Decision trees

Predictive Analytics

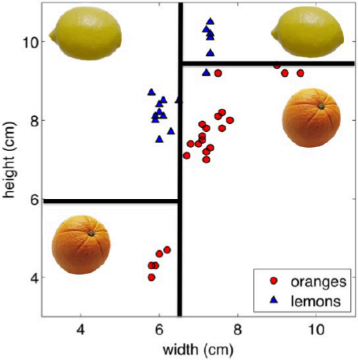
Acknowledgments

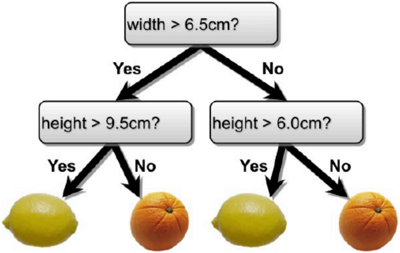
These slides draw upon and adapt selected materials by:

- **Fragkiskos D. Malliaros** (CentraleSupélec, Université Paris–Saclay, France)
- **David Sontag** (MIT CSAIL, USA)

Another Classification Idea (2/2)

- Gives axes aligned decision boundaries



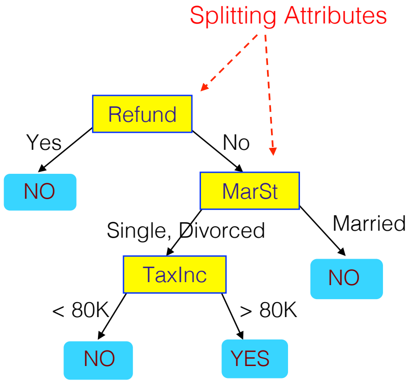


Example of a Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

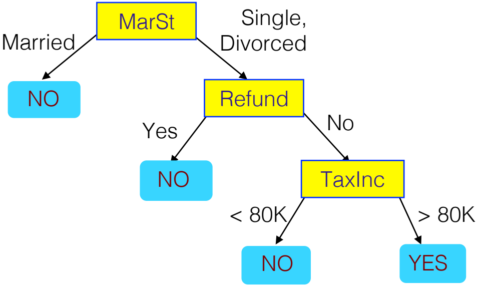


Model: Decision Tree

Another Example of Decision Tree

categorical categorical continuous class

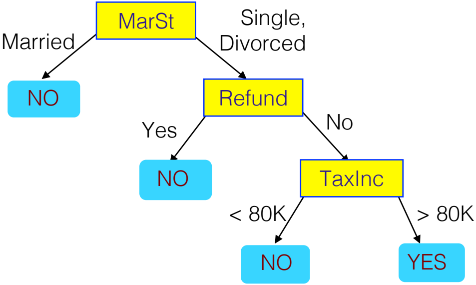
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

Decision Trees – Nodes and Branching

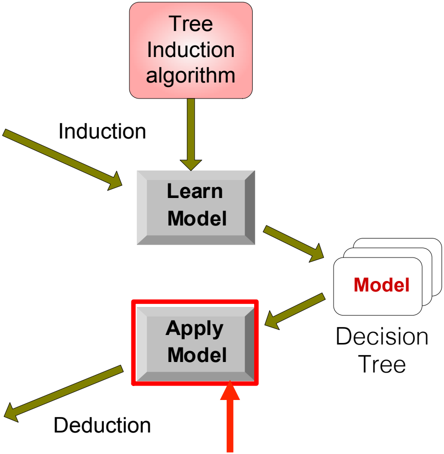
- Internal nodes correspond to test attributes
- Branching is determined by attribute value
- Leaf nodes are outputs (class assignments)
  - YES or NO in this example



Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

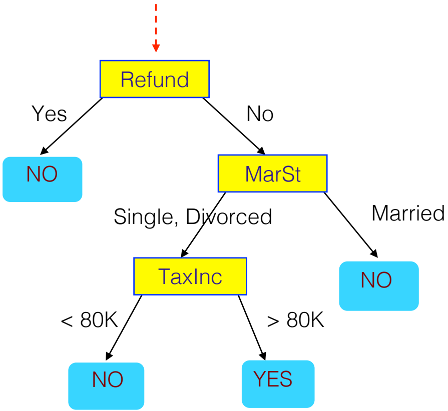


Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Apply Model to Test Data

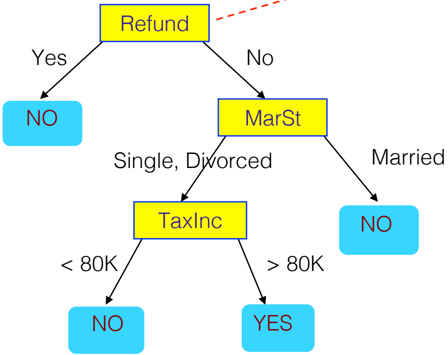
Start from the root of tree



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

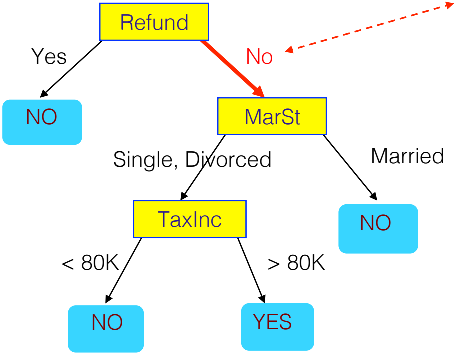
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

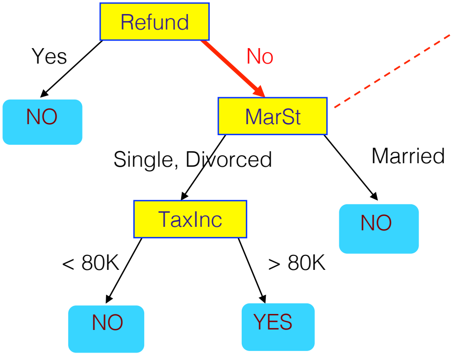
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



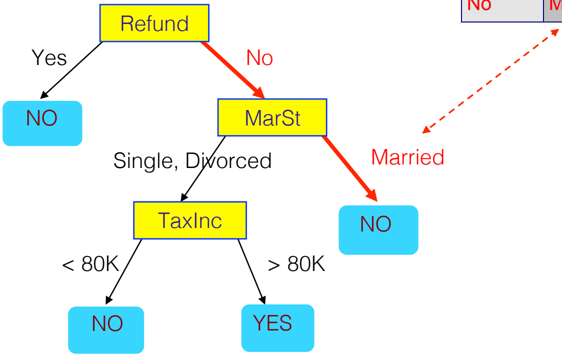
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

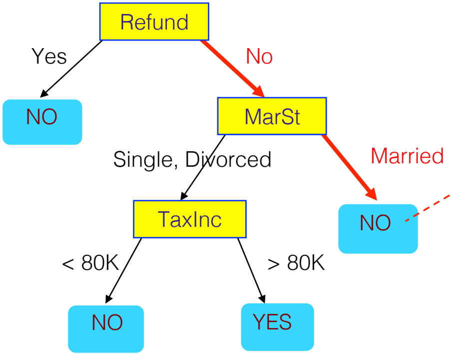
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

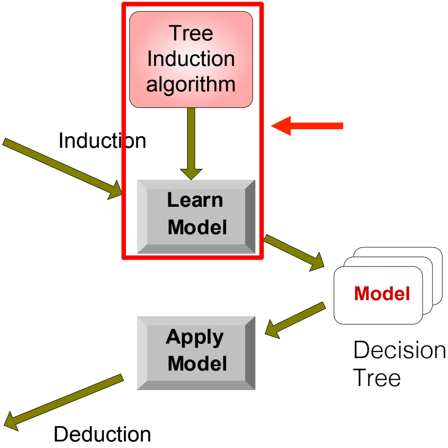


Assign Cheat to "No"

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Decision Tree Induction – The Idea (1/2)

- Basic algorithm
  - Tree is constructed in a top-down recursive manner
  - Initially, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on the selected attributes
  - Split attributes are selected on the basis of a heuristic or statistical measure (e.g., gini index, information gain)

- Most commercial DTs use variations of this algorithm

Decision Tree Induction – The Idea (2/2)

- Simple, greedy, recursive approach, builds up tree node-by-node

1. Pick an attribute to split at a non-terminal node
2. Split examples into groups based on attribute value
3. For each group:
   - If no examples – return class majority from parent node
   - Else, if all examples are in the same class – return class
   - Else, loop to step 1

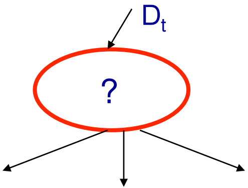
Decision Tree Induction Algorithms

- Many algorithms
  - Hunt's algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT

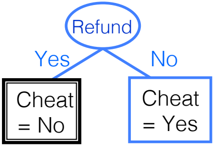
General Structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node t
- General procedure:
  - If D_t contains records that belong to the same class y_t, then t is a leaf node labeled as y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If D_t contains records that belong to more than one classes, use an attribute test to split the data into smaller subsets
    - Recursively apply the procedure to each subset

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

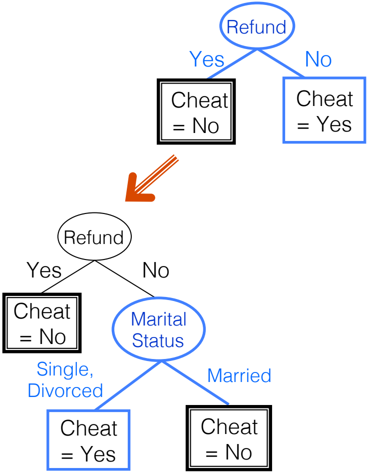


Hunt's Algorithm



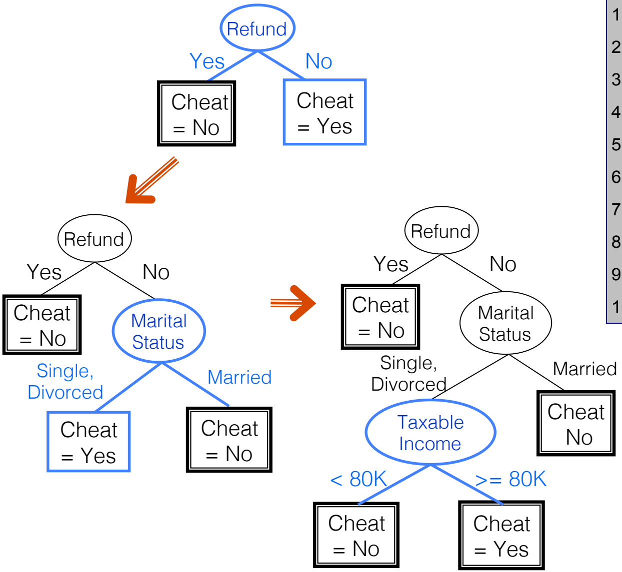
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm



Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

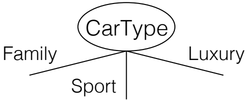
How to Specify Test Condition?

- Depends on attribute types
  - Nominal (categorical)
    - E.g., female, male
  - Ordinal (categorical, but there is an ordering)
    - E.g., economic status: low, medium, high
  - Continuous

- Depends on the number of ways to split
  - 2-way split
  - Multi-way split

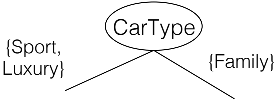
Splitting Based on Nominal Attributes

- Multi-way split: Use as many partitions as distinct values

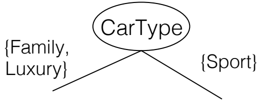


No ordering of the attributes

- Binary split: Divides values into two subsets
  (Need to find optimal partitioning)

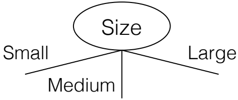


OR

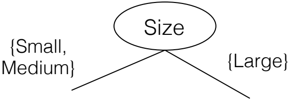


Splitting Based on Ordinal Attributes

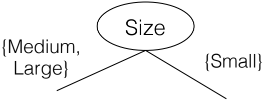
- Multi-way split: Use as many partitions as distinct values



- Binary split: Divides values into two subsets
  (Need to find optimal partitioning)



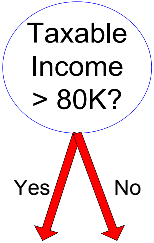
OR



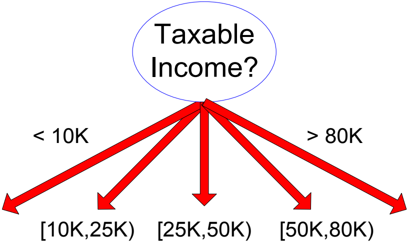
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary decision: $(A < v)$ or $(A \geq v)$
    - Considers all possible splits and finds the best cut
    - Can be more computational intensive

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

Tree Induction

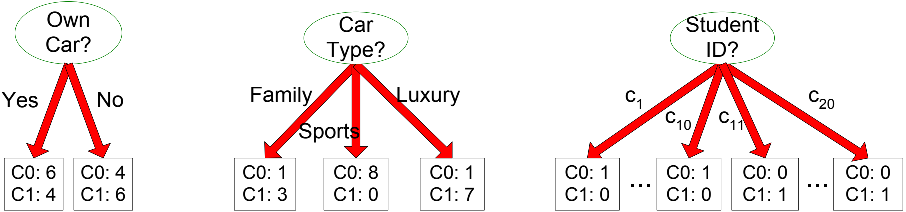
- Greedy strategy
  - Split the records based on an attribute test that optimizes certain criterion

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to Determine the Best Split (1/2)

Before splitting: 10 records of class 0
10 records of class 1

Suppose that we want to predict if a student will get tax return or no?

