Rule based classification

We need labelled data to build a classifier

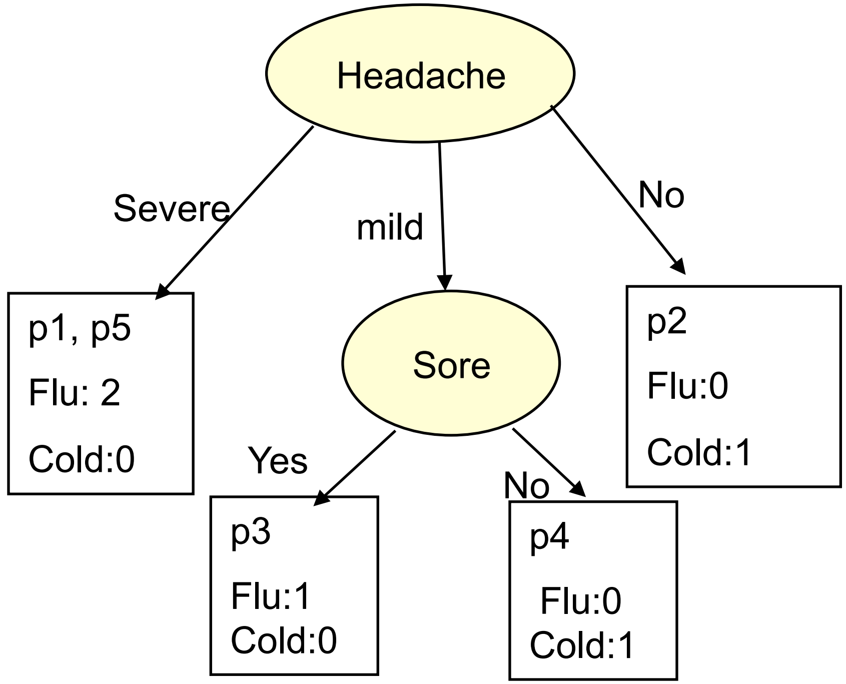
Headache	Cough	Temperature	Sore	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	cold
severe	severe	normal	yes	Flu

Rule based classification

Patient#	Headache	Cough	Temperature	Sore	Diagnosis
p1	severe	mild	high	yes	Flu
P2	no	severe	normal	yes	Cold
P3	mild	mild	normal	yes	Flu
P4	mild	no	normal	no	cold
p5	severe	severe	normal	yes	Flu

