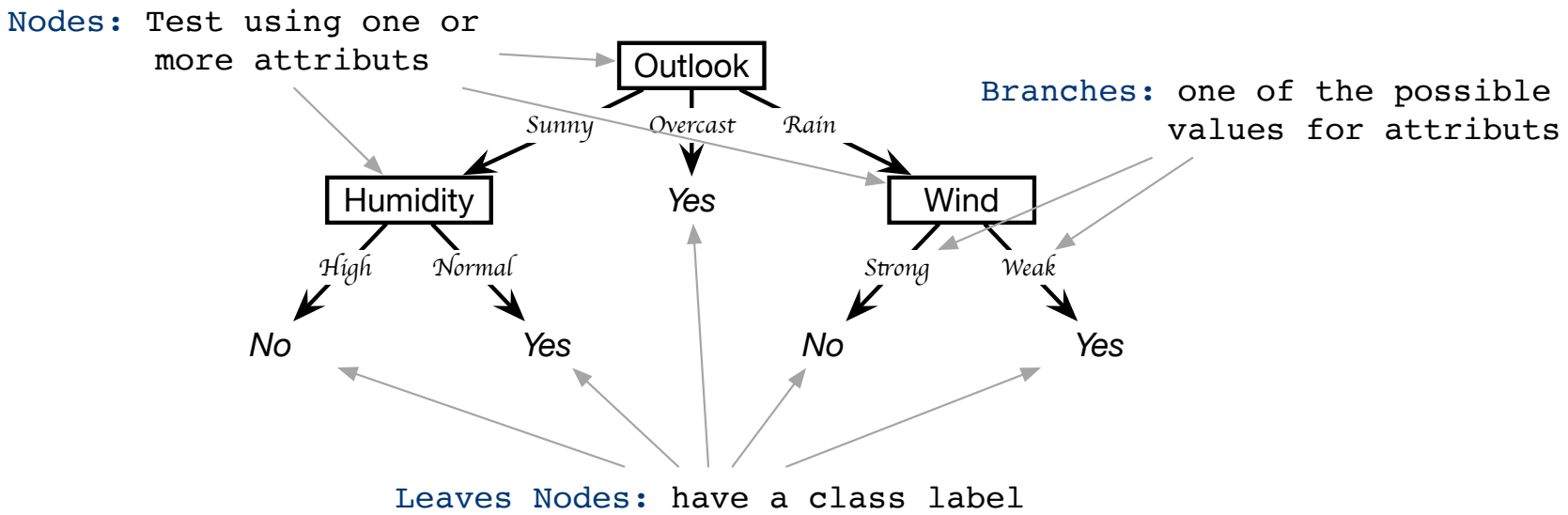
- Decision Trees are supervised learning methods used for classification and regression.
- Learning simple decision rules inferred from the data features.



Human interpretable!

Decision Trees: Use Nodes and Leaves

- Each internal node tests an attribute x_i
- One branch for each possible attribute value $x_i = v$
- Each leaf assigns a class y
- To classify input x: traverse the tree from root to leaf, output the labeled y



Advantages

- Interpretation: explain complex data, result easy to analyse, interpret and understand.
- Adaptation: easy to update to new data.
- Different types of variables: categorical(discrete), numerical(continuous).
- Dealing with missing labels.
- They perfom automatic feature selection.

Hypothesis space: What functions can be represented

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

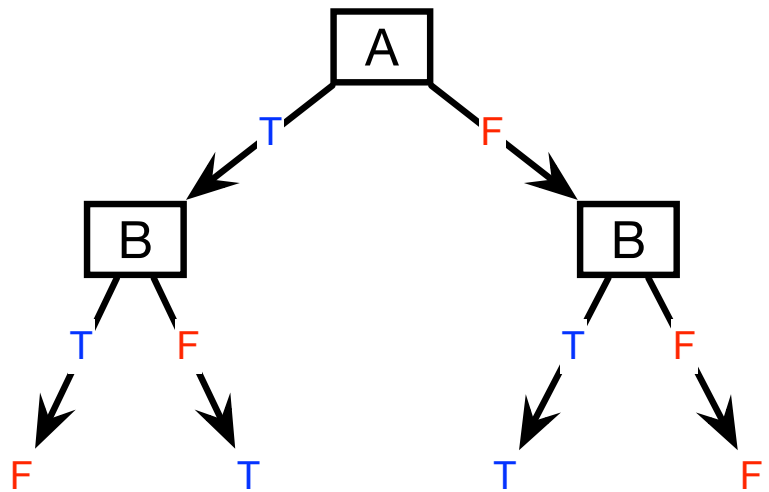
What functions can be represented?

- Decision trees can represent any function of the input attributes!
- For Boolean functions, path to leaf gives truth table rows
- Could require exponentially many nodes

What functions can be represented?

- Decision trees can represent any function of the input attributes!
- For Boolean functions, path to leaf gives truth table rows
- Could require exponentially many nodes

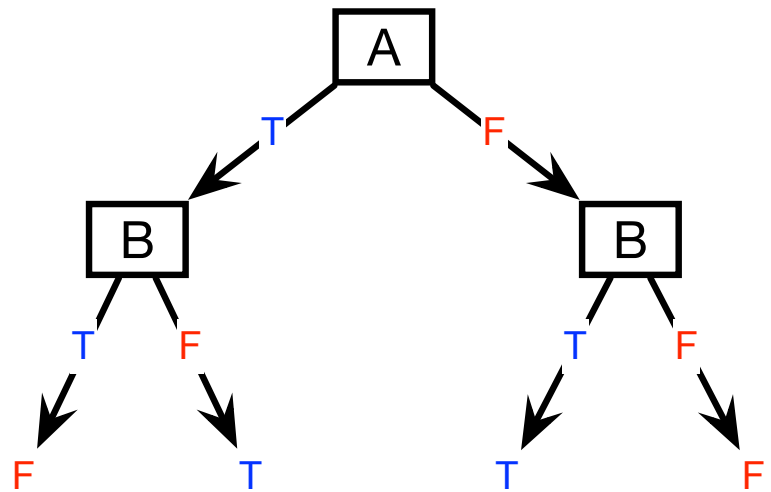
A	B	A **xor** B
T	T	F
T	F	T
F	T	T
F	F	F



What functions can be represented?

- Decision trees can represent any function of the input attributes!
- For Boolean functions, path to leaf gives truth table rows
- Could require exponentially many nodes

