Decision Tree를 이용한 Machine Learning
(Machine Learning by Decision Tree Algorithm)

정원찬*최경수**김정호***
(W.C.Jung, K.S.Choi, J.H.Kim)

필요한 자료의 제공만으로 컴퓨터 스스로 논리 체계를 세워 나가는 Machine Learning은 인공지능의 한 분야로서 여러 방면에서 활발한 연구가 진행되고 있다. 본 고에서는 Machine Learning의 기본적인 여러가지 방식 중의 하나인 Decision Tree 방법을 소개하고 문제점 및 연구 방향을 서술한다.

Ⅰ. 서 론

컴퓨터는 인간이 하는 일의 많은 부분을 대신하여 왔으며 그 영역은 점점 더 다양화되고 전문화되어 가고 있다. 궁극적으로 컴퓨터가 인간의 역할을 대신할 수 있을는 지에 대해서는 아직 판단하기가 쉽지 않으나 인공지능 (Artificial Intelligence) 분야의 과학자들은 여러가지 방법을 동원하여 컴퓨터로 하여금 인간처럼 판단할 수 있도록 하는 연구를 계속하고 있다. 여기서 소개하고자 하는 Machine Learning이라고 하는 분야는 인간이 성장하며 논리체계를 배워 나가는 방법을 도입하여 알고리즘화한 것으로 단순히 데이터만을 제공함으로써 컴퓨터로 하여금 나름대로의 논리체계를 세울 수 있도록 한다.

*위성망감시제어연구실 선임연구원
**위성통신기술연구단 사업개발실 선임연구원
***RF시스템연구실 책임연구원, 실장

Ⅱ. Machine Learning

인간이 유아기에 받는 교육은 학교에서 가르침을 받는 것과는 많은 차이가 있다. 처음 사물의 이름을 배울 때 아기는 그 물건에 대한 설명을 이해하는 능력이 없기 때문에 어머니로부터 '이것은 의자, 저것은 책상…' 하는 식으로 사물의 형태와 이름만을 배울 뿐이다. 아기는 왜 이것은 의자이고 저것은 책상인지 알지 못한 채 여러가지 형태의 의자와 책상의 모양만을 기억하여 나름대로 하나의 논리 체계를 갖추어 의자와 책상에 대한 정의를 내리게 된다.

의자의 경우를 예로 들면, 아기는 여러가지 형태의 의자를 보면서 의자란 다리가 있고 그 위에 판이 있고 그 위에 사람이 앉을 수 있다는 식으로 나름대로의 논리 체계를 갖추게 된다. 이와 같은 논리 체계를 갖추게 된 후 어떤 물체를 보면 그것의 다리가 있는지 판이 있는지 등등을 조사하여 그것이 의자인지 아닌지를 판단할 수 있게 된다. 물론 충분한 의자의 예를 보지 못하여 나름대로 세운 논리 체계가 잘못될 수도 있게 된다. 즉, 등받이가 있는 의자만 본 아기의 경우 등받이가 없는 것은 의자가 아니라고 잘못 판단할 수도 있는 것이다. 그러나 이같은 문제는 많은 수의 의자를 경험하게 되면 자연히 고쳐질 수 있다.

이와 같이 관찰을 통하여 논리 체계를 확립하는 유아들의 학습방법을 컴퓨터에 도입하여 알고리즘화 한 것이 Machine Learning 이다. 이 방법은 많은 데이터를 입력하여 -아기에게 많은 종류의 의자를 보여주듯이- 컴퓨터 스스로 그 데이터를 분석하여 논리체계를 만들 수 있도록 하는 것이다. 여기서 사용되는 data set은 여러 개의 attribute와 한 개의 classification value (class)로 구성되며 class 값은 각 attribute의 값의 조합에 따라 결정된다. 의자의 예를 적용하면 '다리가 있는가?' '판이 있는가?' 등의 사항은 attribute가 되고 그것이 의자인가 아닌가 하는 것은 class가 될 것이다. 이와 같은 data set을 분석하여 각 attribute 들의 값과 class 값의 관계를 찾아내는 알고리즘을 개발하는 것이 Machine Learning에서 요구되는 task이다. 이 Machine Learning 분야에도 여러가지 방법이 있는데 여기서는 논리 체계를 tree 모양으로 구성하는 방법(Decision Tree 알고리즘)을 알아보기로 한다.

