Data Science

04.01-02 의사결정나무

나무모델 생성 알고리즘에서 노드의 분할 (split)

나무 모델 생성에서의 issues

Issues

- 노드를 어떻게 split 할 것인가?
  - 테스트조건을 어떻게 정할 것인가?
  - 최선의 split 을 어떻게 정할 것인가?
- 언제 split 을 중지할 것인가?
  (언제 나무의 성장을 중지할 것인가?)

나무 모델 생성에서의 issues

Issues

- 노드를 어떻게 split 할 것인가?
  - 테스트조건을 어떻게 정할 것인가?
  - 최선의 split 을 어떻게 정할 것인가?
- 언제 split 을 중지할 것인가?
  (언제 나무의 성장을 중지할 것인가?)

테스트 조건

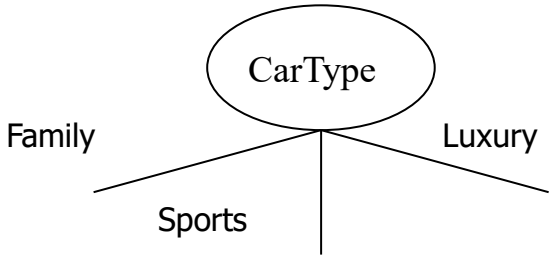
- Split 의 가지 수 에 따라
  - 2-way split
  - Multi-way split

- 속성의 유형 에 따라
  - 명목형
  - 순위형
  - 연속형

명목형 속성을 기반으로 split

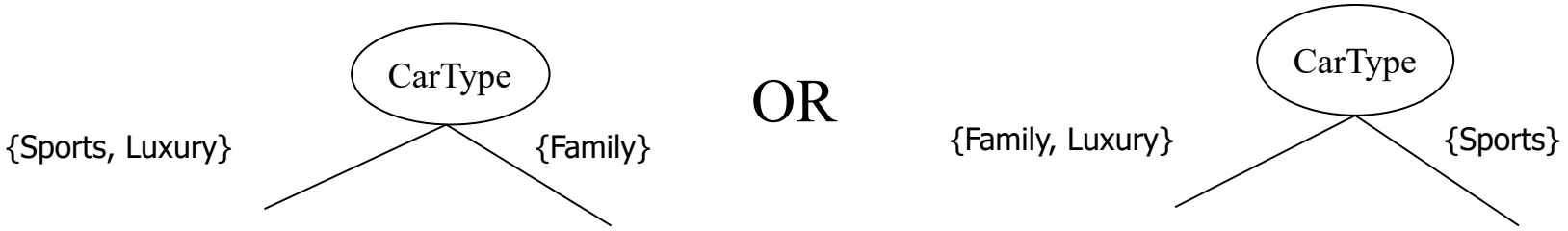
Multi-way split:

- 속성 값마다 하나의 가지로 분할. 데이터셋이 속성값의 종류 수와 같은 부분집합으로 분할됨.



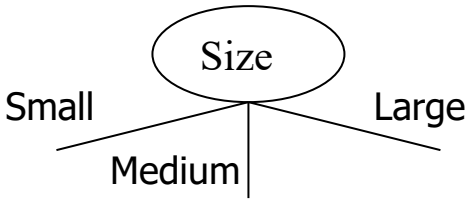
Binary split:

- 두 개의 부분 집합으로 분할 함.
- 최적한 분할을 탐색.