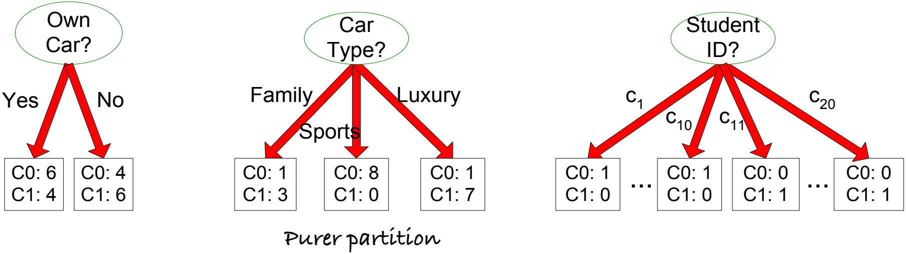


Which test condition is the best?

How to Determine the Best Split (1/2)

Before splitting: 10 records of class 0
10 records of class 1

Suppose that we want to predict if a student will get tax return or no?

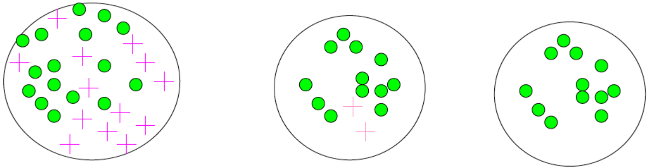


Which test condition is the best?

How to Determine the Best Split (2/2)

- Greedy approach:
  - Nodes with homogeneous (pure) class distribution are preferred
- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	Homogeneous, Low degree of impurity



Very impure   Medium impure   Pure

Measures of Node Impurity

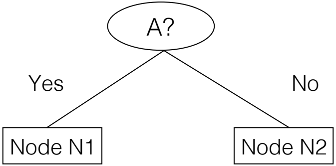
- Gini Index
- Entropy and Information Gain
- Misclassification error

How to Find the Best Split – Gini Index

Before Splitting:

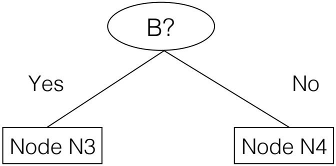
C0	**N00**
C1	**N01**

Two possible splits on attributes A or B



C0	**N10**
C1	**N11**

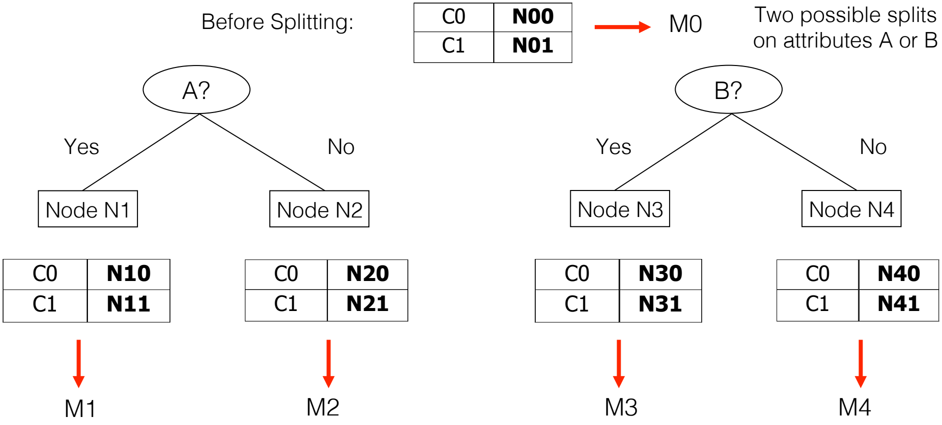
C0	**N20**
C1	**N21**



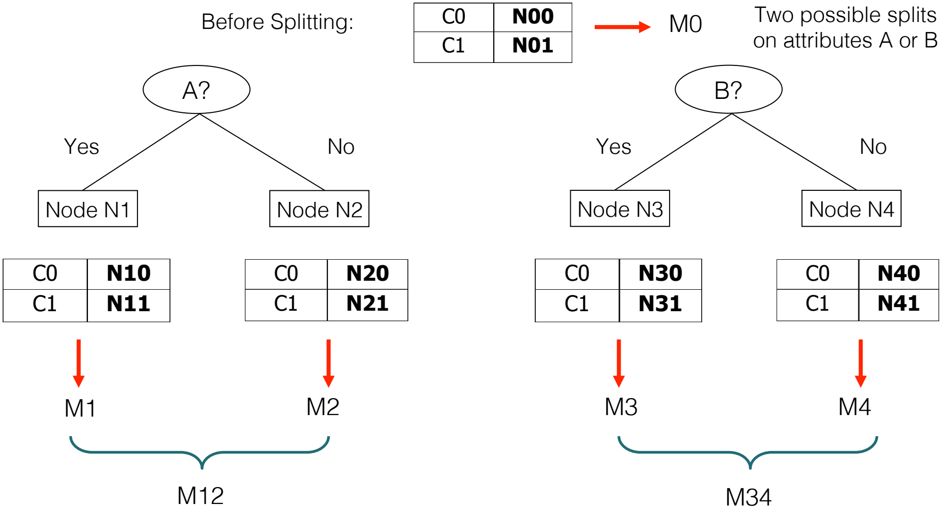
C0	**N30**
C1	**N31**

C0	**N40**
C1	**N41**

How to Find the Best Split – Gini Index



How to Find the Best Split – Gini Index



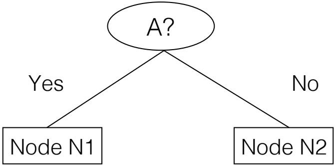
How to Find the Best Split – Gini Index

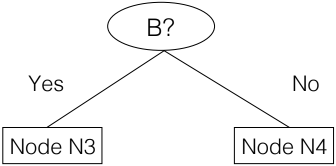
Before Splitting:

C0	**N00**
C1	**N01**

→ M0

Two possible splits on attributes A or B





C0	**N10**
C1	**N11**

↓ M1

C0	**N20**
C1	**N21**

↓ M2

$\underbrace{\text{M1} \qquad \text{M2}}_{\text{M12}}$

C0	**N30**
C1	**N31**

↓ M3

C0	**N40**
C1	**N41**

↓ M4

$\underbrace{\text{M3} \qquad \text{M4}}_{\text{M34}}$

Gain = M0 – M12 vs. M0 – M34

Reduction in impurity

Measure of Impurity: Gini Index

Gini index for a node t

$$Gini(t) = 1 - \sum_{j=1}^{J} \left(p(j|t)\right)^2$$

- J number of classes
- p(j | t) is the relative frequency of class j at node t

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Gini = $1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Gini = $1 - (1/6)^2 - (5/6)^2 = 0.278$

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Gini = $1 - (2/6)^2 - (4/6)^2 = 0.444$

Select the split with the smallest Gini index

Splitting based on Gini Index

- When a node t is split on attribute A into k partitions (children), the quality of split is computed as

$$\text{Gini}_A = \sum_{i=1}^{k} \frac{n_i}{n} \text{Gini}(i)$$

n_i = number of records at child i
n = number of records at node t

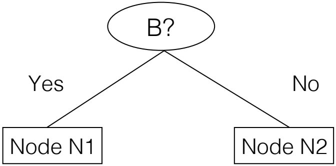
- Reduction in impurity (gain) after splitting node t (on attribute A), is computed as

$$\Delta\text{Gini}_A = \text{Gini}(t) - \text{Gini}_A$$

- The attribute X with the smallest Gini_X or the largest reduction in impurity is chosen to split the node

Example

- Splits into two partitions
  - Goal: reduce impurity



	Parent
C1	**6**
C2	**6**
Gini = 0.500	

Gini(N1)
$= 1 - (5/7)^2 - (2/7)^2$
$= 0.408$

Gini(N2)
$= 1 - (1/5)^2 - (4/5)^2$
$= 0.32$

	N1	**N2**
C1	**5**	**1**
C2	**2**	**4**
Gini=0.371		

Split on attribute B

Gini_B
$= 7/12 * 0.408 +$
$5/12 * 0.32$
$= 0.371$

Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes certain criterion

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values

Decision Tree Based Classification

- Advantages
  - Inexpensive to construct (training phase)
  - Extremely fast at testing phase (classifying unseen data)
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers

- Two approaches to avoid overfitting
  - Pre-pruning: Halt tree construction early; do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Post-pruning: Remove branches from a fully grown tree
    - Get a sequence of progressively pruned trees
    - Use a dataset (different from the training data) to decide which is the best pruned tree

scikit-learn

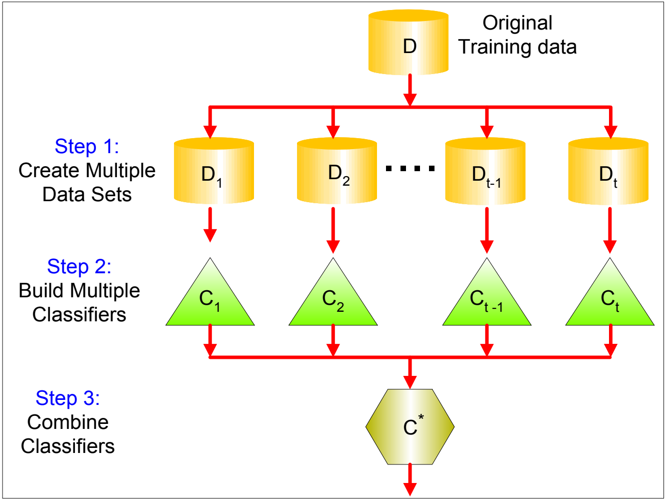
http://scikit-learn.org/stable/modules/generated/
sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Ensemble learning

Ensemble Methods

- Typical application: classification
- Ensemble of classifiers: set of classifiers whose individual decisions are combined in some way to classify new examples
- Simplest approach:
  1. Generate multiple classifiers (e.g., decision trees, logistic regression)
  2. Each classifier votes (decides) on a test instance
  3. Take majority as classification
- Classifiers are different due to different sampling of training data, or randomized parameters within the classification algorithm
- Goal: take a simple algorithm and transform it into a ‘super classifier’

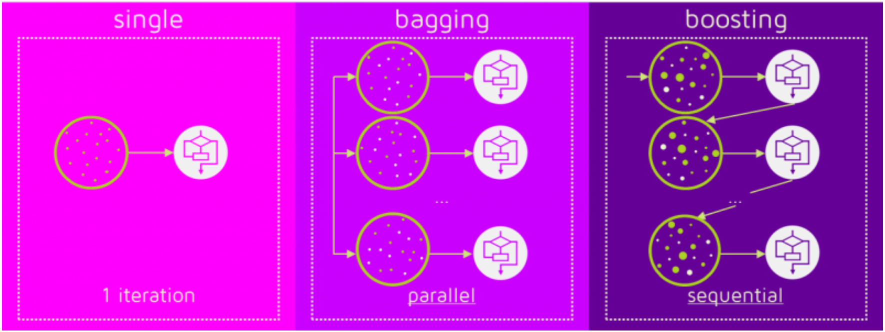
Ensemble Learning – General Idea



Ensemble Methods: Summary

- Differ in training strategy and combination method
- Bagging (bootstrap aggregation)
  - Random sampling with replacement
  - Train separate models on overlapping training sets, average their predictions
  - E.g., random forest classifier
- Boosting
  - Sequential training, iteratively re-weighting the training examples – the current classifier focuses on hard examples
  - E.g., AdaBoost

Bagging vs. Boosting



Source: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

Bagging: Bootstrap Estimation

- Repeatedly draw n samples from D
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance
- Bagging: bootstrap aggregation (Breiman 1994)

Bagging

- Simple idea
  - Generate M bootstrap samples from your original training set
  - Train on each one to get y_m , and average them

$$y_{bag}^{M}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} y_m(\mathbf{x})$$

- Each bootstrap sample is drawn with replacement
  - Each one contains some duplicates of certain training points and leaves out other training points completely
- For regression: average the predictions
- For classification: average class probabilities or take majority vote

Boosting

- Also works by manipulating the training set, but the individual classifiers are trained sequentially
- Each classifier is trained based on knowledge of the performance of previously trained classifiers
  - Focus on difficult examples
- Final classifier: weighted sum of the component classifiers

AdaBoost: Making Weak Learners Stronger

- Suppose you have a weak learning module (a base classifier) that can always get $0.5 + \varepsilon$ correct when given a binary classification task
  - A little bit better than a random classifier
  - Weak learner

- Can you apply this learning module many times to get a strong learner that can get close to zero error rate on the training data?
  - ML theorists showed how to do this and it actually led to an effective new learning procedure (Freund & Shapire, 1996)
  - The AdaBoost algorithm

AdaBoost – The Idea

- Train T weak learners (models) sequentially
- First train the base classifier on all the training data with equal importance weights on each case
- Then, re-weight the training data to emphasize the hard cases and train a second model
  - Instances that were misclassified in the previous step
  - Q: How do we re-weight the data?
- Keep training new models on the re-weighted data
- Finally, use a weighted committee of all the models for the test data
  - How do we weight the models in the committee?