Ⅲ. Decision Tree

Decision Tree 알고리즘이 취하는 data set는 〈표 1〉과 같은 형태를 가지고 있다. 여기에 소개된 data set는 a_1, a_2, a_3의 세 attribute와 classification value C로 이루어져 있으며, a_1, a_2, a_3 및 C는 모두 0 또는 1의 값을 취할 수 있다. 또 〈표 1〉의 data set은 8가지 object로 구성되어 있고 a_1, a_2, 및 a_3 가 취할 수 있는 모든 값의 조합을 포함하고 있다. 모든 형태의 의자를 본 아기라면 '의자란 어떤 것이다'라는 논리체계를 뚜렷이 가질 수 있는 것과 마찬가지로 이렇게 각 attribute가 취할 수 있는 모든 경우를 data set이 포함하고 있을 때에는 완벽한 decision tree가 구성될 수 있다. 여기서는 decision tree의 구성을 설명하는 것이 중요하므로 이와 같은 예를 택하였다.

〈표 1〉에 나타난 Data Set을 자세히 살펴보면, C의 값은 a_3의 값이 1이고, 동시에 a_1이나 a_2의 값이

〈표 1〉 Data Set

a_1	a_2	a_3	C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

1인 경우에만 1이 되며 그 외에는 모두 0이 된다. 이와 같은 조건에 따라 C의 값을 정하려면 다음과 같은 순서를 거쳐야 한다 :

i) a_3의 값이 0이면 C의 값은 0

a_3의 값이 1이면 (ii)항의 조건 check

ii) a_1의 값이 1이면 C의 값은 1

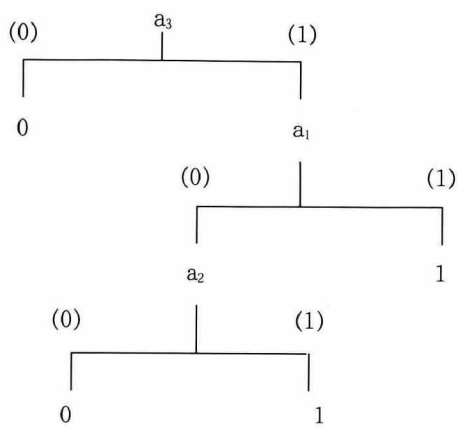
a_1의 값이 0이면 (iii)항의 조건 check

iii) a_2의 값이 1이면 C의 값은 1

a_2의 값이 0이면 C의 값은 0.

Decision Tree 알고리즘은 이와 같은 논리체계를 찾아내어 (그림 1)과 같은 Decision Tree를 구축하게 된다.

이 Decision Tree의 각 node는 그 node가 inner node일 경우 하나의 attribute를 가지고 있으며, 그 attribute의 값에 따라 branch를 형성하고 그 node가 leaf일 경우에는 class 값을 가진다. Inner node에서 파생된 branch 위에 괄호 속의 값은 그 inner node가 취하는 attribute의 값에 해당한다. 따라서 a_1, a_2, a_3의 값을 알면 (그림 1)과 같은 Decision Tree를 통하여 C의 값을 추정해 낼 수 있다.



(그림 1) Data Set 1에 대한 Decision Tree

Ⅳ. Decision Tree의 구성

이와 같은 Decision Tree는 다음과 같은 알고리즘[1]을 통하여 만들어진다.

■ 정의

D를 전체 data set을 구성하는 각 data d_i의 집합이라 한다.

A를 attribute a_i의 집합이라 한다.

■ 초기화

D를 전체 data set으로, A를 data set의 모든 attribute의 집합으로 한다.

■ 알고리즘

i) 알고리즘 최초의 시작은 초기화 값이 설정된 D와 A를 root에서부터 적용한다.

ii) D와 A를 parameter로 받아 D를 검토하여 C의 값이 모두 같으면 그 C의 값을 현재의 node에 기입하고 return 한다.

C의 값이 모두 같지 않으면 (iii)으로 진행한다.

iii) D를 검토하여 C에 가장 크게 영향을 미치는 attribute a_i를 A로부터 택하여 a_i를 현재의 node에 기록한다. (영향을 미치는 정도는 Quinlan의 방법으로 계산하며 그 내용은 V장에서 다루기로 한다.)

iv) a_i가 취할 수 있는 모든 값 v_1, v_2 v_3 …… v_j …… v_m에 대하여 D'와 A'을 다음과 같이 계산하여 (ii)항으로 parameter passing 하여 recursive 하게 진행한다.

D'=set of all $d_k \in D$ such that $a_i = v_j$

$A' = A - \{a_i\}$

V. Quinlan's Method

Ⅳ장의 Decision Tree 알고리즘 (iii)항에서 잠시 소개된 Quinlan의 방법을 소개하기 전에 가장 크게 영향을 미치는 attribute를 찾는 것이 왜 중요한지를 먼저 살펴보기로 한다.