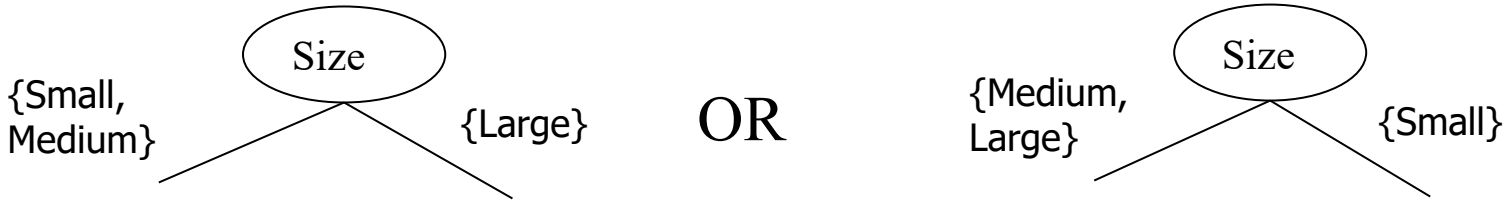


순위형 속성을 기반으로 split

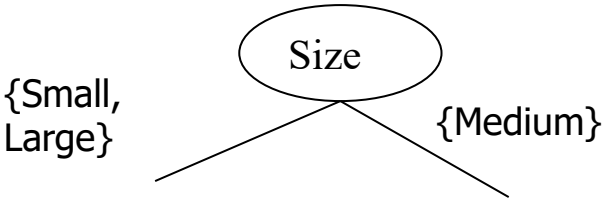
- Multi-way split:



- Binary split:



- 이러한 split는 어떨까?



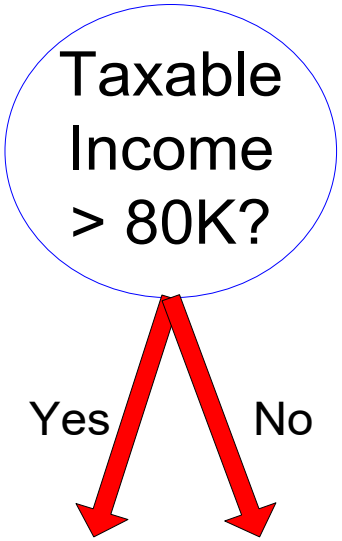
연속형 속성을 기반으로 split

2 가지의 처리 방법

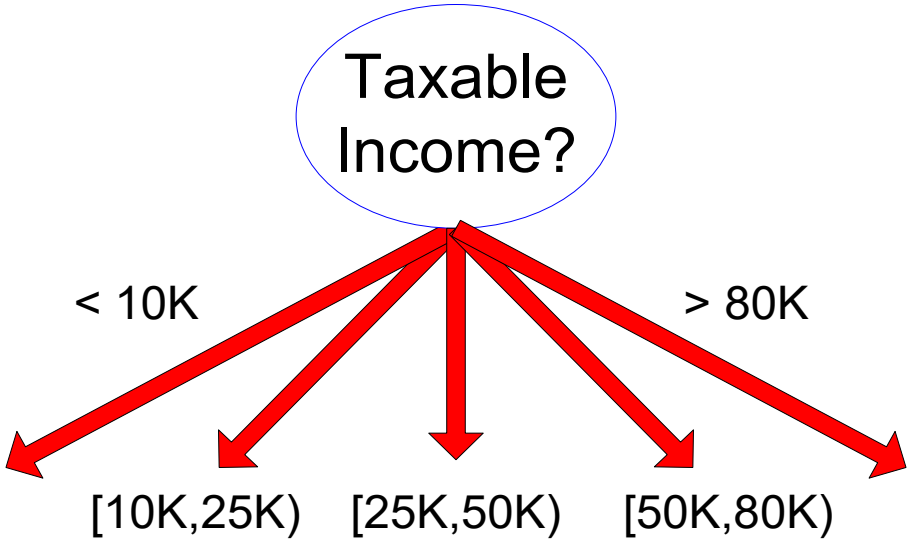
- 이산화 (Discretization) → 순위형 속성으로 변환
  - 정적(static) – 최초에 한번만 이산화
  - 동적(dynamic) – splitting 할 때 마다 이산화
  - 구분 영역(bin): equal interval bucketing, equal frequency bucketing(percentiles), or clustering.

- Binary Decision: $(A < v)$ or $(A \geq v)$
  - 임계점 v 를 기준으로 두 구간으로 나눔.
  - 가능한 split 를 모두 고려 하여 임계점 v (best cut) 을 찾는다.
  - 계산량이 많다.

