Journal of the Korean Institute of Industrial Engineers
Vol. 26, No. 2, pp. 171-182, June 2000.

실시간 기계 상태 데이터베이스에서 데이터 마이닝을 위한 적응형 의사결정 트리 알고리듬

백준걸[1] · 김강호[1] · 김성식[1] · 김창욱[2]

[1]고려대학교 산업공학과 / [2]명지대학교 산업시스템공학부

Adaptive Decision Tree Algorithm for Data Mining in Real-Time Machine Status Database

Jun-Geol Baek[1] · Kang-Ho Kim[1] · Sung-Shick Kim[1] · Chang-Ouk Kim[2]

For the last five years, data mining has drawn much attention by researchers and practitioners because of its many applicable domains. This article presents an adaptive decision tree algorithm for dynamically reasoning machine failure cause out of real-time, large-scale machine status database. Among many data mining methods, intelligent decision tree building algorithm is especially of interest in the sense that it enables the automatic generation of decision rules from the tree, facilitating the construction of expert system. On the basis of experiment using semiconductor etching machine, it has been verified that our model outperforms previously proposed decision tree models.

1. 서 론

컴퓨터 기술의 발달은 다양한 도메인에서 많은 양의 데이터를 실시간으로 생성해내고 있으며, 이러한 데이터 집합에는 대부분 사용자가 필요로 하는 유용한 정보가 내포되어 있다. 이런 점 때문에 최근 들어 대용량 데이터베이스로부터 의사결정에 필요한 정보나 지식을 추출해내는 데이터 마이닝(Data Mining) 분야에 관한 연구가 폭 넓게 진행되고 있다.(Adriaans and Zantinge, 1996)

이 연구에서는 실시간으로 수집되는 기계 상태 데이터로부터 기계 고장 원인을 점진적으로 발견하는 데이터 마이닝 방법으로 적응형 의사결정 트리인 ADT(Adaptive Decision Tree) 알고리듬을 제시한다. 기계 상태 데이터의 분석을 통해 기계 고장 원인에 관한 정보를 추론하는 것은 기계 고장 진단 시스템 구축에 있어서 매우 중요한 연구 과제이다. 이런 문제는 기존에 신경망, 퍼지 이론(Fuzzy Logic), 또는 사례 기반 추론(Case-Based Reasoning) 등 여러 가지 방법을 이용하여 연구되어 왔으나, 의사결정 트리는 이러한 방법들과는 달리 기계 고장의 원인을 규칙(Rule)으로 표현할 수 있기 때문에 사용자가 기계 고장 원인을 쉽게 이해할 수 있고 전문가 시스템 구축을 용이하게 한다는 장점을 지니고 있다. 때문에 기계 고장 진단 시스템 구축에 적합한 기법이 될 수 있으나 시간에 따른 기계 내부의 성능 변화를 점진적으로 반영하여 고장 원인을 동적으로 재추론하는 적응형 의사결정 트리에 관한 연구는 현재까지 거의 없다. 이 연구에서 제시하는 적응형 의사결정 트리인 ADT 알고리듬은 기존에 연구된 의사결정 트리 알고리듬에 비해 실시간으로 수집되는 기계 상태 데이터에 따라 트리의 효율적인 확장 및 변경이 가능하며, 오류 데이터의 감지 및 처리가 가능하다는 장점을 지닌다.

2. 데이터 마이닝 기존 연구 고찰

데이터 마이닝의 연구 과제는 전통적으로 통계학에서 다루어 온 것이다. 그러나 기존의 통계학 방법들은 계산 시간상의 비효율과 수치적 데이터만을 다룰 수 있다는 단점이 있다. 최근 들어 통계학 이론이 인공지능의 한 분야인 기계 학습(Machine Learning) 이론과 상호 보완 작용을 함으로써 수치와 심볼로 구성된 대단위 데이터 집합에서 미지의 패턴을 발견하는 데이터 마이닝이란 분야로 발전하게 된다. 이 분야는 현재까지 천체 영상 인식 시스템(Fayyad *et al.*, 1996), 영업 관리 시스템(Barr and Mani, 1994), 통신 네트워크 경보 시스템(Sasisekharan *et al.*, 1996), 의료진단 시스템(Mangasarian *et al.*, 1995) 등 다양한 분야에 성

공적으로 적용되고 있다.

데이터 마이닝은 데이터를 구성하는 속성(Attribute)들간의 관계를 설명할 수 있는 미지의 패턴을 주어진 데이터 집합으로부터 근사적인 함수(또는 규칙)로 표현하는 방법으로 정의된다. 데이터 마이닝 방법론은 함수 형태를 명확하게 정형화된 함수로 표현하느냐 혹은 함수의 질적(Qualitative) 특징만을 가정하는 비정형화 함수로 표현하느냐에 따라서 정형화 함수 모형과 비정형화 함수 모형으로 나눌 수 있다. 대표적인 정형화 함수 모형으로는 신경망(Bishop, 1995), 회귀분석(Eubank, 1999), 의사결정 트리(Quinlan, 1993), 분할 벡터 기계(Support Vector Machine)(Bernhard *et al.*, 1998) 등이 있으며, 이 모형들은 함수의 모수(Parameter)를 조정함으로써 미지의 패턴을 찾아가게 된다. 비정형화 함수 모형으로는 K-최단 인접(K-Nearest Neighbor)(Duda and Hart, 1989), 비모수 함수 근사 방법(Bertsimas *et al.*, 1999) 등이 있다. 또한 미지의 패턴을 확정적 또는 확률적 함수로 설명하느냐에 따라서 데이터 마이닝 방법론은 확정적 모형과 확률적 모형으로 구분할 수 있다. 확정적 함수 모형은 앞에서 거론한 정형화 함수 모형 등이 있다. 확률적 모형은 다시 시간에 따른 패턴의 변화에 따라서 정적 모형과 동적 모형으로 나눌 수 있으며 정적 모형으로는 베이지안 네트워크(Bayesian Network) (Heckerman, 1996)와 퍼지 이론 등이 있고 동적 모형으로는 은닉 마코프(Hidden Markov) (Rabiner, 1989)와 동적 베이지안 네트워크 모형(Koller and Russell, 1995) 등이 있다. 한편 데이터 마이닝은 적용 도메인에 따라서 분류(Classification), 군집(Clustering), 연관 규칙(Association), 및 최적(Optimization) 문제로도 구분해 볼 수 있다. 이중에서 최적화를 제외한 도메인에서는 앞에서 언급한 모형들이 주로 사용된다. 대표적인 지능적 최적화 모형으로는 유전 알고리듬(Goldberg, 1989), 신경망, 뉴로 동적 계획법(Bertsekas and Tsitsiklis, 1996) 등이 있다.

