Data Mining

Lecture 9

Ananya Jana
CS360

Fall 2024

Design Issues of Decision Tree Induction

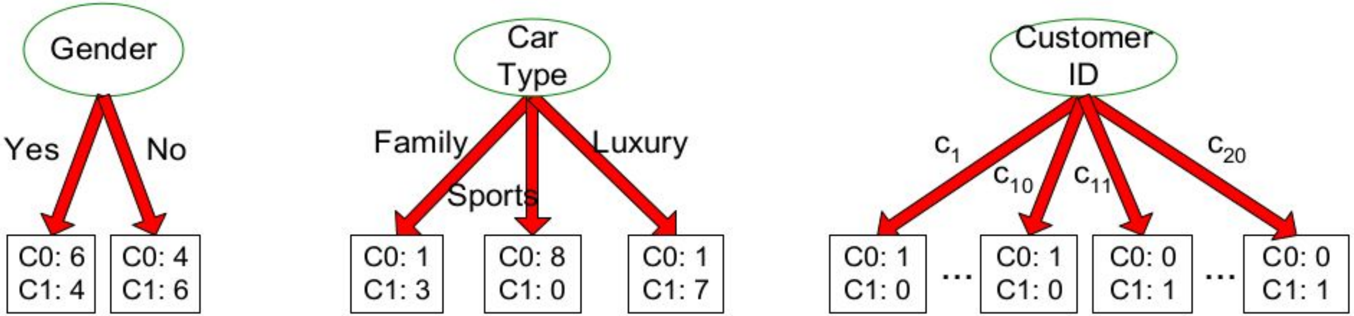
- How should training records be split?
  - Method for specifying test condition
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition

- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

How to determine the best split

Before Splitting: 10 records of class 0,
10 records of class 1

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1



Which test condition is the best?

How to determine the best split

- Greedy approach:
  - Nodes with purer class distribution are preferred

- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
High degree of impurity	**Low degree of impurity**

Measures of Node Impurity

- Gini Index

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

- Entropy

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

- Misclassification error

$$Error(t) = 1 - \max_{i} P(i \mid t)$$

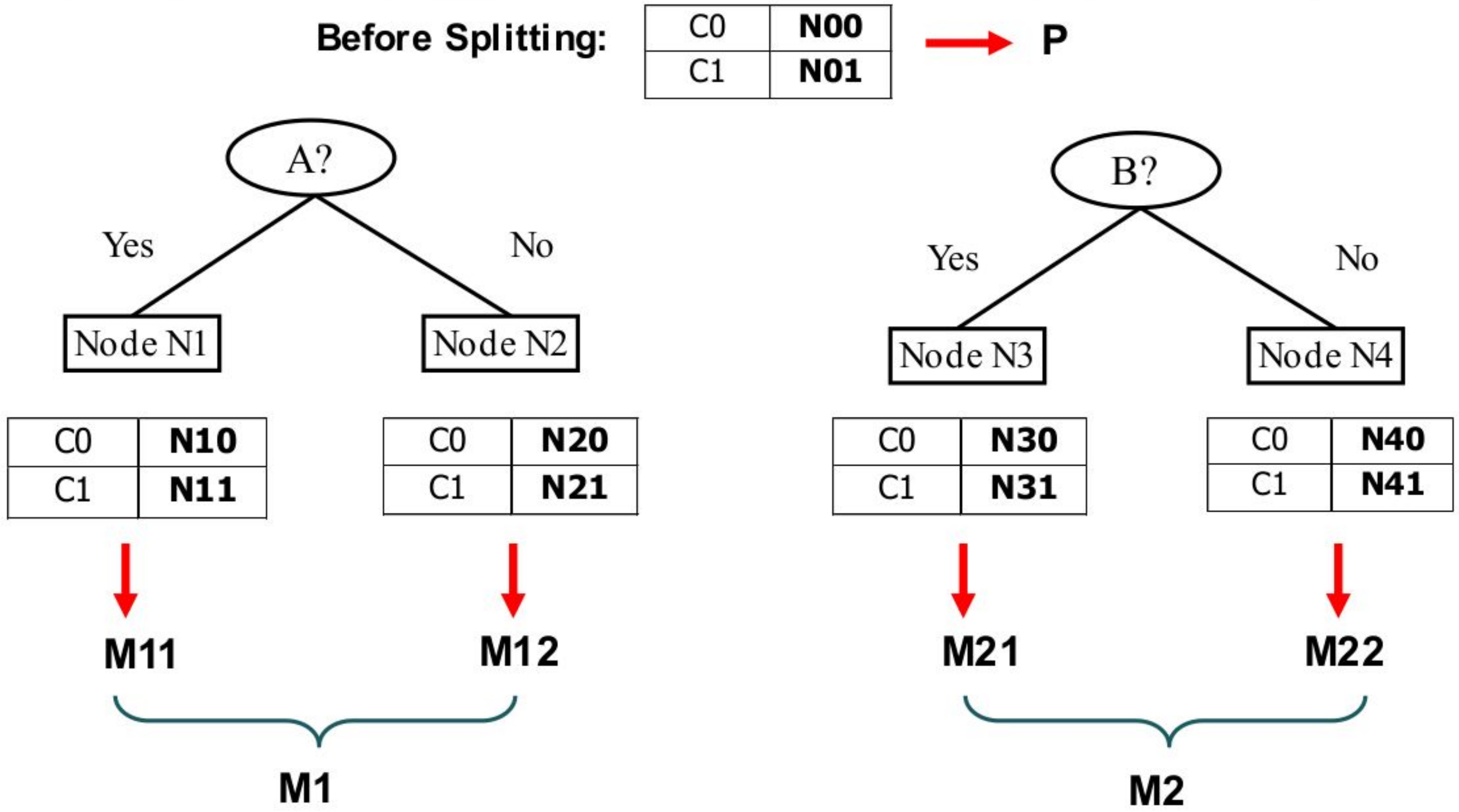
Finding the best split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of children
3. Choose the attribute test condition that produces the highest gain