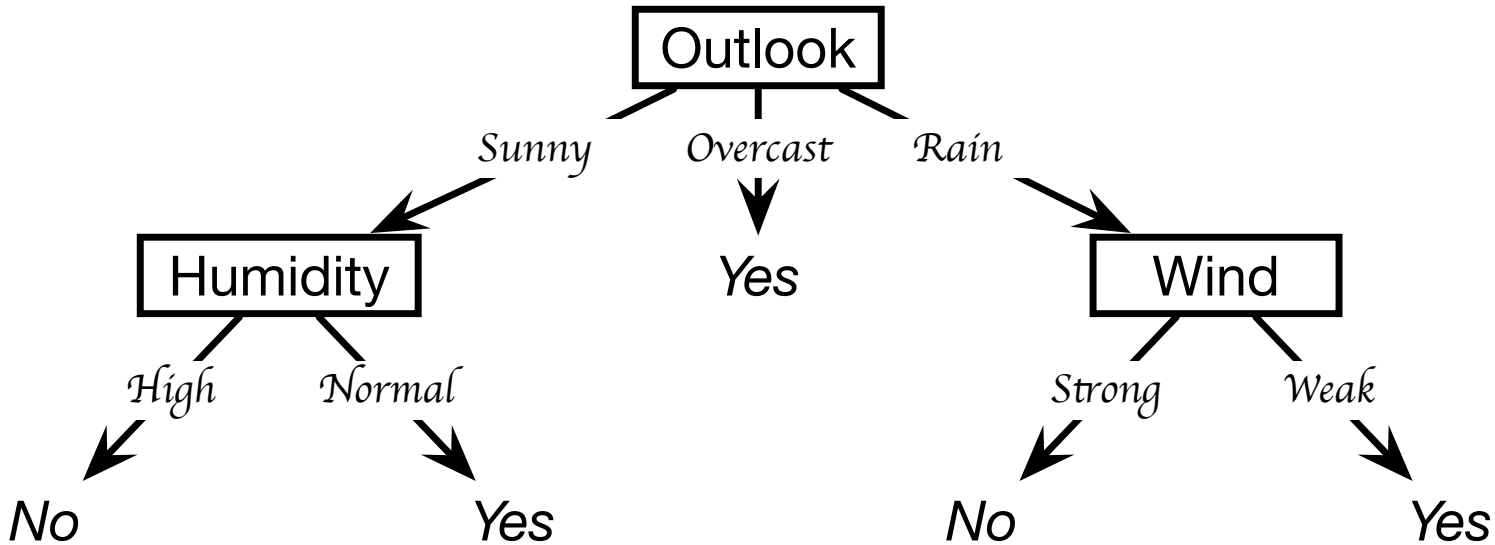
A	B	A **xor** B
T	T	F
T	F	T
F	T	T
F	F	F



$$(A \wedge \bar{B}) \vee (\bar{A} \wedge B)$$

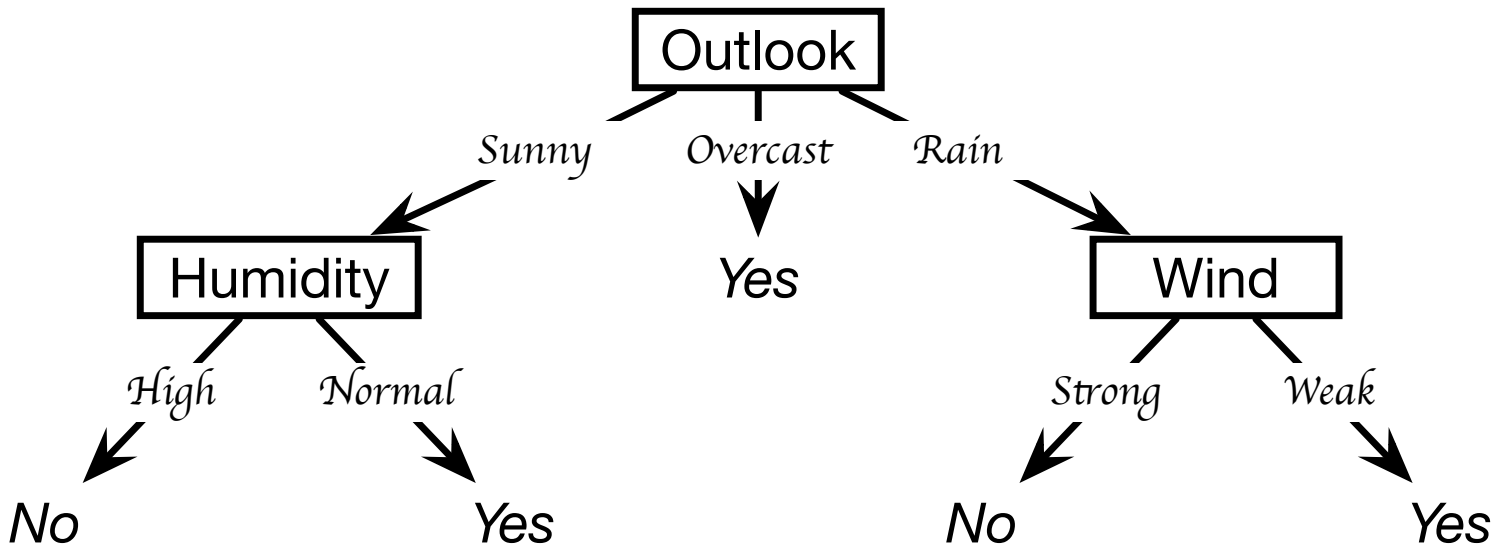
What funcGons can be represented?

- Decision trees can represent any function of the input attributes!
- For Boolean functions, path to leaf gives truth table rows
- Could require exponentially many nodes



What funcGons can be represented?

- Decision trees can represent any function of the input attributes!
- For Boolean functions, path to leaf gives truth table rows
- Could require exponentially many nodes



$$\big(\,(\text{Outlook} = \text{Sunny}) \wedge (\text{Humidity} = \text{Normal})\,\big) \vee (\text{Outlook} = \text{Overcast}) \vee \big(\,(\text{Outlook} = \text{Rain}) \wedge (\text{Wind} = \text{Weak})\,\big)$$

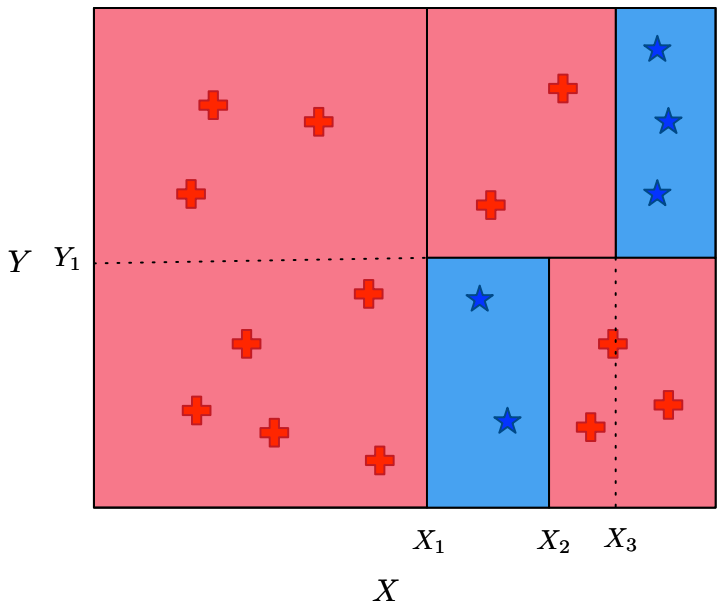
Exercises

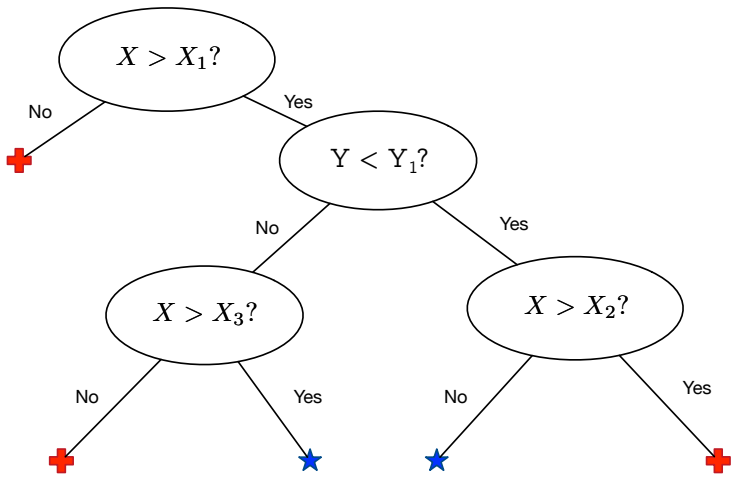
Give the decision trees to represent the following boolean Functions:

- $A \wedge \bar{B}$
- $A \vee [B \wedge C]$
- $A \oplus B$
- $[A \wedge B] \vee [C \wedge \bar{D} \wedge E]$

Different Types of Questions

- Decision trees.
  - $X \in \{\text{blue}, \text{white}, \text{red}\}$: categorical questions.
  - $X \leq a$: continuous variables.
- Binary space partition (BSP) trees:
  - $\sum_{i=1}^{n} \alpha_i X_i \leq a$: partitioning with convex polyhedral regions.


