JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 18, No. 6 : 1275~1282 Jun. 2014

중요한 이벤트만을 검색함으로써 분류기의 최적 성능을 찾는 방법

김동희[1] · 이원돈[2*]

A method of searching the optimum performance of a classifier by testing only the significant events

Dong-Hui Kim[1] · Won Don Lee[2*]

[1]Department of Computer Science and Engineering, Chungnam National University, Daejeon 305-764, Korea
[2*]Department of Computer Science and Engineering, Chungnam National University, Daejeon 305-764, Korea

요 약

유비쿼터스 환경에서는 수많은 정보들이 존재한다. 하지만 이 정보들은 너무 광범위하기 때문에 이로부터 필요에 따라 적절하게 사용 할 수 있는 정보를 얻기란 쉽지가 않다. 이로 인해 의사 결정 트리 알고리즘은 데이터 마이닝 분야 또는 기계 학습 시스템 분야에서 매우 유용하게 사용된다. 왜냐하면 빠르고 정확하게 정보를 분류하여 좋은 결과를 도출하기 때문이다. 하지만 때때로 의사 결정 트리가 매우 작은 데이터나 노이즈 데이터로 구성된 리프 노드들로 인해 좋은 정보를 제공하지 못하는 경우가 있다. 이 논문은 이러한 분류 문제를 해결하기 위해 분류기, UChoo를 사용할 것이고 노이즈 또는 노이즈 형태로 보이는 리프들을 제외하고 오직 중요한 리프들만을 검사하는 효과적인 방법을 제안한다. 그리고 실험을 통하여 의사 결정시 오직 중요한 리프들만을 의사 결정 트리에서 선택함으로써 효과적으로 에러가 줄어드는 것을 보일 것이다.

ABSTRACT

Too much information exists in ubiquitous environment, and therefore it is not easy to obtain the appropriately classified information from the available data set. Decision tree algorithm is useful in the field of data mining or machine learning system, as it is fast and deduces good result on the problem of classification. Sometimes, however, a decision tree may have leaf nodes which consist of only a few or noise data. The decisions made by those weak leaves will not be effective and therefore should be excluded in the decision process. This paper proposes a method using a classifier, UChoo, for solving a classification problem, and suggests an effective method of decision process involving only the important leaves and thereby excluding the noisy leaves. The experiment shows that this method is effective and reduces the erroneous decisions and can be applied when only important decisions should be made.

키워드 : 의사 결정 트리, 분류기, 중요, 이벤트, 유비쿼터스

Key word : decision tree, classifier, significant, event, ubiquitous

접수일자 : 2014. 03. 31 심사완료일자 : 2014. 05. 12 게재확정일자 : 2014. 05. 26

*** Corresponding Author** Won Don Lee(E-Mail: wlee@cnu.ac.kr, Tel:+82-42-821-6058)
Department of Computer Science and Engineering, Chungnam National University, Deajeon 305-764, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2014.18.6.1275 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

오늘날 수많은 정보가 PDA, 테블릿pc, 노트북, 스마트폰 등과 같은 휴대 기기를 통해 만들어 지고 있다.

하지만 이를 통해 만들어지는 데이터는 매우 크기 때문에 이 속에서 중요한 정보를 선택하는 것은 쉬운 일이 아니다. 따라서 빠르고 정확하게 정보를 얻는 방법의 필요성이 증가하고 있다.

이러한 이유 때문에 기계학습, 데이터 마이닝 분야는 더욱 중요한 위치를 차지하고 있다. 특히 최대한 노이즈 데이터를 포함하지 않은 훈련 데이터로부터 분류 규칙(rule)을 만들어 내는 의사 결정 트리 알고리즘은 사람들에게 좋은 정보를 제공해 준다.

이와 같이 의사 결정 트리 알고리즘을 통해 좋은 정보 얻기 위해서는 우선 훈련 데이터가 필요하다. 하지만 예기치 않은 센서의 오작동, 데이터를 수집하는 사람의 실수, 데이터 또는 분류기의 성격과 같은 여러 요인에 의해 훈련데이터에 노이즈 데이터가 포함될 수 있고 이로 인해 의사 결정 트리는 오직 노이즈 이벤트로만 구성된 리프노드나 매우 작은 크기의 리프노드를 가질 수 있다. 이처럼 의사 결정 트리가 중요하지 않은 이벤트를 포함한 리프노드를 가지고 있을 때 사용자에게 상황에 적절치 않은 정보를 제공할 수 있다. 따라서 이 논문은 UChoo 분류기[5,6]를 사용하여 의사 결정 트리를 만들고 이 트리의 각각의 리프 노드들이 충분한 크기의 이벤트들을 가지고 있을 때 중요한 이벤트로 결정하여 이러한 리프노드만 시험하는 방법을 제안 한다.