이 연구에서 다루는 실시간 기계 상태 데이터베이스에서의 점진적 의사결정 문제는 분류 문제의 일종이며, 기계 내부의 상태 변화에 따라 고장 원인이 변하는 동적 모형이기도 하다. 따라서 이러한 문제의 해결을 위해서 이 연구에서는 확정적 함수인 의사결정 트리의 구조와 모수를 동적으로 바꾸면서 기계 고장 원인을 추론하는 방법을 제시한다.

3. 의사결정 트리 정의 및 문제점

3.1 의사결정 트리 정의

의사결정 트리 문제를 구체적으로 정의하면 다음과 같다. 데이터는 n개의 속성(Attribute) 값 a_i, $i=1, 2, \cdots, n$ 와 결과 클래스(Class) 값 c로 정의되며, 데이터 집합 D에 속한 i번째 데이터 d_i, $i=1, 2, \cdots, |D|$는 $(a_{i1}, a_{i2}, \cdots, a_{in}, c_i)$로 표현된다. 각 속성 값은 이산형 또는 연속형 값을 가질 수 있으며, 결과 클래스 값은 이산형 집합 C에서 하나의 원소 값을 갖는다. 따라서 의사결정 트리 문제는 속성들과 결과 클래스 값의 직교 곱(Cartesian Product)으로 표현되는 데이터 공간에서 데이터 집합 D가 주어졌을 때 이 데이터 집합을 각 속성 값에 따라 부분 집합으로 분할하는 것이며, 이 때 분할 기준은 각 부분 집합에 포함된 모든 데이터가 최대한 같은 결과 클래스 값을 갖도록 하는 것이다.

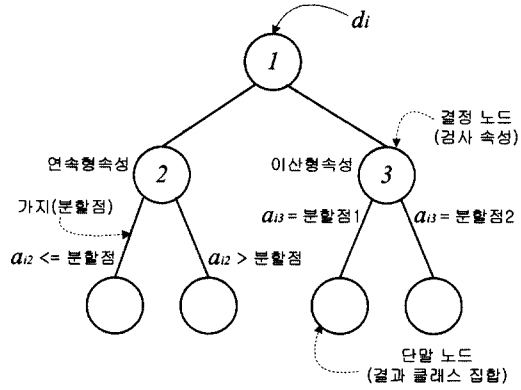


그림 1. 의사결정 트리.

의사 결정 트리는 <그림 1>에서 보듯이 그래프의 일종으로 단말 노드(Node)를 제외한 중간 노드(이하 결정 노드라 함)는 분할 기준이 되는 속성을 의미하며 단말 노드는 상위 결정 노드 패스에 의해 분류된 결과 클래스들의 집합을 의미한다. 가지(Branch)는 결정 노드의 분할 기준이 되는 속성의 분할점(Threshold)이 된다. 이산형 값을 갖는 속성의 분할점은 속성이 가질 수 있는 모든 이산형 값이 되며, 연속형 값을 갖는 속성의 분할점은 속성의 특성을 반영하여 적절히 설정해야 한다. 따라서 의사결정 트리가 주어졌을 때 데이터 집합은 각 결정 노드마다 분할점에 의해서 부분 집합으로 분할되며, 단말 노드에 가서는 최대한 같은 결과 클래스 값을 같은 부분 집합으로 데이터가 구성되어 진다.

이와 같은 문제는 탐욕(Greedy) 알고리듬을 이용한 최적화 문제와 거의 일치한다. 결정 변수는 트리에서 확장될 다음 결정 노드의 속성(이하 검사 속성이라 함)과 검사 속성의 분할점이며[1], 최적 검사 속성은 주어진 데이터들을 최대한 같은 결과 클래스를 갖는 부분 집합으로 분류할 수 있는 것으로 선택한다. 일반적으로 결정 노드에서 최적 검사 속성을 찾기 위해서는 속성 선택의 기준이 되는 척도(Measure)를 정의해야 하는데 이 연구에서는 효율적인 계산을 수행할 수 있는 새로운 속성 선택 척도를 5장에서 제시한다.

3.2 기존 의사결정 트리의 문제점

데이터 마이닝을 위해 사용되는 의사결정 트리의 구축에 관

1) 이산형 속성일 경우는 분할점이 미리 정해져 있으므로 분할점 설정은 결정 변수에서 제외된다.

련된 대표적인 연구로는 C4.5 (Quinlan[17])를 들 수 있다. C4.5는 ID3 알고리듬을 확장한 것으로 결정 노드에서 분할 기준이 되는 최적 검사 속성을 선택하기 위해서 속성 선택 척도로 정보획득량(Information Gain)을 사용한다. 이 척도는 정보 이론의 엔트로피(Entropy) 개념을 사용하는데 엔트로피값은 데이터의 분할이 잘 될수록 작은 값을 갖게 된다. 따라서 검사 속성 후보 중에서 엔트로피값이 최소가 되는 속성을 해당 결정 노드의 검사 속성으로 선택하게 된다. 그러나 C4.5는 고정된 데이터 집합을 기반으로 의사결정 트리를 구축하기 때문에 실시간으로 수집되는 기계 상태 데이터가 주어지는 경우 변화하는 기계 특성을 정확히 반영한 기계 고장을 예측하기 위해서는 데이터가 증가할 때마다 새롭게 의사결정 트리를 구축해야 하는 문제점을 지닌다.

따라서 이러한 문제점을 해결하기 위한 방법으로 기계 상태 데이터가 증가하더라도 기존에 구축된 의사결정 트리를 버리지 않고 데이터가 증가될 때마다 기존의 트리를 확장해 나가는 적응형 의사결정 트리의 구축에 관한 연구가 진행되고 있다. 적응형 의사결정 트리의 구축을 위해서는 실시간으로 수집되는 데이터에 따라 기존의 트리를 재구성해야 하는데 트리의 재구성은 하나의 데이터가 추가될 때마다 처리하는 방법과 데이터 군(Batch)을 형성하여 처리하는 방법이 있다. 적응형 의사결정 트리에 관련된 대표적인 연구로는 ID5R(Utgoff, 1989)을 들 수 있다. ID5R에서는 데이터가 추가될 때마다 기존의 트리를 재구성하는 방법을 사용하고 있으며 C4.5와 같이 엔트로피를 나타내는 E-score를 척도로 사용한다. 또한 풀-업(Pull-Up) 알고리듬을 이용하여 데이터가 추가될 때마다 E-score를 갱신하고 이를 기준으로 기존에 구축된 의사결정 트리를 확장함으로써 현재의 기계 상태를 의사결정 트리에 반영할 수 있다. 그러나 ID5R은 새로운 데이터가 추가될 때마다 이 데이터를 트리의 재구성을 위해 사용하기 때문에 데이터가 수집되는 순서에 매우 민감하고 의사결정 트리의 잦은 재구성으로 인한 계산 부하가 증가하는 단점을 지닌다. 또한 ID5R에서는 수집되는 데이터에 대해 아무런 제약을 두지 않고 그대로 받아들이기 때문에 오류 데이터가 입력되는 경우 의사결정 트리가 비효율적으로 재구성[2]되는 단점을 지닌다.