(그림 1)의 Decision Tree는 4 개의 leaf를 가지고 있는데, 따라서 4가지 조건을 포함하고 있다.

i) $a_3=0$ 이면 C = 0

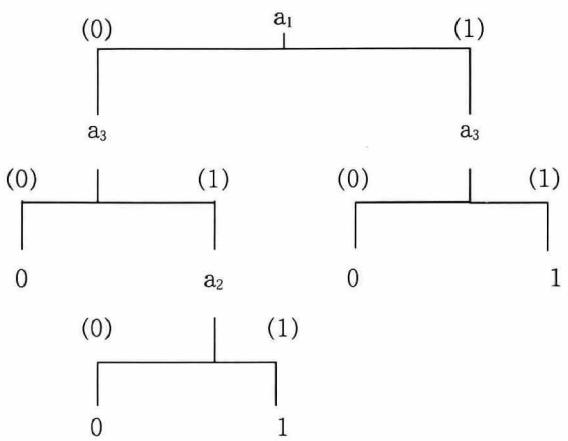
ii) $a_3=1$ 이고 $a_1=1$ 이면 C = 1

iii) $a_3=1$, $a_1=0$ 및 $a_2=0$ 이면 C = 0

iv) $a_3=1$, $a_1=0$ 및 $a_2=1$ 이면 C = 1

이 Tree에서는 a_3를 root에서 check 하고 그 이후 a_1과 a_2를 check 한다. 이는 세 개의 attribute 중 a_3의 C에 미치는 영향이 가장 크다고 판단되었기 때문이다. 만일 Root에서 a_3 대신 a_1이 check 되었다고 하면 (그림 2)와 같은 Tree가 만들어지게 된다.

(그림 2)의 Decision Tree는 5개의 leaf를 가지며, 따라서 5가지 조건을 포함하고 있다.



(그림 2) 〈표 1〉 에 대한 decision tree

i) $a_1=0$ 이고 $a_3=0$ 이면 C = 0

ii) $a_1=0$, $a_3=1$ 및 $a_2=0$ 이면 C = 0

iii) $a_1=0$, $a_3=1$ 및 $a_2=1$ 이면 C = 1

iv) $a_1=1$, $a_3=0$ 이면 C = 0

v) $a_1=1$, $a_3=1$ 이면 C = 1

이 중 (i)과 (iv)의 조건은 "$a_3=0$ 이면 C=0"이라는 하나의 조건으로 나타낼 수 있으므로 (그림 2)의 Decision Tree는 Optimal Tree가 되지 못한다. 따라서 어떤 level에서 C에 가장 큰 영향을 미치는 attribute를 찾아내는 것은 Optimal Tree를 만드는데 아주 중요한 일이다.

가장 중요한 attribute를 찾는 알고리즘은 Quinlan의 방법[1]이 널리 쓰이고 있다. 이는 주어진 data set과 attribute set를 이용하여 각 attribute의 값에 따라 class 값이 정해질 수 있는 확률을 계산한 것이다. Quinlan의 방법을 〈표 1〉의 data set에 대략적으로 적용하여 a_1, a_2, a_3의 C에 대한 영향도를 계산하면 다음과 같다.

$$a_1: P(a_1=0)\times P(C=0 \mid a_1=0)+P(a_1=1)\times P(C=1 \mid a_1=1)=0.625$$

$$a_2: P(a_2=0)\times P(C=0 \mid a_2=0)+P(a_2=1)\times P(C=1 \mid a_2=1)=0.625$$

$$a_3: P(a_3=0)\times P(C=0 \mid a_3=0)+P(a_3=1)\times P(C=1 \mid a_3=1)=0.875$$

위에서 계산한 바에 따르면 a_1이 C 값에 영향을 미칠 확률은 0.625, a_2가 영향을 미칠 확률은 0.625, a_3가 영향을 미칠 확률은 0.875 로 각각 나타난다. 따라서 확률이 가장 높은 a_3를 제일 먼저 check 하게 되는 것이다.

Quinlan의 방법은 classification value에 가장 큰 영향을 미칠 attribute를 미리 구하는 것인데 유감

스럽게도 어떠한 경우에나 모두 적용될 수 있는 것은 아니다.

