결정 트리와 랜덤 포레스트 이해

가천대학교 스마트보안전공 202334841 이가람

lyza@gachon.ac.kr, blog.naver.com/luexr

CONTENTS

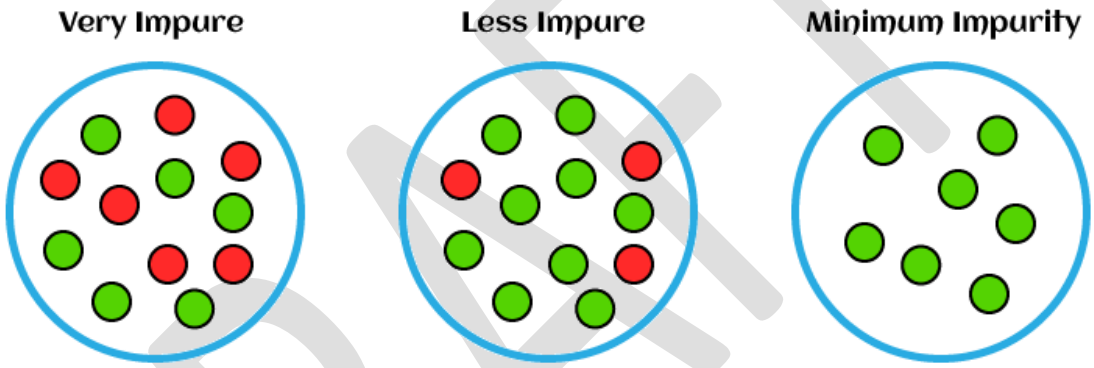
1. 불순도(impurity) 2
2. 엔트로피 이해(understanding entropy) 2
3. 정보 이득과 지니 불순도(information gain and gini impurity) 5
4. 결정 트리(decision tree) 8
5. 트리의 앙상블 그리고 랜덤 포레스트(ensemble of tree and random forest) 13

1. 불순도(IMPURITY)

In chemistry and materials science, impurities are chemical substances inside a confined amount of liquid, gas, or solid, which differ from the chemical composition of the material or compound.

불순도(impurity)란 말 그대로 어떤 대상에 얼마나 다양한 개체들이 섞여 들어가 있는지를 나타내는 척도이다. 예를 들어 설탕과 소금이 있다고 할 때, 한 통 안에 오로지 설탕 또는 소금만 들어 있다면 불순도는 낮고, 설탕과 소금이 1:1 비율로 섞여 있다면 불순도는 상승한다. 만약, 이 상태에서 후추와 같은 다른 요소들을 더 가미하여 더욱 다양한 개체들이 동시에 섞이는 a 상황이 발생하면, 불순도는 더욱 상승한다.

아래 사진 자료는 이런 불순도의 개념적인 정의를 직관적으로 잘 나타내고 있다.



이런 불순도를 수치화 하는 방법으로는 여러가지가 있는데, 정보 이론에서 유명한 엔트로피(entropy), 결정 트리에서 직접적으로 사용하는 지니 계수(Gini index) 등이 있다.

2. 엔트로피 이해(UNDERSTANDING ENTROPY)

엔트로피(entropy)란 물리학 또는 기타 연관된 과학 분야에서 어떤 관찰 대상을 이루는 요소들이 현재 가지고 있는 무질서한 정도(the measure of disorder)로 표현된다. 또한, 자연계에서 시간 흐름에 따라 엔트로피는 필연적으로 증가하는 추세를 따라갈 수밖에 없다고 알려져 있다. 쉽고 직관적인 예시로 시간이 지남에 따라 사람이 사용하는 방은 별도의 에너지를 소모하는 노력(청소)을 가하지 않으면 계속해서 방이 더러워지는 예시를 들 수 있다. 엔트로피가 무질서한 정도를 측정하는데 사용할 수 있다는 점을 감안하면, 시간이 지날수록 청소를 하지 않으면 방의 엔트로피는 꾸준히 증가하는 것으로 볼 수 있다.

열역학적(thermodynamic) 관점에서 엔트로피의 정의를 조금 더 깊게 살펴보면, 평형 상태에 이르러 온도가 T 에 수렴한 계(field: 일정한 상호 작용이나 서로 관련이 있는 물체들의 모임)에 열을 가할 때, 주어진 열이 실제 사용 가능한 일로 얼마나 변환될 수 있는지에 대한 가능성을 나타내는 상대적인 척도이다. 예를 들어 온도가 T에 이른 계에 열 δQ를 가한다고 하면, 엔트로피의 증가량 dS는 아래와 같이 정의될 수 있다.

$$dS = \frac{\delta Q}{T}$$

이 식을 부정적분 하면 아래와 같이 상대적인 척도인 엔트로피의 정도를 알 수 있다.

$$\Delta S = \int \frac{\delta Q}{T}$$

즉, 열역학적 관점에서의 엔트로피는 온도와 관련되어 있으며, 새롭게 가해진 열이 기존의 어떤 안정된 온도를 가지고 있는 계에 얼마나 큰 변화 또는 영향을 가할 수 있는지를 보다 수치적으로 개념화한 척도이다. 예를 들어 같은 크기의 열 에너지를 가한다고 하더라도 이미 뜨거운 상태를 유지하는 계보다 차가운 상태를 유지하는 계에 영향을 더 크게 준다. 다시 말하면, 더 차가운 계에 더 큰 엔트로피의 증가가 발생한다. 이는 이전에 설명한 무질서(disorder)한 정도를 나타낸다는 설명과 부합한다.

미시적인 관점에서, 뜨거운 상태의 계는 분자 운동이 활발하므로, 그만큼 덜 조직적으로 연계되어 있어 무질서도가 높아 엔트로피가 높다고 할 수 있으며, 차가운 상태의 계는 그 반대라고 할 수 있다. 따라서 이 과정에서 어떤 열 에너지를 가하면, 필연적으로 열 에너지 투입의 대상의 되는 계가 가지고 있는 엔트로피의 총량은 증가한다. 그러나 이미 뜨거운 계에서는 투입되는 열 에너지가 실제로 증가시키는 엔트로피의 양이 이미 존재하는 엔트로피의 양에 비해 적으며(적은 영향을 주며), 차가운 계에서는 그 비율이 커 많은 영향을 받는다.