따라서 이 연구에서는 위에서 언급한 ID5R의 문제점을 해결할 수 있는 새로운 적응형 의사결정 트리의 구축 방법을 제시하고, 이를 통해 실시간으로 수집되는 기계 상태 데이터를 효율적으로 분석하여 기계 고장 예측을 동적으로 수행할 수 있도록 한다.

4. 적응형 의사결정 트리

이 연구에서 제시하는 적응형 의사결정 트리인 ADT는 다음과 같은 특징을 갖는다.

- 실시간으로 수집되는 기계 상태 데이터에 따라 트리의 변경 및 확장이 가능한 점진적 의사결정 트리
- 오류 데이터의 감지 및 처리를 통해 불필요한 트리의 갱신을 방지할 수 있는 적응형 의사결정 트리

4.1 적응형 의사결정 트리 구축 절차

ADT의 구축 절차를 단계별로 설명하면 다음과 같다.

단계 1 : 초기 의사결정 트리 구축

이 연구에서 제안하는 검사 속성 선택 척도인 SDM(Sum of Dominance Metric)을 이용하여 이미 수집된 데이터 집합에 대한 초기 의사결정 트리를 구축한다.

단계 2 : 군(Batch) 데이터 구성

너무 잦은 트리의 갱신을 방지하고 오류 데이터의 검사가 용이하도록 실시간으로 수집되는 데이터를 일정 크기의 군으로 형성한다.

단계 3 : 의사결정 트리 갱신 여부 판단

- 단계 3.1 : 데이터 군의 적합도 검사

입력된 데이터 군이 현재의 트리에 얼마나 잘 맞는지를 검사한다. 일정 허용 오차 내에서 데이터 군이 적합하다는 판단이 서면 <단계 2>로 가고(현재 트리를 보존), 그렇지 않으면 <단계 3.2>로 간다.

- 단계 3.2 : 노이즈(Noise) 데이터 검사

데이터 군에 노이즈 데이터가 섞여있는지를 검사한다. 노이즈 데이터가 존재하면 이를 제거하고 <단계 4>로 간다.

단계 4 : 의사결정 트리의 갱신(변경 및 확장)

SDM 척도를 이용하여 트리를 변경하거나 확장한다.

4.2 군 데이터 구성 및 적합도 검사

입력된 데이터 군이 현재의 트리에 얼마나 잘 맞는지를 다음과 같은 오류율(Error Ratio) 계산을 통해 검사한다.

$$e(B)=\frac{\sum_{i=1}^{|B|}\delta(d_i)}{|B|}$$

여기서

$|B|$: 데이터 군 B의 크기

$d_i=(a_{i1}, a_{i2}, \ldots, a_{in}, c_i)$: 데이터 군에 속한 i번째 데이터, $i=1, 2, \ldots, |B|$

$$\delta(d_i)=\begin{cases}1 & \text{if } c_i \neq T(a_{i1}, a_{i2}, \ldots, a_{in})\\ 0 & \text{if } c_i = T(a_{i1}, a_{i2}, \ldots, a_{in})\end{cases}$$

2) 이러한 현상을 과도 적합(Overfitting)이라 한다.

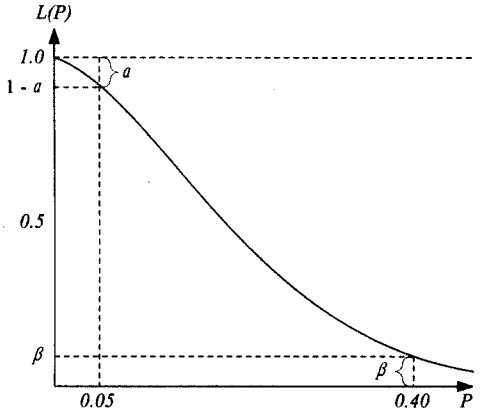


그림 2. 기존 트리 보존 확률 OC 곡선.

T: 현재의 의사결정 트리

입력된 데이터 군에 대한 적합도 검사는 해당 데이터 군의 오류율이 기준율(ψ) 이하이면 현재의 트리를 보전하고 그렇지 않으면 노이즈 데이터 검사로 넘어간다.

트리의 재구성 여부를 결정짓는 기준율(ψ)은 데이터 군의 크기 $|B|$와 기존 트리와의 불일치 데이터 허용 한계 수 k에 의해 결정되는데 $|B|$와 k의 값은 <그림 2>와 같은 OC (Operating Characteristic) 곡선을 이용하여 결정한다. 데이터 검사 방식은 계수형 품질 관리에서 사용하는 ($|B|$, k) 방식을 응용한 것으로 데이터 군(B)의 오류 데이터 수가 k개를 초과하면 현재 트리에 문제가 있는 것으로 판별한다.

<그림 2>에서 가로축 P는 입력되는 데이터가 기존의 트리와 일치하지 않을 확률을 나타내고 세로축 $L(P)$는 P가 주어질 경우 기존의 트리가 보존될 확률을 의미한다. α는 제1종 과오로서 입력되는 데이터 군이 현재의 트리와 일치하지 않을 확률이 0.05로 낮음에도 불구하고 트리가 갱신될 확률을 나타내고 β는 제2종 과오로서 입력되는 데이터가 기존의 트리와 일치하지 않을 확률이 0.4로 높음에도 불구하고 트리가 보존될 확률을 나타낸다. 따라서 α와 β를 만족할 만한 수준으로 보장할 수 있는 데이터 군의 크기($|B|$)는 다음 식과 같이 푸아송 분포를 이용하여 근사적으로 구할 수 있다. ($\alpha=\beta=0.05$인 경우)

$$L(0.05)=\sum_{x=0}^{k}\frac{e^{-0.05|B|}(0.05|B|)^x}{x!}=1-\alpha=0.95$$

$$L(0.4)=\sum_{x=0}^{k}\frac{e^{-0.4|B|}(0.4|B|)^x}{x!}=\beta=0.05$$

위 식에 의하면 데이터 군의 크기($|B|$)는 16이 되고 오류 데이터의 허용한계 수(k)는 2가 된다. 따라서 기준율은 $\psi=k/|B|=2/16=0.125$로 계산되어지고 이에 따른 적합도 검사가 이루어진다. $|B|$와 k 값은 사용자가 입력하는 모수 α와 β에 의해 좌우된다.