Ⅱ. UChoo, 분류기

2.1. 확장된 데이터 표현

표 1은 가중치(Weight)라는 중요도와 함께 훈련 데이터집합(dataset)의 값이 0, 1 값뿐만 아니라 0과 1사이의 확률 값으로 입력되어 있는 확장된 데이터 표현을 보여 주고 있다.

이러한 확장된 훈련 데이터집합(dataset)은 이 분야의 전문가를 통해 만들어 질 수 있고 의사 결정 트리로부터 생성 되어질 수도 있다.

어떤 전문가가 만약 날씨가 sunny이고 바깥온도가 약 화씨 70도씨 정도일 경우 TV가 어떤 프로그램을 상영하던지 상관없이 밖에 나가서 놀 확률은 2/3이고 이에 대한 가중치 (Weight)를 30정도로 주겠다고 생각한다면 이는 표 1의 첫 번째 줄인 event#1과 같이 확장된 훈련 데이터 형태로 표현할 수 있다. 따라서 표 1과 같이 확장된 훈련 데이터 형 태에서는 일반적인 훈련데이터의 튜플(tuple) 또는 인스턴스(instance)(이하 인스턴스)에 대 한 재정의가 필요하다.

이 논문은 확장된 훈련 데이터 형태에서 각각의 열을 이벤트라고 정의한다. 표 1과 같 이 각각의 이벤트들은 가중치(Weight)를 가지 고 있다.

1의 가중치(Weight)을 가진 이벤트는 1개의 인스턴스와 동일하게 볼 수 있다. 표 1을 보 면 첫 번째 이벤트는 가중치(Weight)가 30 이기 때문에 30의 중요도를 가지고 있다. 이는 마치 30개의 인스턴스를 가지고 있는 것과 동일한 효과를 가진다. 결국 w의 가중치(Wei- ght)를 가진 이벤트는 가중치 (Weight)가 1이 고 동일한 분포도 속성들 및 클래스를 가진 w 개의 인스턴스들로 이루어진 훈련 데이터집합 (dataset)과 동일하다. 예로 표 1은 3개의 이벤트를 가지고 있다.

하지만 각 이벤트들의 가중치(Weight)의 합이 32이므로 이는 32개의 인스턴스를 가진 훈련 데이터집합(dataset)와 동일하다. 또한 확장된 데이터 형태는 각각의 속성 및 클래 스가 0 또는 1 뿐만 아니라 확률적인 값들을 가질 수 있다. 예로 표 1의 첫 번째 이벤트를 보면 TV 속성 의 Doc.와 Ani.는 각각 1/2, 클래스의 Play,

표 1. 전문가에 의한 확장된 데이터 표현
Table. 1 Expended Data Expression according To Expert

# Event	Weight	Outlook			Temp(OF)			TV		Class	
		sunny	overcast	rain	60	70	80	Doc.	Ani.	Play	Don't Play
1	30	1	0	0	0	1	0	1/2	1/2	2/3	1/3
2	1	1/3	1/3	1/3	1	0	0	1	0	1	0
3	1	0	0	1	1	0	0	1	0	0	1

Don't Play 는 각각 2/3, 1/3의 확률 값으로 주어져 있다.

2.2. UChoo: 확장된 데이터 표현 분류기

이 논문은 의사 결정 트리의 시험(test) 노드를 선택하기 위한 정보이득비율(Gain_ratio)을 구하기 위해 C4.5 알고리즘의 척도(measure) 를 사용하였다.

하지만 앞서 기술 했듯이 이 확장된 데이터 표현은 각각의 이벤트에 가중치(Weight)라는 중요도가 존재할 뿐만 아니라 각각의 속성 및 클래스가 확률 값을 가질 수 있기 때문에 의사 결정 트리를 만들어 내는데 C4.5 알고리즘과 다른 새로운 계산법이 필요하다.