연속형 속성을 기반으로 split



(i) Binary split



(ii) Multi-way split

나무 모델 생성에서의 issues

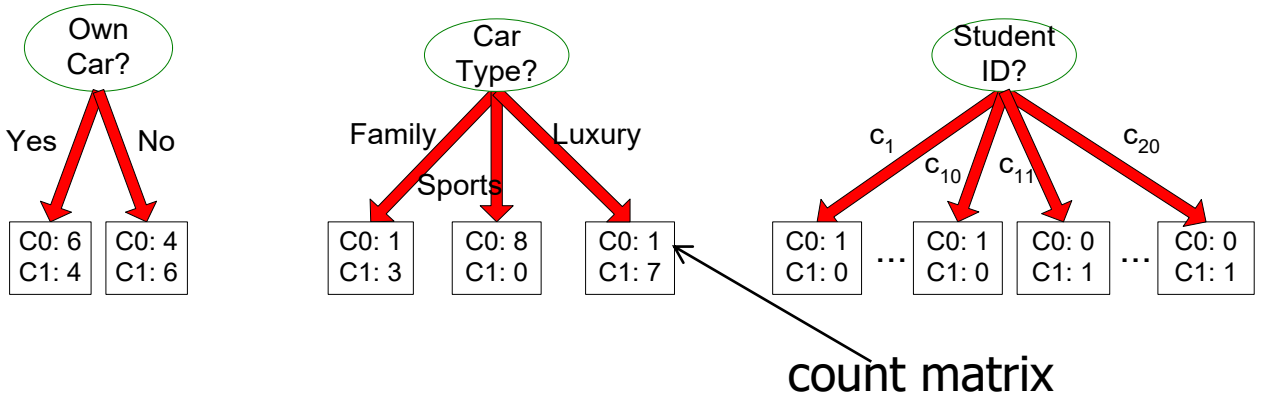
Issues

- 어떻게 split 할 것인가?
  - 테스트조건을 어떻게 정할 것인가?
  - 최선의 split 을 어떻게 정할 것인가?
- 언제 split 을 중지할 것인가?
  (언제 나무의 성장을 중지할 것인가?)

최적의 split 정하기(best split)

Before Splitting:
부모 노드
10 사례가 class 0,
10 사례가 class 1

After Splitting:
자식노드



Which test condition is the best?

최선의 split 정하기

- 탐욕적 접근 방법(Greedy approach):
  - Split 한 후 각 자식노드의 class distribution 이 동질적 (homogeneous, pure)이 되도록 함.

C0: 5
C1: 5

비동질적, 혼합
불순도가 높음

C0: 9
C1: 1

동질적, 순수,
불순도가 낮음

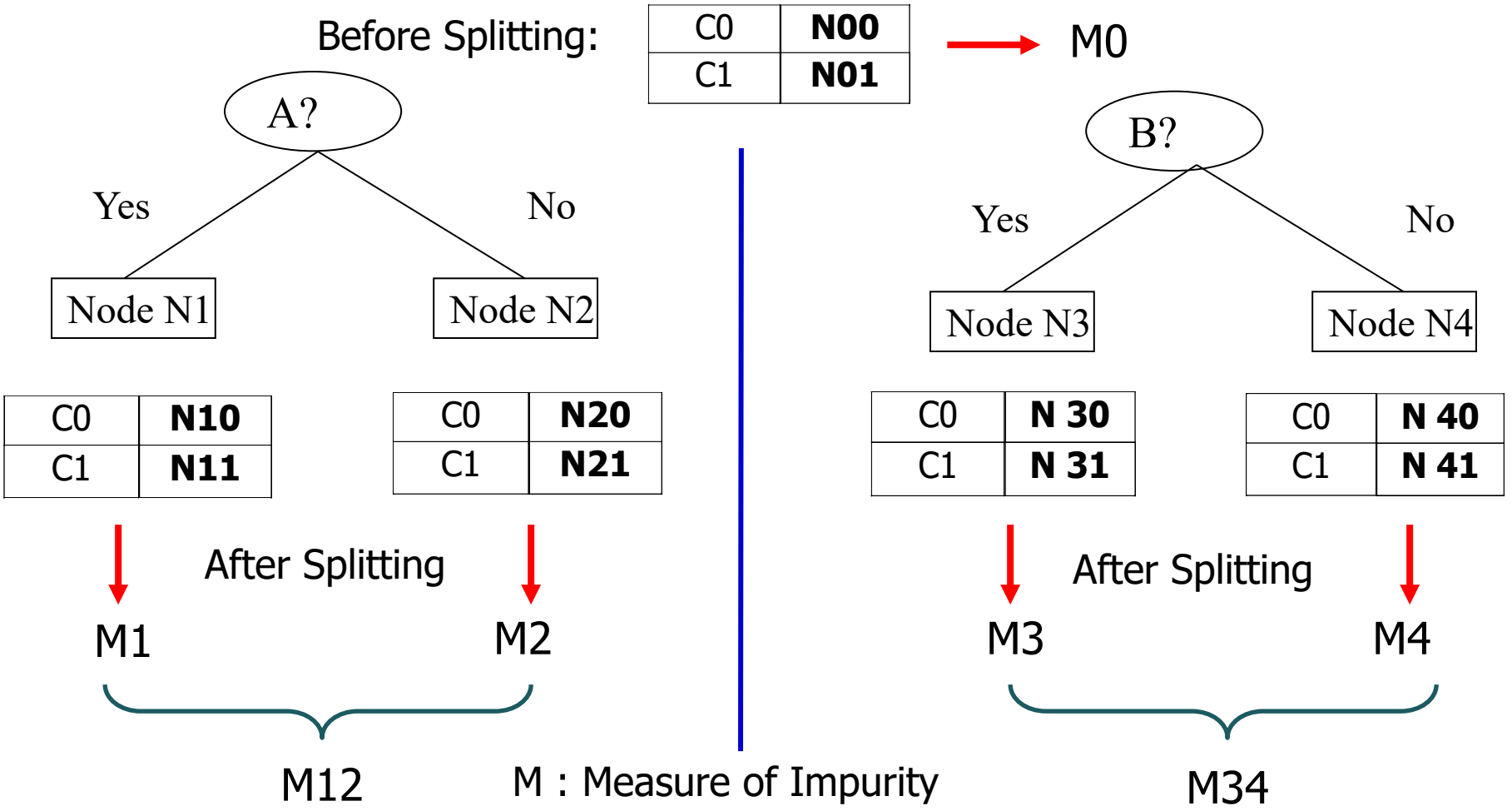
- 동질적인 정도 을 나타내는 불순도(impurity) 척도가 필요함.