Y
Y_1
X_1
X_2
X_3
X


$X > X_1$?
No
Yes
$Y < Y_1$?
No
Yes
$X > X_3$?
$X > X_2$?
No
Yes
No
Yes

Prediction

- In each tree leaf(Region R_t):
  - Classification: majority vote.

$$\hat{y}_t = \underset{y \in Y}{\operatorname{argmax}} \left| \{ x_i \in R_t : i \in [1, m], y_i = y \} \right|$$

  - Regression: Average value.

$$\hat{y}_t = \frac{1}{|R_t|} \sum_{x_i \in R_t} y_i$$

For confident prediction, need enough points in each region

Learning simplest decision tree is NP-hard

How to build a decision tree from data?

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil and Rivest, 1976].
- Resort to a greedy heuristic procedure to compute a locally opimal MLE. Recursive Algorithm:
  1. Start from empty decision tree
  2. Split data using the **next best attribute (feature)** into corresponding subsets.
  3. reiterate with resulting subsets.

Learning simplest decision tree is NP-hard

How to build a decision tree from data?

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil and Rivest, 1976].
- Resort to a greedy heuristic procedure to compute a locally opimal MLE. Recursive Algorithm:
  1. Start from empty decision tree
  2. Split data using the **next best attribute (feature)** into corresponding subsets.
  3. reiterate with resulting subsets.
- Method used by **CART** [Breiman et al., 1984], **C4.5** [Quinlan, 1993] and **ID3** [Quinlan, 1986].

Learning Algorithm

Function BuildTree($\mathcal{I}$, *depth*)**:**

Input: Training set of instances $\mathcal{I}$ belonging to class c_1, c_2

Result: Decision Tree T

T.prediction $= \underset{y \in Y}{\text{argmax}} \left| \{ x_i \in R_t : i \in [1, m], y_i = y \} \right|$

$(j^*, t^*, \mathcal{I}_L, \mathcal{I}_R) = \text{split}(\mathcal{I})$

if *not worthSplittin(depth, cost,* $\mathcal{I}_L$*,* $\mathcal{I}_R$*)* **then**

  return *leaf node* T *with class* T*.prediction*

end

T.test $= x_{j^*} \leq t^*$

T.Left $=$ BuildTree($\mathcal{I}_L$, depth $+ 1$)

T.Right $=$ BuildTree($\mathcal{I}_R$, depth $+1$)

return T

Algorithm 1: BuildTree($\mathcal{I}$, *depth*) : Recursive procedure to learn a classification/regression Tree

Learning

The split function choose the best feature j^*, and the best value t^* for that feature:

$$(j^*, t^*) = \underset{j \in \{1,\dots,D\}, t \in \mathcal{T}_j}{\arg\min} \operatorname{cost}\left(\{x_i, y_i : x_{ij} \leq t\}\right) + \operatorname{cost}\left(\{x_i, y_i : x_{ij} > t\}\right)$$

Stopping Heuristics

- Small reduction in the cost function.

$$\triangleq \text{cost}(\mathcal{I}) - \left(\frac{|\mathcal{I}_L|}{|\mathcal{I}|} \text{cost}(\mathcal{I}_L) + \frac{|\mathcal{I}_R|}{|\mathcal{I}|} \text{cost}(\mathcal{I}_R) \right)$$

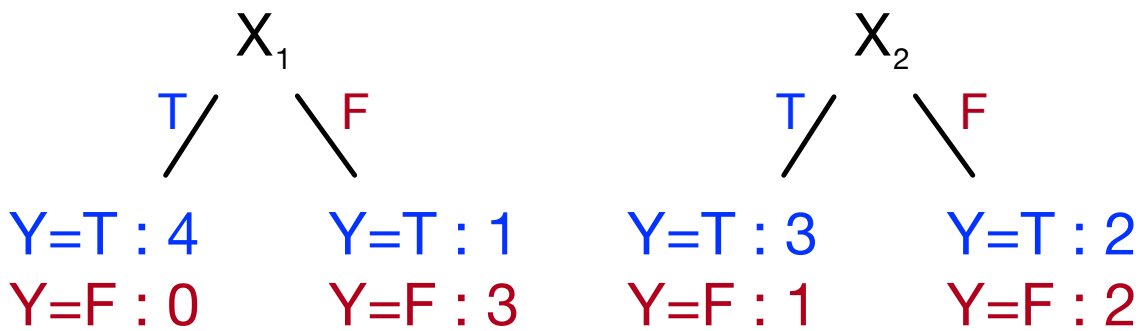
- Maximum desired depth.
- is the distribution in $\mathcal{I}_L$ and $\mathcal{I}_R$ sufficiently homogeneous (all labels are the same, so the distribution is **pure**)?
- Is the number of examples in either $\mathcal{I}_L$ or $\mathcal{I}_R$ too small?

Regression cost

$$\begin{aligned} \text{cost}(\mathcal{I}) &= \sum_{i \in \mathcal{I}} (y_i - \bar{y})^2 \\ \bar{y} &= \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{I}} y_i \end{aligned}$$

Splittng: choosing a good attribute

Would we prefer to split on X_1 or X_2?



Idea: use counts at leaves to define probability distributions, so we can measure uncertainty!