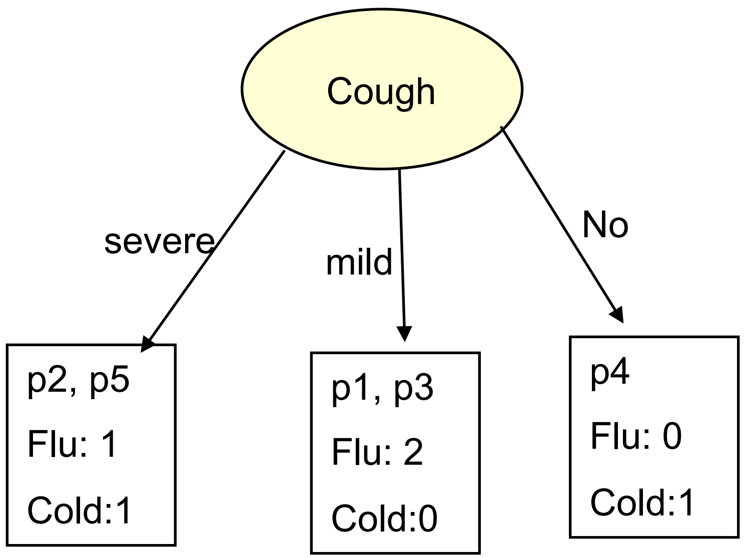
One good approach is

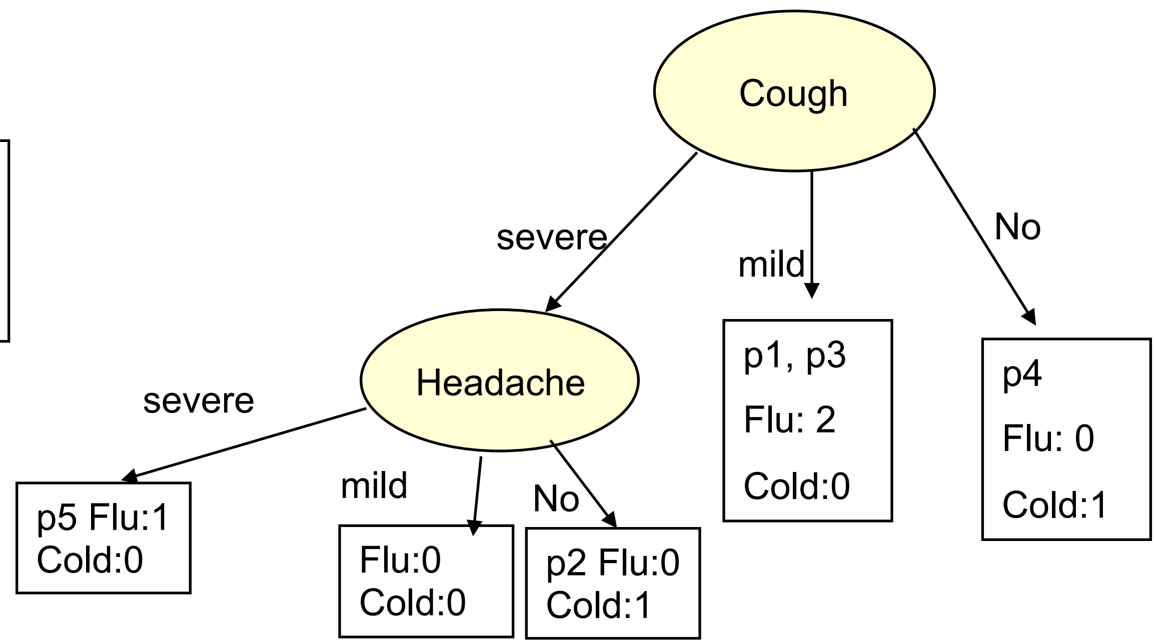
- Construct a decision tree
- Extract one rule for each leaf node
- Example:
  - Rule1: if (headache = severe) then it is Flue
  - Rule2: if (headache = mild) and (Sore = yes) then it is Flu
  - Rule3: if (headache = mild) and (Sore = no) then it is Cold
  - Rule 4: if (headache = no) then it is Cold



