SESSION # 8
< 의사결정나무 >

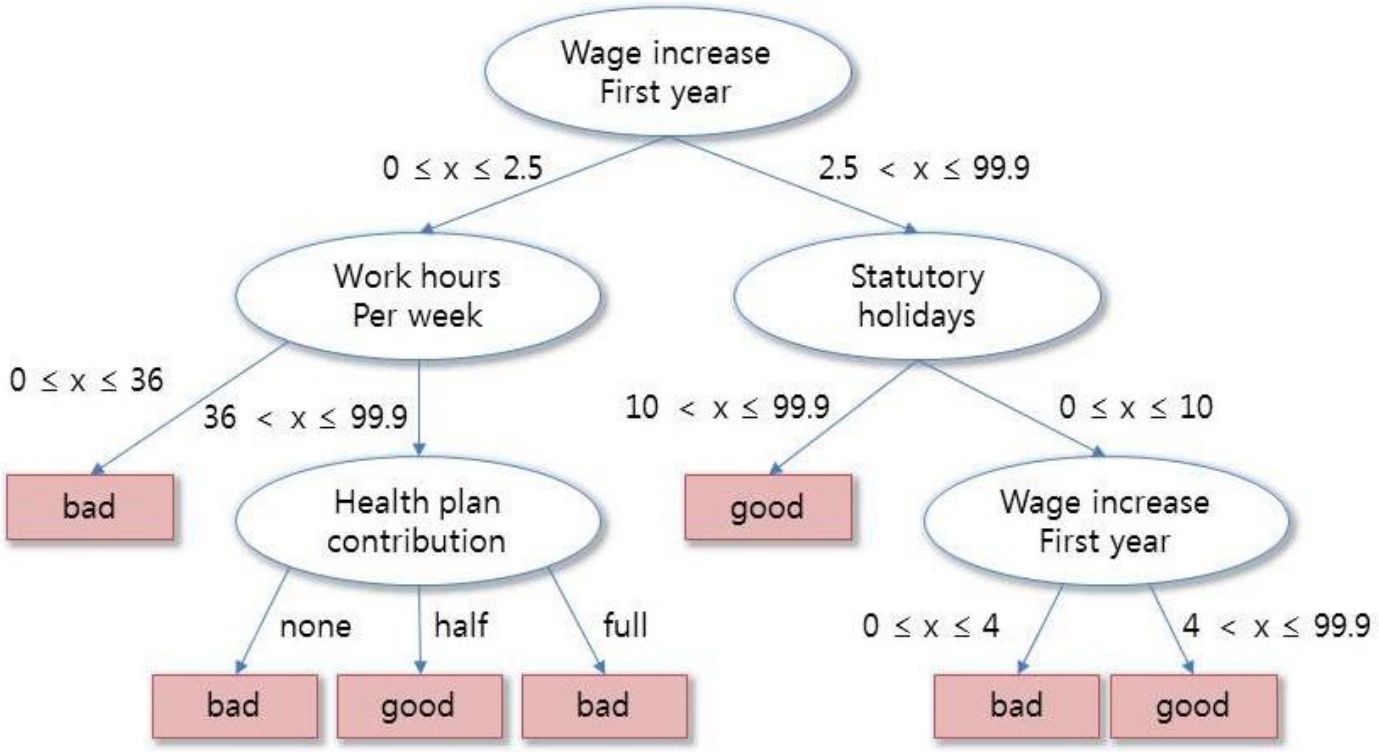
By Team 2
@ 남정우, 신윤기

Date _ 2017.08.22

CONTENTS

01 의사결정나무란?

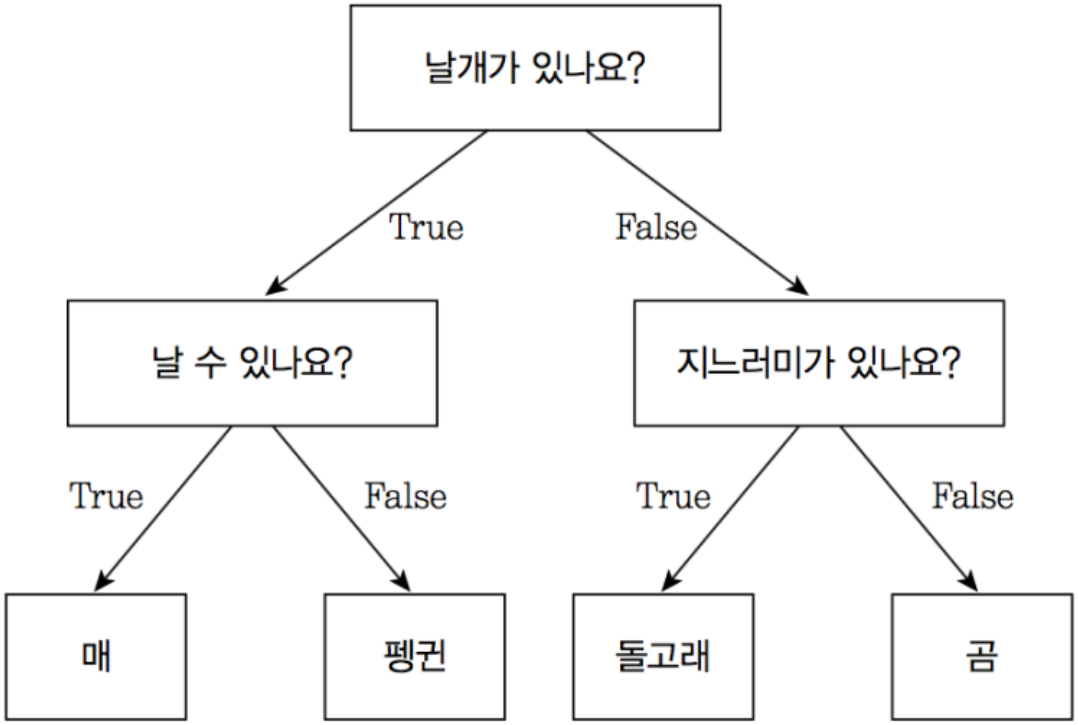
의사결정나무란 무엇인가?



출처 : http://www.dator.co.kr/osung/textyle/360437

01 의사결정나무란?

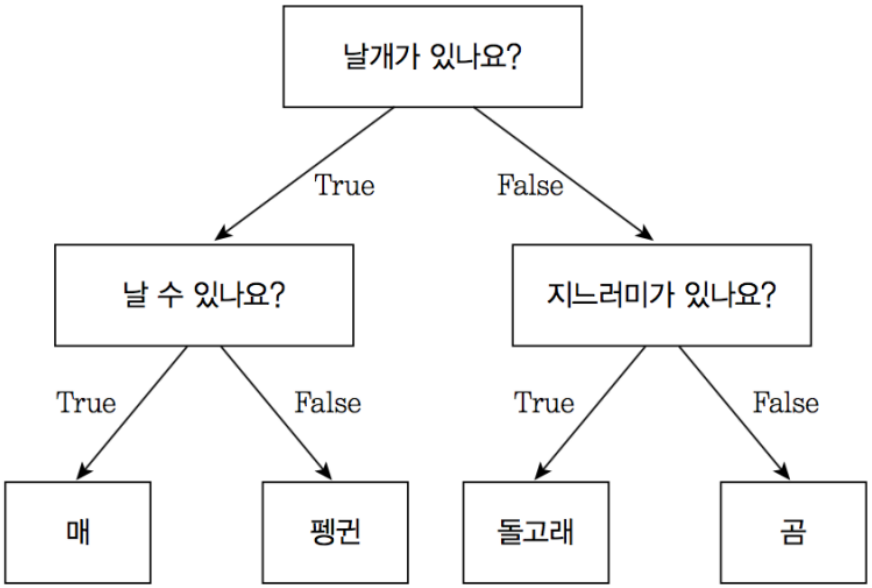
의사결정나무란 무엇인가?



출처 : https://tenserflow.blog 링크

01 의사결정나무란?

의사결정나무란 무엇인가?



출처 : https://tenserflow.blog 링크

-결정 트리는 결정에 다다르기 위해 **'예/아니오'**의 질문을 이어 나가며 학습

-목표는 가능한 한 적은 **'예/아니오'** 질문으로 문제를 해결하는 것.

-다양한 의사결정 경로(decision path)와 결과(outcome)를 나무구조로 나타냄.

-**숫자형** 결과를 반환하는 것을 **회귀나무**(regression tree), **범주형** 결과를 반환하는 것을 **분류나무**(classification tree)

-직접 모델 만들지 않고도 지도학습으로 데이터로부터 학습 가능

01 의사결정나무란?

의사결정나무란 무엇인가?

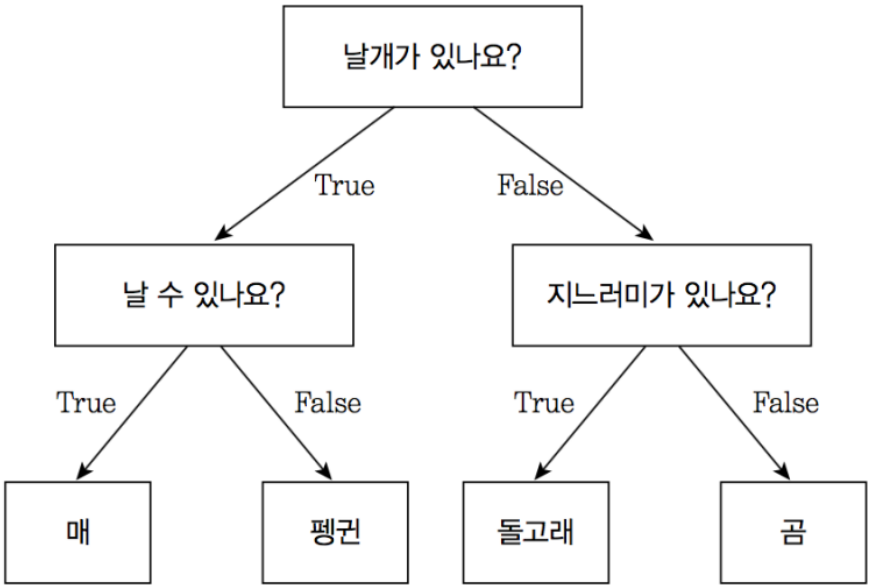
우리는 [매, 펭귄, 돌고래, 곰]
이 네 가지의 동물을 구분하는 의사결정트리를 만들어야 한다!

목표는 가능한 한 적은 **'예/아니오'** 질문으로 문제를 해결하는 것.

⇒ **가장 먼저** 어떤 질문을 던지는 것이 좋을까요?
⇒ 질문의 순서가 매우 중요합니다!

01 의사결정나무란?

의사결정나무란 무엇인가?



출처 : https://tenserflow.blog 링크

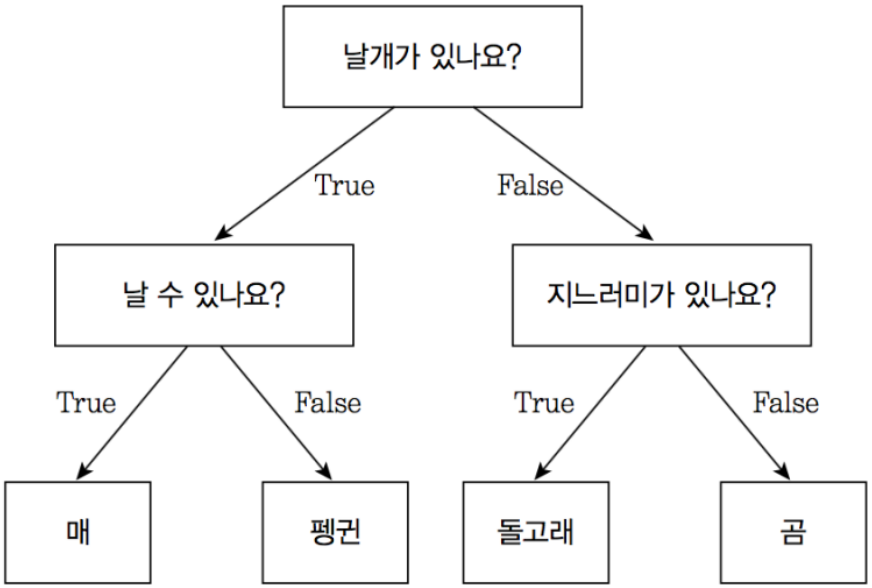
Q. 가장 먼저 어떤 질문을 던지는 것이 좋을까요 ?

A1. 날개가 있는가?
=> yes - 매, 펭귄 / no - 돌고래, 곰

A2. 헤엄 칠 수 있는가?
=> yes - 펭귄, 돌고래, 곰 / no - 매

01 의사결정나무란?

의사결정나무란 무엇인가?



출처 : https://tenserflow.blog 링크

Q. 가장 먼저 어떤 질문을 던지는 것이 좋을까요 ?

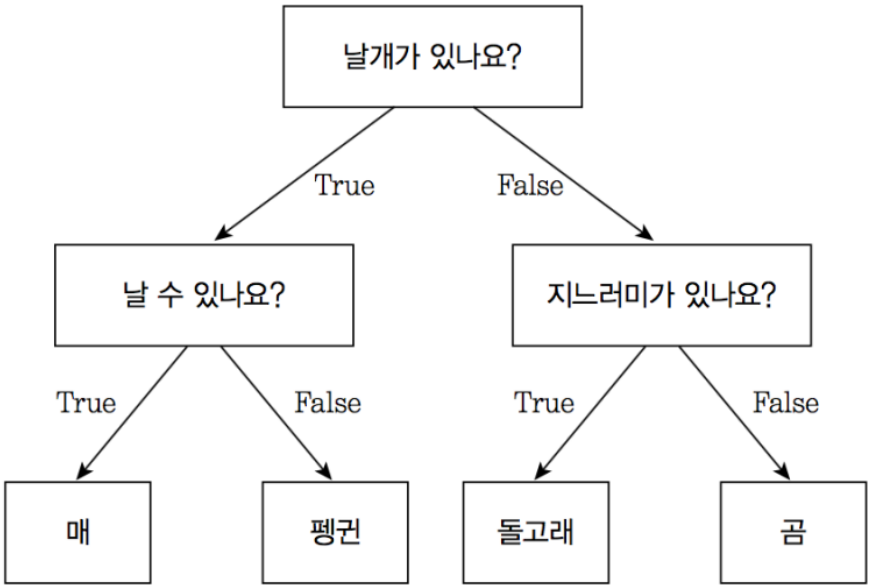
A1. 날개가 있는가?
=> yes - 매, 펭귄 / no - 돌고래, 곰

A2. 헤엄 칠 수 있는가?
=> yes - 펭귄, 돌고래, 곰 / no - 매

→둘 중 어느 질문이 엔트로피를 낮추게 하는 질문인가가 중요!
(전체 엔트로피 낮추게 하는, information gain이 많은 질문)

01 의사결정나무란?