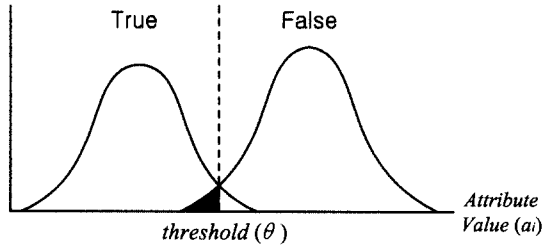


그림 3. 노이즈의 정의.

4.3 노이즈 데이터 검사

이 연구에서 제안하는 ADT는 노이즈 데이터의 입력으로 인한 비효율적인 트리의 재구성을 방지하기 위해 노이즈 데이터 검사를 수행한다.

4.3.1 노이즈 데이터의 정의 및 원인

노이즈 데이터는 연속형 검사 속성의 분할점에 의해 데이터가 부분 집합으로 나뉘어 질 때 사후(Posterior) 확률 분포 $P(c|a_1, a_2, \cdots, a_n)$ 특성상 분할점 부근에서 자연스럽게 발생하는 것을 의미한다. 예를 들어 <그림 3>에서 색칠한 부분과 같이 실제 결과 클래스가 'False'인 데이터가 의사결정 트리에 의해 'True'로 분류되는 것을 의미한다. 이와 같은 현상은 의사결정 트리를 구축하려는 대상 시스템의 결과 클래스가 속성 값들에 의해 확정적인 함수로 표현되는 것이 아니라 확률적으로 표현되는 데서 비롯된다. 따라서 확률 분포의 특성상 자연스럽게 발생하는 노이즈를 확정적 함수인 트리를 사용해서 무리하게 맞추려고 하면 분할점이 무수히 많아지고, 따라서 트리가 과도 적합(Overfitting)하게 되어 오히려 트리의 정확도가 떨어지는 문제점을 발생시킨다.

데이터 군에 속한 노이즈 데이터의 판별 방법은 노이즈를 발생시키는 검사 속성이 연속형이냐 이산형이냐에 따라서 달라진다.

4.3.2 연속형 속성의 노이즈 처리

연속형 속성의 노이즈 데이터 처리를 위해서 이 연구에서는 거리 기준 최단 인접 규칙(Distance Based Neighborhood Rule)을 제시한다. 거리 기준 최단 인접 규칙은 주어진 오류 데이터 $d_i \in B$에서 연속형 검사 속성의 값이 a_{ij}이고 속성의 분할점이 θ_l일 때 다음과 같은 식을 통해 노이즈 데이터 여부를 판별한다.

If $\left((a_{ij}-\theta_l \le offset) \text{ and } \left(\frac{E_c(\theta_l)}{E_c(\theta_l)+N_c(\theta_l)} < \xi_1\right)\right)$

Then 노이즈로 처리

여기서

$$offset = \lambda_l \cdot \omega$$

$$\lambda_l = |\theta_l - mean(a_{\cdot j})|$$

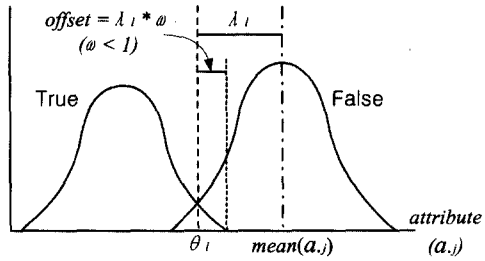


그림 4. *offset* 정의.

$mean(a_{\cdot j})$: j 번째 속성의 평균값

$E_c(\theta_l)$: θ_l의 왼쪽(오른쪽)에 속하면서 결과 클래스가 c인 오류 데이터 수

$N_c(\theta_l)$: θ_l의 오른쪽(왼쪽)에 속하면서 오류 데이터와 동일한 결과 클래스(c)를 갖는 데이터 수

ω, ξ_1 : 모수 ($0 < \omega, \xi_1 < 1$)

위의 첫번째 식에서 *offset*은 <그림 4>와 같이 분할점 θ_l과 연속형 검사 속성 $a \cdot j$의 평균값의 절대 차이인 λ_l에 사용자가 입력하는 허용 모수 ω를 곱하여 계산한다.

따라서 오류 데이터 d_i의 속성값 a_{ij}와 분할점 θ_l의 차이가 미리 설정한 *offset*보다 작으면 a_{ij}는 θ_l 근처의 값을 갖게 되므로 노이즈일 가능성이 커진다. 그러나 최적 θ_l은 <그림 4>와 같이 사후 확률 분포를 알아야만 구할 수 있기 때문에 사후 확률 분포를 추정하는 과정에서는 데이터가 증가함에 따라 θ_l을 변경할 필요가 있다. 즉, <그림 4>에서 'False'인 데이터가 θ_l의 왼쪽에 계속적으로 나타날 때 θ_l은 왼쪽으로 이동해야 한다. 위의 판별식에서 두 번째 식이 이와 같은 현상을 반영하기 위한 것이다. 예를 들어 <그림 4>에서 θ_l의 왼쪽에 속하면서 'False'로 판명된 데이터 수(오류 데이터 수)가 10개($E_F(\theta_l) = 10$)이고 θ_l의 오른쪽에 속하면서 'False'로 판명된 데이터 수(정상 데이터 수)가 100개($N_F(\theta_l) - 100$) 이면 $E_F(\theta_l) / (E_F(\theta_l) + N_F(\theta_l)) = 0.091$이 된다. 이 값이 커질수록 현재의 θ_l은 부적절한 것이며, 반대로 이 값이 작아지면 θ_l은 적절하므로 현재 고려중인 오류 데이터는 노이즈로 판별하게 된다. 기준값 ξ_1은 사용자가 정하는 모수이다.

노이즈를 검사하기 위해서는 데이터가 새로 추가될 때마다 매번 전체 데이터를 재검색하여 결과 클래스 별 데이터 속성의 평균값을 다시 계산해야 *offset*을 알 수 있는데 이 연구에서는 이러한 문제점을 해결하기 위해 각 단말 노드 별로 해당되는 데이터 부분 집합을 저장하는 방법을 사용한다. 이 경우 의사결정 트리를 표현하기 위한 기억공간이 커질 위험이 있지만 상위 결정노드에 의해 분할된 데이터만을 보관함으로써 보관되는 데이터의 크기를 줄일 수 있다. 자세한 내용은 5장에서 다루기로 한다.