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

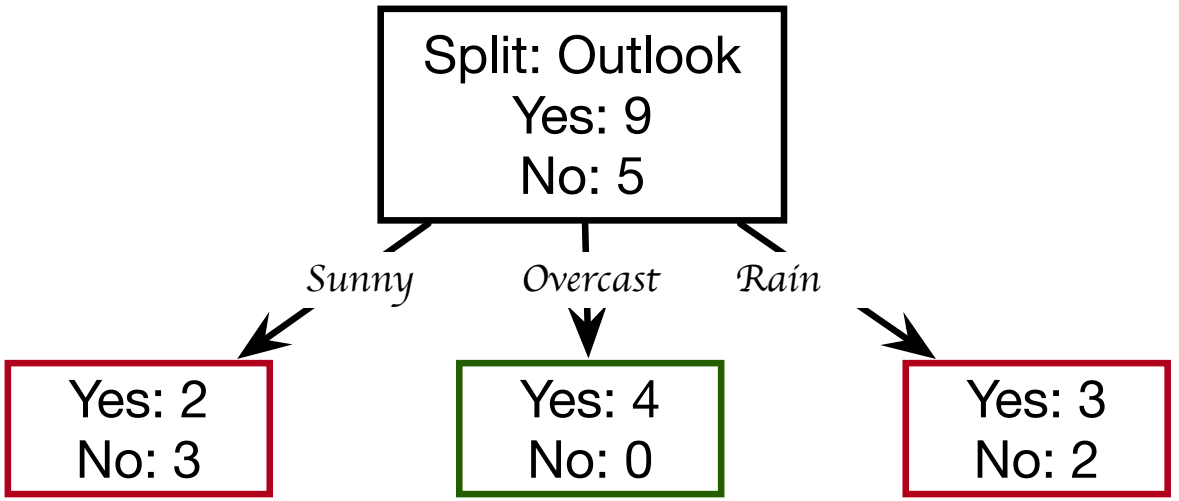
Measuring uncertainty

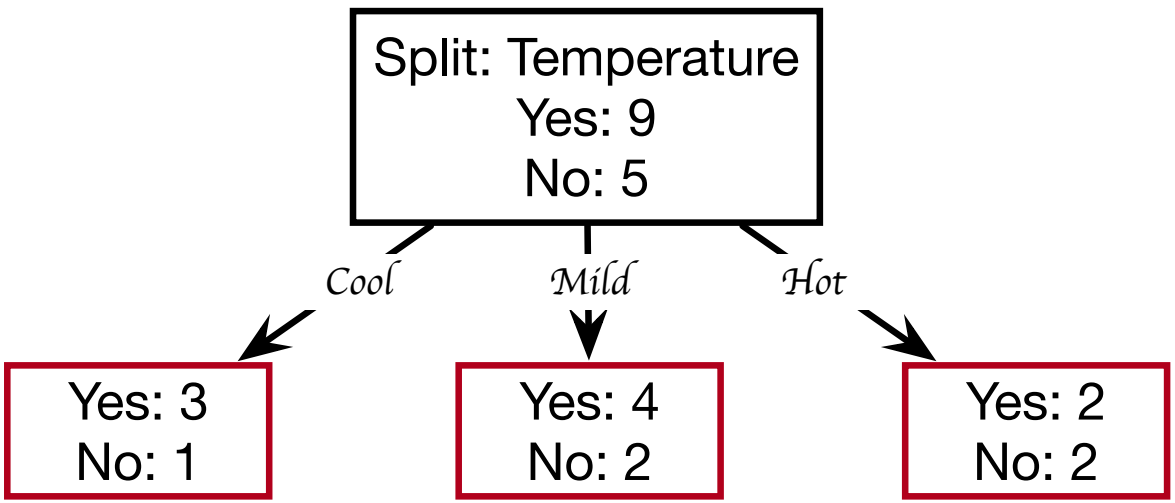
Good split if we are more certain about classificaGon after split

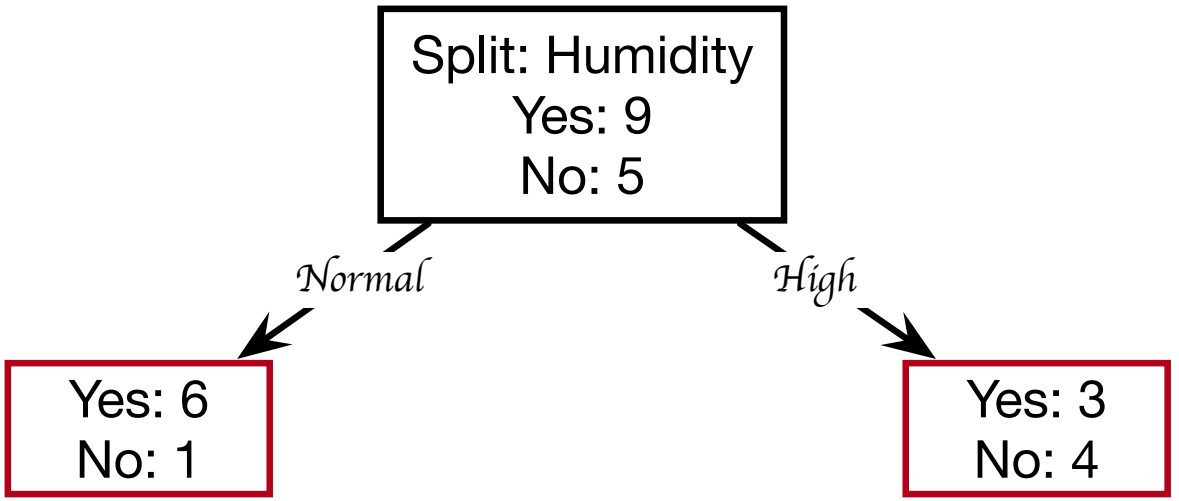
- Deterministic good (all true or all false)
- Uniform distribution bad
- What about distributions in between?

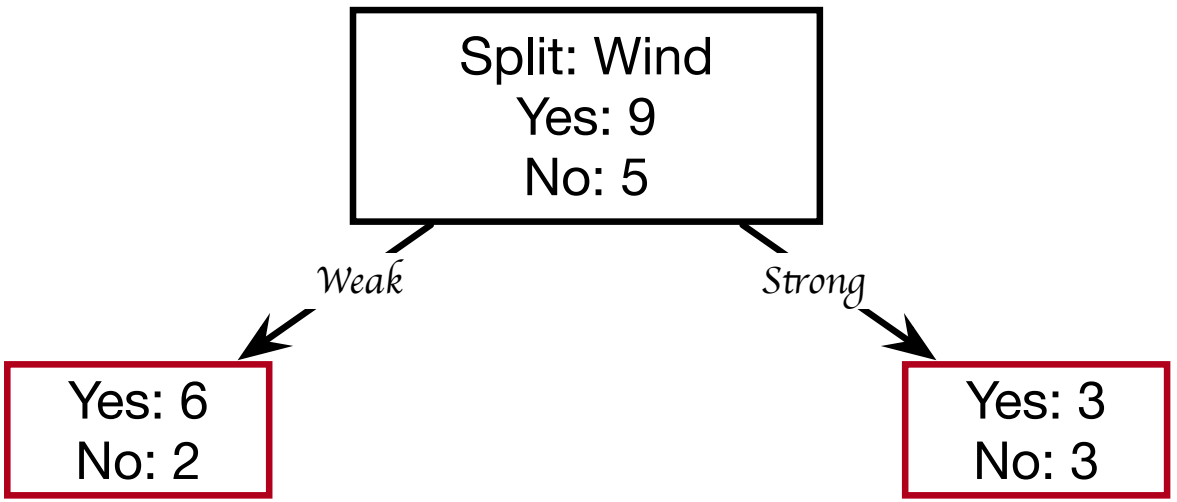
$P(Y = \mathrm{A}) = \frac{1}{2}$	$P(Y = \mathrm{B}) = \frac{1}{4}$	$P(Y = \mathrm{C}) = \frac{1}{8}$	$P(Y = \mathrm{D}) = \frac{1}{8}$

$P(Y = \mathrm{A}) = \frac{1}{4}$	$P(Y = \mathrm{B}) = \frac{1}{4}$	$P(Y = \mathrm{C}) = \frac{1}{4}$	$P(Y = \mathrm{D}) = \frac{1}{4}$


Split: Outlook
Yes: 9
No: 5
Sunny
Overcast
Rain
Yes: 2
No: 3
Yes: 4
No: 0
Yes: 3
No: 2


Split: Temperature
Yes: 9
No: 5
Cool
Mild
Hot
Yes: 3
No: 1
Yes: 4
No: 2
Yes: 2
No: 2