지금까지 여러 번 언급된 이 무질서함(disorder)이, 정보 이론(information theory)으로 넘어오게 되면 모르는 것(something you don't know)의 정도로 변하게 된다. 정보 이론, 통신을 포함한 IT 분야에 지대한 영향을 끼친 클라우드 섀넌(Claude Shannon)은 1948 년 자신의 논문 <A mathematical theory of communication>을 통해 방금 전에도 서술한 열역학에서의 엔트로피를 개념적으로 확장하여 정보량을 수치화 한다는 내용을 발표했다. 섀넌은 여기서 정보량을 어떤 대상의 구체적인 값을 관찰할 때 얻을 수 있는 정보(information)의 양을 얻는지 정량화 하는 방법을 발표했다. 이때, 어떤 대상이 실제로 일어날 수 있는 확률에 반비례하여 얻을 수 있는 정보의 양이 결정된다는 것이 핵심 아이디어이다. 즉, 덜 자주 일어나는 사건일수록 자주 발생하는 사건보다 정보량이 많다는 것이다.

예를 들어 당연하게도 해가 뜰 확률은 뜨지 않은 확률보다 훨씬 크다. 따라서, “내일 해가 뜬다” 라는 메시지보다 “내일 해가 뜨지 않는다”라는 메시지는 정보 이론에 따라 더욱 많은 정보량을 가지고 있다는 소리가 된다. 즉, 확률에 따라 정보량이 결정되는 형태이므로 언제나 일어나리라고 보장된(확률이 100%) 사건은 정보량이 없다.

또한, 어떤 이벤트에 대하여 독립적인 사건(independent event)들은 추가적인 정보량(additive information)을 가지게 된다. 예를 들어, 공정한 동전을 던졌을 때, 앞면이 연속하여 두 번 나올 가능성은 한 번 나올 가능성의 절반이므로 정보량은 그에 반비례하여 2 배가 된다.

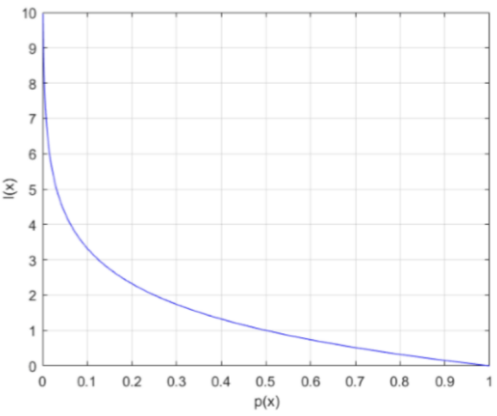
지금까지 서술한 조건을 만족시키는 수식은 아래와 같으며, 어떤 메시지가 일어날 확률에 음의 로그를 취한 값으로 나타낼 수 있다. 확률은 반드시 1 이하이므로 로그 함수는 음수만을 가지게 된다. 그러나 정보량은 양의 정수로 표현되는 것이 관례이므로, 앞에다가 음의 기호(-)를 붙인 것이다. 어떤 메시지가 일어날 확률을 $P(x)$라고 하면, 그 메시지가 가진 정보량은 $I(x)$로 아래와 같이 정의된다.

$$I(x) = -logP(x)$$

이때, 실제 컴퓨터들은 binary 한 시스템을 사용하기 때문에 밑이 2 인 로그를 취하며 이러한 정보량의 단위를 섀넌(Shannon) 또는 비트(bit)라고 하며, 비트(bit)라는 이름은 전세계적으로 널리 통용되어 사용되고 있다.

$$I(x) = -log_2 P(x)$$

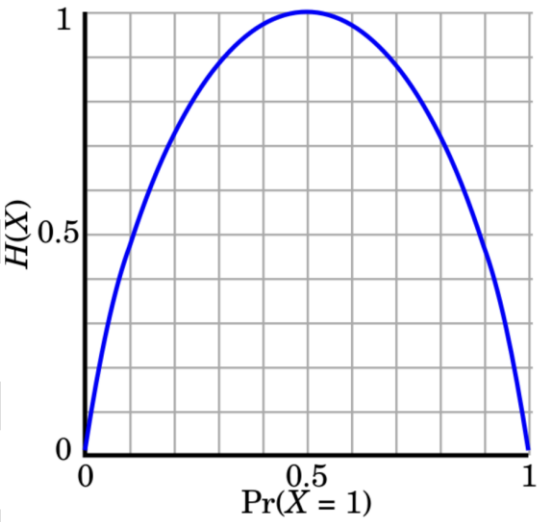
이러한 정보량을 그래프로 나타내면 아래와 같다.



섀넌 엔트로피(Shannon entropy)는 지금까지 설명했던 내용을 바탕으로, 모든 사건 정보량의 기댓값의 총량을 의미한다. 즉, 전체 사건의 확률분포의 불확실성을 의미한다. 어떤 확률분포 P에 대한 섀넌 엔트로피 $H(P)$는 아래와 같다. (각각의 정보량을 더한 형태이다.)

$$H(x) = -\sum_{x} P(x) log_2 P(x)$$

위 수식을 참고하여, binary 한 환경에서 클래스(가짓수)가 2 개인 섀넌 엔트로피의 그래프를 그려보면 아래와 같이 $P(x) = 0.5$일 때 엔트로피 값 $H(x)$가 최대를 가리키는 것을 볼 수 있다. 이는 두 가지 색깔의 공들이 섞인 주머니에서 한 가지를 뽑을 때 가지는 불확실성에 비유할 수 있다. 예를 들어 빨간색과 파란색 공이 있을 때, 빨간색 공을 뽑을 가능성을 $P(x)$라고 하자. 이때 $P(x)$ 극단적으로 낮거나(파란색 공을 뽑을 가능성이 아주 높음), 극단적으로 높은 경우(파란색 공을 뽑을 가능성이 아주 낮음) 공을 뽑는 사람이 어떤 공을 뽑을지 비교적 높은 확률로 예측할 수 있다. 다시 말해 불확실성이 낮고, 예측가능성이 높으니 이전에 설명한 바에 따라 엔트로피가 높다. 그러나, $P(x) = 0.5$인 경우에는 두 가지 공을 뽑는 가운데에서 가장 예측하기 힘들다. $P(x)$, 즉 빨간색 공을 뽑을 가능성이 가장 낮은 상태로, 현재 상황에서는 엔트로피가 가장 높은 상황이 된다.