의사결정나무란 무엇인가?



출처 : https://tenserflow.blog 링크

용어설명

- **노드** : 질문, 정답을 담은 네모 상자 (마지막 노드는 leaf라고 함)
- **엣지** : 질문의 답과 다음 질문을 연결하는 선, 가지
- **attirbute** : 분류 속성, 기준

-세 개의 속성(attributes_날개유무, 지느러미 유무, 비행 가능 유무)를 사용하여 네 개의 클래스(매, 펭귄, 돌고래, 곰)을 구분하는 모델을 만듦.

01 의사결정나무란?

의사결정나무의 장단점

-장점 :

(1) 예측 프로세스가 명확하고 쉽게 시각화할 수 있어 결과 해석, 이해 용이
(2) 각 특성이 개별적으로 처리되어 데이터를 분할하는 데 데이터 스케일의 영향을 받지 않으므로 정규화나 표준화 같은 전처리 과정이 필요 없음.
(3) 숫자형 데이터와 범주형 데이터 혼용 가능
(4) 특정 변수의 값이 누락되어도 사용할 수 있음
-> 각각의 데이터 포인트가 5개값을 갖고 있을 때 어떤건 4개 갖고 있고, 어떤건 3개 갖고 있더라도 이를 분류할 때 에러메시지 X, 완전한 조건을 갖추지 않은 데이터도 정상적으로 처리할 수 있음.

01 의사결정나무란?

의사결정나무의 장단점

-단점 :

(1) **휴리스틱(heuristics) 기법**을 기반으로 하기 때문에 최적의 의사결정나무 보장하기 어려움
(2) 너무 복잡한 결정트리를 만들 가능성
(3) 트레이닝 데이터를 모두 기억하기 때문에 새로운 데이터를 예측할 능력이 없음
-> overfitting(과적합)의 위험

=> 사전 가지치기
=> 그럼에도 불구하고 과대적합되는 경향.
=> 오버피팅을 방지할 수 있는 방법 -> 랜덤 포레스트

Heuristics(발견법):
의사결정을 하려면 다양한 변수를 고려해야 하지만 현실적으로 정보의 부족과 시간제약으로 완벽한 의사결정을 할 수 없다. 제한된 정보와 시간제약을 고려해 실무상 실현 가능한 해답이 필요하다. 발견법은 이런 경우를 위해 가장 이상적인 해답을 구하는 것이 아니라 현실적으로 만족할 만한 수준의 해답을 찾는 것이다.

01 의사결정나무란?

ID3 알고리즘

- 결정나무의 알고리즘 중 하나 (그외 C4.5, CART)
1. 루트노드 생성
2. 현재 트리에서 모든 단말 노드에 대해서 아래를 반복한다.

A. 해당 노드의 샘플들이 같은 클래스이면, 해당 노드는 단말노드(Leaf)가 되고, 해당 클래스로 레이블을 부여
B. 더 이상 사용할 수 있는 속성이 없으면 수행 종료
C. Information Gain이 높은 속성을 선택해서 노드를 분할

01 의사결정나무란?

Entropy

1) 주어진 데이터 집합의 혼잡도, 불확실성 (그 외 지니계수 등이 있음)

2) 얼마만큼의 정보를 담고 있는가

3) 즉, 주어진 데이터 집합에서 서로 다른 레코드들이 섞여 있으면 엔트로피가 높고, 같은 종류의 레코드들이 섞여 있으면 엔트로피가 낮다.

4) 0에서 1사이의 값을 가지며, 가장 혼합도가 높은 상태가 1

지니 불순도는 집합에 이질적인 것이 얼마나 섞였는지를 측정하는 지표이며 CART 알고리즘에서 사용한다.
어떤 집합에서 한 항목을 뽑아 무작위로 라벨을 추정할 때 틀릴 확률을 말한다.
집합에 있는 항목이 모두 같다면 지니 불순도는 최솟값(0)을 갖게 되며 이 집합은 완전히 순수하다고 할 수 있다.

02 Entropy / Information Gain

Entropy

@ 의사결정나무에서 엔트로피란?

- 데이터를 분류할 질문, attribute을 결정하는데 중요한 판단 기준이다.
즉, 잔존하는 데이터의 엔트로피가 높은 상태에서 낮은 상태가 되도록 하는 특정 조건을 찾아 데이터를 쪼갠다.

- 데이터를 분류할 질문은 예측하려는 대상에 대해
가장 많은 정보를 담고 있는 것이 좋다.

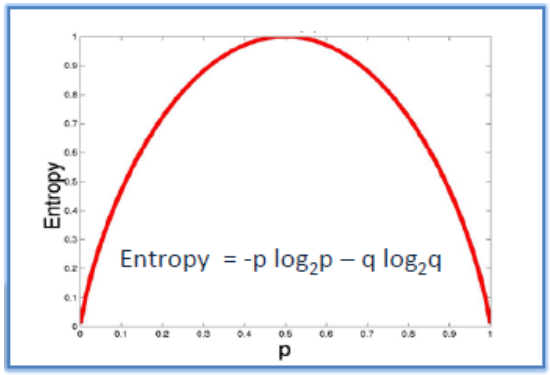
02 Entropy / Information Gain

Entropy

@ 엔트로피 계산

-m 개의 레코드가 속하는 A영역에 대한 엔트로피 계산식

$$Entropy(A) = -\sum_{k=1}^{m} p_k \log_2 (p_k)$$



Entropy = -0.5 $\log_2$0.5 – 0.5 $\log_2$0.5 = 1

https://medium.com/greyatom/decision-tree-intuition-a38669005cb7

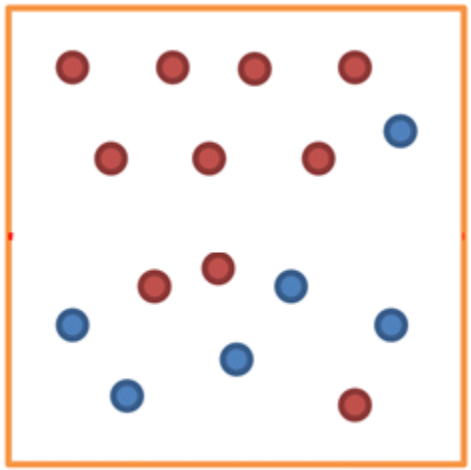
(Pk 는 A 영역에 속하는 레코드 가운데 k 범주에 속하는 레코드의 비율)

02 Entropy / Information Gain

Entropy

@ 엔트로피 계산 예시 I

<오렌지 박스로 둘러싸인 A영역 엔트로피 구하기>



- m = 16
- 빨간색(범주=1)은 10개, 파란색(범주=2)는 6개

-> A 영역의 엔트로피 ?

$$Entropy(A) = -\sum_{k=1}^{m} p_k \log_2 (p_k)$$

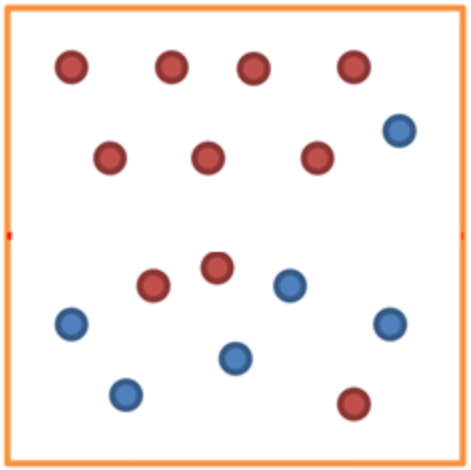
출처 : http://ratsgo.github .io 링크

02 Entropy / Information Gain

Entropy

@ 엔트로피 계산 예시 I

<오렌지 박스로 둘러싸인 A영역 엔트로피 구하기>



- m = 16
- 빨간색(범주=1)은 10개, 파란색(범주=2)는 6개

-> A 영역의 엔트로피 ?

$$Entropy(A) = -\sum_{k=1}^{m} p_k \log_2 (p_k)$$

$$Entropy(A) = -\frac{10}{16}\log_2 (\frac{10}{16}) - \frac{6}{16}\log_2 (\frac{6}{16}) \approx 0.95$$

출처 : http://ratsgo.github .io 링크

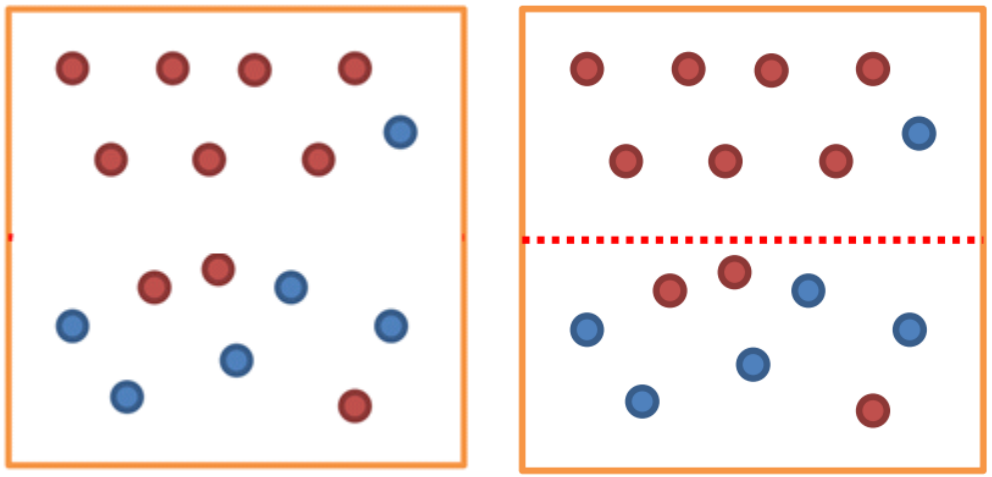
02 Entropy / Information Gain

Entropy

@ 엔트로피 계산 예시 2

<분할된 A영역 엔트로피 구하기>

출처 : http://ratsgo.github .io 링크



-> 그런데!!
의사결정나무의 각 단계에서는 데이터를 여러개의 파티션으로 분할하게 됨

-> 다리가 다섯 개 이상인가?
라는 질문에 yes인 동물(ex 거미), no인 동물(ex. 두더지)에 대한

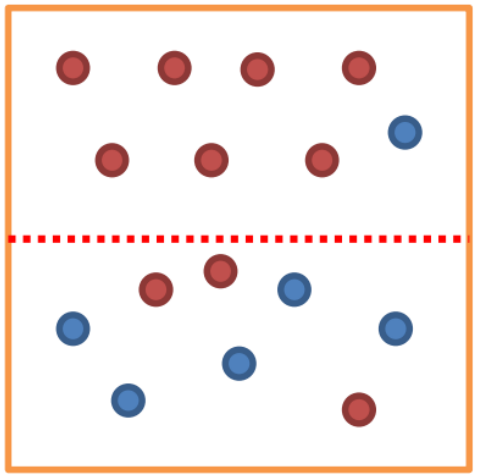
=> 분할된 각 영역의 엔트로피는 어떻게 구할까요?

02 Entropy / Information Gain

Entropy

@ 엔트로피 계산 예시 2

<분할된 A영역 엔트로피 구하기> 출처 : http://ratsgo.github .io 링크



-> 빨간색 점선을 기준으로 두 개 부분집합(R1, R2)로 분할
-> 각 클래스 레이블의 확률은 별도로 계산해주어야 한다.
(가중합)

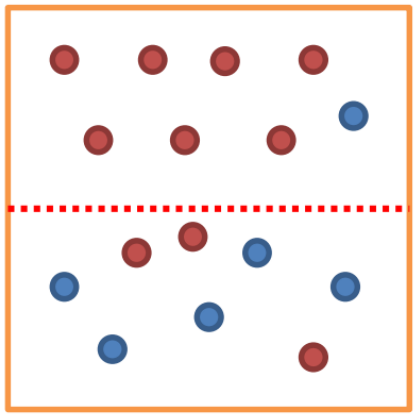
=> 분할된 각 영역 R1, R2의 엔트로피는?

02 Entropy / Information Gain

Entropy

@ 엔트로피 계산 예시 2

<분할된 A영역 엔트로피 구하기> 출처 : http://ratsgo.github .io 링크



-> 빨간색 점선을 기준으로 두 개 부분집합(R1, R2)로 분할
-> 각 클래스 레이블의 확률은 별도로 계산해주어야 한다.
(가중합)

$$Entropy(A) = \sum_{i=1}^{d} R_i \left(-\sum_{k=1}^{m} p_k \log_2 (p_k) \right)$$

$$Entropy(A) = 0.5 \times \left(-\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \right) + 0.5 \times \left(-\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right) \right) \approx 0.75$$

02 Entropy / Information Gain

Entropy

@ 파티션의 엔트로피

(I) 데이터를 분할하는 좋은 기준(attribute) 이란?

-> 모든 파티션들의 엔트로피를 낮출 수 있는 질문
-> 하나의 파티션의 엔트로피가 아주 낮아졌으나 다른 파티션의 엔트로피가 매우 높으면 그것은 좋은 분할 기준X

(ex. '호주 5센트 동전' 질문이 처음부터 던져졌다면
S1 = {바늘두더지}, S2 = {바늘두더지가 아닌 모든 동물} 로
나뉘기 때문에 좋은 질문이 아니다.

02 Entropy / Information Gain

Entropy

@ 파티션의 엔트로피

(2) Overfitting의 위험 고려 필요성

-> 다양한 값을 가질 수 있는 변수를 사용하여 파티션을 나누는 경우

(ex. 은행 고객이 대출을 제대로 갚을 수 있을 것인지에 대한 의사결정 나무 -> 변수를 고객의 주민등록번호로 선택 -> 파티션 당 사람이 한 명씩만 속함 -> 엔트로피 매우 낮지만 new data 처리 불가능)

-> 다양한 값을 가질 수 있는 변수를 최대한 피하기
or 변수에 속한 값을 최대한 적은 수의 bucket으로 clustering

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain

-어떤 속성을 선택함으로 인해서 데이터를 더 잘 구분하게 되는 것.
ex) 수능 등급을 구분하는데 있어 수학점수가 체육 점수보다 변별력이 더 높다.
이 경우 수학점수가 체육점수보다 정보이득이 높은 속성이다.

- base entropy – new entropy
- 원래의 엔트로피와 세부 클래스로 분할된 후의 엔트로피의 차이이다.