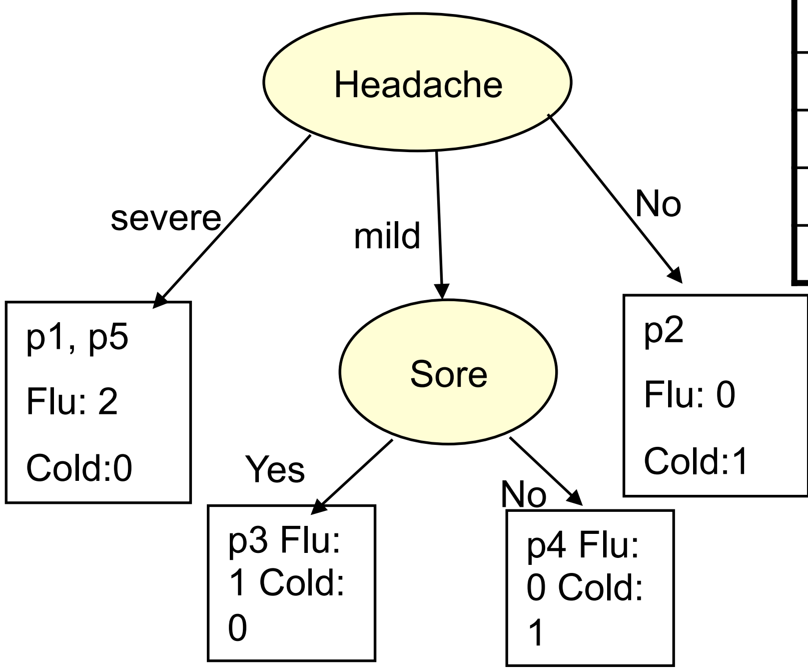
Decision Trees

Patient#	Headache	Cough	Temperatu re	Sore	Diagnosis
p1	severe	mild	high	yes	Flu
P2	no	severe	normal	yes	Cold
P3	mild	mild	normal	yes	Flu
P4	mild	no	normal	no	cold
p5	severe	severe	normal	yes	Flu





Decision Trees

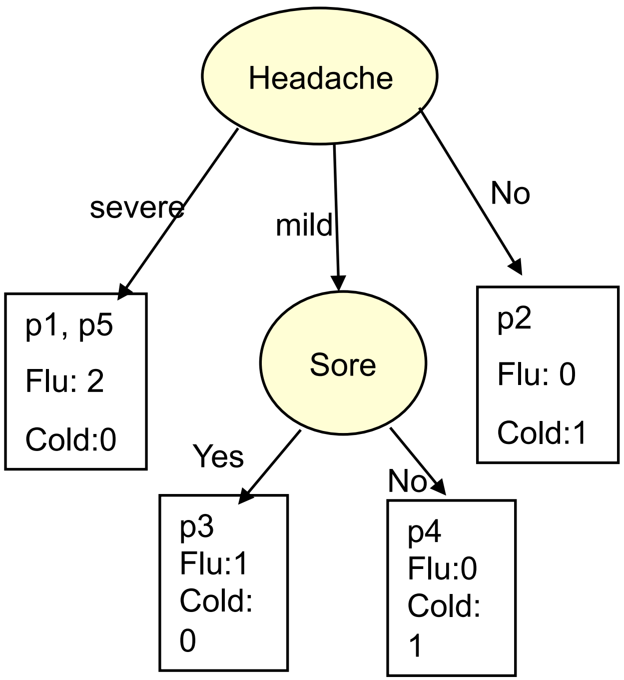


Patient#	Headache	Cough	Temperature	Sore	Diagnosis
p1	severe	mild	high	yes	Flu
P2	no	severe	normal	yes	Cold
P3	mild	mild	normal	yes	Flu
P4	mild	no	normal	no	cold
p5	severe	severe	normal	yes	Flu

Issues:

- How to build optimal Decision Tree?
- How to choose attribute values at each decision point (node)?
- How to choose number of branches at each node and attribute values for partitioning the data?
- When to stop the growth of the tree?

Decision Trees



Issues:

- How to build optimal Decision Tree for a given training data set?
- How to choose attribute values at each decision point (node)?
- How to choose number of branches at each node and attribute values for partitioning the data?
- When to stop the growth of the tree?

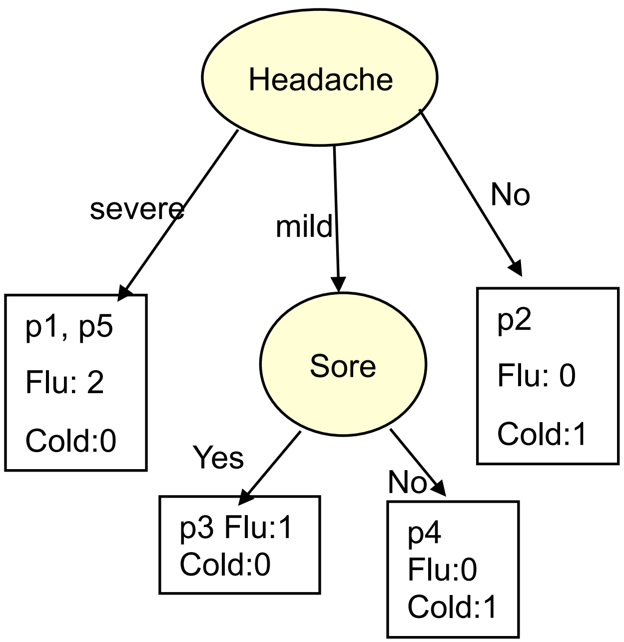
Optimal construction of a Decision Tree is NP (non-deterministic polynomial) hard.