How to Train Each Classifier

- Input: $(D_n = \{(\mathbf{x_i}, t_i)\}_{i=1}^{n})$
- Output: $y(\mathbf{x_i}) \in \{-1, 1\}$
- Weight of instance (e.g., data point) x_i for classifier t: $w_i^{(t)}$
- Cost function for classifier t:

$$J_t = \sum_{i=1}^{n} w_i^{(t)} \underbrace{I(y_t(\mathbf{x_i}) \neq t_i)}_{\text{1 if error, 0 otherwise}} = \sum \text{weighted errors}$$

Weight of Instances for Classifier t

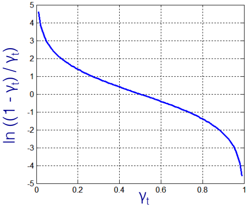
- Weighted error rate of a classifier t

$$\gamma_t = \frac{J_t}{\sum_{i=1}^{n} w_i^{(t)}}$$

- The quality (coefficient) of classifier t is

It is zero if the classifier has weighted error rate of 0.5 and infinity if the classifier is perfect

$$\alpha_t \leftarrow \ln\left\{\frac{1-\gamma_t}{\gamma_t}\right\}$$

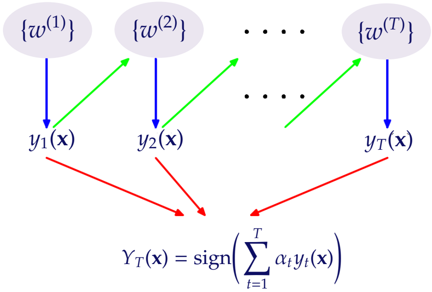


- Update the weights for the next classifier (t+1)

$$w_i^{(t+1)} \leftarrow w_i^{(t)} \ \exp\{\alpha_t \underbrace{I(y_t(\mathbf{x_i}) \neq t_i)}_{\text{1 if error, 0 otherwise}}\}$$

The weights 'inform' the training of the weak learner
(decision trees can be grown that favor splitting sets of samples with high weights)

How to Make Predictions



- Weight the binary prediction of each classifier by the quality of that classifier

$$Y_T(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t y_t(\mathbf{x})\right)$$

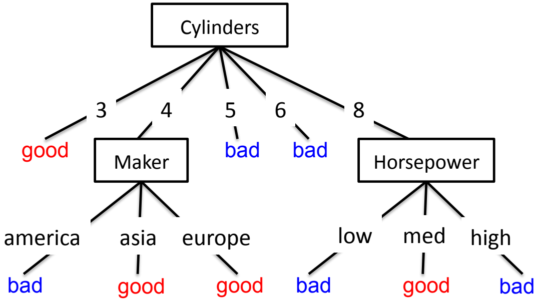
scikit-learn

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

http://scikit-learn.org/stable/modules/ensemble.html

Hypotheses: decision trees $f : X \to Y$

- 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

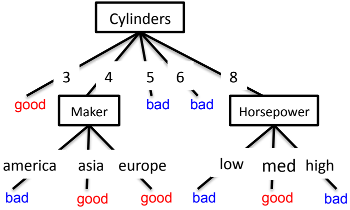


Human interpretable!

Hypothesis space

- How many possible hypotheses?
- What functions can be represented?

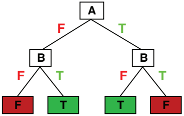
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe



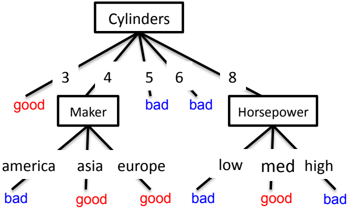
What functions can be represented?