이는 1 절에서 설명한 불순도(impurity)와도 연계되는 부분이다. 불순도라 함은 어떤 대상에 얼마나 다양한 개체들이 섞였는지를 측정하는 측도라고 하였다. 위 상황에서는 $P(x) = 0.5$일 때 불순도가 가장 높다. 이럴 경우, 어떠한 한 색깔의 공만 분류하는 것이 어렵다. 다시 말해 엔트로피가 높다는 말은 불순도가 높고, $P(x)$, 즉 특정한 확률은 감소한다는 것을 알 수 있다.

3. 정보 이득과 지니 불순도(INFORMATION GAIN AND GINI IMPURITY)

2 절에서 설명하였던 개념을 통해 정보 이익 또는 정보 이득(IG: Information Gain)이란 개념을 만들 수 있다. 정보 이익이란, 정보의 가치라고 할 수 있으며, 정보가 가지고 있는 엔트로피의 값, 즉 불확실성이 얼마나 줄어들었는지를 계산한다. (즉, "데이터를 얼마나 더 잘 구분하게 될 수 있는지"에 대한 척도이다.) 한 단계당 2 가지로 분류하는 경우, 아래와 같이 이전 엔트로피에서 (분류 후) 현재 엔트로피의 차로 계산하여 정보 이득을 쉽게 계산할 수 있다. (상위 노드 엔트로피에서 하위 노드 엔트로피를 뺀 거)

$$IG = E(before) - E(after)$$

데이터를 분류하는 입장에서 바라보면, 정보 이득이 커지면 커질수록 분류할 대상의 불순도가 감소하게 되었다는, 즉 데이터가 더욱 깔끔하고 명확하게 분류되었다는 의미가 된다. 결정 트리와 같은 분류 알고리즘의 경우에는 이 정보 이득을 최대한 크게 하는 방향으로 알고리즘을 수행한다.

만약 나누는 범주가 2 개 이상이라면(가지가 2 개 이상으로 뻗어 나간다면) 위에서 말한 정보이득(IG) 계산 방법에 문제가 발생한다. 여러 가지로 분류할수록 불순도가 높다 하더라도 더 명확하게 분류할 수 있으므로 알고리즘은 가지를 더 많이 치는 쪽만 더 중요하게 판단할 수 있다. 따라서 이럴 경우에는 이득율(IGR: Information Gain Ratio)라는 척도를 사용한다. 이득율은 IG(Information Gain)을 IV(Intrinsic Value)로 나눈 값으로, 어떤 속성이 취할 수 있는 분류의 가짓수가 증가할수록 IV 의 값이 크게 증가하여 전반적으로 이득율의 값이 감소하게 된다. 이득율은 아래와 같은 수식으로 계산한다.

$$IV(x) = -\sum_{x} \frac{1}{x} log_2\left(\frac{1}{x}\right)$$

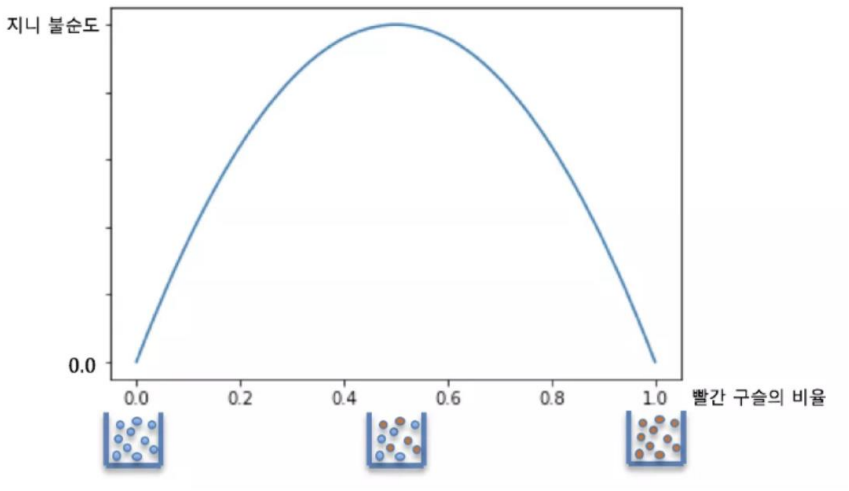
$$IGR(x) = \frac{IG(x)}{IV(x)}$$

위의 경우에는 분기되어지는 가짓수에 따라 일종의 penalty 가 부과되는 형태이므로 IG 의 단점을 보완할 수 있다.

True / False 형태로 계속 질문하여 대상을 분류하는 CART (Classification and Regression Tree)알고리즘의 경우에는 지니계수(gini index) 또는 지니 불순도(gini impurity)를 통해

불순도(통계적인 분산의 정도)를 계산하고, 이를 통해 정보 이득을 계산하여 정보 이득이 최대가 되도록 노드를 분산시킨다.

아래 그림은 불순도에 따른 지니 불순도의 변화를 단순화하여 그린 그림이다. 결국 지니 불순도 또한 위에서 서술한 엔트로피처럼 데이터의 불순도를 측정하는 척도이므로 엔트로피에서 설명한 것과 같은 그림이 그려진다.