Ⅵ. F-family data set

Machine Learning 분야에서 가장 널리 쓰이는 data set은 F-family라고 하는 data set이다. F-family는 F_2, F_6, F_{11}, F_{20}, …… 등 data set들의 포괄적인 이름이다. F_n data set에서의 n은 attribute의 수를 의미하는데 각 attribute들과 class C는 0 또는 1의 값을 취할 수 있다. 여기서 n은 어떤 자연수 m에 대하여 $m+2^m$의 식으로 표현될 수 있어야 하며, 이 attribute들 중 처음 m개의 attribute는 address bit, 나머지 2^m개의 attribute는 data bit으로 불린다. C의 값을 정하는 방법(decision tree 알고리즘이 찾아낼)은 처음 m bit가 정하는 address를 찾아 그 address가 지정하는 data bit의 값을 읽으면 된다. 예를 들어 F_6의 경우를 살펴보면 a_1와 a_2는 address bit이 되고, a_3, a_4, a_5, 및 a_6는 data bit이 되며 C의 값은 다음과 같은 방법으로 결정된다.

if $a_1=0$ and $a_2=0$ then $C=a_3$

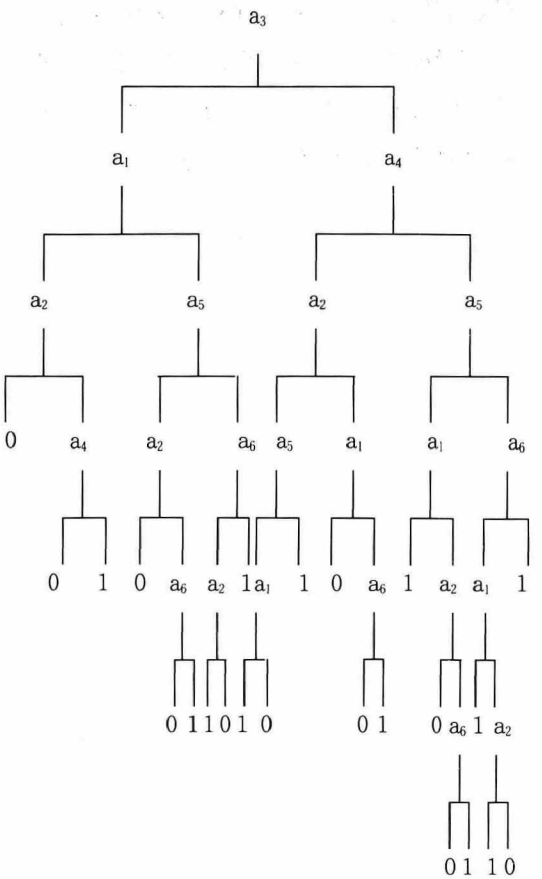
if $a_1=0$ and $a_2=1$ then $C=a_4$

if $a_1=1$ and $a_2=0$ then $C=a_5$

if $a_1=1$ and $a_2=1$ then $C=a_6$

$n=m+2^m$이 되는 F_n data set에서 C의 값을 정하려면 모두 m+1개의 attribute 값만을 찾아보면 간단히 구할 수 있으나 그 방법을 알지 못할 경우에는 그다지 쉽게 C의 값을 찾을 수 없다. 이 때문에 F-family는 Machine Learning 분야에서 가장 널리 사용되고 있는 data set이다.

재미있는 것은 이 F-family data set에 Quinlan의 방법을 적용할 경우 절대 optimal tree가 만들어지지 않는다는 점이다. F-family data set에서의 optimal tree를 구하려면 먼저 address bit들을 check 하고, 그 address bit들이 지정하는 data bit 한 개만 check 하여야 한다. 그러나 Quinlan의 방법을 적용할 경우 root에서는 언제나 address bit보다는 data bit의 Classification Value에 대한 영향도가 높다. 왜냐하면 address bit는 data bit를 지정하는 역할을 하지만 C 값에는 직접적인 영향을 미치지 못하기 때문에 address bit에 의해 C 값이 결정될 확률은

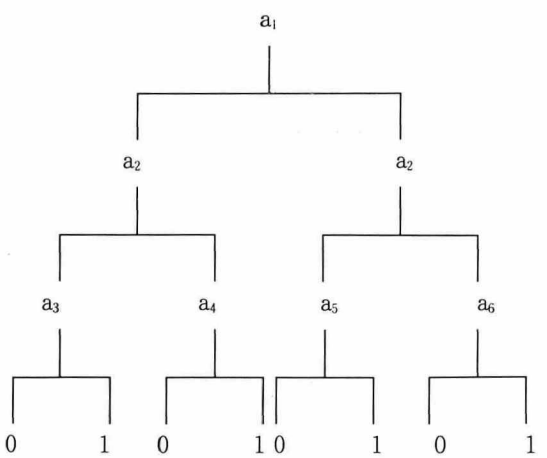


(그림 3) Quinlan의 방법을 사용한 F_6의 decision tree

50%인 반면, data bit가 C 값에 영향을 미치는 확률은 address bit이 그 data bit을 지정할 경우 100%, 그렇지 않을 경우 50%로서 평균으로 계산하여 50%를 상회한다. 따라서 Quinlan의 방법을 F-family에 적용할 경우 언제나 data bit 중의 하나가 root에 오게 되고 따라서 optimal tree가 되지 못한다.