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

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



(Figure from Stuart Russell)



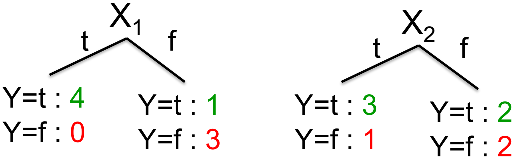
cyl=3 ∨ (cyl=4 ∧ (maker=asia ∨ maker=europe)) ∨ …

Learning *simplest* decision tree is NP-hard

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest ’76]
- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on **next best attribute (feature)**
  - Recurse

Splitting: 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

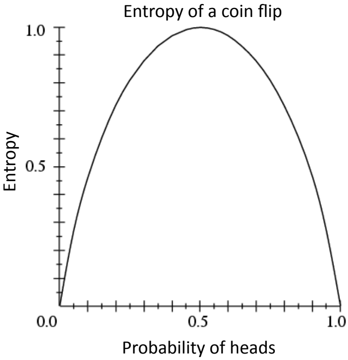
Entropy

Entropy *H(Y)* of a random variable Y

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

More uncertainty, more entropy!

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



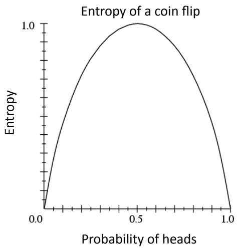
High, Low Entropy

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

(Slide from Vibhav Gogate)

Entropy Example

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



P(Y=t) = 5/6

P(Y=f) = 1/6

H(Y) = - 5/6 $\log_2$ 5/6 - 1/6 $\log_2$ 1/6

= 0.65

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

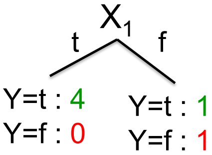
Conditional Entropy $H(Y|X)$ of a random variable Y conditioned on a random variable X

$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

Example:

$P(X_1=t) = 4/6$

$P(X_1=f) = 2/6$



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

$H(Y|X_1) = -\ 4/6\ (1 \log_2 1 + 0 \log_2 0)$

$\qquad -\ 2/6\ (1/2 \log_2 1/2 + 1/2 \log_2 1/2)$

$= 2/6$

Information gain

- Decrease in entropy (uncertainty) after splitting

$$IG(X) = H(Y) - H(Y \mid X)$$

In our running example:

$IG(X_1) = H(Y) - H(Y|X_1)$
$= 0.65 - 0.33$

$IG(X_1) > 0 \rightarrow$ we prefer the split!

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

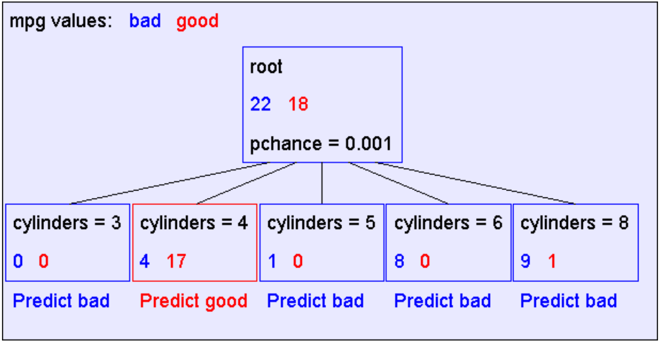
Learning decision trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute:

$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$$

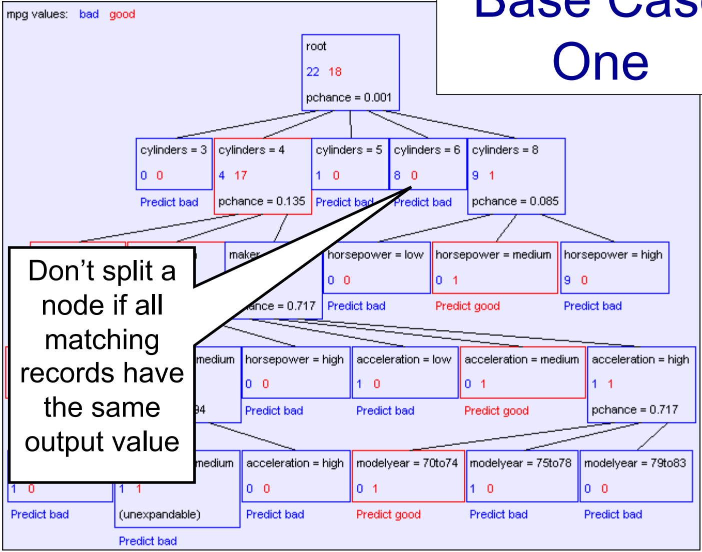
- Recurse

When to stop?



First split looks good! But, when do we stop?

Base Case One



Don’t split a node if all matching records have the same output value

Base Case Two