어떤 조사 대상의 세트로부터 지니 불순도를 구하는 수식은 아래와 같다.

$$G(x) = 1 - \sum_{x} P(x)^2$$

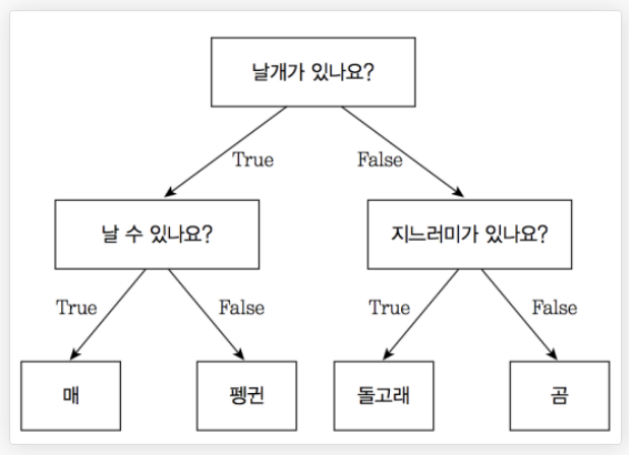
예를 들어 구슬이 25 개 들어있는 주머니를 생각해보자. 이때 빨간 구슬이 10 개, 파란 구슬이 15 개 들어있다고 해 보자. 이 경우 아래와 같이 지니 불순도를 구할 수 있다.

$$G(pocket) = 1 - \left(\left(\frac{10}{25}\right)^2 + \left(\frac{15}{25}\right)^2\right) = 0.48$$

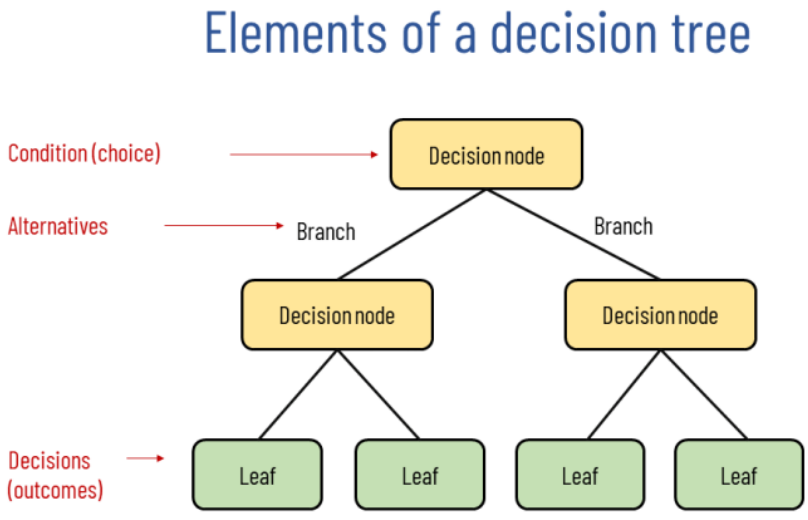
이러한 지니 계수는 경제학에서도 소득의 불평등을 측정할 때 널리 사용되며, 대중에게도 비교적 널리 알려져 있다. 기본적으로 결정 트리는 이 지니 계수를 통해 정보 이득을 계산해 트리를 만들며, 원한다면 DecisionTreeClassifier 클래스에서 직접 엔트로피 방식을 사용하도록 할 수도 있다. 중요한 점은, 데이터의 불순도를 계산하고, 분류 기준에 따라 정보 이득이 어떻게 변하는지 살펴보고 이를 통해 정보 이득을 최대화하는 과정이다.

4. 결정 트리(DECISION TREE)

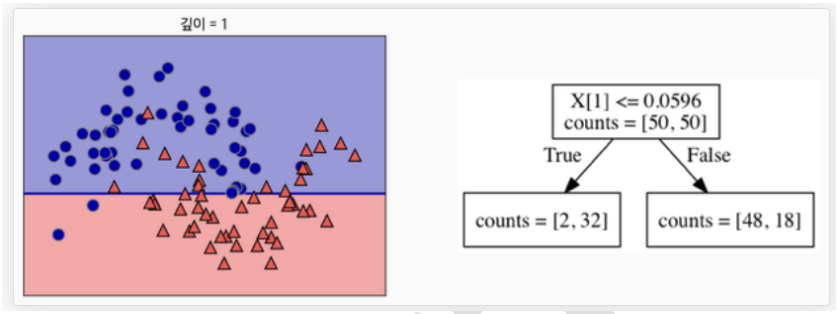
결정 트리(decision tree)란 스무고개 게임과 같이, 어떤 척도를 기준으로 한 질문을 반복적으로 하여 대상을 분류하는 알고리즘으로, 아래 그림과 같이 동작한다. 예를 들어 아래 그림에서는 처음에 입력으로 동물들이 전부 주어진다. 그리고 날개가 있는지, 있으면 날 수 있는지, 아니면 지느러미가 있는지를 통해 질문을 이어나가 분류한다. 이런 과정에서 분류되는 데이터는 점점 불순도가 낮아지며 결국 원하는 목적에 맞게 분류되게 된다.

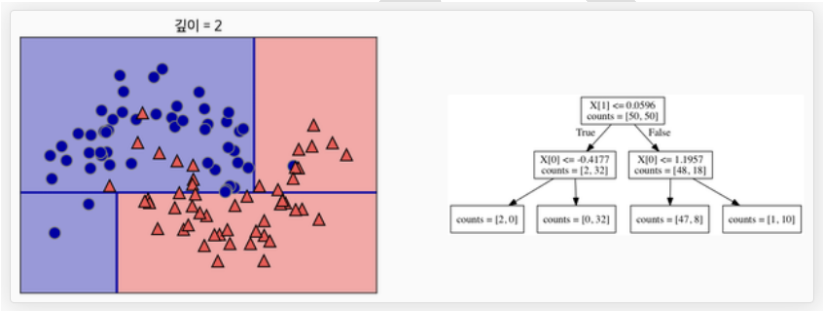


일반적으로는 True / False 와 같이 한 번에 두가지로 분기하며 , 이렇게 특정 기준에 따라 분류하는 모델을 결정 트리 모델(decision tree model)이라고 한다. (자료구조에서 트리(tree) 구조라고 하여 굉장히 자주 나온다.)