Split: Humidity
Yes: 9
No: 5
Normal
High
Yes: 6
No: 1
Yes: 3
No: 4


Split: Wind
Yes: 9
No: 5
Weak
Strong
Yes: 6
No: 2
Yes: 3
No: 3

uncertainty: Impurity Criteria

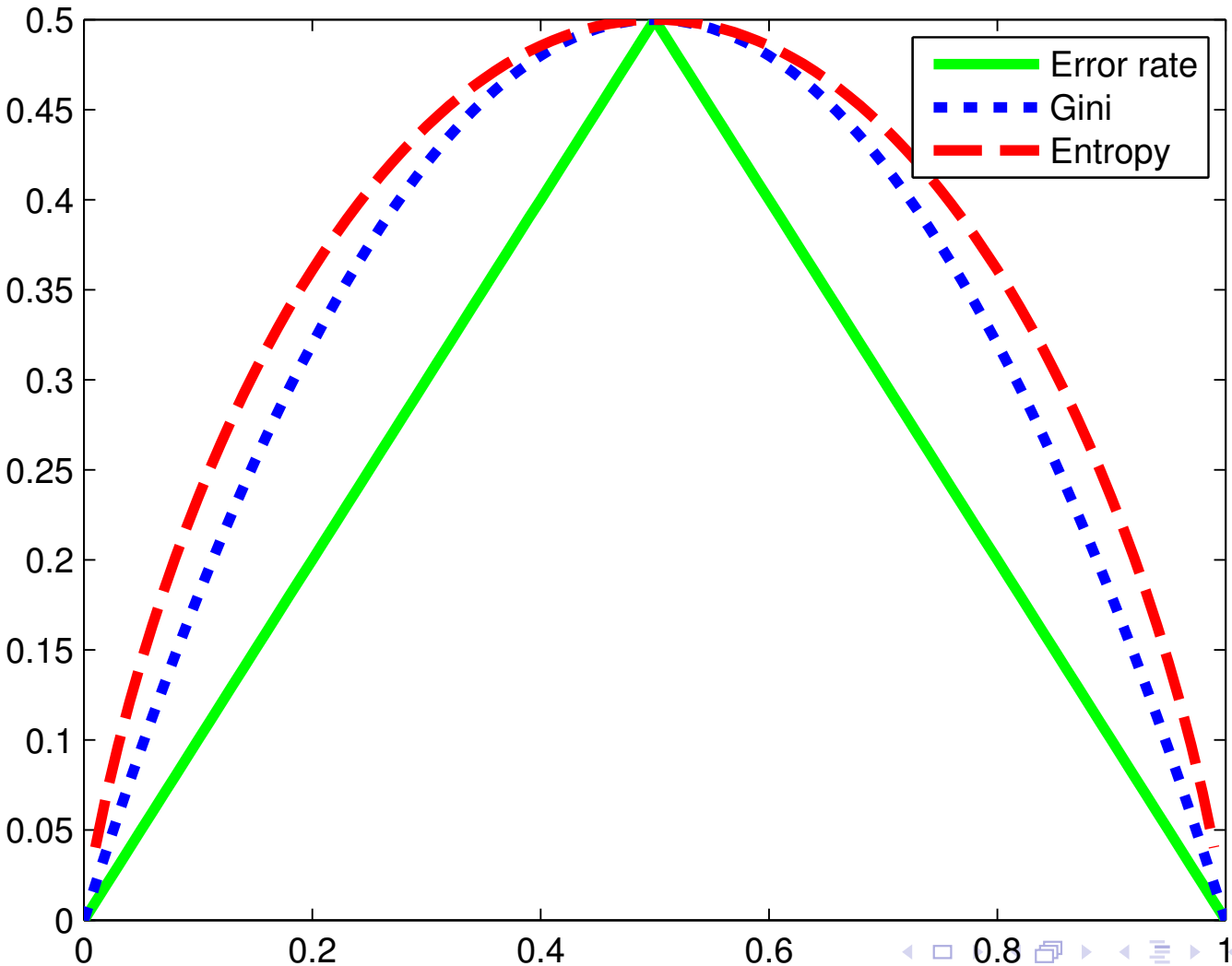
We define the estimation of the class-conditional probabilities

$$p_k = \frac{1}{\mathcal{I}} \sum_{i \in \mathcal{I}} \mathbb{1}(y_i = k)$$

- Misclassification rate: $(1 - p_{\hat{y}})$
- Entropy: $H(P) = -\sum_{k=1}^{K} p_k \log_2(p_k)$
- Gini index:

$$\begin{aligned} \sum_{k=1}^{K} p_k (1 - p_k) &= \sum_{k=1}^{K} p_k - \sum_{k=1}^{K} p_k^2 \\ &= 1 - \sum_{k=1}^{K} p_k^2 \end{aligned} \quad (1)$$

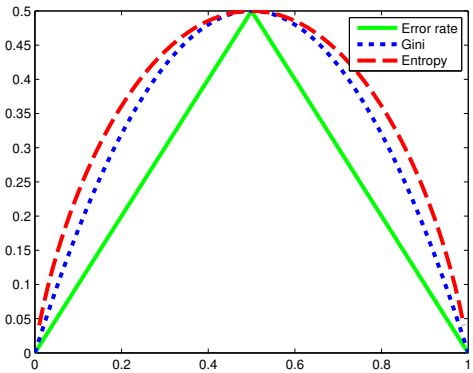
uncertainty: Impurity Criteria



Entropy

Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{k} P(X = x_i) \log_2 P(X = x_i)$$



More uncertainty, more entropy

Information Theory interpretation: $H(X)$ is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

High, Low Entropy

- High Entropy:
  - X is from a uniform like distribution
  - Flat histogram
  - Values sampled from it are less predictable
- Low Entropy
  - X is from a varied (peaks and valleys) distribution
  - Histogram has many lows and highs
  - Values sampled from it are more predictable

Entropy Example

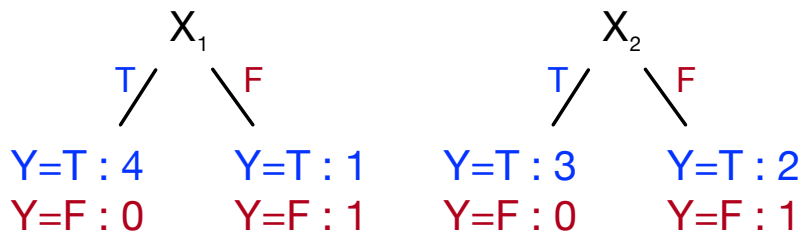
$$
\begin{aligned}
H(Y) &= -\sum_{y=F,T} P(Y=y)\log_2 P(Y=y) \\
P(Y=T) &= \frac{5}{6} \\
P(Y=F) &= \frac{1}{6} \\
H(Y) &= -\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6} \\
&= 0.65
\end{aligned}
$$

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Conditional Entropy

Conditional Entropy $H(Y|X)$ of a random variable Y conditionned on a random variable X

$$H(Y|X) = -\sum_{i=1}^{I} P(X = x_i) \sum_{j=1}^{J} P(Y = y_j|X = x_i) \log_2 P(Y = y_j|X = x_i)$$



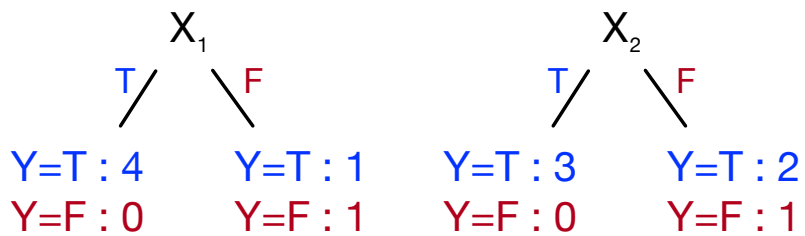
X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$$\begin{aligned} H(Y|X_1) &= -\frac{4}{6}(1 \log_2 1 + 0 \log_2 0) \\ &\quad -\frac{2}{6}(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}) \\ &= \frac{2}{6} \end{aligned}$$