So we use heuristics:

- Choose an attribute to partition the data at the node such that each partition is as homogeneous (least impure) as possible. This means we would like to see most of the instances in each partition belonging to as few classes as possible and each partition should be as large as possible.

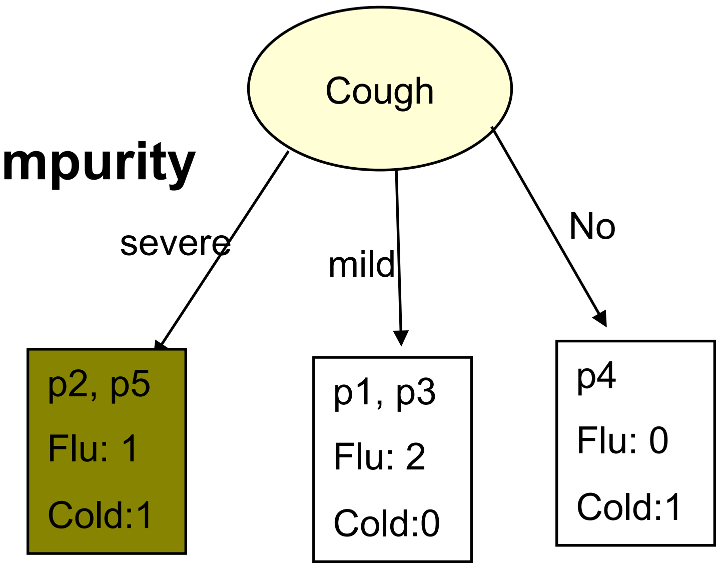
- We can stop the growth of the tree if all the leaf nodes are largely dominated by a single class (that is the leaf nodes are nearly pure).