- information gain이 클수록 **분류하기 좋은 속성**임을 의미
- ID3 알고리즘은 information gain을 사용하여 노드를 결정함.
(information gain이 큰 질문(속성)으로 data를 분류함)

➔ Information gain 의사결정나무 학습과정의 메커니즘

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain

(출처 : http://jihoonlee.tistory.com/16)

$$Gain(T,\ X) = Entropy(T) - Entropy(T,\ X)$$

$$Entropy(T,\ X) = \sum P(i)E(i)$$

- base entropy – new entropy
-원래의 엔트로피와 세부 클래스로 분할된 후의 엔트로피의 차이이다.
-T가 상위 노드, X는 'T에 잔존한 데이터를 분류할 속성'
-> Gain(T, X) = T의 데이터들을 X라는 속성으로 나누었을 때 얻을 수 있는 정보량
-X 라는 속성으로 나누었을 때 엔트로피
-> 나누어진 m개의 하위노드에서의 각각의 엔트로피에다가
해당 노드의 레코드 개수를 곱한 가중합을 모두 더한 것. (하위 노드의 엔트로피들의 합)
- (원래 노드의 엔트로피) – (그 노드에서 나온 m개의 하위 노드의 전체적인 엔트로피)

02 Entropy / Information Gain

17.2 엔트로피

 # 1. entropy(class_probabilities)

```
# 1

def entropy(class_probabilities):
    """클래스, 즉 레이블(True / False)에 속할 확률을 입력하면 엔트로피를 계산하는 공식"""

    return sum(-p * math.log(p, 2) for p in class_probabilities if p)

    # if 조건문 - '숫자'자료형: 숫자가 0이 아니면 True, 0이면 False
    # 즉 여기서는 확률이 0인 경우는 제외한 것이다
```

02 Entropy / Information Gain

17.2 엔트로피

 # 2. class_probabilities(labels)

```
# 2

def class_probabilities(labels):
    """어떤 클래스, 즉 레이블(True / False)에 속할 확률을 구함 """

    total_count = len(labels) # labels 라는 전체 데이터 리스트에서, 데이터의 총 개수 구함

    return [count / total_count
            for count in Counter(labels).values()]

    # 데이터들이 각 레이블에 속할 확률을 리스트로 반환
    # Counter(parameter): 주어진 파라미터의 element는 key로,
                          element들의 count횟수는 value로 포함하는 딕셔너리를 만든다
    # values() : 딕셔너리의 method로, 딕셔너리 내 value들의 리스트를 반환
    # Counter(labels).values() 의 결과는 {'label1' : 'label1 count 횟수', 'label2' : 'label2 count 횟수'}
    # 결과값이 이진인 경우만 고려하자. 그러므로 label의 종류에는 True와 False 두 가지만 존재한다.
```

02 Entropy / Information Gain

17.2 엔트로피

 # 3. data_entropy(labeled_data)

```
# 3

def data_entropy(labeled_data):
    """데이터 전체에 대한 엔트로피 계산"""

    labels = [label for _, label in labeled_data]
    # 전체 데이터를 의미하는 labeled_data는 (input, label)라는 튜플들로 구성됨.
      그 중 label만 갖고 와서 labels 리스트를 만듦
    probabilities = class_probabilities(labels)

    return entropy(probabilities)
    # 위에서 만든 labels 리스트, 즉 전체 데이터에 대해 엔트로피 구함.
```

02 Entropy / Information Gain

17.3 파티션 후의 엔트로피

 # 4. partition_entropy(subsets)

```
# 4

def partition_entropy(subsets):
    """subsets는 레이블(True/False)이 있는 데이터를 포함하는 list이다. 여기서는 subsets = 전체 집합.
    이런 전체집합에 대한 파티션 엔트로피,
    즉 전체집합을 몇 가지 부분집합으로 나누었을 때의 엔트로피를 구하라 " " "

    total_count = sum(len(subset) for subset in subsets)
 # subsets = [ [subset 1], [subset 2], … , [마지막 subset] ] 이런 모양이다
 # 각 subset들에 포함된 데이터들의 개수를 모두 더해야 총 데이터의 개수인 total_count를 구할 수 있다

    return sum( data_entropy(subset) * len(subset) / total_count
            for subset in subsets )
    # 파티션 후, 부분집합에 해당하는 각 subset의 엔트로피에다 해당 subset에 속할 확률을 곱해주어
    # 가중평균시킨 엔트로피의 값을 구함
    # 이 값은 기존의, 분할 이전 데이터 전체에 대해 구한 엔트로피 값보다 낮을 것이다

                -> "Information Gain"이 발생!
```

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

Information gain 알고리즘(ID3)을 통하여
이제 직접 의사결정나무를 만들어 봅시다!

자료가 범주형/연속형일 때
이 두 가지 경우로 살펴봅시다!

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	FALSE	No
sunny	hot	high	TRUE	No
overcast	hot	high	FALSE	Yes
rain	mild	high	FALSE	Yes
rain	cool	normal	FALSE	Yes
rain	cool	normal	TRUE	No
overcast	cool	normal	TRUE	Yes
sunny	mild	high	FALSE	No
sunny	cool	normal	FALSE	Yes
rain	mild	normal	FALSE	Yes
sunny	mild	normal	TRUE	Yes
overcast	mild	high	TRUE	Yes
overcast	hot	normal	FALSE	Yes
rain	mild	high	TRUE	No

Q. Outlook, Temp, Humidity, Windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지를 분류해보자!

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	FALSE	No
sunny	hot	high	TRUE	No
overcast	hot	high	FALSE	Yes
rain	mild	high	FALSE	Yes
rain	cool	normal	FALSE	Yes
rain	cool	normal	TRUE	No
overcast	cool	normal	TRUE	Yes
sunny	mild	high	FALSE	No
sunny	cool	normal	FALSE	Yes
rain	mild	normal	FALSE	Yes
sunny	mild	normal	TRUE	Yes
overcast	mild	high	TRUE	Yes
overcast	hot	normal	FALSE	Yes
rain	mild	high	TRUE	No

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

⇒ 먼저 무엇을 해야 할까요?

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	FALSE	No
sunny	hot	high	TRUE	No
overcast	hot	high	FALSE	Yes
rain	mild	high	FALSE	Yes
rain	cool	normal	FALSE	Yes
rain	cool	normal	TRUE	No
overcast	cool	normal	TRUE	Yes
sunny	mild	high	FALSE	No
sunny	cool	normal	FALSE	Yes
rain	mild	normal	FALSE	Yes
sunny	mild	normal	TRUE	Yes
overcast	mild	high	TRUE	Yes
overcast	hot	normal	FALSE	Yes
rain	mild	high	TRUE	No

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

: information gain이 큰 질문(속성)으로 data를 분류해야 함

-> 어떤 속성으로 데이터를 분류했을 때 가장 information gain이 큰지 하나씩 살펴봐야 함

-> 순서 또한 중요! 어떤 속성 먼저 분류 기준으로 선택하는지에 따라 결정나무 완전히 달라짐

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	FALSE	No
sunny	hot	high	TRUE	No
overcast	hot	high	FALSE	Yes
rain	mild	high	FALSE	Yes
rain	cool	normal	FALSE	Yes
rain	cool	normal	TRUE	No
overcast	cool	normal	TRUE	Yes
sunny	mild	high	FALSE	No
sunny	cool	normal	FALSE	Yes
rain	mild	normal	FALSE	Yes
sunny	mild	normal	TRUE	Yes
overcast	mild	high	TRUE	Yes
overcast	hot	normal	FALSE	Yes
rain	mild	high	TRUE	No

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

각각의 속성을 기준으로 나누었을 때 Information gain을 비교!

➔ 가장 큰 gain을 얻게 하는 속성을 해당 클래스에서의 분류 기준으로 선택!

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ 한번 더! Information Gain (출처 : http://jihoonlee.tistory.com/16)

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

$$Entropy(T, X) = \sum P(i)E(i)$$

- base entropy – new entropy
-원래의 엔트로피와 세부 클래스로 분할된 후의 엔트로피의 차이이다.
-T가 상위 노드, X는 'T에 잔존한 데이터를 분류할 속성'
-> Gain(T, X) = T의 데이터들을 X라는 속성으로 나누었을 때 얻을 수 있는 정보량
-X 라는 속성으로 나누었을 때 엔트로피
-> 나누어진 m개의 하위노드에서의 각각의 엔트로피에다가
해당 노드의 레코드 개수를 곱한 가중합을 모두 더한 것. (하위 노드의 엔트로피들의 합)
- (원래 노드의 엔트로피) – (그 노드에서 나온 m개의 하위 노드의 전체적인 엔트로피)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	FALSE	No
sunny	hot	high	TRUE	No
overcast	hot	high	FALSE	Yes
rain	mild	high	FALSE	Yes
rain	cool	normal	FALSE	Yes
rain	cool	normal	TRUE	No
overcast	cool	normal	TRUE	Yes
sunny	mild	high	FALSE	No
sunny	cool	normal	FALSE	Yes
rain	mild	normal	FALSE	Yes
sunny	mild	normal	TRUE	Yes
overcast	mild	high	TRUE	Yes
overcast	hot	normal	FALSE	Yes
rain	mild	high	TRUE	No

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

(I) Outlook 일 때 Entropy

- Informaition gain
= 원래 entropy – 분류된 하위 노드들의 전체적인 entropy

- Gain(T, X)
= entropy(T) - entropy(T, X)
- Gain(Play, outlook)
= entropy(Play) - entropy(Play, outlook)

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

(I) Outlook 일 때 Entropy

- Gain(T, X) = E(T) - E(T, X)
⇒Gain(Play, outlook) = E(Play) - E(Play, outlook)

-E(Play)란?
=> 초기 분기 시, 아직 가지를 한 번도 뻗어나가지 않았음(데이터들 모두 분류되지 않고 나열 되어 있는 상태)
=> 이 경우, 궁극적인 분류하고자 한 기준인 'play 유무' 를 노드(속성)으로 둔다.

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

(I) Outlook 일 때 Entropy

- Gain(T, X) = E(T) - E(T, X)
⇒Gain(Play, outlook) = E(Play) - E(Play, outlook)

-. 전체 데이터 중 Play 유무에 대해 응답 [yes – 9 / no – 5]

=> 먼저 E(Play)를 구해보세요!

$$Entropy(A) = -\sum_{k=1}^{m} p_k \log_2 (p_k)$$

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

(I) Outlook 일 때 Entropy

- Gain(T, X) = E(T) - E(T, X)
⇒Gain(Play, outlook) = E(Play) - E(Play, outlook)

-. 전체 데이터 중 Play 유무에 대해 응답 [yes – 9 / no – 5]

=> 먼저 E(Play)를 구해보세요!

$$E(Play) = \left(-\frac{5}{14}\log_2\frac{5}{14}\right) + \left(-\frac{9}{14}\log_2\frac{9}{14}\right)$$

$$E(Play) = 0.94$$

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

(I) Outlook 일 때 Entropy

- Gain(T, X) = E(T) - E(T, X)
⇒Gain(Play, outlook) = E(Play) - E(Play, outlook)

-. => E(Play, Outlook)을 구해보세요!

E(Play, Outlook)

		Play		
		Yes	No	Tot
Outlook	sunny	3	2	5
	overcast	4	0	4
	rain	3	2	5

파티션 엔트로피 구하기!

$$Entropy(A) = \sum_{i=1}^{d} R_i \left(- \sum_{k=1}^{m} p_k \log_2 (p_k) \right)$$

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

(I) Temperature 일 때 Entropy

=> E(Play, Teperature)을 직접 구해보세요!

E(Play,Temperature)

		Play		
		Yes	No	Total
Temp..	hot	2	2	4
	mild	4	2	6
	cool	3	1	4

파티션 엔트로피 구하기!

$$Entropy(A) = \sum_{i=1}^{d} R_i \left(- \sum_{k=1}^{m} p_k \log_2 (p_k) \right)$$

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

(2) Humidity 일 때 Entropy

=> E(Play, Humidity)을 직접 구해보세요!

E(Play, Humidity)

		Play		
		Yes	No	Total
Humidity	high	3	4	7
	normal	6	1	7

파티션 엔트로피 구하기!

$$Entropy(A) = \sum_{i=1}^{d} R_i \left(- \sum_{k=1}^{m} p_k \log_2 (p_k) \right)$$

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

(3) Windy 일 때 Entropy

=> E(Play, Windy)을 직접 구해보세요!

E(Play, Windy)

		Play		
		Yes	No	Total
Windy	TRUE	3	3	6
	FALSE	6	2	8

파티션 엔트로피 구하기!

$$Entropy(A) = \sum_{i=1}^{d} R_i \left(- \sum_{k=1}^{m} p_k \log_2 (p_k) \right)$$

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

➔ 어느 것이 가장 information gain이 높은 속성인가?
(= 어느 질문을 가장 먼저 하는 것이 좋은가?)

$$E(Play) - E(Play, Outlook) = 0.25$$
$$E(Play) - E(Play, Temp) = 0.02$$
$$E(Play) - E(Play, Humidity) = 0.1514$$
$$E(Play) - E(Play, Windy) = 0.047$$

=> Outlook이 가장 information gain 높은 분류 속성
=> Outlook으로 나누었을 때 new entropy 가장 낮아짐

(출처 : http://jihoonlee.tistory.com/16)

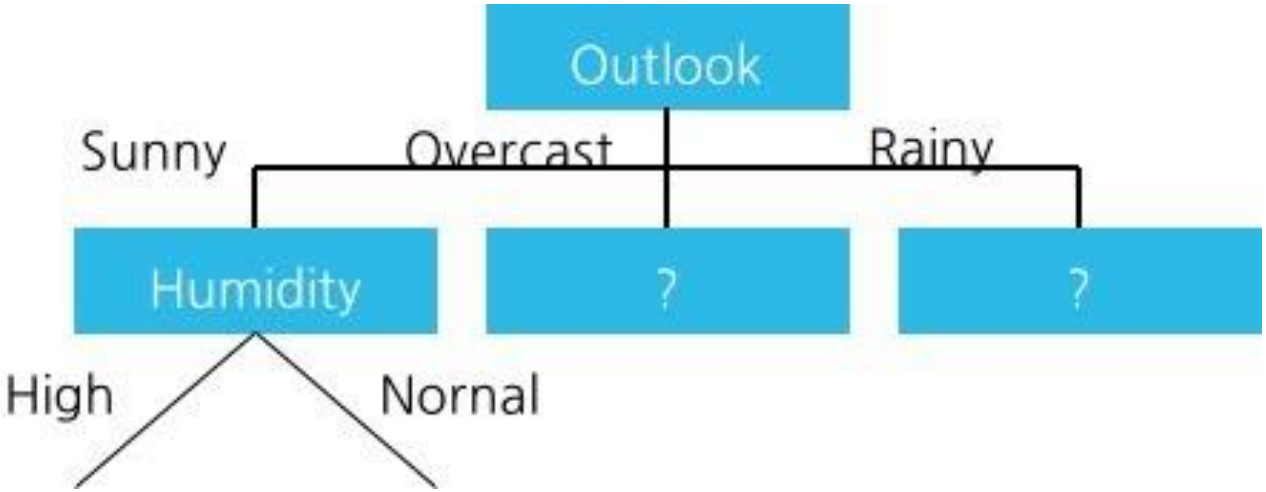
02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