노드에 대한 불순도 척도

- GINI Index
- Entropy
- Misclassification error (error rate)

최선의 split 찾기 – split에서의 Gain

Which split is better, A or B ?



GINI index

- GINI Index for a given node t :

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

(NOTE: $p(j \mid t)$ 는 node t 의 데이터 세트에서 class j 의 비율)

- Minimum (0) : 모든 사례가 한 개의 class 에 속하는 경우
  → most interesting information

- Maximum (1 - $1/n_c$) : 각 class 마다 record 수가 동일하게 분포
  → least interesting information

C0	**0**
C1	**6**
Gini=0.000	

C0	**1**
C1	**5**
Gini=0.278	

C0	**2**
C1	**4**
Gini=0.444	

C0	**3**
C1	**3**
Gini=0.500	

GINI 예제

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

C0	0
C1	6

P(C0) = 0/6 = 0 P(C1) = 6/6 = 1
Gini = 1 – P(C0)^2 – P(C1)^2 = 1 – 0 – 1 = 0

C0	1
C1	5

P(C0) = 1/6 P(C1) = 5/6
Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278

C0	2
C1	4

P(C0) = 2/6 P(C1) = 4/6
Gini = 1 – (2/6)^2 – (4/6)^2 = 0.444

GINI 를 기반으로 한 split

- CART, SLIQ, SPRINT 에서 사용.
- 노드 *t* 가 *k* 개의 자식노드로 split(분할) 되는 경우의 분할 후 의 GINI는 다음 식과 같이 가중평균으로 계산함.

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

where, n_i = 자식노드 *i* 에서의 데이터셋의 사례수

n = 노드 *t* 에서의 데이터셋의 사례수

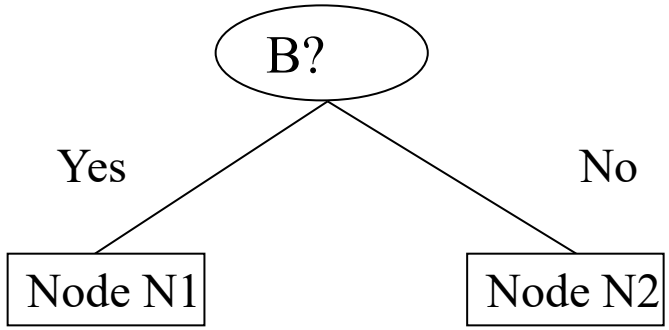
- GINI Gain = 부모 node 의 GINI – split 후 의 GINI
- GINI Gain 를 최대로 하는 split를 선택!!!