Decision Trees

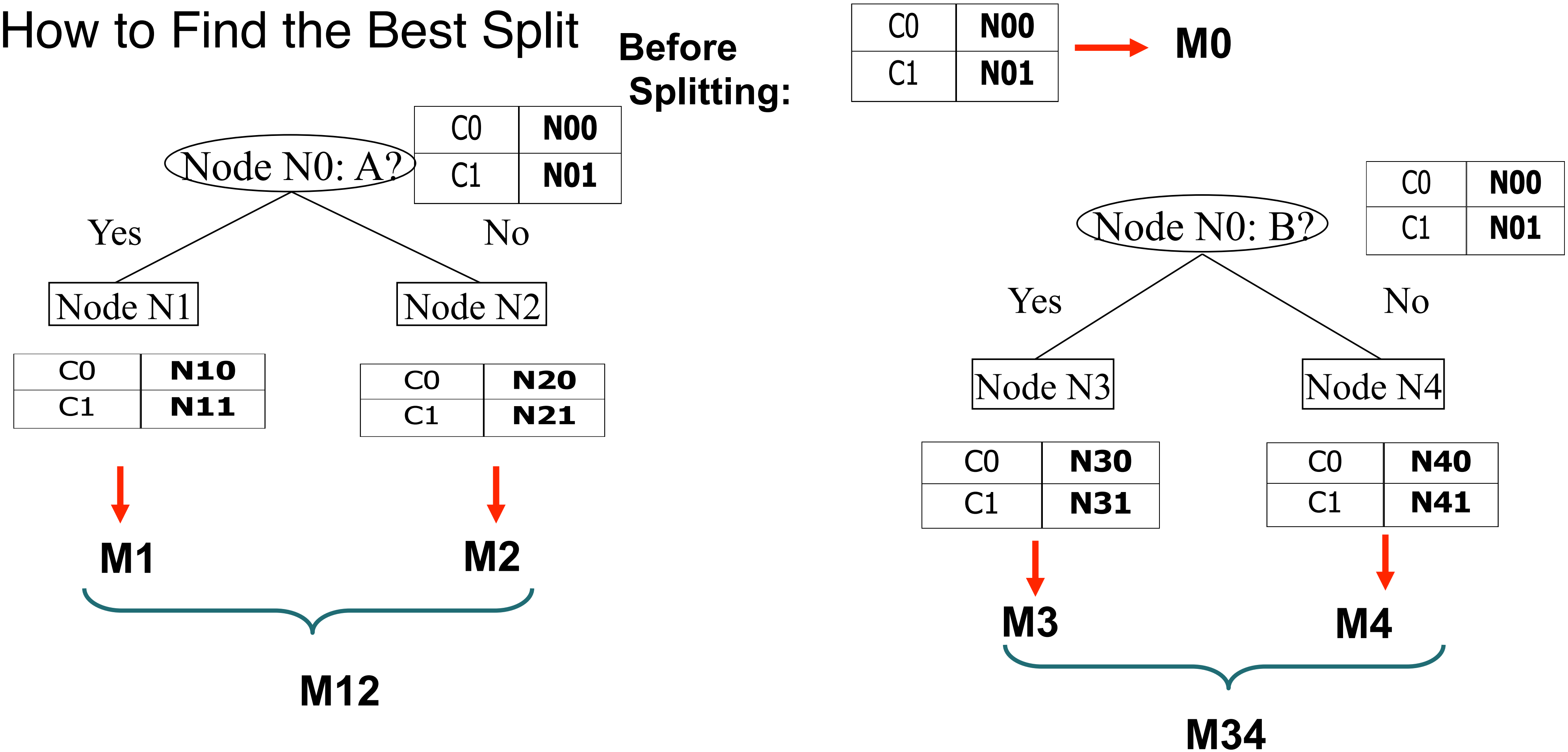


Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error



How to Find the Best Split



Gain = (M0 – M12) vs (M0 – M34)

Where M is some measure of impurity (discussed later).

One Measure of Impurity: GINI Index

Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

(Where *p(j | t)* is the relative frequency of class j at node t).

- Maximum value of Gini index = $(1 - 1/n_c)$ when records are equally distributed among all classes, implying least interesting information or most impure.
- Minimum is (0.0) when all records belong to one class, implying most interesting information or most pure or most homogeneous
- Examples:

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

$1 - (0/6)^2 - (6/6)^2 = 0$ $1 - (1/6)^2 - (5/6)^2 = 0.278$ $1 - (2/6)^2 - (4/6)^2 = 0.444$ $1 - (3/6)^2 - (3/6)^2 = 0.5$

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

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

Splitting Based on GINI

Used in CART, SLIQ, SPRINT.
When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,
n = number of records at parent node p.

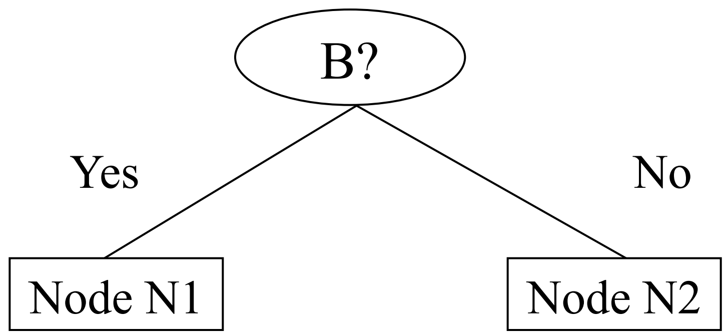
If $GINI(j) - GINI_{split}(j) > delta$ then split the node j.