➔ Outlook 으로 분류 했을 때 트리 구조



(출처 : http://jihoonlee.tistory.com/16)

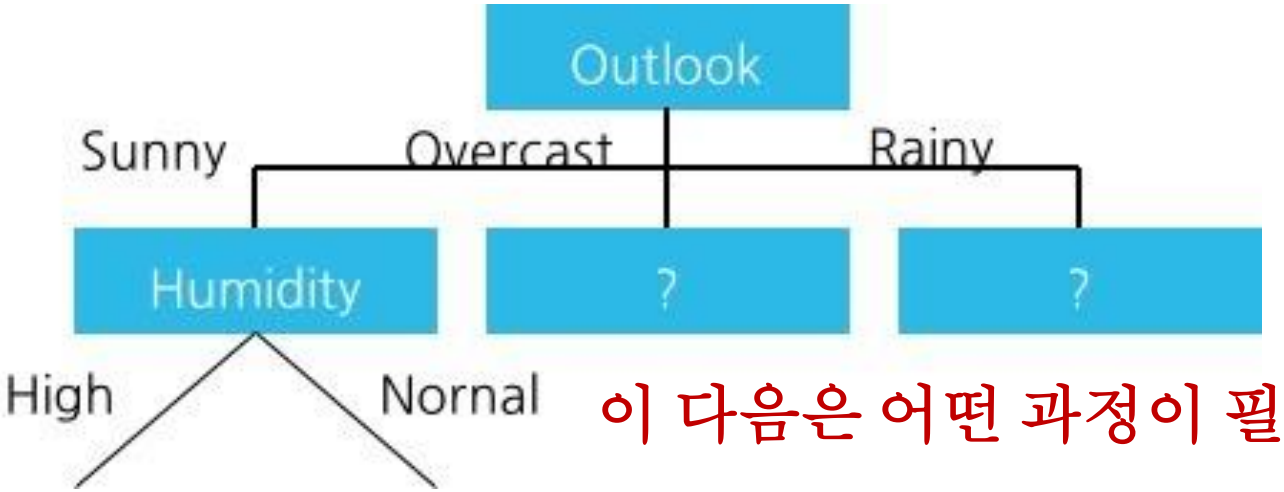
02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

➔ Outlook 으로 분류 했을 때 트리 구조



이 다음은 어떤 과정이 필요할까?

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

➔이 다음은 어떤 과정이 필요할까?

=> 각각의 하위 노드에서 최적의 분류기준(information gain이 높은 분류기준)을 찾아 분류함

⇒언제까지?
: 잎 노드가 나올 때 까지 (각 데이터의 클래스 레이블이 모두 같아질 때 까지, 더 이상 데이터들을 세부적으로 나눌 수 없을 때 까지)

=> 더 뻗어나갈 가지 없을 때 완성되는 트리

(출처 : http://jihoonlee.tistory.com/16)

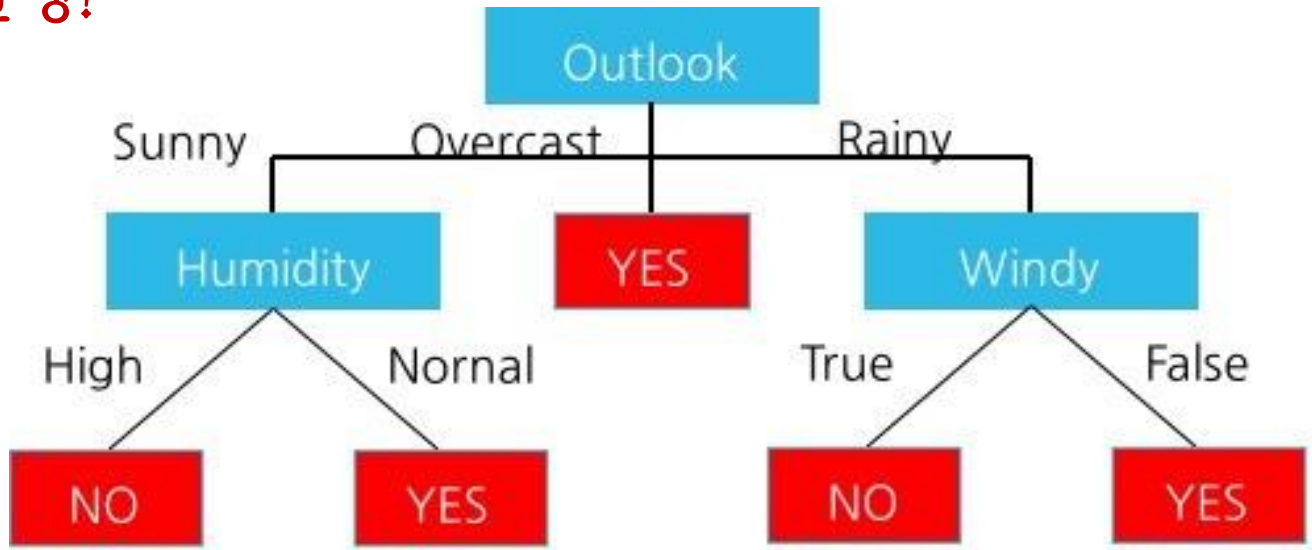
02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 범주형 자료

Q. outlook, temperature, humidity, windy에 관한 자료를 통해 어떤 날씨에 테니스를 치는지에 대해 분류해보자!

완성!



(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ 2. 연속형 자료

Income	Lot size	Ownership	Income	Lot size	Ownership
60.0	18.4	Owner	75.0	19.6	Non-owner
85.5	16.8	Owner	52.8	20.8	Non-owner
64.8	21.6	Owner	64.8	17.2	Non-owner
61.5	20.8	Owner	43.2	20.4	Non-owner
87.0	23.6	Owner	84.0	17.6	Non-owner
110.1	19.2	Owner	49.2	17.6	Non-owner
108.0	17.6	Owner	59.4	16.0	Non-owner
82.8	22.4	Owner	66.0	18.4	Non-owner
69.0	20.0	Owner	47.4	16.4	Non-owner
93.0	20.8	Owner	33.0	18.8	Non-owner
51.0	22.0	Owner	51.0	14.0	Non-owner
81.0	20.0	Owner	63.0	14.8	Non-owner

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디깎기 기계 구입 여부 (Ownership)를 조사한 데이터가 있다.

이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

(출처 : https://ratsgo.github.io/machine%20learning/2017/03/26/tree/)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ 2. 연속형 자료

Income	Lot size	Ownership	Income	Lot size	Ownership
60.0	18.4	Owner	75.0	19.6	Non-owner
85.5	16.8	Owner	52.8	20.8	Non-owner
64.8	21.6	Owner	64.8	17.2	Non-owner
61.5	20.8	Owner	43.2	20.4	Non-owner
87.0	23.6	Owner	84.0	17.6	Non-owner
110.1	19.2	Owner	49.2	17.6	Non-owner
108.0	17.6	Owner	59.4	16.0	Non-owner
82.8	22.4	Owner	66.0	18.4	Non-owner
69.0	20.0	Owner	47.4	16.4	Non-owner
93.0	20.8	Owner	33.0	18.8	Non-owner
51.0	22.0	Owner	51.0	14.0	Non-owner
81.0	20.0	Owner	63.0	14.8	Non-owner

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디깎기 기계 구입 여부(Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

⇒ 먼저 무엇을 해야 할까요?

(출처 : https://ratsgo.github.io/machine%20learning/2017/03/26/tree/)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ 2. 연속형 자료

Income	Lot size	Ownership	Income	Lot size	Ownership
60.0	18.4	Owner	75.0	19.6	Non-owner
85.5	16.8	Owner	52.8	20.8	Non-owner
64.8	21.6	Owner	64.8	17.2	Non-owner
61.5	20.8	Owner	43.2	20.4	Non-owner
87.0	23.6	Owner	84.0	17.6	Non-owner
110.1	19.2	Owner	49.2	17.6	Non-owner
108.0	17.6	Owner	59.4	16.0	Non-owner
82.8	22.4	Owner	66.0	18.4	Non-owner
69.0	20.0	Owner	47.4	16.4	Non-owner
93.0	20.8	Owner	33.0	18.8	Non-owner
51.0	22.0	Owner	51.0	14.0	Non-owner
81.0	20.0	Owner	63.0	14.8	Non-owner

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디깎기 기계 구입 여부(Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

: information gain이 큰 질문(속성)으로 data를 분류해야 함

-> 어떤 속성으로 데이터를 분류했을 때 가장 information gain이 큰지 하나씩 살펴봐야 함

-> 순서 또한 중요! 어떤 속성 먼저 분류 기준으로 선택하는지에 따라 결정나무 완전히 달라짐

(출처 : https://ratsgo.github.io/machine%20learning/2017/03/26/tree/)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ 2. 연속형 자료

Income	Lot size	Ownership	Income	Lot size	Ownership
60.0	18.4	Owner	75.0	19.6	Non-owner
85.5	16.8	Owner	52.8	20.8	Non-owner
64.8	21.6	Owner	64.8	17.2	Non-owner
61.5	20.8	Owner	43.2	20.4	Non-owner
87.0	23.6	Owner	84.0	17.6	Non-owner
110.1	19.2	Owner	49.2	17.6	Non-owner
108.0	17.6	Owner	59.4	16.0	Non-owner
82.8	22.4	Owner	66.0	18.4	Non-owner
69.0	20.0	Owner	47.4	16.4	Non-owner
93.0	20.8	Owner	33.0	18.8	Non-owner
51.0	22.0	Owner	51.0	14.0	Non-owner
81.0	20.0	Owner	63.0	14.8	Non-owner

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디깎기 기계 구입 여부(Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

: information gain이 큰 질문(속성)으로 data를 분류해야 함

-> 어떤 을 때 가장 inf 살펴봐야 함

-> 순서 분류 기준으 나무 완전히

의사결정나무 메커니즘은 범주형 자료와 같음!

(출처 : https://ratsgo.github.io/machine%20learning/2017/03/26/tree/)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ 2. 연속형 자료

Income	Lot size	Ownership	Income	Lot size	Ownership
60.0	18.4	Owner	75.0	19.6	Non-owner
85.5	16.8	Owner	52.8	20.8	Non-owner
64.8	21.6	Owner	64.8	17.2	Non-owner
61.5	20.8	Owner	43.2	20.4	Non-owner
87.0	23.6	Owner	84.0	17.6	Non-owner
110.1	19.2	Owner	49.2	17.6	Non-owner
108.0	17.6	Owner	59.4	16.0	Non-owner
82.8	22.4	Owner	66.0	18.4	Non-owner
69.0	20.0	Owner	47.4	16.4	Non-owner
93.0	20.8	Owner	33.0	18.8	Non-owner
51.0	22.0	Owner	51.0	14.0	Non-owner
81.0	20.0	Owner	63.0	14.8	Non-owner

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디깎기 기계 구입 여부(Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

(I) 첫 번째 속성 Lot size!

- Informaition gain

-= 원래 entropy – 분류된 하위 노드들의 entropy

-Gain(T, X)

= entropy(T) - entropy(T, X)

-Gain(Ownership, Lot size)

= entropy(Ownership) - entropy(Ownership, Lot size)

(출처 : https://ratsgo.github.io/machine%20learning/2017/03/26/tree/)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 연속형 자료

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디깎기 기계 구입 여부(Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

(I) 첫 번째 속성 Lot size!

- Gain(T, X) = E(T) - E(T, X)
- Gain(Ownership, Lot size)
= E(Ownership) - E(Ownership, Lot size)

- E(Ownership)
⇒ 초기 분기 노드 결정 단계임.
⇒ 이때는 궁극적으로 분류하고자 하는 Ownership 유무를 노드로 둔다.
⇒ (Owner 12개, Non-owner 12개)

$$\text{분기 전 엔트로피} = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 연속형 자료

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디깎기 기계 구입 여부(Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

(I) 첫 번째 속성 Lot size!

- Gain(T, X) = E(T) - E(T, X)
- Gain(Ownership, Lot size)
= E(Ownership) - E(Ownership, Lot size)

➔ E(Ownership, Lot size)
- 가능한 모든 분기점에 대해 순차적으로 엔트로피를 구해 분기 전과 비교해 정보획득을 조사한다. (연속형 자료)
- **가장 큰 정보획득을 제공하는 분기점을 찾는다.**

(출처 : http://jihoonlee.tistory.com/16)

Income	Lot size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
47.4	16.4	Non-owner
85.5	16.8	Owner
64.8	17.2	Non-owner
108.0	17.6	Owner
84.0	17.6	Non-owner
49.2	17.6	Non-owner
60.0	18.4	Owner
66.0	18.4	Non-owner
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner
69.0	20.0	Owner
81.0	20.0	Owner
43.2	20.4	Non-owner
61.5	20.8	Owner
93.0	20.8	Owner
52.8	20.8	Non-owner
64.8	21.6	Owner
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디 깎기 기계 구입 여부 (Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

(I) 첫 번째 속성 Lot size!

➔ E(Ownership, Lot size)

- 가능한 모든 분기점에 대해 순차적으로 엔트로피를 구해 분기 전과 비교해 정보획득을 조사한다.
- **가장 큰 정보획득을 제공하는 분기점을 찾는다.**

⇒ Lot size를 기준으로 정렬한다.

⇒ 첫 번째로 첫 레코드 vs 나머지 23개 레코드 사이를 분기지점으로 두고 엔트로피를 구해본다!

⇒ 연속형 자료이기 때문에

⇒**결정 노드는 '16<', of '17=>'** 등의 형태가 될 것

(출처 : http://jihoonlee.tistory.com/16)

Income	Lot size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
47.4	16.4	Non-owner
85.5	16.8	Owner
64.8	17.2	Non-owner
108.0	17.6	Owner
84.0	17.6	Non-owner
49.2	17.6	Non-owner
60.0	18.4	Owner
66.0	18.4	Non-owner
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner
69.0	20.0	Owner

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디 깎기 기계 구입 여부 (Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

(I) 첫 번째 속성 Lot size!

➔ E(Ownership, Lot size)

⇒ Lot size를 기준으로 정렬한다.

⇒ 첫 번째로 첫 레코드 vs 나머지 23개 레코드 사이를 분기지점으로 두고 엔트로피를 구해본다!

⇒ 연속형 자료이기 때문에

⇒**결정 노드는 '16<', or '17=>' 등의 형태가 될 것**

$$\text{분기 후 엔트로피} = \frac{1}{24}(-\log_2 1) + \frac{23}{24}\left(-\frac{12}{23}\log_2\left(\frac{12}{23}\right) - \frac{11}{23}\log_2\left(\frac{11}{23}\right)\right) \approx 0.96$$

$$\text{정보획득} = 1 - 0.96 = 0.04$$

64.8	21.6	Owner
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

(출처 : http://jihoonlee.tistory.com/16)

Income	Lot size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
47.4	16.4	Non-owner
85.5	16.8	Owner
64.8	17.2	Non-owner
108.0	17.6	Owner
84.0	17.6	Non-owner
49.2	17.6	Non-owner
60.0	18.4	Owner
66.0	18.4	Non-owner
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner
69.0	20.0	Owner
64.8	21.6	Owner
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디 깎기 기계 구입 여부 (Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

(I) 첫 번째 속성 Lot size!