이진형 속성: GINI index 계산

두 개의 노드로 split 될 때

가중 평균의 효과:

- 크고 순수한 데이터셋으로의 split 이 선호됨.



	Parent
C0	6
C1	6
Gini = 0.50	

GINI(N1)
$= 1 - (5/7)^2 - (2/7)^2$
$= 0.41$

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

GINI(Children)
$= 7/12 * 0.41 + 5/12 * 0.32$
$= 0.37$

	N1	N2
C0	5	1
C1	2	4
Gini=0.37		

명목형 속성에서의 GINI 의 계산

- Count matrix 생성:
  - 각 속성 값에 따라 데이터셋을 서브 데이터셋으로 분할하고 각 서브데이터셋에 대하여 각각의 class 에 속하는 사례의 수를 센다.
- Count matrix 를 이용하여 GINI index 계산 split 을 선택

Multi-way split

	CarType		
	Family	Sports	Luxury
C0	1	2	1
C1	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C0	3	1
C1	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C0	2	2
C1	1	5
Gini	0.419	

연속형 속성: GINI index 계산

- 하나의 임계값(cut-off)을 정하여 Binary split하는 경우
- 최적의 임계값을 탐색
  - 가능한 split 값의 개수 = 서로 다른 속성 값의 갯수.
- 임계값 마다 count matrix 를 계산할 수 있음.
  - $A < v$, $A \geq v$ 의 두 분할 영역의 count
- v 에 대한 최적 값 탐색 방법
  - v 가 취할 수 있는 각 값마다, DB 를 scan 하여 count matrix 와 GINI index 를 계산함.
  - 반복 계산 → 계산적으로 비효율적

연속형 속성: GINI index 계산

■ 효율적인 방법

For each attribute,

Sort the attribute on values.

Linearly scan these values, each time updating(not counting again) 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
	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

Only one example difference between two neighboring count matrix

Entropy 와 INFO

- node t 에서의 entropy:

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

(NOTE: $p(j \mid t)$ 는 node t 에서 class j 의 비율).

  - Minimum (0) : 모든 사례 가 하나의 class 에 속할 경우 → most information
  - Maximum ($\log_2 n_c$) : 데이터 셋이 class 별로 동일 비율의 사례를 가지고 있을 때 → least information

Entropy 계산의 예

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

C0	0
C1	6

P(C0) = 0/6 = 0 P(C1) = 6/6 = 1
Entropy = – {0 log 0 + 1 log 1} = – {0 + 0} = 0

C0	1
C1	5

P(C0) = 1/6 P(C1) = 5/6
Entropy = – {(1/6) $\log_2$ (1/6) + (5/6) $\log_2$ (5/6)} = 0.65

C0	2
C1	4

P(C0) = 2/6 P(C1) = 4/6
Entropy = – {(2/6) $\log_2$ (2/6) + (4/6) $\log_2$ (4/6)} = 0.92

INFO 를 기반으로 split 하기

- Information Gain:
  - split 전, 후의 Entropy 의 차이
  - $$GAIN_{split} = Entropy(t) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

    임의의 노드가 k 개의 자식노드로 split 됨;

    n_i 는 자식 노드 i 에 할당된 데이터셋의 사례의 수
  - Split에 대한 entropy의 감소량을 계산함.
  - 감소량을 최대로 하는(maximizes GAIN) split 를 선택함.
  - ID3 and C4.5에서 사용.
  - 단점: 분할 영역 의 개수가 많을 수록 GAIN 값이 커짐 → 소규모의 분할영역을 많이 만든다.

INFO 를 기반으로 split 하기

- Gain Ratio:

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

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

노드 t, 가 k 개의 자식 노드로 split 됨;
n_i 는 자식노드 i 의 할당된 데이터셋의 사례의 수.

- SplitINFO 는 splitting 에 대한 entropy 를 나타냄. 작은 데이터 셋일 수록 높은 값.
- GainRATIO 는 Information Gain 을 SplitINFO (entropy of splitting)로 나눈값. → 작은 데이터 셋 여러 개로 나누는 것에 penalty
- Information Gain 의 단점을 극복.
- C4.5 에서 사용.