Binary Attributes: Computing GINI Index

Splits into two partitions

Effect of Weighing partitions:

- Larger and Purer Partitions are sought for.



	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		

Gini(Children)
= 7/12 * 0.408 +
5/12 * 0.32
= 0.371

Categorical Attributes: Computing Gini Index

For each distinct value, gather counts for each class in the dataset

Use the count matrix to make decisions if the parent node has instances: 5 Family; 3 Sports and 2 Luxury its Gini Index is 0.62.

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split (find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes: Computing Gini Index

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

Use Binary Decisions based on one value

Several Choices for the splitting value

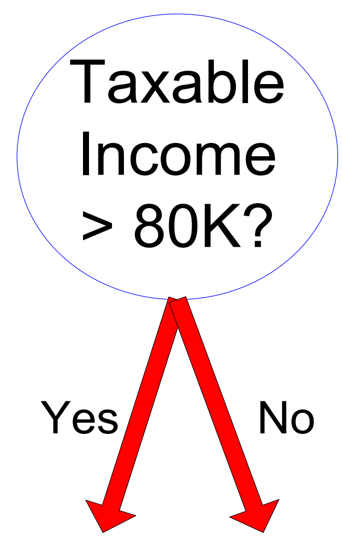
- Number of possible splitting values = Number of distinct values

Each splitting value has a count matrix associated with it

- Class counts in each of the partitions, A < v and A ≥ v

Simple method to choose best v

- For each v, scan the database to gather count matrix and compute its Gini index
- Computationally Inefficient! Repetition of work.



Continuous Attributes: Computing Gini Index...

For efficient computation: for each attribute,

- Sort the attribute on values
- Linearly scan these values, each time updating the count matrix and computing gini index at points where class label changes (at points A and B)
- Choose the split position that has the least gini index

A B

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values	60	70	75	85	90	95	100	120	125	220

Split Positions	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

Alternative Splitting Criteria based on INFORMATION gain

Entropy at a given node t:

$$Entropy(t) = -\sum_{j} p(j \mid t) \log p(j \mid t)$$

(Where *p(j | t)* is the relative frequency of class j at node t).

- Measures homogeneity of a node.
  - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information
- Entropy based computations are similar to the GINI index computations

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

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

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

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

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

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

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

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

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

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Based on INFO...

Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

n_i is number of records in partition i

- Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3 and C4.5
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Splitting Based on INFO...

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

Parent Node, p is split into k partitions

n_i is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain

Splitting Criteria based on Classification Error

Classification error at a node t :

$$Error(t) = 1 - \max_i P(i \mid t)$$

Measures misclassification error made by a node.

- Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

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

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

Error = 1 – max (0, 1) = 1 – 1 = 0

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

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

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

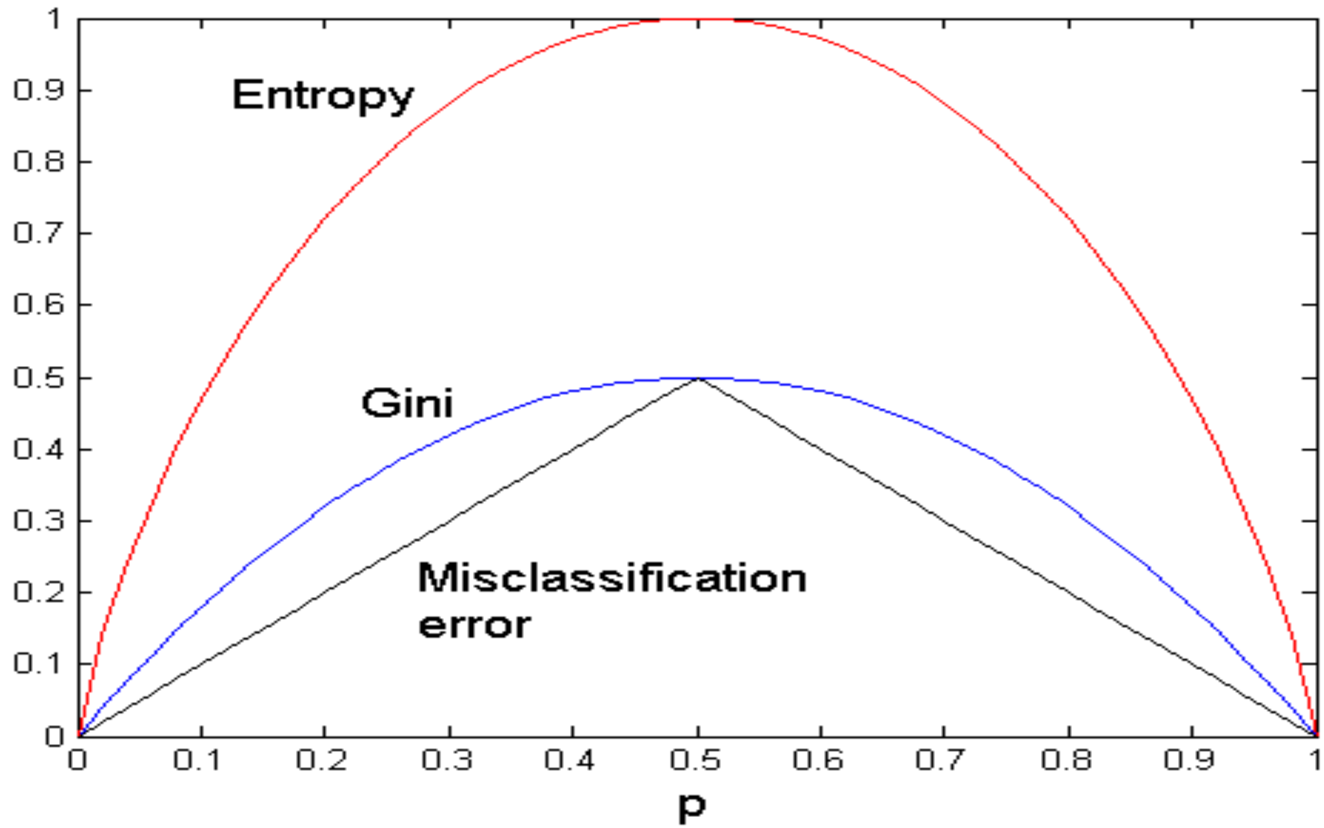
C1	**2**
C2	**4**

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

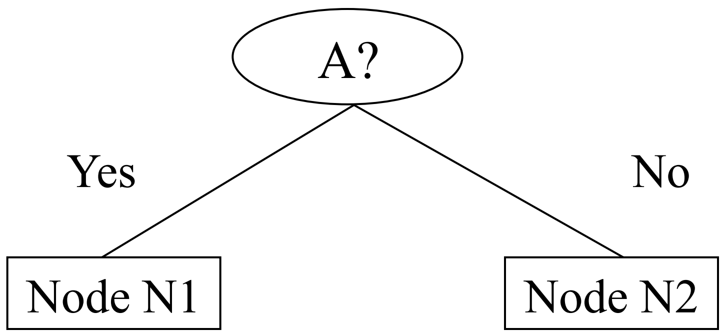
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

For a 2-class problem:



Misclassification Error vs Gini



	Parent
C1	7
C2	3
Gini = 0.42	
MissClass = 0.3	

Gini(N1)
$= 1 - (3/3)^2 - (0/3)^2$
$= 0$

Gini(N2)
$= 1 - (4/7)^2 - (3/7)^2$
$= 0.489$

MissClass(N1) = 1 –(3/3) = 0

MissClass(N2) = 1 – 4/7 = 3/7

	N1	N2
C1	3	4
C2	0	3
Gini = 0.342		
MissClass =0.3		

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

Gini improves !!

MissClass =3/10 *0+(7/10)*(3/7)=0.3

Missclassification unchanged!