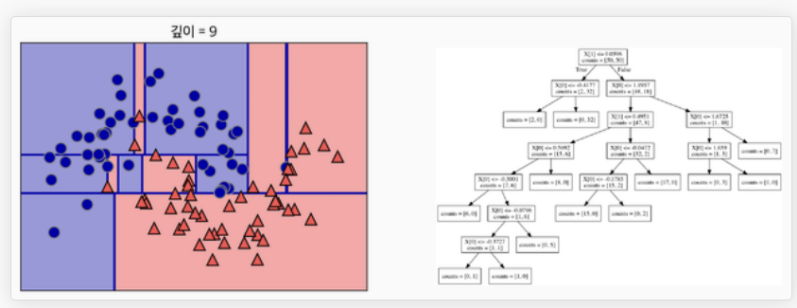


작동 방식은 이전에 설명하였듯이 데이터를 잘 구분할 수 있는 질문을 준비하여 나누고, 또 나누어 각각의 불순도를 줄이고 정보 이득을 최대화하는 방식을 사용한다.



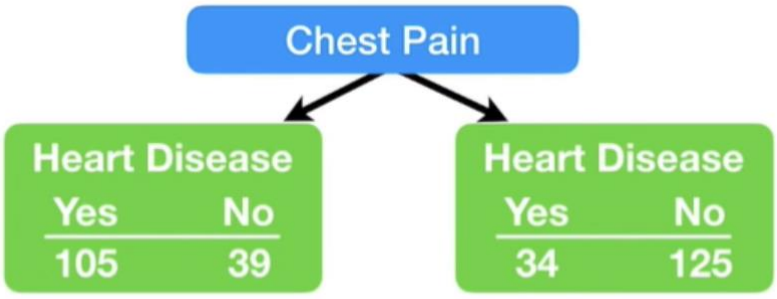


그러나 아래 그림처럼 아무런 제약 없이 트리 알고리즘을 작동시키면 질문이 끝도 없이 진행된다. 트리 구조에서는 이렇게 노드가 계속 아래로 한 단계씩 자랄 때마다 트리의 깊이(depth)가 1 씩 증가한다고 하는데, 이 깊이를 적절히 조절하지 않으면 과적합(overfitting) 현상이 발생한다. 그래서 이 깊이를 제한하는 것을 가지치기(pruning)라고 하는데, 사전에 미리 모델을 돌릴 때 정해줄 수 있다.



이전에 설명하였듯이 결정 트리는 불순도를 최소화하는 방식으로 노드를 나눈다. 예를 들어 심장병에 대한 데이터를 트리 알고리즘으로 분류하고자 하는데, 예를 들어 현재 상황이 아래와

같다고 하자. “가슴 통증(Chest Pain)”이 있는 경우, 심장병을 가진 환자가 105 명, 그렇지 않은 환자가 39 명이었고 가슴 통증이 없다고 한 경우에는 심장병을 가진 환자가 34 명, 그렇지 않은 환자가 125 명이었다는 내용이 아래 도식에 그려져 있다.



이때 좌측 노드의 지니 계수는 아래와 같이 구할 수 있다.

$$1 - \left(\left(\frac{105}{105+39}\right)^2 + \left(\frac{39}{105+39}\right)^2\right) \sim 0.3949$$

같은 방법으로 우측 노드의 지니 계수는 아래와 같이 구할 수 있다.

$$1 - \left(\left(\frac{34}{34+125}\right)^2 + \left(\frac{125}{34+125}\right)^2\right) \sim 0.3362$$

트리 노드는 여러 질문 자체의 지니 계수를 비교하고 가장 낮은 지니 계수를 보이는 질문을 먼저 물어보아 정보 이득을 최대한 높여야 하므로, 해당 질문을 구성하는 두 노드 각각의 지니 계수를 가중평균을 구하여 다시 구한다. 좌측 노드의 총 개수는 105 + 39 = 144 개, 우측 노드의 총 개수는 34 + 125 = 159 개이다. 따라서 저 노드의 지니 계수는 아래와 같다.

$$\left(\frac{144}{144+159}\right) * 0.3949 + \left(\frac{159}{144+159}\right) * 0.3362 \sim 0.36409$$

위와 같은 과정을 다른 질문에도 반복한다. 예를 들어 다른 어떤 질문을 설정하고 지니 계수를 계산하였는데 0.33 이 나왔다고 해 보자. 그러면 이전 질문의 지니 계수 0.36409 보다 낮으므로 더 좋은 질문이라 할 수 있다. 이 경우 결정 트리는 지니 계수가 0.33 인 보다 더 “우수한” 질문을 먼저 물어보게 된다. (그래야 더 빠르게 불순도를 낮출 수 있기 때문이다.)

같은 방식을 수치형 변수를 기반으로 구분해야 할 때도 적용할 수 있다. 예를 들어 심장병 환자 데이터를 분석하는데, 아래 표와 같은 데이터가 있다고 해 보자. 체중(weight)과 그 체중을 가진 환자가 실제 심장병을 가지고 있었는지에 대한 여부이다.

①

Weight	Heart Disease
220	Yes
180	Yes
225	Yes
190	No
155	No

이 경우 아래와 같이 오름차순으로 정렬한 다음, 범주형 변수로 변환하기 위해 평균을 낸다. 그리고 위에서 설명한 대로 지니 계수를 계산한다. 이때 범주를 어떻게 나누는지에 따라 지니 계수가 달라지므로 여러 번 수행하여 가장 지니 계수가 낮은(정보 이득이 큰) 질문을 선택하여 물어본다.

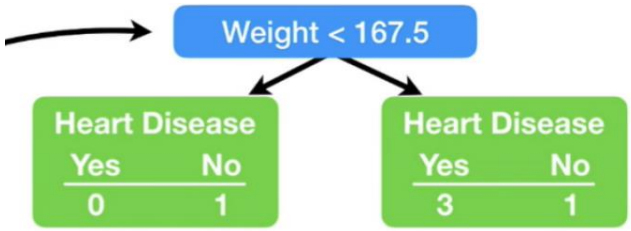
②

	Weight	Heart Disease
Lowest	155	No
	180	Yes
	190	No
	220	Yes
Highest	225	Yes

③

Weight	Heart Disease
155	No
167.5	
180	Yes
185	
190	No
205	
220	Yes
222.5	
225	Yes

예를 들어 아래와 같이 체중이 167.5(범주형 변수로 변환됨)임을 물어봐서 심장병 여부를 판단하고자 한다고 해 보자. 그럴 경우, 아래와 같은 형태의 트리가 만들어지게 된다.