➔ E(Ownership, Lot size)

⇒ Lot size를 기준으로 정렬한다

⇒ 첫 번째로 첫 [redacted] 를 분기지점으로

⇒ 연속형 자료이

⇒결정 노드는 '16

$$\text{분기 후 엔트로피} = \frac{1}{24}(-\log_2 1) + \frac{23}{24}\left(-\frac{1}{2}\right.$$ 6

$$\text{정보획득} = 1 - 0.96 = 0.04$$

이후
속성이 Income일 때의
information gain과 비교
⇒더 큰 gain을 갖는
속성을 첫 분기 노드로 결정!

(출처 : http://jihoonlee.tistory.com/16)

Income	Lot size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
47.4	16.4	Non-owner
85.5	16.8	Owner
64.8	17.2	Non-owner
108.0	17.6	Owner
84.0	17.6	Non-owner
49.2	17.6	Non-owner
60.0	18.4	Owner
66.0	18.4	Non-owner
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner
69.0	20.0	Owner
81.0	20.0	Owner
43.2	20.4	Non-owner
61.5	20.8	Owner
93.0	20.8	Owner
52.8	20.8	Non-owner
64.8	21.6	Owner
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디깎기 기계 구입 여부 (Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

(2) 두 번째 속성 Income!

➔ E(Ownership, Income)

⇒ Income을 기준으로 정렬한다.

⇒ 이 또한 연속형 자료이기 때문에 결정 노드는 '80<', of '75=>' 등의 형태가 될 것이다.

⇒ 가능한 모든 분기점에 대해 엔트로피를 구해 분기 전과 비교해 정보획득을 조사한다.

⇒ 가장 큰 정보획득을 제공하는 분기점을 찾는다.

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : information gain

@ Information Gain _ I. 연속형 자료

Q. 24개 가정을 대상으로 소득(income), 주택크기(Lot size), 잔디깎기 기계 구입 여부(Ownership)를 조사한 데이터가 있다. 이를 토대로 기계 구입 여부에 대한 의사결정나무 모델을 학습시켜라!

어느 질문이 가장 좋은가?

=> Lot size를 기준으로 정렬한 자료에서 정보획득이 가장 큰 분기점과, Income을 기준으로 정렬한 자료에서 정보획득이 가장 큰 분기점을 비교하여 더 큰 정보획득인 속성을 먼저 선택함!
=> 만약 Lot size(=<18.4)의 IG = 0.04, Income(=<80) 의 IG = 0.43 이라면 Income(=<80)을 최초의 분기 노드로 채택!

=> I회 분기를 위해 계산 해야하는 수는 총 몇 번인가?
: 개체가 n개 이고, 변수가 d개일 때 경우의 수는 d(n-I)개이다.
: 분기를 하지 않는 경우를 제외하고 모든 개체와 변수를 고려하는 것이다!

(출처 : http://jihoonlee.tistory.com/16)

02 Entropy / Information Gain

의사결정나무 학습과정 : Pruning

@ 결정 트리의 복잡도 제어하기 _ 가지치기

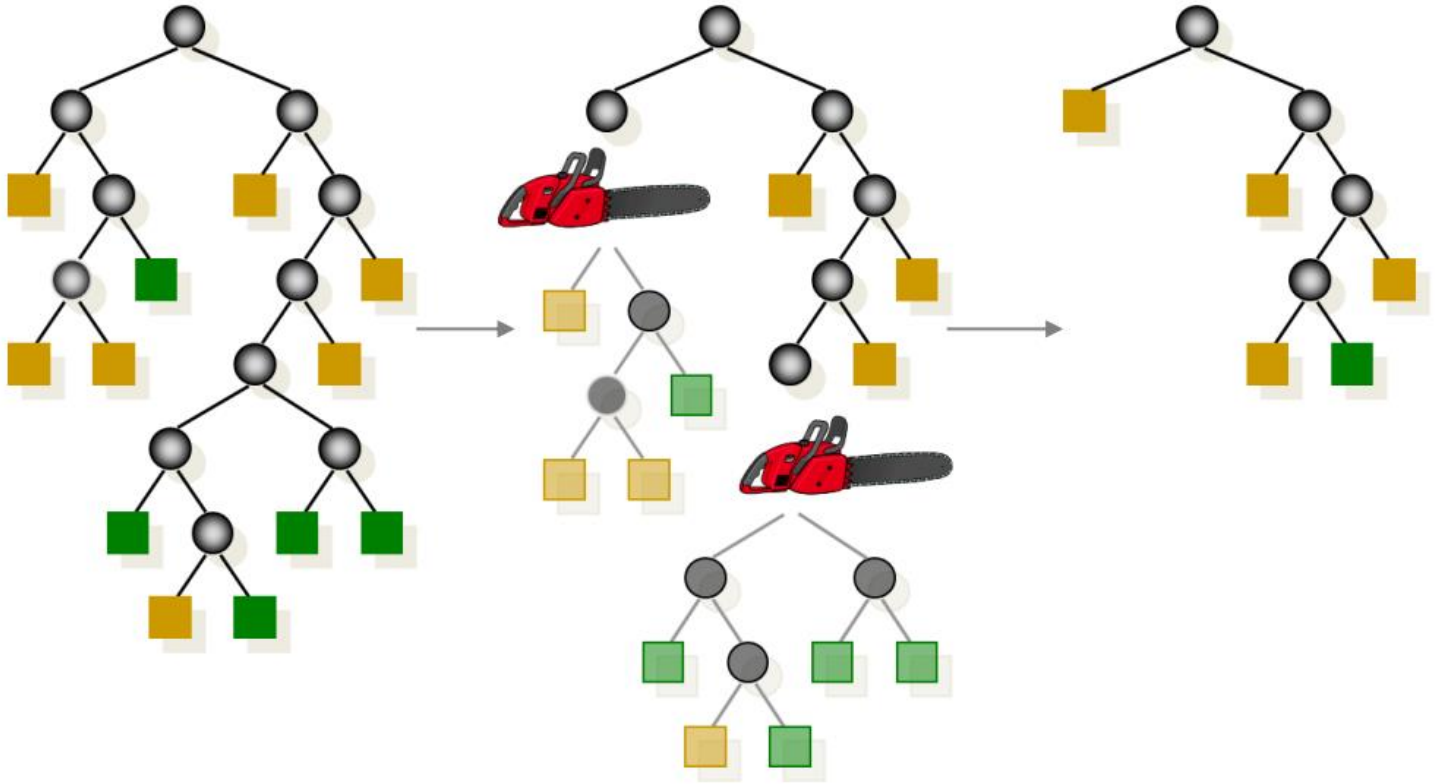
-모든 terminal node의 순도가 100%인 상태를 Full tree라고 함
-Full tree 생성 후에는 적절한 수준에서 terminal node를 결합해주어야 한다.

➔ Pruning

-> 분기가 너무 많으면 학습데이터에 **과적합(overfitting)할 염려**
-> 분기 수가 증가할 때 처음에는 새로운 데이터에 대한 오분류율이 감소,
일정 수준 이상이 되면 오분류율이 되레 증가.
-> 검증데이터에 대한 **오분류율이 증가하는 시점에서 적절히 가지치기**를 수행

02 Entropy / Information Gain

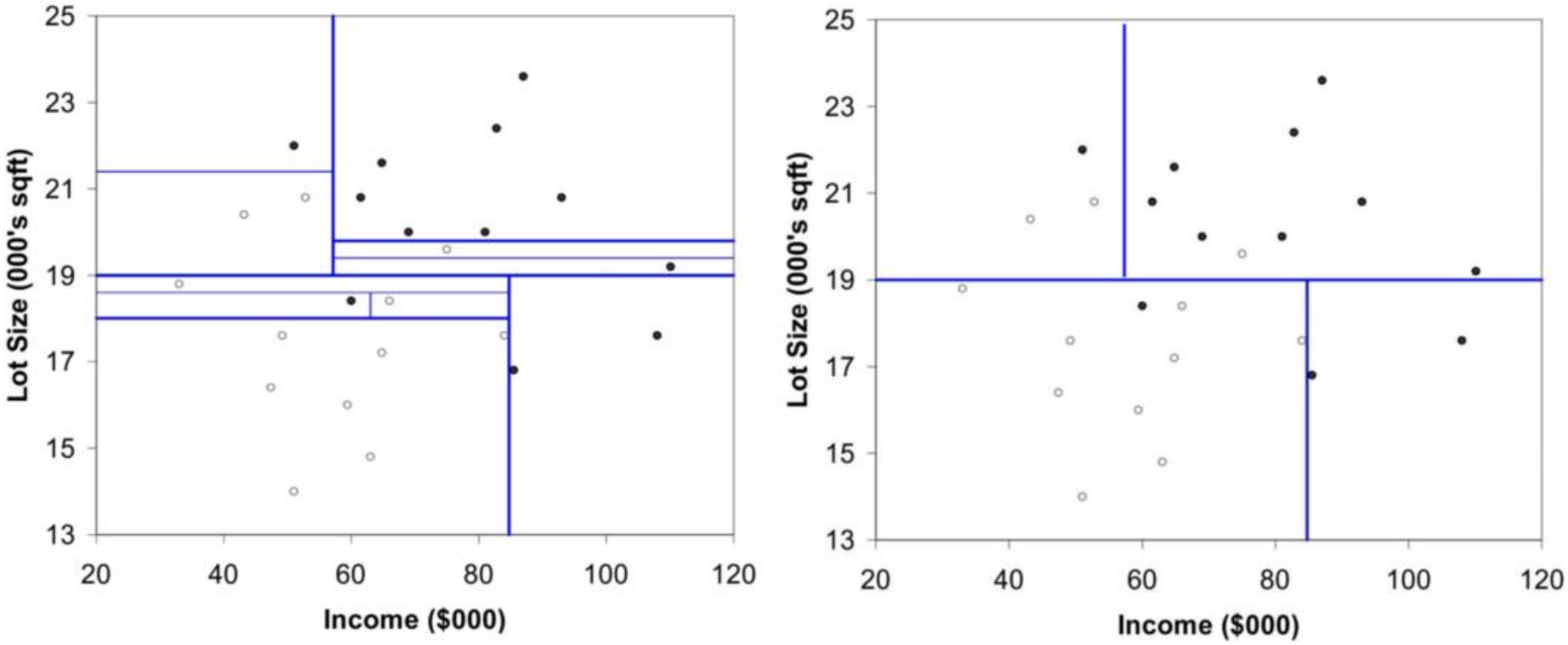
의사결정나무 학습과정 : Pruning



(출처 : http://imgur.com/MVFcKwz)

02 Entropy / Information Gain

의사결정나무 학습과정 : Pruning



(출처 : http://imgur.com/MVFcKwz)

02 Entropy / Information Gain

의사결정나무 학습과정 : Pruning

@ 가지치기의 비용함수 (cost function)

$$CC(T) = Err(T) + \alpha \times L(T)$$

-> CC(T) = 의사결정나무의 비용 복잡도
(오류가 적으면서 terminal node 수가 적은 단순한 모델일 수록 작은 값)

-> ERR(T) = 검증데이터에 대한 오분류율

-> L(T) = terminal node의 수(구조의 복잡도)

-> Alpha = ERR(T)와 L(T)를 결합하는 가중치
(사용자에 의해 부여됨, 보통 0.01~0.1의 값을 씀)
(여기서 α ≥ 0 는 나무모형의 크기와 자료에 대한 적합도를 조절하는 조율모수로
α 값이 크면(작으면) Tα 의 크기는 작아(커)진다.
α = 0 이면 가지치기는 일어나지 않고 T0 를 최종모형으로 준다.)

02 Entropy / Information Gain

의사결정나무 학습과정 : Pruning

@ 가지치기 종류

➔ 가지치기, 즉 과대적합을 막는 전략은 크게 두 가지

(1) 트리 생성을 일찍 중단하는 전략 : **사전 가지치기(prepruning)**
-> 트리의 최대 깊이나 리프의 최대 개수를 제한(ex. max_depth=4),
-> 또는 노드가 분할하기 위한 포인트의 최소 개수 지정(ex. min_samples_leaf=16)

(2) 트리를 만든 후 데이터 포인트가 적은 노드를 삭제하거나 병합하는 전략
: **사후 가지치기 (post-pruning)** 또는 그냥 가지치기(pruning).

03 Decision Tree Algorithm

"밑바닥부터 시작하는 데이터 과학"

: 17장. 의사결정나무
- Page 205 ~ 213 까지의 코드 설명

03 Decision Tree Algorithm

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	FALSE	No
sunny	hot	high	TRUE	No
overcast	hot	high	FALSE	Yes
rain	mild	high	FALSE	Yes
rain	cool	normal	FALSE	Yes
rain	cool	normal	TRUE	No
overcast	cool	normal	TRUE	Yes
sunny	mild	high	FALSE	No
sunny	cool	normal	FALSE	Yes
rain	mild	normal	FALSE	Yes
sunny	mild	normal	TRUE	Yes
overcast	mild	high	TRUE	Yes
overcast	hot	normal	FALSE	Yes
rain	mild	high	TRUE	No

03 Decision Tree Algorithm

17.2 엔트로피

 # 1. entropy(class_probabilities)

```
# 1

def entropy(class_probabilities):
    """클래스, 즉 레이블(True / False)에 속할 확률을 입력하면 엔트로피를 계산하는 공식"""

    return sum(-p * math.log(p, 2) for p in class_probabilities if p)

    # if 조건문 - '숫자'자료형: 숫자가 0이 아니면 True, 0이면 False
    # 즉 여기서는 확률이 0인 경우는 제외한 것이다
```

03 Decision Tree Algorithm

17.2 엔트로피

 # 2. class_probabilities(labels)

```
# 2

def class_probabilities(labels):
    """어떤 클래스, 즉 레이블(True / False)에 속할 확률을 구함"""

    total_count = len(labels) # labels 라는 전체 데이터 리스트에서, 데이터의 총 개수 구함

    return [count / total_count
            for count in Counter(labels).values()]

    # 데이터들이 각 레이블에 속할 확률을 리스트로 반환
    # Counter(parameter): 주어진 파라미터의 element는 key로,
                          element들의 count횟수는 value로 포함하는 딕셔너리를 만든다
    # values() : 딕셔너리의 method로, 딕셔너리 내 value들의 리스트를 반환
    # Counter(labels).values() 의 결과는 {'label1' : 'label1 count 횟수', 'label2' : 'label2 count 횟수'}
    # 결과값이 이진인 경우만 고려하자. 그러므로 label의 종류에는 True와 False 두 가지만 존재한다.
```