Conditional Entropy

Conditional Entropy $H(Y|X)$ of a random variable Y conditionned on a random variable X

$$H(Y|X) = -\sum_{i=1}^{I} P(X = x_i) \sum_{j=1}^{J} P(Y = y_j|X = x_i) \log_2 P(Y = y_j|X = x_i)$$



X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$$\begin{aligned} H(Y|X_2) &= -\frac{3}{6}\left(1 \log_2 1 + 0 \log_2 0\right) \\ &\quad -\frac{3}{6}\left(\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3}\right) \\ &= 0.46 \end{aligned}$$

ID3: Iterative Dichotomiser 3

- Information Theory (Shannon) considerations are used for the selection of the partitioning criterion
- Maximizing the information gain

$$IG(A, C) = \sum_{a,\omega \in \mathbb{D}_a \times \mathbb{D}_c} \mathrm{P}(A = a, C = c) \log \left(\frac{\mathrm{P}(A = a, C = c)}{\mathrm{P}(A = a)\,\mathrm{P}(C = c)} \right)$$

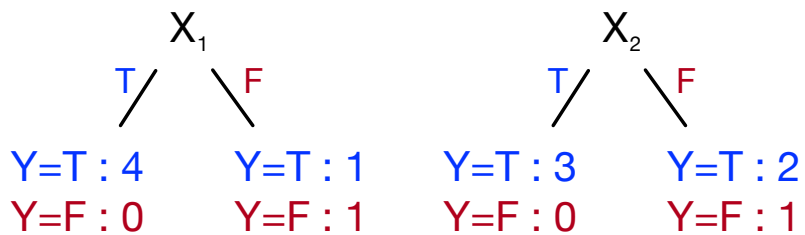
Entropy vs Information Gain

$$
\begin{aligned}
IG(A, C) &= \sum_{a,\omega \in \mathbb{D}_a \times \mathbb{D}_c} \mathrm{P}(A = a, C = c) \log \left(\frac{\mathrm{P}(A = a, C = c)}{\mathrm{P}(A = a)\,\mathrm{P}(C = c)} \right) \\
H(A) &= -\sum_{a \in \mathbb{D}_a} \mathrm{P}(A = a) \log \left(\mathrm{P}(A = a) \right) \\
H(A|C) &= -\sum_{a,\omega \in \mathbb{D}_a \times \mathbb{D}_c} \mathrm{P}(A = a, C = c) \log \left(\mathrm{P}(A = a|C = c) \right) \\
IG(A, C) &= H(A) - H(A|C) \qquad (2)
\end{aligned}
$$

Information Gain

Decrease in entrpy (uncertainty) after splitting

$$IG(X) = H(Y) - H(Y|X)$$



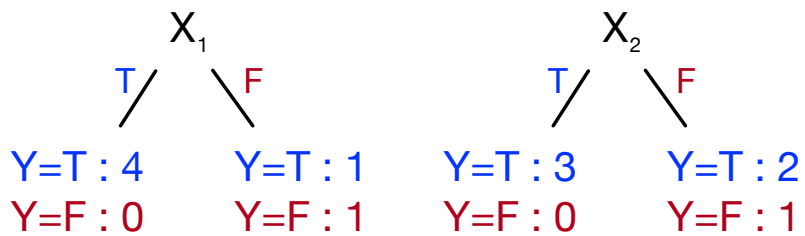
X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$$\begin{aligned} H(Y|X_1) &= \frac{2}{6} \\ IG(X_1) &= H(Y) - H(Y|X_1) \\ &= 0.65 - 0.33 \\ &= 0.32 \end{aligned}$$

Conditional Entropy

Decrease in entrpy (uncertainty) after splitting

$$IG(X) = H(Y) - H(Y|X)$$



X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$$\begin{aligned} H(Y|X_2) &= 0.46 \\ IG(X_2) &= H(Y) - H(Y|X_2) \\ &= 0.65 - 0.46 \\ &= 0.19 \end{aligned}$$

Exercise:

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Remarks

- Attribute list may be empty
- Work on symbolic data only
- Very sensitive to noise!
- Learning Bias: shorter solutions are preferred on longer ones
  Occam's razor: *When you have two competing theories that make exactly the same predictions, the simpler one is the better.*
- No backtracking; local optimum!

C4.5 [Quinlan, 1993]

- C4.5 introduces a number of extensions of the original ID3 algorithm.
- We can deal with training sets that have records with unknown attribute values.
- We can classify records that have unknown attribute values by estimating the probability of the various possible results

Gain Ratio as the Splitting Criterion

- Problem of Information gain approach
  - Biased towards tests with many outcomes (attributes having a large number of values)
  - Attribute acting as unique:
    - identifier produce a large number of partitions (1 tuple per partition)
    - Each resulting partition $\mathcal{I}$ is pure Info($\mathcal{I}$)=0.
- Extension to Information Gain
  - C4.5 uses an extension to information gain known as **gain ratio**.
  - Overcomes the bias of Information gain.
  - Applies a kind of normalization to information gain using a split information value

Gain Ration as the Splitting Criterion

- The **split information value** represents the potential information generated by splitting the training data set $\mathcal{I}$ into k partitions, corresponding to k outcomes on attribute A

$$SplitInformation(\mathcal{I}, A) \equiv -\sum_{i=1}^{k} \frac{|\mathcal{I}_i|}{|\mathcal{I}|} \log_2 \frac{|\mathcal{I}_i|}{|\mathcal{I}|}$$

where $\mathcal{I}_i$ is subset of $\mathcal{I}$ for which A has value v_i

  - **High splitInformation:** partitions have more or less the same size (uniform)
  - **Low splitInformation:** few partitions hold most of the tuples(peaks)

Gain Ration as the Splitting Criterion

- The gain ratio is defined as

$$GainRatio(\mathcal{I}, A) \equiv \frac{IG(\mathcal{I}, A)}{SplitInformation(\mathcal{I}, A)}$$

- The attribute with the maximum gain ratio is selected as the splitting attribute

Dealing with Continuous Valued Attributes

Create a discrete attribute to test continuous

- $Temperature = 82.5$
- $(Temperature > 72.3) = t, f$

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

When should you do the discretization ?

Dealing with missing attributes

- Use a variation of the information gain defined by:

$$IG(\mathcal{I}, A) = \frac{|\mathcal{I} - \mathcal{I}_0|}{|\mathcal{I}|} IG(\mathcal{I} - \mathcal{I}_0, A)$$

  with $\mathcal{I}_0$ the subset of $\mathcal{I}$ with missing Attribute A.
- During Classification, assign a Probability for each value of the attribute, given the known attributes

CART: Classification and Regression Trees [Breiman et al. 1984]

- Binary Decision Tree
- Use the Gini Impurity (also called variance impurity for binary problems)

$$\begin{aligned} \text{Gini}(\mathcal{I}) &= \sum_{k=1}^{K} p_k(1-p_k) \\ &= 1 - \sum_{k} P_k^2 \end{aligned} \tag{3}$$

- The derivative of the Gini impurity is chosen as splitting criterion:

$$\delta\text{Gini}(\mathcal{I}) = \text{Gini}(\mathcal{I}) - p_L\text{Gini}(\mathcal{I}_L) - (1-p_L)\text{Gini}(\mathcal{I}_R)$$

Comparing Attribute Selection Measures

- Information Gain
  - Biased towards multivalued attributes
- Gain Ratio
  - Tends to prefer unbalanced splits in which one partition is much smaller than the other
- Gini Index
  - Biased towards multivalued attributes
  - Has difficulties when the number of classes is large
  - Tends to favor tests that result in equal-sized partitions and purity in both partitions