다음은 시험(test) 노드를 선택하기 위한 정보이득비율(Gain_ratio)을 구하는 공식이다[1,2].

Gain_ratio(A)=Gain(A)/Split_info(A) (1)

여기서, A는 훈련 데이터의 속성을 의미 한다.

확장된 데이터 표현에 따른 새로운 정보이득 비율(Gain_ratio)을 구하기 전에 먼저 각각의 과정을 돕기 위한 용어 정의는 다음과 같다.

TS : 전체 훈련 데이터의 집합(set)

T : 의사 결정 트리 생성 시 각 노드에서 인스턴스들의 집합(set). 루트노드에서 T는 전체 훈련 데이터 집합(dataset), TS이다.

T_{Aj} : T의 서브 집합(set)이고 이것은 특정 속성 A의 outcome 값 j를 가진다.

Class : {C_1,C_2,……. C_{k-1},C_k}

여기서 k는 클래스 수이다.

특정 속성 A에 대해서

Outcome : {O_{A1},O_{A2},…… $O_{A(n-1)}$,O_{An}}

여기서 n은 특정 속성의 outcome 수이다.

Outcome 소속값

: {$O_{A1(m)}$,$O_{A2(m)}$,…… $O_{A(n-1)(m)}$,$O_{An(m)}$}

$O_{Aj}(m)$는 집합(set) T에서 m번째 이벤트의 속성 A에서 outcome j가 갖는 값이다.

여기서, j ∈{ 1 … n }, $\sum_{j=1}^{n} O_{Aj}(m) = 1$

Weight(m,T) : 집합(set) T에서 m번째 이벤트의 중요도

|T| : 집합(set) T에서 인스턴스의 수 1개의 이벤트가 중요도, W을 가지고 있다면 이는 동일한 분포의 속성과 클래스를 가진 W개의 인스턴스와 같다.

|E(T)| : 집합(set) T에서 이벤트의 수

$|E(T_{Aj})|$: 집합(set) T_{Aj}에서 이벤트의 수

$|T_{Aj}|$: 집합(set) T_{Aj}에서 인스턴스의 수

$$|T_{Aj}| = \sum_{m=1}^{|E(T_{Aj})|} Weight(m, T_{Aj}) \cdot O_{Aj}(m) \quad (2)$$

Class 소속값

:{C1(m),…,Ck(m)} i∈{1 … k}, $\sum_{i=1}^{k} C_i(m) = 1$

C_i(m)는 m번째 이벤트의 C_i클래스에 속한 정도를 나타낸다.

freq(Ci,T)

: 집합 T 안에서 클래스 Ci에 속해 있는 인스턴스들의 개수.

$$freq(C_i, T) = \sum_{m=1}^{|E(T)|} Weight(m, T) \cdot C_i(m) \quad (3)$$

freq(C_i,T_{Aj})

: 집합 T_{Aj}안에서 클래스 C_i에 속해 있는 인스턴스들의 개수.

$$freq(C_i, T_{Aj}) = \sum_{m=1}^{|E(T_{Aj})|} Weight(m, T_{Aj}) \cdot C_i(m) \cdot O_{Aj}(m) \quad (4)$$

정보이득비율(Gain_ratio)을 구하기 전 먼저 Gain(A)를 구하면 다음과 같다.

Gain(A)= info(T) – $info_A$(T) (5)

여기서, A 는 훈련 데이터의 속성이다.

$$info(T) = -\sum_{i=1}^{k} (freq(C_i, T)/|T|) \cdot \log_2(freq(C_i, T)/|T|) \quad (6)$$

$$info_A(T) = \sum_{j=1}^{n} (|T_{Aj}|/|T|) \cdot info(T_{Aj}) \quad (7)$$

$$\text{info}(T_{Aj}) = -\sum_{i=1}^{k}(freq(C_i, T_{Aj})/|T_{Aj}|) \cdot \log_2(freq(C_i, T_{Aj})/|T_{Aj}|) \quad (8)$$

여기서, k 는 클래스 수이다.

의사 결정 트리 생성 시 Gain(A)만으로도 충분히 좋은 시험(test) 노드를 뽑아 낼 수 있다. 하지만 만일 특정 기호적 속성이 매우 많은 결과(outcome)를 가지고 있을 경우 상대 적으로 작은 결과(outcome)을 가진 기호적 속성이나 2개의 하위집합(subset)로만 나누어지는 수치적 속성은 시험(test) 노드로 선택되어 질 확률이 낮아진다[1].