03 Decision Tree Algorithm

17.2 엔트로피

 # 3. data_entropy(labeled_data)

```
# 3

def data_entropy(labeled_data):
    """데이터 전체에 대한 엔트로피 계산"""

    labels = [label for _, label in labeled_data]
    # 전체 데이터를 의미하는 labeled_data는 (input, label)라는 튜플들로 구성됨.
      그 중 label만 갖고 와서 labels 리스트를 만듦
    probabilities = class_probabilities(labels)

    return entropy(probabilities)
    # 위에서 만든 labels 리스트, 즉 전체 데이터에 대해 엔트로피 구함.
```

03 Decision Tree Algorithm

17.3 파티션 후의 엔트로피

 # 4. partition_entropy(subsets)

```
# 4

def partition_entropy(subsets):
    """subsets는 레이블(True/False)이 있는 데이터를 포함하는 list이다. 여기서는 subsets = 전체 집합.
    이런 전체집합에 대한 파티션 엔트로피,
    즉 전체집합을 몇 가지 부분집합으로 나누었을 때의 엔트로피를 구하라 " " "

    total_count = sum(len(subset) for subset in subsets)
 # subsets = [ [subset 1], [subset 2], … , [마지막 subset] ] 이런 모양이다
 # 각 subset들에 포함된 데이터들의 개수를 모두 더해야 총 데이터의 개수인 total_count를 구할 수 있다

    return sum( data_entropy(subset) * len(subset) / total_count
            for subset in subsets )
    # 파티션 후, 부분집합에 해당하는 각 subset의 엔트로피에다 해당 subset에 속할 확률을 곱해주어
    # 가중평균시킨 엔트로피의 값을 구함
    # 이 값은 기존의, 분할 이전 데이터 전체에 대해 구한 엔트로피 값보다 낮을 것이다

                -〉 "Information Gain"이 발생!
```

03 Decision Tree Algorithm

17.4 의사결정나무 만들기

5. partition_by(inputs, attribute)

```
# 5

def partition_by(inputs, attribute):
    """attribute에 따라 inputs의 파티션 하기"""

    # inputs 모양 : [ ({key1 : value1, key2 : value2 }, True), ({key1 : value1, key2 : value2}, False) ]
    # input 모양 : ({key : value}, True/False)

    groups = defaultdict(list) # value값이 기본적으로 빈 list의 형식을 가지는 groups라는 dictionary

    for input in inputs: # 주어진 iterable한 데이터, inputs의 각 원소인 input에 대해
        key = input[0] [attribute]
        # 각 input의 [0]번째 원소인 딕셔너리에서 attriute라는 키에 해당하는 value를 'key'로 정의
        groups[key].append(input)
        # groups 라는 딕셔너리에서,
        # 위에서 구한 결과값 key를 key로 두고, 그에 대응되는 value 리스트에 input을 추가시킨다.

    return groups # groups 라는 딕셔너리 반환
```

03 Decision Tree Algorithm

17.4 의사결정나무 만들기

 # 6. partition_entropy_by(inputs, attribute)

```
# 6

def partition_entropy_by(inputs,attribute):
    """주어진 attribute로 파티션(부분집합 만들기) 한 후의 엔트로피를 계산"""

    partitions = partition_by(inputs, attribute)
    # 주어진 attribute로 partition_by을 사용해 파티션 함.
    # 그 결과는 앞서 구한 groups라는 딕셔너리인데, 이름을 partitions로 바꾼 후

    return partition_entropy(partitions.values())
    # partitions 딕셔너리의 각 key값들의 value, 즉 리스트로 표현된 각 부분집합들을 이용해
    # 파티션 후의 entropy를 구한다
      -> "Information Gain"이 발생!
```

03 Decision Tree Algorithm

17.5 종합하기

 # 7. classify(tree, input) - 앞 부분

```
# 7.

def classify(tree, input):
    """새로 주어진 개별 데이터 'input'을 의사결정나무(tree)를 이용하여 분류하자 """

    # input 모양 : ({key : value}, True/False)

    if tree in [True, False]:
        return tree
    # 잎 노드, 즉 의사결정나무(tree) 자체가 바로 True나 False 값을 갖고 있으면 그 값을 반환

    # 그게 아니라면 데이터의 변수(attribute)로 파티션 하자.
    # key로 변수값(attribute), value으로 서브트리(subtree_dict) 를 사용하면 된다
    attribute, subtree_dict = tree # tree는 (attribute, subtree_dict)의 tuple 형식이다

    subtree_key = input.get(attribute)
    # 위의 tree (attribute, subtree_dict)에서
    # 주어진 변수(attribute)를 key로 하는 value값을 subtree_key로 둔다.
```

03 Decision Tree Algorithm

17.5 종합하기

 # 7. classify(tree, input) - 뒷 부분

```
# 7 (이어서)

If subtree_key not in subtree_dict:
    subtree_key = None
    # subtree_key가 서브트리(subtree_dict)내에 존재하지 않을 때
    # None 서브트리를 사용
    # 즉 입력된 데이터의 변수 중 하나가 기존에 관찰되지 않았다면 None이 된다.

subtree = subtree_dict[subtree_key]

return classify(subtree, input)
# 위에서 정의한 subtree_key를 사용해 적절한 서브트리(subtree_dict)를 선택
# 그리고 입력된 데이터를 계속, 끝까지 분류 - 재귀적(recursive)!
```

03 Decision Tree Algorithm

17.5 종합하기

 # 8. build_tree_id3(inputs, split_candidates = None) - (1)

```
# 8

def build_tree_id3(inputs, split_candidates=None):
    """학습용 데이터로부터 실제 나무를 구축하기"""

    # 만약 split_candidates가 주어져 있지 않다면
    # 입력된 데이터의 모든 변수(attribute)가 파티션 기준 후보
    if split_candidates is None:
        split_candidates = inputs[0] [0].keys()

    # dict.keys()는 딕셔너리의 key값만 모아서 dict_keys 개체를 반환한다
    # 아래 주어질 예시의 경우 dict_keys(['Outlook', 'Temperature', 'Humidity', 'Windy'])
```

03 Decision Tree Algorithm

17.5 종합하기

8. build_tree_id3(inputs, split_candidates = None) - (2)

```
# 입력된 데이터에서 True와 False의 개수를 세어 본다
# inputs 모양 : [ ({key1 : value1, key2 : value2 }, True), ({key1 : value1, key2 : value2}, False) ]
# 전체 데이터 inputs 안의 각 tuple이 개별 데이터 input

num_inputs = len(inputs) # 개별 데이터의 갯수
num_trues = len([label for item, label in inputs if label])
# if 조건문 : 참/ 거짓 판단 후 참일 시 앞의 작업 시행
# 여기서는 label이 True이면 앞의 작업 시행
# 즉, 해당 True를 리스트에 추가하고, 후에 리스트 내의 True의 총 갯수를 센다

num_falses = num_inputs - num_trues

if num_trues == 0:              # True가 하나도 없다면 False 잎을 반환
   return False

if num_falses == 0:             # False가 하나도 없다면 True 잎을 반환
   return True

if not split_candidates:        # 만약 파티션 기준으로 사용할 변수가 없다면
   return num_trues >= num_falses  # 다수결로 결과를 결정
```

03 Decision Tree Algorithm

17.5 종합하기

 # 8. build_tree_id3(inputs, split_candidates = None) - (3)

```
# 아니면 가장 적합한 변수를 기준으로 파티션

best_attribute = min(split_candidates,
   key=partial(partition_entropy_by, inputs))
# partial(func, argu) 메소드는 func(argu)를 실행해준다
# partition_entropy_by(inputs, attribute) :
# 주어진 attribute로 파티션한 뒤 각 부분집합의 엔트로피 계산
# partition_entropy_by(inputs, split_candidates)의 최소값을 만드는
# split_candidate가 best_attribute

partitions = partition_by(inputs, best_attribute)
# inputs를 best_attribute에 따라 파티션. partitions는 딕셔너리 자료형

new_candidates = [a for a in split_candidates
            if a != best_attribute]
# new_candidates는 앞서 best_attribute가 아니였던 attribute들을 모아놓은 리스트
```

03 Decision Tree Algorithm

17.5 종합하기

 # 8. build_tree_id3(inputs, split_candidates = None) - (4)

```
# 재귀적으로 서브트리를 구축

subtrees = { attribute : build_tree_id3(subset, new_candidates)
        for attribute, subset in partitions.items() }
# dict.item은 딕셔너리의 key와 value를 모아둔 '튜플'을 가진
# dict_items([('key1', 'value1'), ('key2', 'value2')]) 객체를 반환
# partition.items 객체의 (attribute, subset)에서 attribute를 가져와서
# attribute를 key로 하고, build_tree_id3(subset, new_candidates)의 결과값을 value로 하는
# subtrees 라는 dict를 만듦

subtrees[None] = num_trues > num_falses
# subtrees에서 "None"이라는 key값의 value는 다수결에 따라 True, False 결정

return (best_attribute, subtrees)
# (best_attribute, subtrees)라는 tuple을 반환
```

03 Decision Tree Algorithm

17.6 실습

http://jihoonlee.tistory.com/16 링크를 참고할 것!
(해당 링크의 도표를 살펴보며 실습을 진행할 것임)

03 Decision Tree Algorithm

17.6 실습

```
""" 17.7 예제 """
if __name__ == "__main__":

    inputs = [
        ({'Outlook':'sunny','Temperature':'hot','Humidity':'high','Windy':'FALSE'}, False),
        ({'Outlook': 'sunny', 'Temperature': 'hot', 'Humidity': 'high', 'Windy': 'TRUE'}, False),
        ({'Outlook': 'overcast', 'Temperature': 'hot', 'Humidity': 'high', 'Windy': 'FALSE'}, True),
        ({'Outlook': 'rain', 'Temperature': 'mild', 'Humidity': 'high', 'Windy': 'FALSE'}, True),
        ({'Outlook': 'rain', 'Temperature': 'cool', 'Humidity': 'normal', 'Windy': 'FALSE'}, True),
        ({'Outlook': 'rain', 'Temperature': 'cool', 'Humidity': 'normal', 'Windy': 'TRUE'}, False),
        ({'Outlook': 'overcast', 'Temperature': 'cool', 'Humidity': 'normal', 'Windy': 'TRUE'}, True),
        ({'Outlook': 'sunny', 'Temperature': 'mild', 'Humidity': 'high', 'Windy': 'FALSE'}, False),
        ({'Outlook': 'sunny', 'Temperature': 'cool', 'Humidity': 'normal', 'Windy': 'FALSE'}, True),
        ({'Outlook': 'rain', 'Temperature': 'mild', 'Humidity': 'normal', 'Windy': 'FALSE'}, True),
        ({'Outlook': 'sunny', 'Temperature': 'mild', 'Humidity': 'normal', 'Windy': 'TRUE'}, True),
        ({'Outlook': 'overcast', 'Temperature': 'mild', 'Humidity': 'high', 'Windy': 'TRUE'}, True),
        ({'Outlook': 'overcast', 'Temperature': 'hot', 'Humidity': 'normal', 'Windy': 'FALSE'}, True),
        ({'Outlook': 'rain', 'Temperature': 'mild', 'Humidity': 'high', 'Windy': 'TRUE'}, False),
    ]
# 총 14개의 데이터
```

03 Decision Tree Algorithm

 # 1) 전체 엔트로피 계산 & 각 attribute로 파티션한 후 엔트로피 계산 - *링크의 첫 번째 도표, 첫번째 의사결정나무*

```
print('inputs', format(data_entropy(inputs), '.4f'))
for key in ['Outlook','Temperature','Humidity','Windy']:
    print(key, format(partition_entropy_by(inputs, key), '.4f'))
print()

"""Outlook의 엔트로피가 가장 낮으며, Outlook으로 파티션 했을 경우의 Information gain이 가장 높다"""
```

 # 2) Outlook으로 파티션 한 뒤 그 안에서 Sunny로 또 파티션 하기 - *링크의 두번째 도표, 두번째 의사결정나무*

```
Out_Sunny_inputs = [(input, label)
                    for input, label in inputs if input["Outlook"] == "sunny"]
print(Out_Sunny_inputs)
print('Out_Sunny_inputs', format(data_entropy(Out_Sunny_inputs), '.3f'))
for key in ["Temperature", "Humidity", "Windy"]:
    print(key, partition_entropy_by(Out_Sunny_inputs, key))
print()

"""Outlook으로 파티션 한 후 다시 Sunny로 파티션 하고,
그 다음 각 attribute로 파티션을 해 보았더니 Humidity로 파티션 한 결과의 엔트로피가 가장 낮았다"""
```

03 Decision Tree Algorithm

3) *링크의 세번째, 네번째 도표*

```
Out_Sunny_Humid_inputs = [(input, label)
                  for input, label in Out_Sunny_inputs if input["Humidity"] == 'high']
print(Out_Sunny_Humid_inputs)
print('Out_Sunny_Humid_inputs', format(data_entropy(Out_Sunny_Humid_inputs), '.3f'))

for key in ["Temperature", "Windy"]:
    print(key, partition_entropy_by(Out_Sunny_Humid_inputs, key))
print()
```

03 Decision Tree Algorithm

4) Outlook에서 overcast 선택 시 - *링크의 다섯번째 도표*

```
Out_Over_inputs = [(input, label)
                   for input, label in inputs if input["Outlook"] == "overcast"]
print(Out_Over_inputs)
print('Out_Over_inputs', format(data_entropy(Out_Over_inputs), '.3f'))
print()
```

5) Outlook에서 rain 선택 시 - *링크의 여섯번째 도표*

```
Out_Rain_inputs = [(input, label)
                   for input, label in inputs if input["Outlook"] == "rain"]
print(Out_Rain_inputs)
print('Out_Rain_inputs', format(data_entropy(Out_Rain_inputs), '.3f'))
for key in ["Temperature", "Humidity", "Windy"]:
    print(key, partition_entropy_by(Out_Rain_inputs, key))
print()
```

03 Decision Tree Algorithm

 # 6) 전체 의사결정 나무 그리기 - *링크의 마지막 의사결정나무 그림*

```
print("building the tree")
tree = build_tree_id3(inputs)
print(tree)
print()
```

 # 7) 새로 입력된 하나의 데이터 값을 의사결정 나무에 따라 분류해보기

```
print("sunny / mild / high / TRUE", classify(tree,
    {"Outlook" : "sunny",
     "Temperature" : "mild",
     "Humidity" : "high",
     "Windy" : "TRUE"} ))
```

04 Random Forest Algorithm

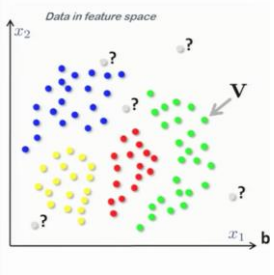
지금까지 전체 데이터를 가지고
트리를 만들었다.