   Gain = P – M

   or equivalently, lowest impurity measure after splitting (M)

Finding the best split



Gain = P – M1 vs P – M2

Measure of impurity: Entropy

- Entropy at a given node t:

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

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- 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

Computing Entropy of a single node

$$Entropy(t) = -\sum_{j} p(j \mid t) \log_2 p(j \mid t)$$

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 $= - (2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$

Computing Information Gain after Splitting

- 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

  - Choose the split that achieves most reduction (maximizes GAIN)

  - Used in the ID3 and C4.5 decision tree algorithms

Measure of Impurity: Gini

- Gini Index for a given node t :

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

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  – Maximum ($1 - 1/n_c$) when records are equally distributed among all classes

  – Minimum (0.0) when all records belong to one class,

Measure of Impurity: Gini

- Gini Index for a given node t :

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

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- For 2-class problem (p, 1 – p):
  - GINI = $1 - p^2 - (1 - p)^2 = 2p(1-p)$

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	

Computing Gini Index of a single node

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

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

Computing Gini Index for a collection of nodes

- When a node p is split into k partitions (children)

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

where, n_i = number of records at child i,

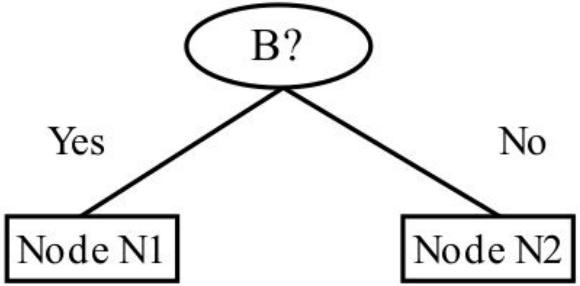
n = number of records at parent node p.

- Choose the attribute that minimizes weighted average Gini index of the children

- Gini index is used in decision tree algorithms such as CART, SLIQ, SPRINT

Binary Attributes: Computing Gini Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.



	Parent
C1	7
C2	5
Gini = 0.486	

Gini(N1)
$= 1 – (5/6)^2 – (1/6)^2$
$= 0.278$

Gini(N2)
$= 1 – (2/6)^2 – (4/6)^2$
$= 0.444$

	N1	N2
C1	5	2
C2	1	4
Gini=0.361		

Weighted Gini of N1 N2
$= 6/12 * 0.278 +$
$6/12 * 0.444$
$= 0.361$

Gain = 0.486 – 0.361 = 0.125

Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

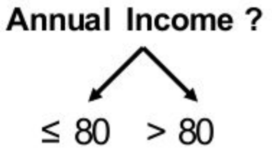
	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

Which of these is the best?

Continous Attributes: Computing Gini Index

- 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.

ID	Home Owner	Marital Status	Annual Income	Defaulted
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



	≤ 80	> 80
Defaulted Yes	0	3
Defaulted No	3	4

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
  - Choose the split position that has the least gini index

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

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
  – Choose the split position that has the least gini index

	Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
		Annual 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
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >

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
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Annual 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														
No							3	4														
Gini							0.343															

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
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual 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	1	2												
No							3	4	3	4												
Gini							0.343		0.417													

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
  - Choose the split position that has the least gini index

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

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

Task

1. Compute the initial Gini index.
   Next, compute GINI index after you split based on the attribute
   i) Gender,
   ii) Car Type (assume the split is 3-way)

2. Do the same exercise as 1 with entropy as the measure of impurity.

3. Show that the maximum Gini index of a node is (1 - ($1/n_c$)) when all records are equally distributed among the n_c different classes. [n_c the number of classes]

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1