이 경우 지니 불순도를 구해보면 아래와 같다.

$$G(left) = 1 - \left(\left(\frac{1}{1+0}\right)^2 + \left(\frac{0}{1+0}\right)^2\right) = 0$$

$$(right) = 1 - \left(\left(\frac{3}{3+1}\right)^2 + \left(\frac{1}{3+1}\right)^2\right) = 0.375$$

$$G(question) = \left(\frac{1}{1+4}\right) * G(left) + \left(\frac{4}{1+4}\right) * G(right) = 0.3$$

따라서 위의 경우, “weight < 167.5”로 나누어지는 경우 지니 불순도는 0.3 이다. 실제로 반복 계산을 해 보면 “weight < 205”로 나눌 때 지니 불순도가 0.27 이 된다. 그렇다면 트리 알고리즘은 이것을 고르게 된다.

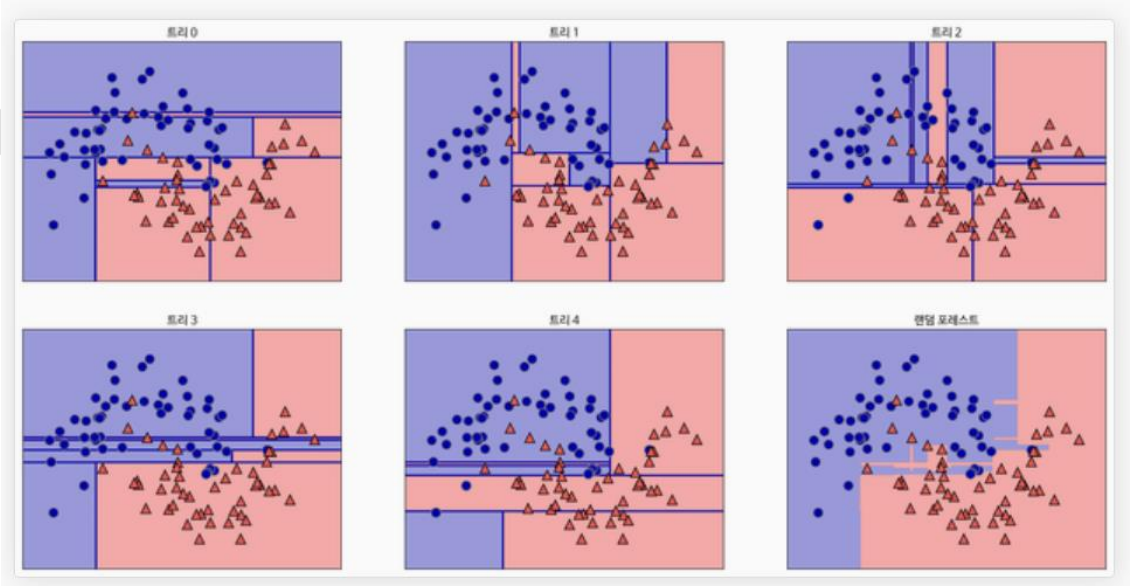
5. 트리의 앙상블 그리고 랜덤 포레스트(ENSEMBLE OF TREE AND RANDOM FOREST)

랜덤 포레스트(random forest)란 결정 트리(decision tree)를 확장시킨 개념으로써, 트리의 앙상블(ensemble)이라고도 한다. 본디 앙상블(ensemble)이란 합주단 또는 집단이나 총체와 같은 의미로, 같은 목적을 공유하는 비슷한 개체들이 한데 모인 연합을 의미한다.

기존의 결정 트리의 경우, 결정의 기본 요소가 되는 데이터들의 특성(feature)의 개수가 많아지면 복잡도가 크게 증가할 뿐만 아니라 과적합이 매우 쉽게 발생할 수 있다는 문제가 발생한다. 이에 수많은 특성 중 몇 개를 임의로 뽑고 이것으로 (작은) 결정 트리를 만들어 결과를 확인하는 프로세스를 여러 번 반복하여 큰 추세를 파악하거나, 결과들을 집계하여 유용한 항목들을 추출해 볼 수 있다. 이렇게 여러 번의 시도로부터 나온 다양한 트리로부터 결과를 집계하여 그 결과를 통합하여 보다 더 좋은 결과를 도출하는 것을 머신러닝에서는 앙상블(ensemble)이라고 한다.

랜덤 포레스트는 이때 여러 개의 특성을 받으면 무작위로 뽑힌 몇 개의 특성들(총 특성이 n개라면 보통 트리당 $log_2 n$개의 특성을 사용한다.)을 각각의 작은 트리들에게 할당하고 이를 가동하는데, 이때 프리스트에 소속된 트리의 수가 많아진다면 이를 통해 모든 특성들을 고려해 볼 수 있다.

아래의 도식은 이런 랜덤 포레스트의 작동 방식을 시각화하여 보여주고 있다.

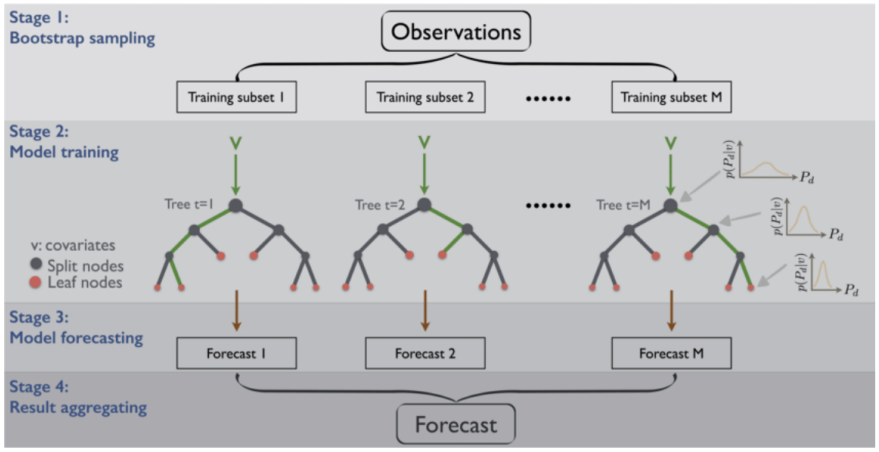


위 랜덤 포레스트에는 0 번 트리부터 4 번 트리까지 있다. 각각의 트리 자체는 상당히 복잡할 뿐만 아니라(과적합이 되어있다고 할 수 있다), 데이터를 제대로 분류하고 있다고 하기도 어렵다. 그러나

이 트리들을 모두 합쳐(앙상블) 랜덤 포레스트라는 하나의 결과로 도출하게 하면 훨씬 정확도가 높고 경계가 깔끔하게 나누어져 있는 것을 확인할 수 있다.