따라서 아래의 공식처럼 Split_info(A)를 계산하여 식 (1)과 같이 Gain(A)를 나누어줌으로써 이 문제를 해결할 수 있다.

$$\text{Split\_info}(A) = -\sum_{j=1}^{n}(|T_{Aj}|/|T|) \cdot \log_2(|T_{Aj}|/|T|) \quad (9)$$

Ⅲ. 중요 이벤트만을 검색 방법

규칙(rule)이나 규칙(rule)의 조합으로부터 더 좋은 성능을 얻기 위한 방법은 수많은 정보가 쏟아지는 요즘 점점 더 중요해지고 있다. 하지만 때때로 예기치 않은 노이즈 정보를 데이터집합(dataset)가 가지고 있을 경우 분류기는 그 성능과 상관없이 좋은 규칙(rule)을 생성하기가 어렵다.

이는 정보를 수집하는 센서에 문제가 생기거나 이를 수집하는 사람이 실수를 하는 것과 같이 여러 가지 요인으로 인해 노이즈 정보, 즉 원치 않는 데이터가 데이터집합(dataset)에 포함될 수 있기 때문이다.

이뿐 아니라 데이터집합(dataset) 그 자체가 노이즈를 포함하지 않고 완벽하더라도 의사 결정 트리가 만들어졌을 때 상위레벨의 노드에서 보면 하위레벨의 노드가 마치 노이즈 데이터로 구성되어 있는 것처럼 보일 수 있다. 이는 아무리 좋은 의사 결정 트리 알고리즘을 사용하더라도 규칙(rule)을 생성할 때 반드시 생기는 내재적 문제이다. 이러한 문제로 인해 생긴 리프들은 노이즈 데이터로 이루어져 있거나 이와 유사한 형태를 띠는 경우가 많다. 따라서 이러한 중요하지 않은 리프들은 에러율을 높이는데 큰 공헌하기 때문에 의사 결정 트리를 통해 의사를 결정 하는 과정에서 반드시 제외되어야 한다.

그림 1처럼 의사 결정 트리 알고리즘을 통해 나누어진 데이터 공간(data space)이 있을 때 분류되어진 데이터 공간(data space)을 보면 Leaf Node 2, Leaf Node 4는 Leaf Node 1, Leaf Node 3에 비해 훨씬 더 작은 이벤트들을 가지고 있는 것을 볼 수 있다.이처럼 데이터 공간(data space) 내에서 상대적으로 매우 작은 수의 이벤트들을 가지고 있는 노드들은 노이즈 이벤트들로 간주해도 문제가 되지 않을 것이다. 따라서 Leaf Node 2 와 Leaf Node 4 같은 이러한 약한 리프들이 의사 결정 과정에서 포함되어 지지 않는다면 분명히 전반적인 성능이 향상되어 질것이라는 것은 쉽게 예측 가능하다.

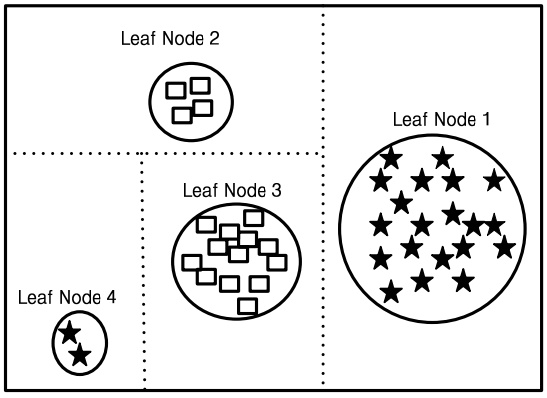

그림 1. 의사 결정 트리에 의해 분류된 데이터 스페이스
Fig. 1 Classified Data Space according to Decision Tree

의사 결정 트리의 최종 규칙(rule)을 관찰 하여 노이즈 데이터를 효과적으로 찾음으로써 이러한 데이터들을 선택하지 않고 가치 있는 데이터들만 선택하는 것이다. 즉, 의사 결정 트리를 통해 의사 결정하는 과정에서 시험(test) 데이터들을 통해 발생하는 모든 결과들을 받아들이지 않고 잘 군집 되어 있고 중요 하다고 여겨지는 노드들만을 선택하면 되 는 것이다.