Misclassification Error 를 기반으로 split

- 노드 t 에서의 misclassification Error:

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

데이터셋에서 가장 많은
사례를 가지는 class의 비율

- Minimum (0) : 모든 사례 가 동일한 class 에 속할 경우 → most information
- Maximum (1 - $1/n_c$) : 모든 class 가 동일한 숫자의 사례를 가지고 있을 때 → least information

Examples for Computing Error

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

C0	0
C1	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1
Error = 1 – max (0, 1) = 1 – 1 = 0

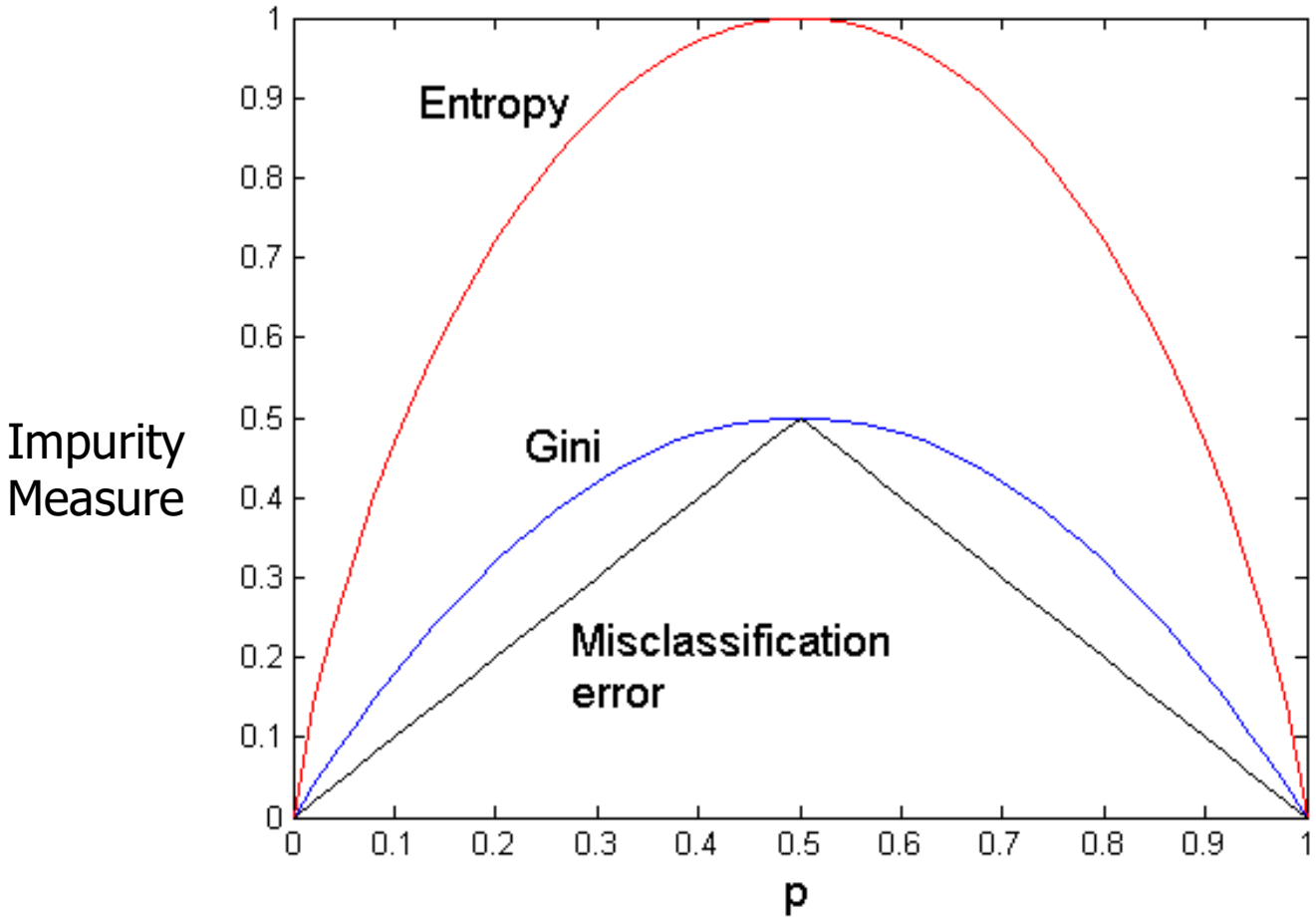
C0	1
C1	5

P(C1) = 1/6 P(C2) = 5/6
Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C0	2
C1	4