4.3.2 이산형 속성의 노이즈 처리

앞에서 언급한 거리를 기준으로 한 최단 인접 규칙은 연속형 확률 분포를 기반으로 정의된 것이기 때문에 이산형 검사 속성에 사용하기는 부적절하다. 따라서 이 연구에서는 이산형 검사 속성에 대한 노이즈 데이터를 검사하기 위해 순수 베이지안 분류식(Naive Bayesian Classifier)을 이용한다. 순수 베이지안 분류식(Mitchell, 1997)은 $c_j \in C$가 주어졌을 때 결합 우도 확률(Joint Likelihood Probability) $P(a_1, a_2, \cdots, a_n | c_j)$가 조건부 독립(Conditional Independence)을 보장한다는 가정 하에 베이지안 공식을 사용하여 다음과 같이 사후 확률 $P(c_j | a_1, a_2, \cdots, a_n)$를 계산한다.

$$P(c_j|a_1, a_2, \ldots, a_n) = \frac{\prod_{i=1}^{n} P(a_i|c_j) P(c_j)}{P(a_1, a_2, \ldots, a_n)}$$

위의 계산식에서 $P(c_j)$는 수집된 데이터의 결과 클래스(c_j) 별 비율로서 쉽게 계산되어 진다. 또한 $P(a_1, a_2, \cdots, a_n)$는 $P(c_j | a_1, a_2, \cdots, a_n)$가 확률적 개념을 만족시킬 수 있도록 정규화 시키는 데 사용되는 값으로서 큰 의미는 없다. 순수 베이지안 분류식은 각 결과 클래스마다 사후 확률을 계산한 후 확률값이 가장 큰 클래스를 최대 사후 결과 클래스 c_{MAP}[3]로 선정한다. 이를 수식화하면 다음과 같다.

$$c_{MAP} = argmax_{c_j \in C} P(c_j|a_1, a_2, \ldots, a_n)$$

현재까지의 연구 결과를 보면 순수 베이지안 분류식은 $P(a_1, a_2, \cdots, a_n | c_j)$가 c_j에 대해 조건부 독립이라는 가정에도 불구하고 상당히 정확한 결과를 보인다(Domingos and Pazzani, 1997).

순수 베이지안 분류식은 일종의 비모수 확률 분포 추정 기법으로 의사결정 트리 대신 분류 문제에 사용할 수 있다. 그러나 데이터 집합의 크기가 클 때는 계산 시간이 오래 걸리기 때문에 비효율적이다. 따라서 이 연구에서는 데이터 군에 속한 오류 데이터에 대해서만 순수 베이지안 분류식을 이용하여 최대 사후 결과 클래스를 추론하며, 노이즈 여부는 다음과 같이 판별한다. 이산형 검사 속성에 의해 분류된 데이터의 부분 집합을 최대 사후 결과 클래스 c_{MAP}와 기타 결과 클래스 $c_j \in C - \{c_{MAP}\}$로 구분하고 이들의 사후 확률의 차이를 다음 식과 같이 계산하여 그 차이가 기준값 ξ_2보다 작을 경우 트리의 갱신을 실시한다.

$$min_{c_j \in INC - \{c_{MAP}\}} (P_{c_{MAP}} - P_{c_j}) < \xi_2$$

연속형 검사 속성과 마찬가지로 기준값 ξ_2는 사용자가 정하는 모수이다.

4.4 의사결정 트리의 갱신

노이즈가 아니라고 판명된 데이터를 기존 트리에 반영할 때는 다음과 같은 선택이 존재한다. 첫째는 새로운 검사 속성을

3) MAP는 최대 사후확률(Maximum a Posterior)을 뜻한다.

고려하여 트리를 확장하는 것, 둘째는 기존 트리의 속성 분할점을 수정하는 것, 그리고 셋째는 기존 트리의 검사 속성 순서를 바꾸는 것이다. 참고로 이산형 검사 속성의 경우 분할점 수정은 해당되지 않는다. 트리의 갱신 및 초기 구축은 다음 장에 소개될 SDM (Sum of Dominance Metric)이라는 속성 선택 척도를 사용하여 이루어진다.

5. 검사 속성 선택 척도

초기 트리 구성 및 기존 트리 갱신에는 검사 속성 선택 기준이 있어야 한다. 이 연구에서는 이를 위해 SDM(Sum of Dominance Metric)이라는 척도를 제시한다. SDM은 실시간으로 증가하는 데이터를 기반으로 적응형 의사결정 트리를 구축하는데 있어서 기존의 엔트로피 방식에 비해 빠른 시간에 검사 속성을 선택할 수 있는 장점을 지닌다. SDM 척도를 사용하기 위해서는 각 결정 노드마다 DM_Array(Dominance Metric Array)와 AV_List (Attribute Value List)라는 자료 구조가 필요하며 이에 대한 설명은 다음과 같다.

5.1 DM_Array

DM_Array는 각 결정 노드에 속한 검사 속성별로 존재한다. DM_Array는 검사 속성이 가질 수 있는 속성값과 그 속성값에 의해 분류된 결과 클래스 수를 나타낸 것이다. DM_Array의 구성 방법은 이산형 속성과 연속형 속성에 따라 달라지며, 자세한 설명은 다음과 같다.

5.1.1 이산형 검사 속성의 DM_Array 구성

이산형 검사 속성의 경우 속성이 가질 수 있는 값이 미리 정해져 있기 때문에 속성값과 결과 클래스별 데이터의 수를 통해 다음과 같이 DM_Array를 쉽게 구성할 수 있다.

$$DM\_Array(i) = (n_{i1}, n_{i2}, \ldots, n_{i|C|}, n_{i21}, n_{i22}, \ldots, n_{ijc}, \ldots, n_{i|a_i||C|})$$

여기서,

i : i 번째 속성

n_{ijc} : 속성 i의 값이 j번째 속성값이고 결과 클래스 값이 c인 데이터의 개수

$|a_i|$: a_i가 가질 수 있는 속성값의 개수

예를 들어 속성 a_1이 이산형으로 가질 수 있는 값이(*short, long*)이고 결과 클래스 집합이 (*true, false*)라고 하면, a_1에 대한 DM_Array의 크기는 (속성 값 수) × (결과 클래스 수) = 2×2 = 4가 되며, DM_Array는 <그림 5>와 같이 나타낼 수 있다. 이 예제에서 N(*short, true*)은 속성 값이 '*short*'이고 결과 클래스가 '*true*'인 데이터의 개수를 나타낸다.