그림 2는 UChoo 분류기를 사용하여 만들 어진 의사 결정 트리이다. 그리고 이 의사 결 정 트리는 7개의 리프 노드들이 존재하고 두개 의 클래스 C1과 C2가 존재한다[2,3]. 문제는 7개의 리프 노드들 중에서 노이즈 이

벤트들로 만들어진 것처럼 보이는 적은 수의 이벤트를 포함 하는 리프 노드를 어떻게 선택 하느냐는 것이 다. 특정 리프 노드가 받아 들여 져야 하는지 그렇지 않은 지를 구별하기 위한 경계점(threshold)이 찾아야 하고 이 이상의 이벤트들을 가지고 있는 리프들만을 의사 결정 과정에서 포함 시켜야 하는 것이다. 또한 이 경계점(threshold)은 일반적으로 훈련 데이터의 성격을 잘 파악할 수 있는 디자이너에 의해 결정 되어 질 수 있다.

다음은 중요 리프노드와 그렇지 않은 리프 노드 사이의 경계점(threshold)을 찾는 과정이다[4].

1) 첫 번째, 모든 리프 노드들에 대해서 이벤트 개수의 합을 계산하다. 그리고 경계점(threshold)값은 디자이너에 의해 설정될 수 있다. 모든 리프 노드들의 이벤트 총 개수를 V_{total},경계점은 F로 명시한다.
   예) 그림 2에서 전체 이벤트 개수 즉, 리프노드들의 클래스 개수 합인 V_{total}는 20이다. 단, 경계점 F는 0.1이라고 가정한다.
2) 두 번째, 경계점보다 작은 개수의 이벤트를 가진 리프 노드를 찾기 위해 V_{total}와 F를 곱한다.
   예) $V_{total*}F=20*0.1=2$
3) 세 번째, 의사 결정트리의 성능을 측정하는데 있어서 오직 $V_{total*}F$보다 크거나 같은 리프 노드들만 결과를 얻기 위한 시험(test)에 참여시킨다.
   예) 그림 2에서 이벤트 개수가 2($V_{total*}F$) 보다 작은 LN7을 제외한 나머지 리프노드들을 중요 노드로 결정한다.

위 과정을 통해서 $V_{total*}F$보다 작은 개수의 이벤트를 가진 리프 노드들은 에러 결과를 뽑 아 내기 위한 성능 통계를 낼 때 제외시킨다. 이와 같은 방법으로 의사 결정 과정에서 노 이즈 리프 노드들을 효과적으로 정제함으로써 의사 결정 트리의 성능이 증가될 것이라는 것을 기대 할 수 있다. 이처럼 이 알고리즘이 전반적으로 생각하는 점은 노이즈 데이터들은 의사 결정 트리에서 많은 이벤트를 가진 리프 노드의 형태로 존재하지 않을 것이라는 사실에 근거 하고 있다.

실세계에서 이 점을 토대로 적용한다면 어떤 데이터를 가지고 의사 결정 트리를 만들더라도 시험(test)과정에서 이러한 노이즈 리프들이 제외됨으로써 성능이 분명히 증가할 것이고 이 로 인해 좋은 정보를 제공할 수 있을 것이다.

따라서 $V_{total*}F$ 보다 작은 수의 이벤트를 가 진 리프들은 노이즈 리프로 간주해야 하고 성 능 측정 시 포함 시켜서는 안 된다.

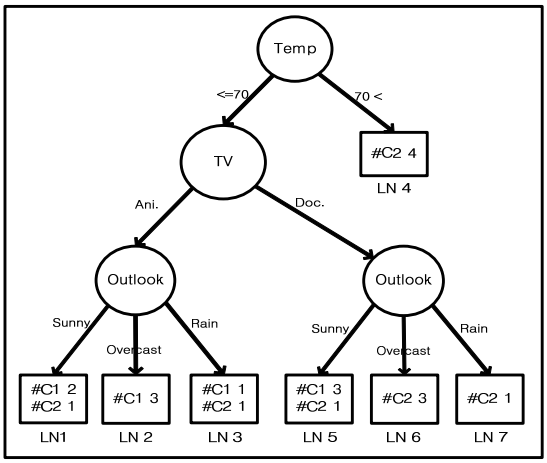

그림 2. 의사 결정 트리
Fig. 2 Decision Tree

다시 말하자면 충분히 큰 이벤트를 가지고 있는 리프들만을 의사 결정하기 위한 좋은 후보들로 여겨야 한다.