그런데,

전체 데이터를 가지고 트리를 만들면 생기는
문제가 있다!

04 Random Forest Algorithm

지금까지 전체 데이터를 가지고 트리를 만들었다.
그런데 전체 데이터를 가지고 트리를 만들면 생기는 문제가 있다!



1. 모든 데이터(data point)들의 parameter을 비교해야해서 데이터를 로딩하고 계산하는데 컴퓨팅 코스트가 많이 듦

2. 모든 변수(parameter)들의 information gain point를 계산해야하고 비교해야해서 컴퓨팅 코스트가 많이 듦

ex) Information gain 가장 많은 split 포인트, 어떤 변수로 할 것인지

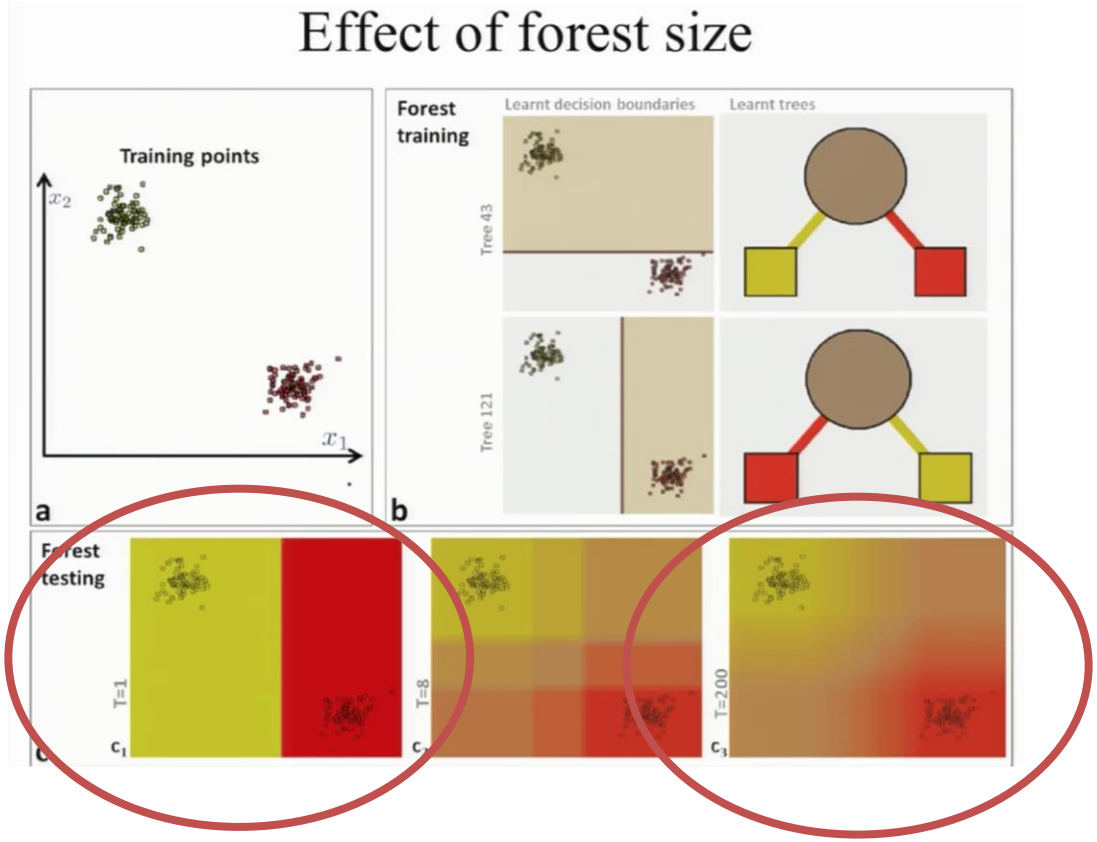
Bagging을 통해 N개의 데이터에서 $\sqrt{N}$ 개의 데이터를 뽑아서 씀

04 Random Forest Algorithm

그래서 우리는 하나의 나무만 기르는 대신 여러 나무를 길러서 숲을 키우기로 했다!

04 Random Forest Algorithm

그리고 Randomness(무작위성)가 주는 장점이 있다. 복잡성(complexity)을 제어 가능



출처 : UBC Machine learning coure
https://www.youtube.com/watch?v=3kYujfDgmNk&index=12&list=PLE6Wd9FR--EdyJ5lbFl8UuGjecvVw66F6

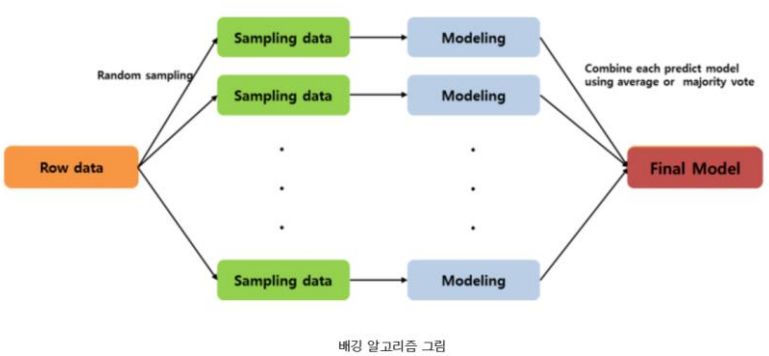
04 Random Forest Algorithm

https://www.youtube.com/watch?v=D_2LkhMJcfY

Ensemble Machine Learning Algorithm
따로 쓰는 경우에 비해 더 좋은 예측 성능을 얻기위해 다수의 학습 알고리즘을 사용하는 방법

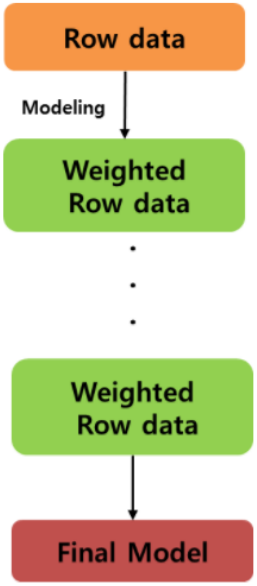
http://haandol.github.io/2017/02/06/ensemble-bagging-boosting.html

Bagging



배깅 알고리즘 그림

Boosting

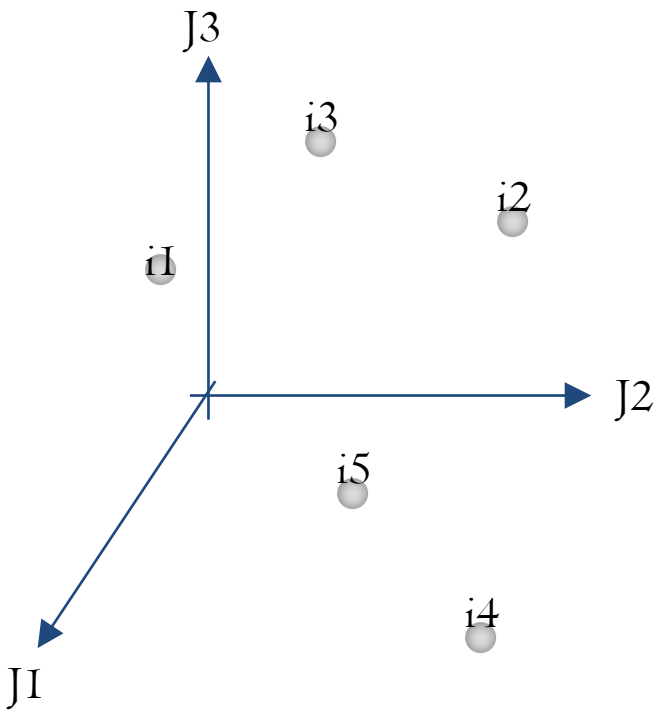


부스팅 알고리즘 그림

Out-Of-Bag(OOB)
Oob 샘플은 샘플링 과정에서 추출되지 않은 관측치들을 말하며 이 샘플들은 주로 평가용 데이터에서의 오분류율을 예측하는 용도 및 변수 중요도를 추정하는 용도로 많이 이용

04 Random Forest Algorithm

랜덤포레스트 생성 알고리즘

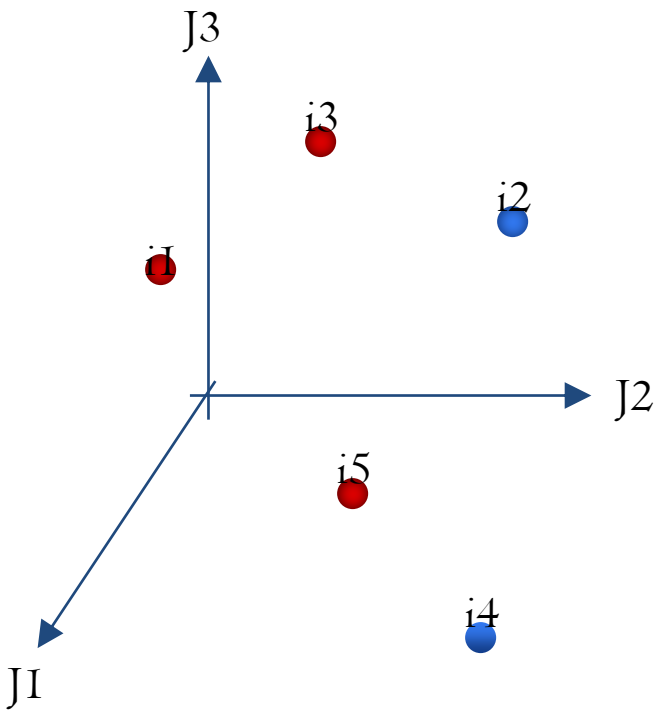


Original Data Set

$$X = \begin{bmatrix} 1 & 3 & 0 & 8 & 5 \\ 0 & 6 & 2 & 9 & 5 \\ 2 & 1 & 4 & 0 & 1 \end{bmatrix}$$

04 Random Forest Algorithm

I) 중복을 허용하여 랜덤하게 데이터를 뽑는다



Original Data Set

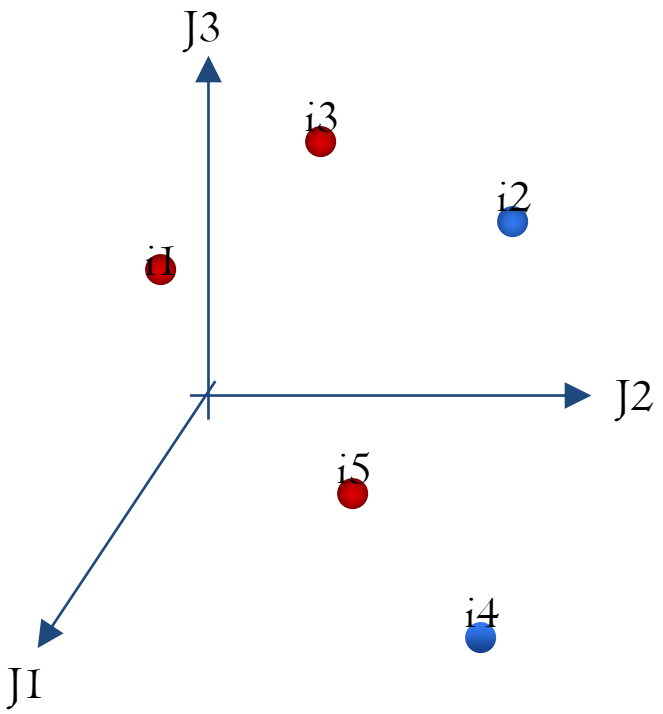
$$X = \begin{bmatrix} & \underset{i=1}{\bigcirc} & i=2 & \underset{i=3}{\bigcirc} & \underset{i=4}{\bigcirc} & i=5 \\ j=1 & 1 & 3 & 0 & 8 & 5 \\ j=2 & 0 & 6 & 2 & 9 & 5 \\ j=3 & 2 & 1 & 4 & 0 & 1 \end{bmatrix}$$

$$Y = \begin{bmatrix} i=1 & i=2 & i=3 & i=4 & i=5 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

* 1이면 Blue 0이면 Red

04 Random Forest Algorithm

2) 중복을 허용하여 랜덤하게 feature를 뽑는다



Original Data Set

$$X = \begin{bmatrix} & i=1 & i=2 & i=3 & i=4 & i=5 \\ j=1 & 1 & 3 & 0 & 8 & 5 \\ j=2 & 0 & 6 & 2 & 9 & 5 \\ j=3 & 2 & 1 & 4 & 0 & 1 \end{bmatrix}$$

$$Y = \begin{bmatrix} i=1 & i=2 & i=3 & i=4 & i=5 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

* 1이면 Blue 0이면 Red

04 Random Forest Algorithm

Bootstraping

데이터를 균등 랜덤하게 추출해 데이터셋을 만들고, 그 데이터셋을 훈련하여 다른 모델을 만드는 것.

다른 데이터를 사용하여 모델을 만들면 완전히 다른 모델이 나오게 된다.
Correlation이 거의 없다

-> 좋은 parameter를 추정하는 테크닉으로 머신러닝에서 많이 씀.
원래 만든 모델과 parameter가 괜찮은지 확신할 수 있게 해줌.

(I) 복잡성(complexity)를 제어할 수 있음 - 몇개의 데이터를 뽑을지 정할 수 있음

(I) 통계학자들이 cross validation 방식 대신 많이 쓰는 방법

04 Random Forest Algorithm

의사 코드(Pseudocode)

[I] 랜덤 트리를 기른다 (Tb_I)

I)사이즈 N 의 bootstrap 샘플 데이터를 랜덤하게 뽑아 (z개) training data 로 설정한다 (앞에서 N = 5, z = 3)

2)P개의 feature(parameter, 변수)중에 m개의 변수를 랜덤하게 뽑아 split point를 정한다.