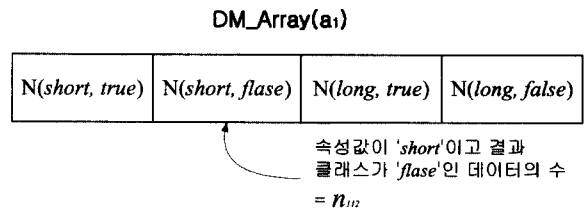


그림 5. 이산형 검사 속성의 DM_Array 구성.

5.1.2 연속형 검사 속성의 DM_Array 구성

연속형 검사 속성의 경우 이산형 속성과 달리 가질 수 있는 값이 무한히 많기 때문에 <그림 5>와 같이 검사 속성값과 결과 클래스별 데이터의 수로 DM_Array를 구성하는 데는 문제가 있다. 따라서 연속형 속성은 가질 수 있는 값의 범위를 구간으로 나누고 각 구간에 속한 결과 클래스 수를 통해서 DM_Array를 구성한다. 속성이 가질 수 있는 값의 구간을 나누기 위해서는 기준이 되는 분할점 $\theta \in \Theta$가 필요하다.

이런 방식을 연속형 검사 속성의 이산형 변환(Discretization)이라고 하며 현재까지 널리 쓰이는 방식은 두 가지가 있다. 첫째는 C4.5에서 제안한 중간점(Mid Point) 방식(Quinlan, 1993)으로 <그림 6>에서 보듯이 속성이 갖는 값들을 크기 순으로 정렬한 후 중간점들을 분할점 집합 Θ로 선택한다. 그러나 C4.5 방식은 속성값의 개수가 커지면 너무 많은 분할점들이 생성된다는 단점을 지닌다. 둘째는 중간점 방식의 단점을 해결하기 위해 제시된 경계점(Boundary Point) 방식(Fayyad and Irani[10])으로 <그림 6>에서 보듯이 결과 클래스의 값이 바뀌는 중간점들 만을 분할점 집합 Θ로 선택한다. 이 연구에서는 <그림 6>과 같은 AV_List를 정의한 후 경계점 방식을 이용하여 분할점을 선택한다.

경계점 방식을 이용하여 연속형 검사 속성에 대한 DM_Array

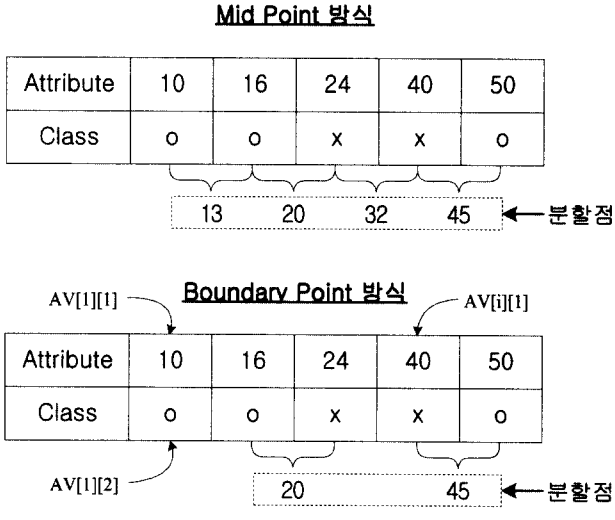


그림 6. 분할점 선택을 위한 AV_List.

를 구성하는 경우 새로운 데이터가 추가될 때 AV_List가 변하게 되고 이에 따라 DM_Array가 갱신되어야 하는데, 이 연구에서는 다음과 같은 절차에 의해 데이터의 추가에 따른 DM_Array의 갱신을 수행한다.

[연속형 속성의 DM_Array 갱신(구성) 절차]

Precondition:

new attribute is inserted at ith position in ascending order AV_List of size N

Procedure:

Case 1 $i = 1$

If $(AV\_List[2][i] \neq AV\_List[2][i+1])$

Then • create new threshold θ

$$\theta = \frac{AV\_List[1][i] + AV\_List[1][i+1]}{2}$$

• update DM_Array for θ

Case 2 $i = N+1$

If $(AV\_List[2][i-1] \neq AV\_List[2][i])$

Then • create new threshold θ

$$\theta = \frac{AV\_List[1][i-1] + AV\_List[1][i]}{2}$$

• update DM_Array for θ

Case 3 $1 < i < N+1$

Case 3.1 $AV\_List[2][i-1] \neq AV\_List[2][i+1]$

If $(AV\_List[2][i-1] = AV\_List[2][i])$

Then • update threshold θ

$$\theta = \frac{AV\_List[1][i] + AV\_List[1][i+1]}{2}$$

• update DM_Array for θ

If $(AV\_List[2][i+1] = AV\_List[2][i])$

Then • update threshold θ

$$\theta = \frac{AV\_List[1][i] + AV\_List[1][i-1]}{2}$$

• update DM_Array for θ

Otherwise

• create new thresholds θ_1, θ_2

$$\theta_1 = \frac{AV\_List[1][i-1] + AV\_List[1][i]}{2}$$

$$\theta_2 = \frac{AV\_List[1][i] + AV\_List[1][i+1]}{2}$$

• update DM_Array for θ_1, θ_2

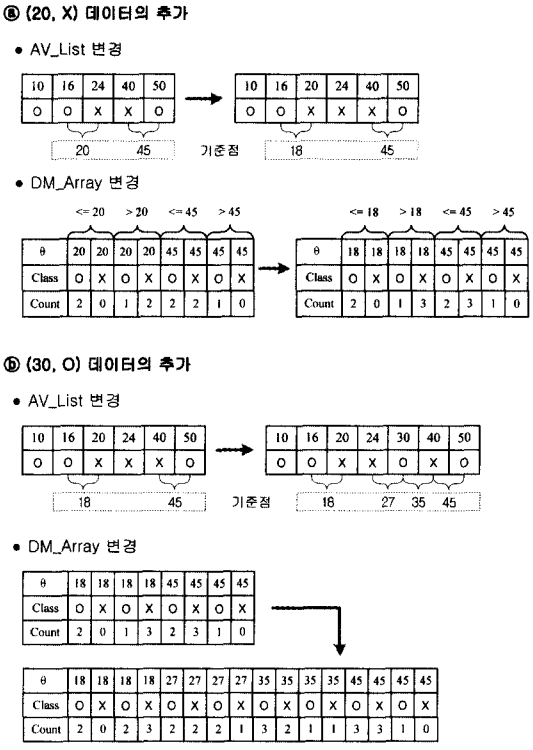


그림 7. DM_Array의 갱신.