그림 2에서 보면 $V_{total*}F(20*0.1)$은 2값 을 가진다. LN7은 경계점($V_{total*}F$) 2보다 작은 오직 한 개의 이벤트를 가지고 있다. 따라서 시험(test)할 때 이 노드로 어떤 시험(test) 데이터가 들어오면 성능 측정에서 이를 제외시킨다. 결국 오직 경계점($V_{total*}F$)보다 크거나 같은 이벤트를 가진 리프 노드들 즉 그림 2 에서 리프노드 LN7을 제외한 다른 리프 노드에 들어가는 시험(test) 데이터에 한해서만 성 능 통계를 내는 것이다.

이처럼 만약 작은 개수의 이벤트들로 이루 어진 리프 노드가 노이즈 데이터라는 가정이 합리적이라면 분명 분류기의 성능은 증가할 것이라는 것을 충분히 기대해 볼 수 있다.

Ⅳ. 실험 및 결과

이 논문에서 제안한 방법의 성능을 측정하기 위해 UCI Machine Learning Repository에서 수집한 데이터

집합(dataset)을 실험을 하는데 사용하였다[7]. 그리고 표 2와 같이 속성이나 클래스에 누락된 값이 존재하지 않은 데이터 집합(dataset)를 사용 하였다.

실험 결과 값이 공신력을 갖도록 각각의 데이터집합(dataset)에 대해 10-fold cross validation을 적용하였다. 전체 훈련 데이터집합(dataset)의 90%는 의사결정 트리를 구성하는 데 사용하였고 10%는 시험(test) 데이터로 사용하였다. 의사 결정 트리를 생성하기 위해 분류기, UChoo를 사용하였고 모든 훈련 데이터의 가중치(Weight)는 동일하게 1을 부여하였다. 그리고 의사 결정 트리가 구성될 때 만일 특정 시험(test) 노드에서 가장 큰 클래스가 나머지 클래스의 합보다 90% 이상 클 때 가지를 늘리지 않고 중단하였다. 예로 특정 시험(test) 노드에서 클래스1, 클래스2, 클래스3가 있을 때 클래스1의 개수가 91, 클래스2의 개수가 6, 클래스3의 개수가 3이면 가지치기(pruning)를 하였다. 이는 모든 실험 데이터집합(dataset)에서 동일하게 적용하였다. 또한 이 논문에서 제안한 방법이 여러 환경에서 적용됨을 보이기 위해 표 2와 같이 기호적 속성을 가진 데이터집합(dataset)뿐만 아니라 수치적 속성을 가진 데이터집합(dataset)을 사용해 실험을 하였다. 그리고 이에 대한 성능은 각 훈련 데이터집합(dataset)에 의해 의사 결정 트리가 만들어 질 때마다 측정하였다.

중요 이벤트만을 선택함으로써 분류기의 최적의 성능을 보이는 경계점(F)을 찾기 위해 경계점(F)을 0 ~ 0.3까지 0.001씩 증가 시켰고 이를 바탕으로 3가지 정보를 측정하였다.

가) 중요 이벤트를 가진 리프 노드 개수
나) 중요 이벤트 리프노드에 참여한 시험(test) 데이터 개수
다) 에러율

그리고 각각의 값들은 그 범위가 차이가 많이 나기 때문에 한 개의 그래프로 표현하여 비교해 볼 수 있도록 평균 분류성능을 0~1사이로 정규화를 하였다. 이때 경계점(F)를 증가시키면서 리프 노드중에서 가장 많은 이벤트 개수가 V_{total}*F보다 작거나 같을 때 경계점(F)증가를 중단하였다. 이는 경계점(F)가 0.3이 되기 전에 실험이 종료되었음을 실험 결과를 통해 확인 할 수 있다.

그림 3, 그림 4, 그림 5, 그림 6의 X축은 경계점(F)이고 Y축은 3개의 평균 성능 결과 값을 0~1로 정규화한 값이다. 즉, Y축이 정규화한 값이기 때문에 3개의 그래프 간에 상대적인 높낮이는 의미가 없고 그래프 형태를 개별적으로 보고 의미를 파악해야 한다. 여기서 경계점(F)이 0인 경우는 모든 시험(test) 데이터가 성능 측정에 참여하였음을 의미한다.