랜덤포레스트에서 중요한 개념으로 부트스트랩(bootstrap) 또는 부트스트랩 방법(bootstrap method)이 있다. 포레스트를 이루는 트리를 훈련하기 위해서는 데이터가 필요한데, 이 데이터는 입력값으로 주어진 훈련 데이터에서 랜덤하게 샘플을 추출하여 훈련 데이터로써 사용한다(이를 부트스트랩 샘플(bootstrap sample)이라고 한다.). 이때 중복을 고려하지 않기 때문에 샘플들이 중복되어 나올 수 있으며, 그러므로 한 번도 훈련 데이터로써 사용되지 않고 남아있는 데이터가 존재할 수 있다. 여기서는 training sets 가 부트스트랩 샘플, test sets 가 남은 데이터이다.

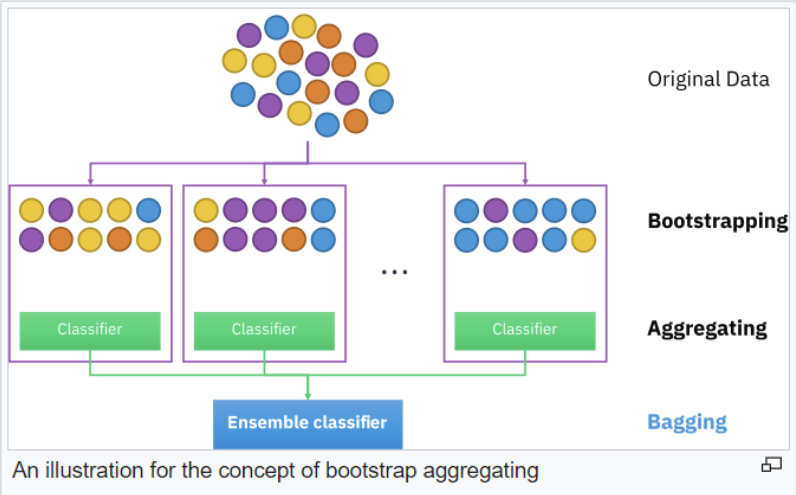
그 이후 훈련 데이터로 랜덤 포레스트를 훈련하고 결과를 취합한다. 즉 똑똑한 한 개인보다 다른 수많은 사람들의 지혜의 총합이 더 우세하다는 설계 아이디어를 볼 수 있다. 랜덤 포레스트의 경우 분류(classification) 문제일때는 작은 결정 트리가 예측한 값 중 가장 많이 나온 값을, 회귀(regression) 문제일때는 트리들이 예측한 값의 평균을 구한다. 이런 일련의 과정을 bagging(bootstrap aggregating)이라고 한다.



이전에 설명했듯이 부트스트랩 샘플링 과정은 중복을 상관하지 않으므로 실제 훈련 데이터로써 선택받지 못한 데이터가 존재한다. 이러한 데이터들을 OOB(Out-Of-Bag)이라고 한다. 이때 부트스트랩 크기가 n이라고 하면, 그 중 하나의 샘플이 훈련 세트에 포함되지 않을 확률은 $\frac{n-1}{n}$이 된다. 따라서 충분히 큰 n을 두게 되면 어떤 샘플이 추출되지 않을 확률, 즉 전체 입력 데이터에서 OOB 로 빠지는 데이터의 비율은 대략 36.8%임을 알 수 있다.

$$OOB\ Ratio = \lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n} = e^{-1} \sim 0.368$$

이 OOB 데이터들은 실제 훈련에 사용되지 않았으므로 검증 세트로서 활용될 수 있다. 지금까지 설명한 내용을 요약하면 아래와 같다.



An illustration for the concept of bootstrap aggregating

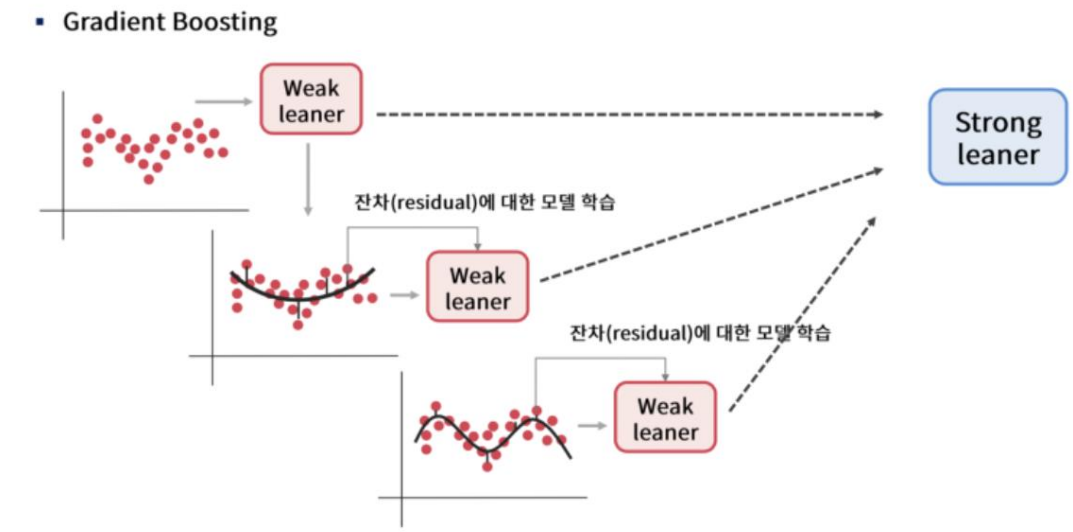
또한 랜덤 포레스트는 특성 중요도(gini importance)라는 통계를 제공한다. 이 중요도는 이전 절에서 이미 설명했던 (트리에 소속된 노드들의) 지니 불순도(gini impurity)를 기반으로 계산하는데, 불순도가 크게 감소하는 노드일수록 그 노드에 사용된 특성이 더욱 예측에 큰 기여를 했다는 점을 이용한다.