Decision Tree Overfit

- Standard decision trees have no learning bias
  - Training set error is always zero! (If there is no label noise)
  - Lots of variance
  - Must introduce some bias towards simpler trees
- Many strategies for picking simpler trees
  - Fixed depth
  - Minimum number of samples per leaf
  - Pruning
- Ensemble Methods: Random Forests

Prepruning:

- Halt tree construction early: do not split a node if this would result in the cost measure falling below a threshold
- Statistical significance, information gain, Gini index are used to assess the goodness of a split
- The leaf may hold the most frequent class among the subset tuples
- Problem: Difficult to choose an appropriate threshold

Postpruning:

- Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
- A subtree at a given node is pruned by replacing it by a leaf
- The leaf is labeled with the most frequent class
- Example: cost complexity pruning algorithm
  - Cost complexity of a tree is a function of the number of leaves and the error rate.
  - At each node N compute
    - The cost complexity of the subtree at N
    - The cost complexity of the subtree at N if it were to be pruned

    If pruning results is smaller cost, then prune the subtree at N
  - Use a set of data different from the training data to decide which is the **best pruned tree**

What you need to know about decision trees

- Decision trees are one of the most popular ML tools
  - Easy to understand, implement, and use
  - Computationally cheap (to solve heuristically)
- Information gain to select aIributes (ID3, C4.5,...)
- Presented for classification, can be used for regression and density estimation too
- Decision trees will overfit!!!
  - Must use tricks to find "simple trees":
    - Fixed depth/Early stopping
    - Minimum number of samples per leaf
    - Pruning
  - Or, use ensembles of different trees (random forests)

Bootstrap Aggregating (Bagging) [Breiman, 1996]

- Generate multiple versions of a predictor and use them to get an aggregated predictor
- Improve the performances of unstable classifiers
- Train N instances of the same classifier on bootstrap sample of the dataset
- Decision is done through a vote
- Could be use in classification or regression

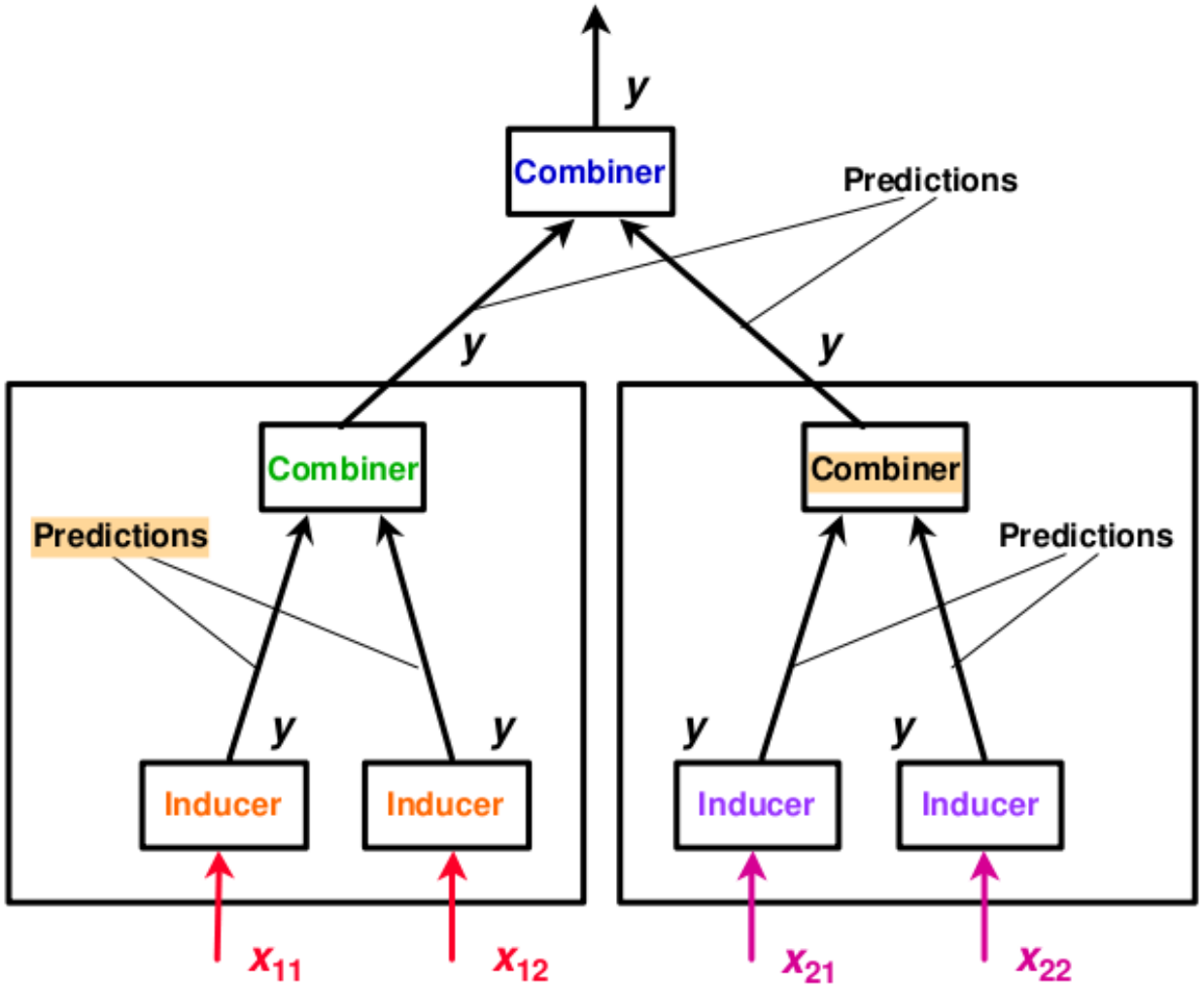
Random Forests [Breiman, 2001]

- **Problem with Bagging:** re-running the same algorithm on different subsets of the data $\longleftrightarrow$ Highly correlated predictors.
- **Random Forest:** Decorrelate the base learners by learning trees based on a randomly chosen subset input features, as well as a randomly chosen subset of data cases.
- Many Applications: Microsoft's Kinect sensor...

Stacked Generalization (Stacking) [Wolpert, 1992]

- improve stable algorithms
- build Hierarchical classifiers:
  - divide dataset into a training set and a validation set
  - consider K subsamples of the training set
  - train K instances of the same classifier with those subsamples
  - train an combiner with the predictions of those K classifiers on the validation set
- Combining different kind of classifiers:
  - divide dataset into a training set and a validation set
  - train K different classifiers on the training set
  - train a combiner with the predictions of those K classifiers on the validation set