표 2. 실험을 위한 훈련 데이터 집합(dataset)
Table. 2 Training Data Set for Experiment

name	# event	# attribute		# class
		nominal	numerical	
Vehicle	850		18	4
Letter	20,000		16	26
KingRook VS KingPawn	3196	36		2
connect-4	67,555	42		3

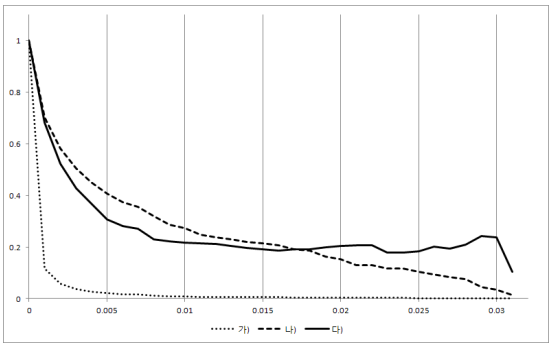

그림 3. Letter
Fig. 3 Letter

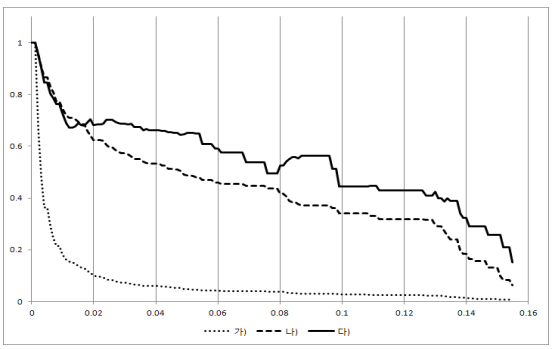

그림 4. Vehicle
Fig. 4 Vehicle

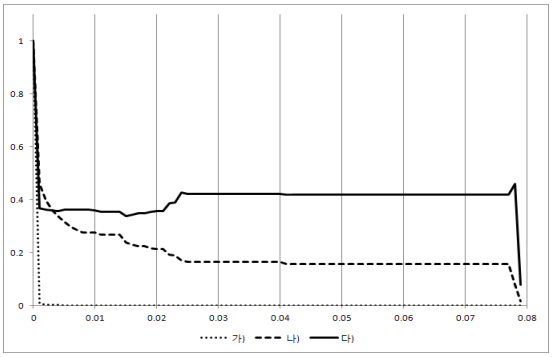

그림 5. Connect-4
Fig. 5 Connect-4

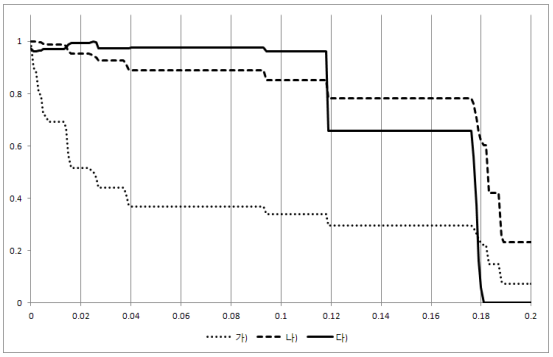

그림 6. KingRook VS KingPawn
Fig. 6 KingRook VS KingPawn

가) 중요 이벤트 노드의 개수
나) 중요 이벤트 리프노드에 참여한 시험(test) 데이터 개수
다) 에러율

즉, 경계점(F)이 점점 증가함에 따라 성능 측정에 참여한 시험(test) 데이터의 개수가 점점 적어지고 있음을 알 수 있다.

경계점(F)가 증가할 때 중요 이벤트를 가진 리프노드의 개수가 초기에 급격히 떨어짐을 볼 수 있다. 이는 매우 작은 이벤트 개수를 가진 리프 노드들 즉, 중요하지 않은 리프 노드들이 많이 존재함을 보여주고 있다. 이와 더불어 에러율도 같이 떨어지는데 이는 결국 중요하지 않은 리프 노드를 의사 결정시 제외함으로써 의사 결정 트리의 성능이 증가되었음을 보여주고 있다.