Tree Induction

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

Greedy strategy

- Choose the Attribute and Splitting test to partition the records that optimizes certain criterion. E.g. GINI index, Entropy, MissClassification Rate.

Stopping Criteria for Tree Induction

Stop expanding a node when all the records belong to the same class. That is the node is pure. One can stop e.g. when the GINI index is close to zero.

Stop expanding a node when all the records have similar attribute values.

This means we cannot partition the data anymore even if the node is impure.

Early termination (to be discussed later)

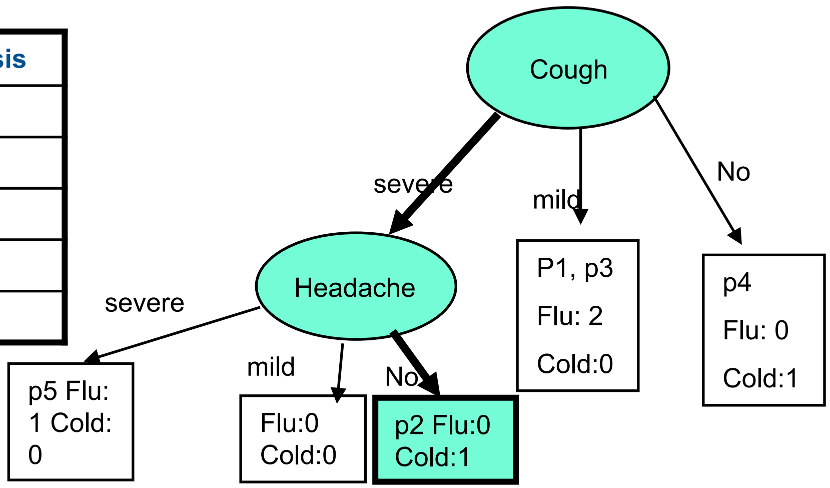
Decision Making using Decision Tree

Traverse the branch of the decision tree from the root node matching the corresponding attribute values of the test record and the traversal reaches the Leaf Node. Make the decision based on the Leaf Node Class distribution. E.g. label the test data as the class of highest frequency class in the Leaf Node.

Decision Trees

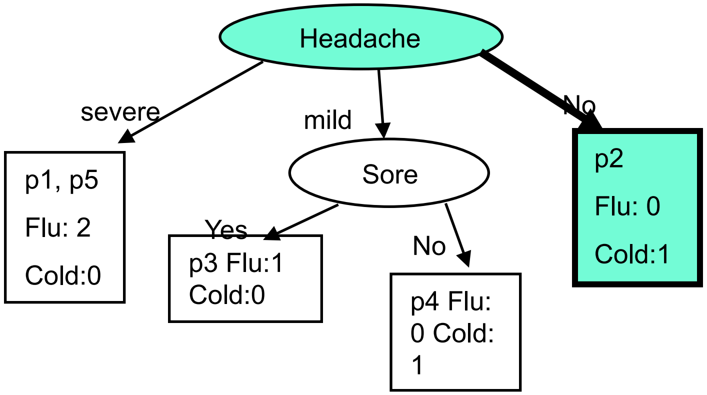
Training Data

Patient#	Headache	Cough	Temperature	Sore	Diagnosis
p1	severe	mild	high	yes	Flu
P2	no	severe	normal	yes	Cold
P3	mild	mild	normal	yes	Flu
P4	mild	no	normal	no	cold
p5	severe	severe	normal	yes	Flu



Test data

Patient#	Headache	Cough	Temperature	Sore	Diagnosis
px	no	severe	high	yes	?



Decision Tree Based Classification

Advantages:

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

Example: C4.5

Simple depth-first construction.

Uses Information Gain

Sorts Continuous Attributes at each node.

Needs entire data to fit in memory.

Unsuitable for Large Datasets. But this problem can be addressed easily.

- Needs disk based sorting.

You can download the software from:
http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

Practical Issues of Classification

Underfitting and Overfitting

Missing Values

Costs of Classification