Case 3.2 $AV\_List[2][i-1] = AV\_List[2][i+1]$

If $(AV\_List[2][i-1] = AV\_List[2][i])$

Then • create new thresholds θ_1, θ_2

$$\theta_1 = \frac{AV\_List[1][i-1] + AV\_List[1][i]}{2}$$

$$\theta_2 = \frac{AV\_List[1][i] + AV\_List[1][i+1]}{2}$$

• update DM_Array for θ_1, θ_2

<그림 7>은 <그림 6>과 같은 AV_List가 주어진 경우 새로운 데이터 (20, X)와 (30, O)가 추가될 때 AV_List의 변경과 그에 따른 분할점의 변경, 그리고 DM_Array의 갱신을 나타낸 그림이다.

<그림 7>에서 보는 바와 같이 (20, X)라는 새로운 데이터가 추가되면 이 데이터는 AV_List에서 (16, O)와 (24, X) 사이에 위치하게 된다. 따라서 DM_Array에서 분할점이 20에서 18로 변경되고 이에 따라 DM_Array가 갱신된다. 또한 (30, O)라는 데이터가 추가되면 새로운 두 개의 분할점이 추가되고 이에 따른 DM_Array가 변경된다.

5.2 SDM

DM_Array가 구성되면 이로부터 SDM 값을 계산할 수 있다. SDM 값은 결정(또는 단말) 노드에서 결과 클래스의 동질성 (Homogeneity)을 나타내는 것으로, SDM 값이 클수록 해당 노드

의 엔트로피가 낮음을 의미한다. 이 연구에서는 데이터를 구성하는 각각의 속성에 대해서 SDM 값을 계산한 후 SDM 값이 가장 큰 속성을 확장될 결정 노드의 검사 속성으로 사용함으로서 엔트로피가 낮은 효율적인 의사결정 트리를 구축할 수 있도록 한다. SDM 값의 계산은 이산형 속성일 경우와 연속형 속성일 경우에 따라서 달라진다. 이산형 속성일 경우 분할점이 이미 주어져 있기 때문에 최적 분할점을 계산할 필요가 없으나, 연속형 속성일 경우에는 이산형 변환을 수행해야 하기 때문에 최적 분할점 계산을 SDM 값을 계산할 때 동반해야 한다.

이산형 속성일 경우 SDM 값은 다음과 같은 식을 통해 계산할 수 있다.

$$SDM\_Value(i) = \sum_{j=1}^{|a_i|} \{w_j \cdot |n_{ijc^*} - \sum_{c \in C-\{c^*\}} n_{ijc}|\}$$

여기서

i : i 번째 속성

$|a_i|$: 속성 i가 가질 수 있는 속성 값의 수

$$w_j = \frac{n_{ic^*}}{\sum_{c \in C} n_{ijc}} \quad for\ each\ \ j \in |a_i|$$

$$c^* = argmax_{c \in C}(n_{ijc})$$

위 식에서 절대값 안의 식은 우세(Dominant) 결과 클래스(c^*)를 갖는 데이터의 수(n_{ijc^*})와 우세 결과 클래스를 제외한 나머지 결과 클래스($c \in C-\{c^*\}$)를 갖는 데이터의 수 $\left(\sum_{c \in C-\{c^*\}} n_{ijc}\right)$의 차이를 의미하며 w_j는 같은 차이라 하더라도 동질성이 큰 속성의 선택 비중을 높이기 위한 가중치를 의미한다. 예를 들어 <그림 8>의 트리 i)과 ii)의 경우 SDM 값을 계산할 때 절대값 부분은 4로 같지만 트리 ii)가 트리 i)보다는 동질성이 좋기 때문에 가중치 w_j를 이용하여 트리 ii)의 SDM 값이 높아지도록 한다.

연속형 속성일 경우에는 이산형 변환을 수행해야 한다. 따라서 최적 분할점 계산을 SDM 값을 계산할 때 동반해야 하기 때문에 다음과 같은 SDM 공식을 사용한다.

$$SDM\_Value(i) = Max_{\theta_l \in \Theta}\ SDM\_Value(i, \theta_l)$$

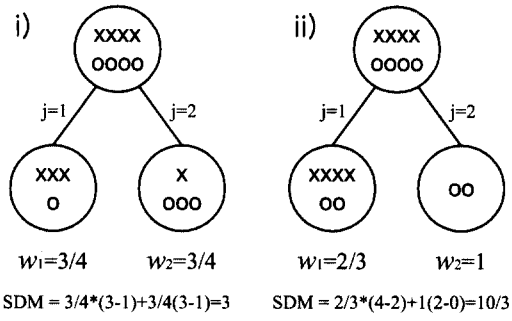


그림 8. 가중치의 의미.

여기서

$$SDM\_Value(i, \theta_l) = \sum_{j \in \{L(\theta_l), R(\theta_l)\}} \{w_j \cdot |n_{ijc^*} - \sum_{c \in C-\{c^*\}} n_{ijc}|\}$$

$L(\theta_l)$: θ_l을 중심으로 왼쪽 구간

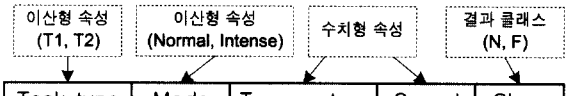
$R(\theta_l)$: θ_l을 중심으로 오른쪽 구간

$SDM_Value(i, \theta_l)$는 주어진 θ_l을 중심으로 왼쪽 구간에 속한 우세 결과 클래스를 갖는 데이터 수와 나머지 결과 클래스를 갖는 데이터 수의 합의 차이에 가중치를 곱한 것과 θ_l을 중심으로 오른쪽 구간에 속한 우세 결과 클래스를 갖는 데이터의 수와 나머지 결과 클래스를 갖는 데이터 수의 합의 차이에 가중치를 곱한 것을 더한 값으로 동질성을 고려한 분류가 제대로 된다면 이 값은 커질 것이다. $SDM_Value(i)$는 모든 가능한 분할점 $\theta_l \in \Theta$에 대한 $SDM\_Value(i, \theta_l)$ 중에서 가장 큰 값을 선택한다. 따라서 최적 분할점 θ_l^*는 다음과 같은 식으로 정의된다.

$$\theta_l^* = argmax_{\theta_l \in \Theta}\ SDM\_Value(i, \theta_l)$$

6. 적응형 의사결정 트리 구축 예제

실시간으로 수집되는 데이터에 따라 기존의 의사결정 트리가 변경되는 점진적 의사결정 트리는 다음과 같은 예제를 통해 설명할 수 있다.