그리고 경계점(F)값 초기에 중요 이벤트를 가진 리프 노드의 개수가 급격히 감소하는 것과 다르게 중요 이벤트에 참여한 시험(test) 데이터 개수는 완만한 경사를 이루고 있음을 볼 수 있다. 특히 그림 6을 보면 경계점(F)이 0.18이 될 때까지 중요 이벤트에 참여한 시험(test) 데이터 개수는 거의 줄어들지 않고 있다. 이는 중요 이벤트들이 큰 공간(space)에 모여 있고 작은 이벤트들로 이루어진 리프 노드에는 시험 데이터(test)가 거의 접근하지 않음을 보여준다. 결국 작은 이벤트들로 이루어진 리프 노드는 에러율을 증가시키는 요인임을 볼 수 있고 시험(test)할 때 이를 제외함으로써 성능이 향상됨을 알 수 있다. 즉, 이 논문이 제시한 가정이 매우 합리적이었다는 것을 보여주는 부분이다.

Ⅴ. 결 론

이 논문은 의사 결정 트리에서 중요한 이벤트들을 가지고 있는 리프 노드들을 결정하여 의사 결정시 이러한 리프 노드들에 한해서 시험(test)하는 방법을 제안하였다. 여기에 선택되어지지 않은 리프 노드들은 매우 작은 이벤트를 가지고 있을 것이고 이는 노이즈 데이터 또는 그러한 형태를 가짐으로써 분류기의 성능을 저하시킬 것이라는 가정에서 출발한 것이다.

이 논문이 제안하는 방법을 검증하기 위해 의사 결정 트리를 사용하였다. 왜냐하면 어느 분할 공간(space)이 중요 이벤트들로 구성되어 있는지 여부를 파악 하려면 이벤트 개수를 쉽게 계산 할 수 있어야 하는데 의사 결정 트리는 지역적으로뿐만 아니라 전역적으로도 노드에 포함된 이벤트 개수를 쉽게 계산 할 수 있기 때문이다. 이는 의사 결정 트리가 레벨이 증가하면서 내재적으로 데이터 공간(space)을 하위 공간(sub- space)로 나누기 때문에 가능한 부분이다.

의사 결정시 원치 않은 리프 노드들을 시험(test)하는 것을 제외하면 시험(test) 데이터의 개수가 줄어야 하는 희생을 감수해야 하지만 분류기의 성능은 증가하였음 실험을 통해서 알 수 있었다. 따라서 데이터의 성격에 맞추어 중요 이벤트를 가지고 있는 잘 분류된 리프 노드의 개수를 바탕으로 경계점(F)을 잘 선택한다면 의사 결정 트리로부터 최적의 성능을 얻을 수 있을 것이다.

REFERENCES

[1] J. R. Quinlan, "C4.5:Program for Machine Learning ," San Mateo, Calif, Morgan Kaufmann, 1993

[2] T. S. Lim, W. Y. Loh, and Y. S. Shih, "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Tree Old and New Classification Algorithms," *Machine Learning*, vol. 40, no. 3, pp. 203-228, Sept.2000.

[3] Paul E. Utgoff, "Incremental Induction of Decision Trees", *Machine Learning*, vol. 4, no. 2, pp. 161-186, 1989

[4] Dong-Hui Kim, Won Don Lee, "The performance of a classifier by testing only the significant events", *2014 International Conference on Information Science and Applications. technically Co-Sponsored by IEEE*, pp.372~375, May. 2014

[5] Dong-Hui, Dong-Hyeok Lee and Won Don Lee, "Classifier using Extended Data Expression", *IEEE Mountain Workshop on Adaptive and LerningSystems*, pp.154-159, July.2006

[6] Dong-Hui Kim, Dong-Hun Seo, Yingrong Li, Won Don Lee, "A Classifier Capable of Rule Refinement", *2008 IEEE International Conference on Service Operations and Logistics, and Informatics*, pp.168~173, Oct.2008

[7] K. Bache,M. Luchiman, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/]

김동희(Dong-Hui Kim)

2004년 충남대학교(학사)
2006년 충남대학교(석사)
2010년 충남대학교(박사수료)
※관심분야 : 데이터마이닝, 신경회로망

이원돈(Won Don Lee)

1979년 서울대학교(학사)
1982년 U.of Illinois(석사)
1986년 U.of Illinois(박사)
1987년~현재 충남대학교 컴퓨터공학과 교수
※관심분야 : 신경회로망, 데이터마이닝, 멀티미디어