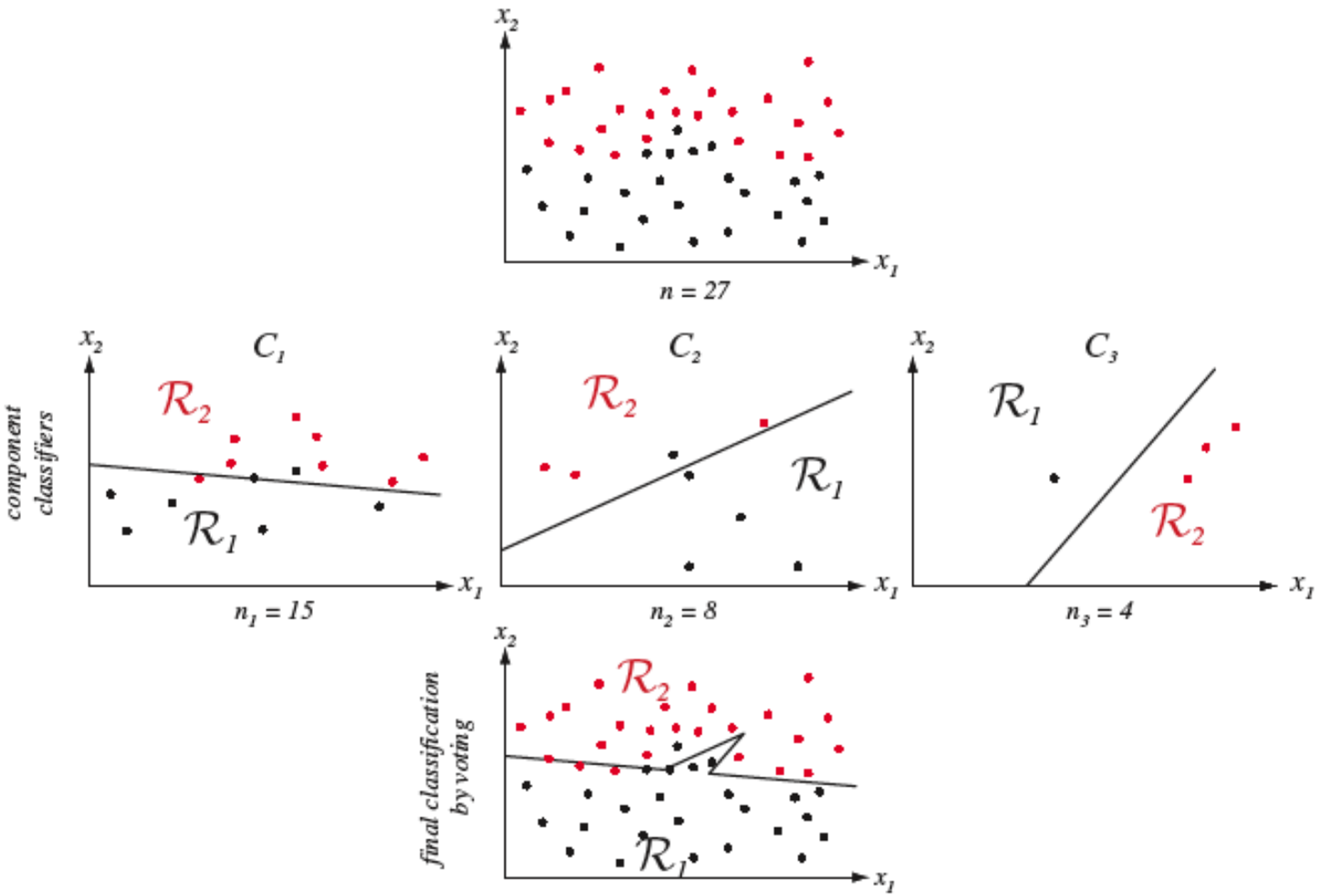
Stacking


y
Combiner
Predictions
y
y
Combiner
Combiner
Predictions
Predictions
y
y
y
y
Inducer
Inducer
Inducer
Inducer
x_{11}
x_{12}
x_{21}
x_{22}

Boosting [Schapire, 1990, Schapire and Freund, 2012]

- use a set of weak learner to create a strong learner
- the accuracy of a weak learner need only to be better than chance
- Many applications : Face Detector [Viola and Jones, 2001].

Example


x_2
x_1
$n = 27$
C_1
C_2
C_3
$\mathcal{R}_1$
$\mathcal{R}_2$
component classifiers
$n_1 = 15$
$n_2 = 8$
$n_3 = 4$
final classification by voting

Boosting: AdaBoost

Input: Training set of N instances $\mathcal{I}$

$p_i^1 = \frac{1}{N}$

for m=1 **to** M **do**

Generate a set $\mathcal{I}_m$ from $\mathcal{I}$ according to the distribution p^m

Learn a classifier h_m using $\mathcal{I}_m$ dataset.

Compute the error: $\epsilon_m = \frac{\sum_{i=1}^{N} p_i^m \mathbb{1}(y_i \neq h_m(x_i))}{\sum_{i=1}^{N} p_i^m}$

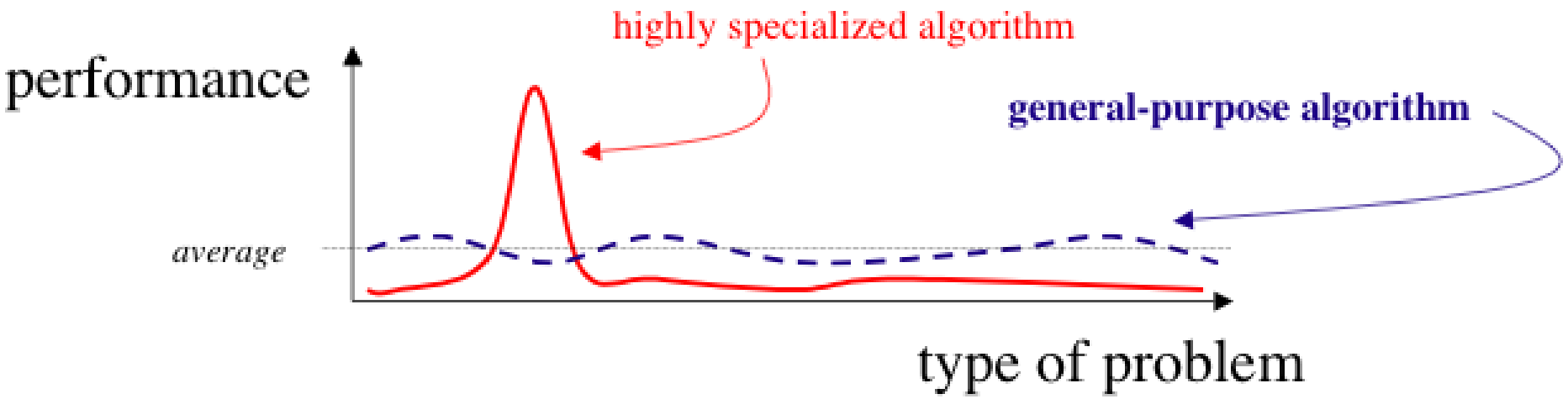
Compute $\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$

for i=1 **to** N **do** $p_i^{m+1} = p_i^m \exp\left(\alpha_m \mathbb{1}\left(y_i \neq h_m(x_i)\right)\right)$;

end

Algorithm 2: AdaBoost for binary classifier with exponential loss

No Free Lunch Theorem



For any two algorithms, A and B, there exist datasets for which algorithm A outperform algorithm B in prediction accuracy on unseen instances.

Proof: Take any Boolean concept. If A outperforms B on unseen instances, reverse the labels and B will outperforms A.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2): 123–140, 1996. doi: 10.1007/BF00058655. URL http://dx.doi.org/10.1007/BF00058655.

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A:1010933404324.

Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision tree is np-complete. *Information Processing Letters*, 5 (1):15–17, 1976.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1 (1):81–106, 1986. doi: 10.1023/A:1022643204877. URL http://dx.doi.org/10.1023/A:1022643204877.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. ISBN 1-55860-238-0.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990. doi: 10.1007/BF00116037. URL http://dx.doi.org/10.1007/BF00116037.

Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012. ISBN 0262017180, 9780262017183.

Paul A. Viola and Michael J. Jones. Robust real-time face detection. In *ICCV*, page 747, 2001.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5: 241–259, 1992.