F_6 data set에 Quinlan의 방법을 적용하여 decision tree를 만들어 본 것이 (그림 3)과 같은 모양의 decision tree이다.

(그림 3)에 나타난 바와 같이 Quinlan의 방법을 사용하여 만든 decision tree는 모두 45개의 node를 포함하고 있다. 상식적으로 생각할 수 있는 F6의 decision tree는 (그림 4)에 나타난 바와 같이 단 15개의 node로 표현될 수 있다.



(그림 4) F6의 optimal decision tree

VII. Decision Tree의 특징 및 연구방향

Machine Learing 방법에는 본 고에 소개된 Decision Tree 방법이외에도 여러가지 방법이 있다. Genetic Algorithm과 Neural network 방법이 decision tree 방법과 함께 Machine Learning의 대표적인 방법으로 손꼽힌다. 이러한 방법들에 비해 decision tree가 가지는 장점은 그 방법의 단순성에 있다. 간단한 test set만 주어지면 그 test set을 이용하여 논리체계를 쉽게 구성할 수 있고, 그 구성 알고리즘이 복잡하지 않으며 또한 tree 구성에 필요한 computation의 양이 많지 않다. 또한 구성된 논리체계를 user가 쉽게 이해할 수 있다는 점도 Decision Tree 방법이 지니는 장점이다. 그러나 모든 논리체계를 tree 형식으로 구성해야 하기 때문에 learning을 할 수 있는 분야가 한정되어 있다는 단점도 있다.

Decision Tree 알고리즘의 생명은 중요한 attribute가 다른 attribute보다 먼저 check 되어야 한다는 점이다. 만일 만들어진 decision tree가 optimal tree가 되지 못하는 경우 이 만들어진 tree의 node 수를 줄여 optimal tree가 되도록 하는 것이 optimization이다. 이 optimization의 방법에는 여러가지 알고리즘이 소개되었고 현재에도 소개되고 있다. 대표적인 방법으로는 topology를 사용한 방법[3]과 look ahead & top-down search를 사용한 방법[4] 등이 있으며, 각 방법마다 장단점이 있어 계속적인 연구가 요구되고 있다.

VIII. 결 론

컴퓨터로 하여금 인간의 두뇌와 같은 역할을 담당하게 하려는 노력은 컴퓨터가 발명된 이래 계속

되어 왔다. 이와 같은 노력의 일환으로 인공지능의 여러 분야가 소개되었고, 그 중 몇 분야는 한계에 부딪치고 다른 분야는 활발한 연구가 계속되고 있다. Machine learning은 인간의 유아기적 학습방식을 도입한 분야로서 당장의 효과보다는 수많은 데이터를 분석하게 되면 그 데이터에 관한 한 전문가에 준하는 수준에 도달할 수 있다는 점에서 관심의 대상이 되고 있다. 그 중 한 방법인 Decision Tree를 이용한 learning 방식은 주어진 data를 분석하여 나름대로의 논리체계를 tree 형태로 갖추는 방식으로서 활발한 연구가 진행되고 있다. 이 Decision Tree를 만들기 위해서는 가장 중요한 attribute를 찾아내는 것이 중요하며 여기에는 Quinlan [1]의 방법이 사용되고 있다. Quinlan의 방법으로 optimal decision tree를 구성하지 못할 경우를 위하여 여러 가지 optimization에 대한 연구가 활발히 진행 중이다.

참 고 문 헌

[1] Quinlan, J.R., "Learning Efficient Classification Procedures and Their Application to Chess End Games", In Michalski,R.S., Carbonell,J., and Mitchell,T.M.(eds), Machine Learning : An Artificial Intelligence Approach, Vol. I, Palo Alto, Tioga Press, pp.463－482.

[2] Quinlan, J.R., "An Empirical Comparison of Genetic and Decision-Tree Classifiers," Proceedings of the 5th International Conference on Machine learning, San Mateo, CA, 1988, pp.135－141.

[3] Van de Velde, W. "Incremental Induction of Topologically Minimal Trees," Proceedings of the 7th Internatinal Conference on Machine Learning, Austin, TX, 1990, pp.66－74.

[4] Won Chan Jung, J. Bush Jones, and Jianhua Chen, "Optimization of the Decision Tree," Proceedings of the 3rd International Conference on Tools for Artificial Intelligence, San Jose, CA, pp.522－523, 1991.