이처럼 랜덤 포레스트는 부트스트랩을 사용하여 결정 트리를 보완하였다면, 엑스트라 트리(extra tree) 라는 것도 존재한다. 엑스트라 트리는 부트스트랩 샘플을 사용하지 않고 전체 훈련 세트를 사용하고, 노드를 분할(분기)할 때 무작위로 분할하게 된다. 특성을 고려하지 않으므로 성능은 낮아지지만 많은 트리가 하나의 앙상블로서 사용되기 때문에 과대적합이 억제되고 검증 과정에서의 모델 성능이 높아진다는 장점이 있다. 또한 위에서 서술한 랜덤 포레스트는 결국 결정 트리의 집합이므로 이전에 설명한대로 지니 불순도 등을 계산하는 절차를 거쳐 결정 트리가 분할을

찾는 과정을 수없이 반복해 시간과 자원이 많이 소모되지만 엑스트라 트리는 그런 것을 고려하지 않으므로 훨씬 속도가 빠르다는 점도 있다.

결정 트리를 앙상블 하는 방법으로, 그레디언트 부스팅(gradient boosting)이라는 방법도 존재한다. 일종의 feedback 을 통해 계속해서 발전하는 형태로, 이전 학습에서 나온 오차를 다음 학습 때 적용하는 구조를 계속 반복하여 성능을 향상시킨다. 경사 하강법을 사용하여 트리를 앙상블에 소속시키며, 분류(classification)에서는 로지스틱 손실 함수(logistic loss function)를, 회귀에서는 평균 제곱 오차 함수(mean squared error function)을 사용한다.

Gradient Boosting

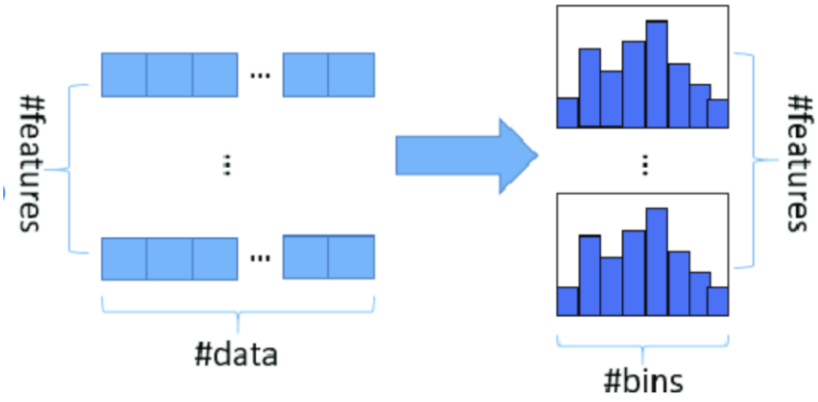


이전의 손실을 보완하고 다음에 반영하는 것으로써 순차적인 면이 있고, 랜덤 포레스트 방법보다 매개변수에 보다 더 민감하지만 잘만 조정한다면 더욱 높은 성능을 낼 수 있다. 이때 그레디언트 부스팅은 조금씩 손실함수의 극솟값으로 이동하는 경사하강법(gradient descent)을 사용하므로, 작은 결정 트리를 여러 번 이용한다. 이러한 작은(깊이가 얕은) 트리를 약한 학습기(weak learner)라고 하며, 각각의 트리는 그래서 일부 항목에 대한 예측만 더 잘 할 수 있고 이런 트리가 많아질수록 성능이 개선된다.

그레디언트 부스팅에서는 이전 트리에서 발생한 오차를 얼마나 강하게(rigid) 보정할지를 제어하는 learning_rate 와 앙상블에 얼마나 더 많은 트리를 추가할지 결정하는 n_estimators 항목이 있다. 이 둘을 키우면 각각 오차 보정이 강하게 이루어지고 훈련 과정에서 실수를 더 많이 바로잡을 수 있게 되지만 전반적인 모델의 크기가 커지고 복잡도가 증가한다는 단점이 있다. XGBoost, CatBoost, LightGBM 등이 이러한 계열의 부스팅 알고리즘으로 널리 알려져 있다.

히스토그램 기반 그레디언트 부스팅(Histogram-based gradient boosting)은 정형 데이터를 다루는 머신러닝 알고리즘 중 가장 높은 인기를 가지고 있는 알고리즘으로, 기존의 그레디언트 부스팅의 속도와 성능적인 부분을 더욱 개선한 것이다.

데이터의 특성을 이해하기 위해 우선 데이터를 히스토그램으로 변환하는데 이때 256 개의 구간(bin)으로 나눈다. 이런 방식을 통해 많은 양의 데이터가 들어와도 메모리를 적게 차지하고, 그래서 속도가 더욱 빠르다.



그리고 히스토그램의 정의 답게, 각 구간에 속하는 데이터 포인트의 개수를 기록한다. 처음에는 한 개의 트리에 모든 데이터를 다 집어넣고 시작하며, 이후 그레디언트 부스팅을 사용하여 새로운 트리를 추가하고, 히스토그램을 이용해 데이터의 (구간별) 분포를 고려해 각 구간의 예측 값을 조정해나가는 방식으로 트리를 구성한다. 트리가 다 만들어지면 실제 값과의 차이 즉 오류를 개선해 다음 트리를 만들고 이를 반복하여 성능을 높인다.

이렇게 히스토그램 기반 그레디언트 부스팅을 활용하면 입력 데이터가 나누어진 256 개의 구간으로 축약되기 때문에 과적합을 완화하고, 범주형 변수를 쉽게 처리할 수 있다.