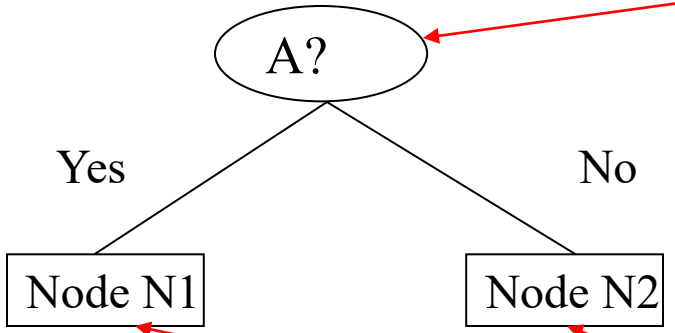
P(C1) = 2/6 P(C2) = 4/6
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

불순도 척도의 비교



2 class classification 문제에서 impurity measure 의 비교
p: 한 개 노드에 주어진 dataset 에서 하나의 class 가 차지하는 비율

Misclassification Error vs Gini



	Parent
C0	7
C1	3
Gini = 0.42	
C.E. = 0.3	

Gini(N1)
= 1 – $(3/3)^2$ – $(0/3)^2$
= 0
Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

	N1	N2
C0	3	4
C1	0	3
Gini = 0.342		
C.E. = 0.3		

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

Split 한 후 Gini 는 개선 되나
Missclassification Error 는 그대로.

Note:

- Split(분할)은 예측 변수의 값을 기준으로 함.
- 불순도 측정은 목표 변수의 값의 비율(분포)를 이용함.
- 불순도가 더 커지는 split 는 손해...

나무 모델 생성에서의 issues

Issues

- 어떻게 split 할 것인가?
  - 테스트조건을 어떻게 정할 것인가?
  - 최적의 split 을 어떻게 정할 것인가?
- 언제 split 을 중지할 것인가?
  (언제 나무의 성장을 중지할 것인가?)

나무 성장을 중지하는 조건

- 아래의 경우에는 노드를 더 이상 split 하는 것이 불가능하며 해당 노드는 잎노드가 됨

- 노드에 부여되는 데이터셋의 사례가 모두 동일한 class 에 속할 경우 stop
  - 해당 노드를 잎노드로 하고 class label 을 붙임.

- 노드에 부여되는 데이터셋의 사례들이 모든 속성에 대하여 비슷한 값을 가질 경우 stop.
  - 이 경우에는 노드에 부여된 데이터 셋에서 다수 클래스로 잎노드의 label 을 붙임

사전 가지 치기 - prepruning

- 사전 가지치기
  - 나무가 불필요하게 크게 자라는 것(overfitting – 과적합)을 방지하기 위하여 노드의 split 를 제한함
  - 아래 조건에 해당 되는 경우 노드의 split 를 하지 않는다.
  - 나무의 깊이가 정해진 값 이상,
  - 노드 에서의 데이터 셋의 크기(레코드의 수)가 정해진 값 이하,
  - 노드에서의 불순도가 정해진 값 이하인 경우
  - 위의 값들은 실험에 의하여 최적값을 정한다.

결정나무에 기반한 분류기법의 장점

- 장점:
  - 모델 생성이 저비용이며 분류의 속도가 빠르다.
  - 작은 크기의 트리일 경우 모델에 대한 해석이 용이함
    - 입력(Input)변수 후보들과 타겟(Target)변수의 관계에 대한 통찰을 얻기 쉬움
      →데이터 탐색에 유용
    - 분기되는 변수는 중요한 변수로 간주할 수 있음
    - 분류의 이유를 쉽게 설명할 수 있다
    - 분류 규칙을 추출할 수 있음
    - SQL과 같은 데이터베이스 언어로도 쉽게 표현가능
  - 데이터 준비과정의 노력이 상대적으로 덜 필요로 함
    - 변수 변환과정을 필요로 하지 않음 (정규화과정)
    - 비선형관계도 의사결정나무 성능에 영향을 주지 않음
  - 모델 자체 내에서 특성 선택(feature selection) 또는 변수 가려내기 (variables screen) 수행함
  - 간단한 데이터 세트 일 경우 정확도는 다른 분류 방법에 뒤지지 않는다.