여기서 랜덤추출은 중복을 허용한 무작위 추출이다. (왜냐하면 현실에서 데이터는 중복인 데이터도 많고 중복이 많으면 빈도가 높음을 반영하기 때문에 어떤 feature나 분류나 예측이 많이 나오는지와 같은 것을 자연스럽게 반영하게 된다)

3) Information gain이 가장 큰 split point의 값으로 노드를 2개의 자식 노드로 나눈다. (각 노드에서 반복)

04 Random Forest Algorithm

의사 코드(Pseudo code)

[2] 랜덤 트리를 B개 길러 포레스트(각각의 랜덤 트리들이 모여 숲)를 만든다. Tb_I~Tb_B개 존재

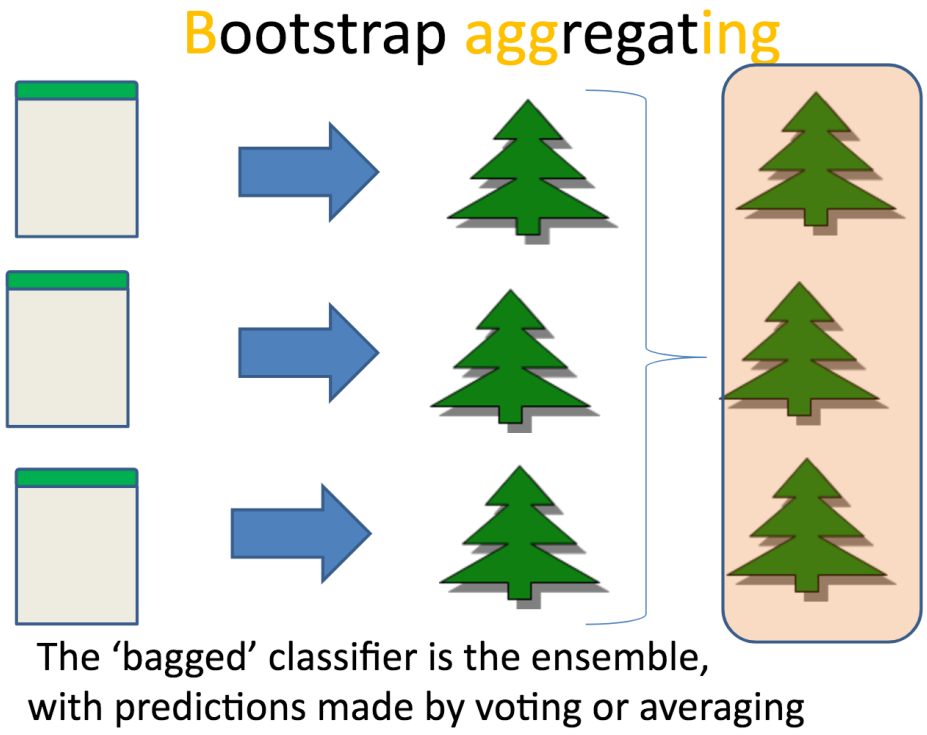
[3] 각 트리를 평균낸다. (Bagging)

각 트리는 완전히 다른 형태로 생겼다. 가지들도 다르고 node도 다르다. 그래서 트리를 평균낸다는 의미는 새로운 데이터셋 v가 들어오면 그 데이터셋이 특정 깊이(depth)에서 어떤 decision에 있는지 확인하고 각각의 트리에서 같은 깊이에서 어떤 decision에 있는지를 확인하여 산술평균을 낸다는 것을 뜻한다. (다 합하고 합친 수만큼 나누기, 혹은 다수결 투표같은 방식. 쉽게 말하면 "각 트리에서 동일한 새로운 입력 값에 대해 어떤 선택이 더 많은가?")

04 Random Forest Algorithm

Bagging

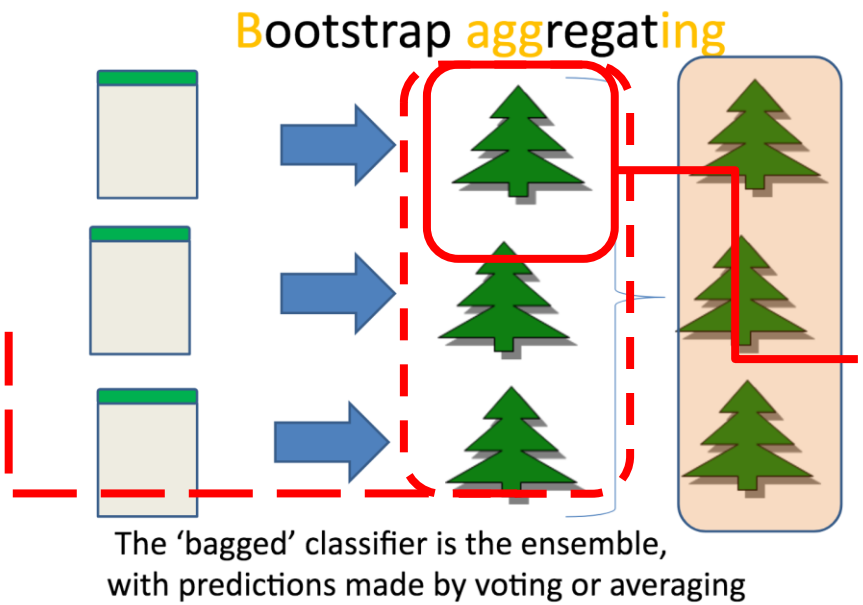
많은 랜덤 트리의 estimator(추정량)을 산술평균이나 투표로 합치는 프로세스



04 Random Forest Algorithm

Bagging

각 나무들의 데이터와 변수가 랜덤하게 추출되므로 각 트리의 예측값 간에 correlation은 거의 없다



하나하나의 트리는 랜덤하게 데이터를 뽑고 랜덤하게 feature를 뽑아 아주 랜덤하기 때문에 variance가 굉장히 높은 high variance classifier이다.

그런데 bagging을 통해 산술평균을 하면 variance를 없애준다!

04 Random Forest Algorithm

각 랜덤트리는 랜덤한 데이터와 랜덤한 파라미터로 만들어지므로 극도로 high variance classifier

각 랜덤트리는 High variance Low bias

- Variance : 모든 training set 데이터와 예측 값의 차이
  Variance가 높으면 모형도 복잡해짐
  (모델의 정확성과 상관있는 것은 아님)

- Bias : 델타. 실제값과 예측값의 차이.
  Underfitting => High Bias
  Overfitting => Low Bias (각 랜덤트리는 그래서 Low bias)

Bagging을 통해 averaging하면 variance가 낮아짐!

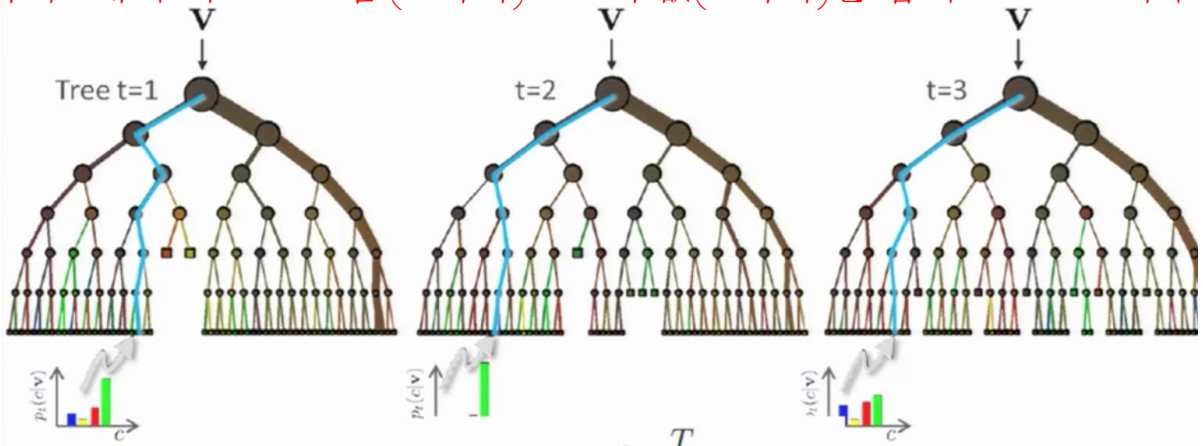
04 Random Forest Algorithm

내 숲에 3개의 트리 t=1,2,3가 있고 새로운 데이터 v가 들어갔을 때 v는 트리별로 다른 root에 다다름

Building a forest (ensemble)

In a forest with T trees we have $t \in \{1, \cdots, T\}$. All trees are trained independently (and possibly in parallel). During testing, each test point $\mathbf{v}$ is simultaneously pushed through all trees (starting at the root) until it reaches the corresponding leaves.

각 트리에 4개의 히스토그램 (4 벡터) 3 트리 값(3 벡터)을 합치고 3으로 나누기



$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{v})$$

출처 : UBC Machine learning course

[Criminisi et al, 2011]

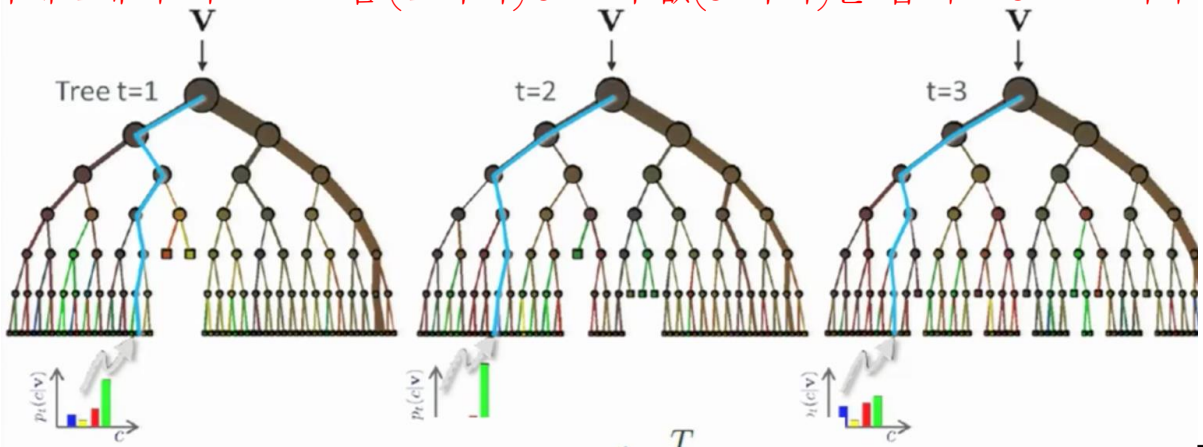
04 Random Forest Algorithm

내 숲에 3개의 트리 t=1,2,3가 있고 새로운 데이터 v가 들어갔을 때 v는 트리별로 다른 root에 다다름

Building a forest (ensemble)

In a forest with T trees we have $t \in \{1, \cdots, T\}$. All trees are trained independently (and possibly in parallel). During testing, each test point $\mathbf{v}$ is simultaneously pushed through all trees (starting at the root) until it reaches the corresponding leaves.

각 트리에 4개의 히스토그램 (4 벡터) 3 트리 값(3 벡터)을 합치고 3으로 나누기



$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{v})$$

[Criminisi et al, 2011]

$$\frac{1}{3}\sum_{t=1}^{3} P_t(c|v)$$

04 Random Forest Algorithm

각 트리는 다른 식으로 만들어져서 같은 node들의 구성이 아닌데 어떻게 averaging을 하는가?

트리를 만들고 같은 깊이의 노드에서 decision이나 히스토그램을 확인하여 각 나무에서 똑같이 구한 값을 averaging하는 것

* 같은 깊이인지 확인하는 방법
ex) 트리에 데이터를 넣고 5분이 지난 후 그 데이터가 어떤 decision에 있는지 확인

04 Random Forest Algorithm

실제 적용

```
def forest_classify(trees, input):
    votes = [classify(tree, input) for tree in trees]
    vote_counts = Counter(votes)
    return vote_counts.most_common(1)[0][0]

    if len(split_candidates) <= self.num_split_candidates:
        sampled_split_candidates = split_candidates
    else:
        sampled_split_candidates = random.sample(split_candidates,
                                                 self.num_split_candidates)

    best_attribute = min(sampled_split_candidates,
                         key=partial(partition_entropy_by, inputs))

    partitions = partition_by(inputs, best_attribute)
```

04 Random Forest Algorithm

실제 적용(scikit-learn) – Iris Data

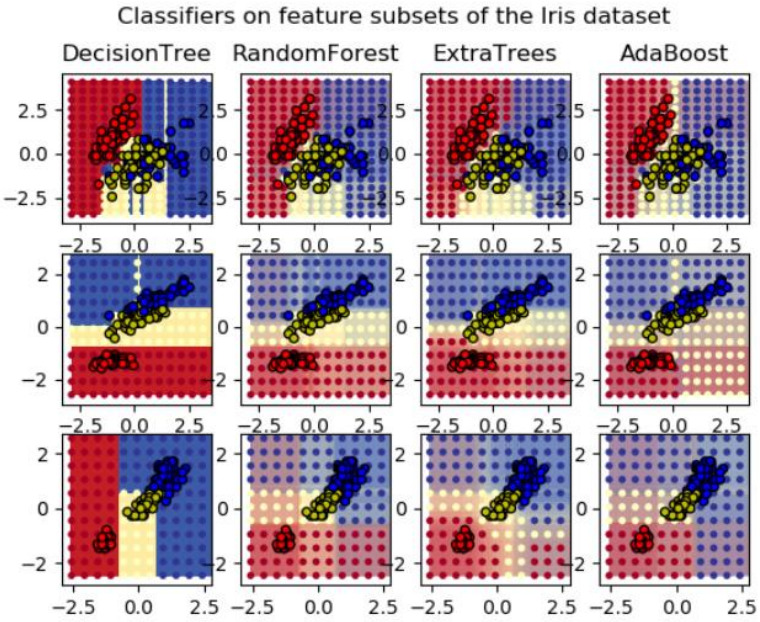
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=1000, n_features=4,
                           n_informative=2, n_redundant=0,
                           random_state=0, shuffle=False)
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X, y)




print(clf.feature_importances_)

print(clf.predict([[0, 0, 0, 0]]))
```

04 Random Forest Algorithm

실제 적용(scikit-learn)

앞에 나온 합격 불합격 범주형 자료 벡터화 시킨 후

sciki-learn의 DecisiontreeClassifier를 쓰는 내용을 다룰 예정!



04 Random Forest Algorithm

활용 가능 분야

- Task Classifier
- Face Detection
- Text Classifier & Spam detector

… 너무 많은 것을 할 수 있을 것이다!

ex) 세상 모든 단어들의 딕셔너리에 200만개의 단어가 있다고 하면 중복허용 랜덤추출 50개의 단어 부분집합을 만듦. 어떤 글자는 더 많이 뽑힘 => 더 중요한 feature를 뽑는 방법
50개의 각 단어에 대해 information gain을 구함. 이 메시지가 스팸인지 아닌지 구분하는 단어의 수가 트리의 노드 수가 될 것임.

04 Random Forest Algorithm

1. 어떤 feature가 중요한지 반영되는가?

중복 허용 랜덤 추출을 하므로 현실세계를 반영하게 된다.

2. 얼마나 많은 트리가 필요한가?

많을수록 좋다. 200개 정도의 트리가 있으면 decision boundary가 부드러워진다. 우리는 확률을 계산중이어서 등고선처럼 probability distribution(확률분포도)이 나옴

3. Classification 시에 레이블 수(class 수)가 트리의 성능에 영향을 주는가?

Binary(클래스가 2개)든 몇개든 클래스 수와 상관없이 잘 분류한다. 현실에서는 노이즈가 있고 클래스끼리 중첩이될도 있다. 하지만 그마저도 잘 분류해냄.

4. 트리 깊이의 효과는?

트리가 얕으면 depth=3 언더피팅, 트리가 깊으면 depth=15 오버피팅

Thank you !