Task_type	Mode	Temperature	Speed	Class
T1	Normal	100	1200	F
T2	Intense	116	1000	F
T1	Normal	102	1600	N
T1	Intense	120	1700	N

그림 9. 초기 데이터.

초기 데이터가 <그림 9>와 같이 주어질 경우 점진적 의사결정 트리의 구축을 위해서는 우선적으로 <그림 10>과 같은 초기 의사결정 트리를 구축해야 한다.

초기 의사결정 트리가 <그림 10>과 같이 구축되었을 때 새로운 데이터 (*T2, Intense, 108, 2100, F*)가 입력되면 기존의 트리가 확장되어져야 하는데 <그림 11>은 새로운 데이터에 의한 DM_Array의 변경과 트리 확장을 나타낸 것이다.

새로운 데이터 (*T1, Normal, 112, 2300, F*)가 추가로 입력되었을 때 <그림 11>의 의사결정 트리는 입력된 데이터에 적합한 트리가 된다. 따라서 트리의 형태는 변화가 없고 <그림 12>와

각 속성별 DM Array 작성

* DM_Array(Task_type)

T1		T2	
N	F	N	F
2	1	0	1

* DM_Array(Mode)

Intense		Normal	
N	F	N	F
2	1	0	1

* DM_Array(Temperature)

101		101		109		109		118		118	
N	F	N	F	N	F	N	F	N	F	N	F
0	1	2	1	1	1	1	1	2	1	0	

AV_List

100	102	116	120
F	N	F	N

* DM_Array(Speed)

1400		1400	
N	F	N	F
0	2	2	0

AV_List

1000	1200	1600	1700
F	F	N	N

SDM Value 계산에 의한 검사속성 선택 및 트리 구축

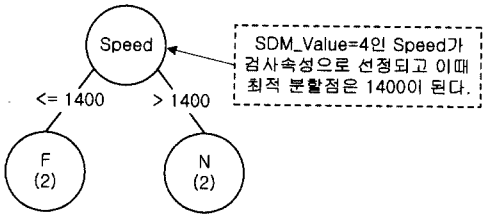


그림 10. 초기 의사결정 트리 구축.

각 속성별 DM Array 갱신

* DM_Array(Task_type)

T1		T2	
N	F	N	F
2	0	0	1

SDM_Value(Task_type) =1*(2-0)+1*(1-0)=3

* DM_Array(Mode)

Intense		Normal	
N	F	N	F
1	1	1	0

* DM_Array(Temperature)

105		105		114		114	
N	F	N	F	N	F	N	F
1	0	1	1	1	1	1	0

AV_List

102	108	120
N	F	N

* DM_Array(Speed)

1900		1900	
N	F	N	F
2	0	0	1

SDM_Value(Speed) =1*(2-0)+1*(1-0)=3

AV_List

1600	1700	2100
N	N	F

SDM Value에 의한 검사속성 선택 및 트리 변경

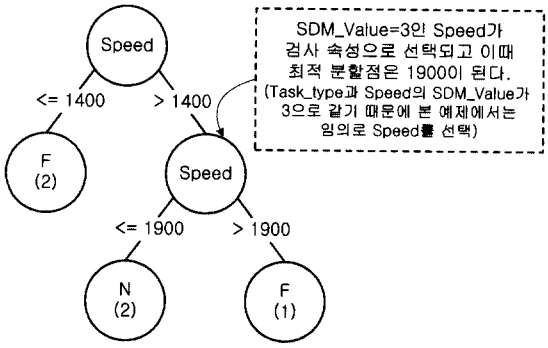


그림 11. 새로운 데이터의 입력에 따른 트리의 확장.

각 속성별 DM Array 갱신

* DM_Array(Task_type)

T1		T2	
N	F	N	F
2	2	0	2

* DM_Array(Mode)

Intense		Normal	
N	F	N	F
1	2	1	2

* DM_Array(Temperature)

101		101		105		105		118		118	
N	F	N	F	N	F	N	F	N	F	N	F
0	1	2	3	1	1	1	3	1	4	1	0

AV_List

100	102	108	112	116	120
F	N	F	F	F	N

* DM_Array(Speed)

1400		1400		1900		1900	
N	F	N	F	N	F	N	F
0	2	2	2	2	2	0	2

AV_List

1000	1200	1600	1700	2100	2300
F	N	F	F	F	N

트리의 단말 노드 정보 변경

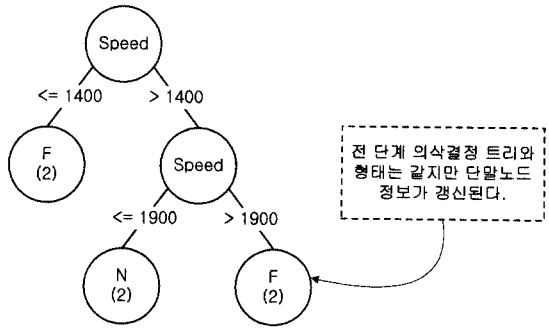


그림 12. 의사결정 트리의 유지.

각 속성별 DM Array 작성

* DM_Array(Task_type)

T1		T2	
N	F	N	F
0	1	1	1

* DM_Array(Mode)

Intense		Normal	
N	F	N	F
0	1	1	1

* DM_Array(Temperature)

123		123	
N	F	N	F
0	2	1	0

SDM_Value(Temperature) =1*(2-0)+1*(1-0)=3

AV_List

100	116	130
F	F	N

* DM_Array(Speed)

1050		1050		1150		1150	
N	F	N	F	N	F	N	F
0	1	1	1	1	1	0	1

AV_List

1000	1100	1200
F	N	F

SDM Value에 의한 검사속성 선택 및 트리 확장

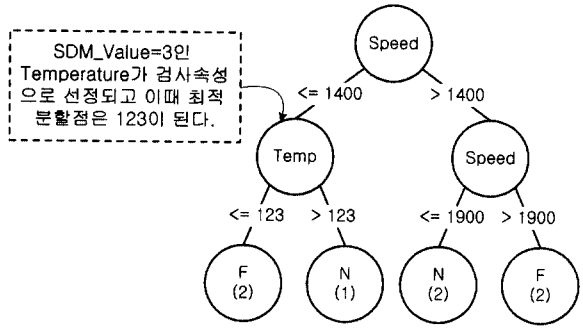


그림 13. 의사결정 트리의 확장.

같이 트리 정보의 변화만이 발생한다.

마지막으로 데이터 (*T2, Normal, 130, 1100, N*)가 입력되는 경우 <그림 12>의 의사결정 트리는 <그림 13>과 같이 확장되어 진다.

7. 사례 연구

이 연구에서 제시한 적응형 점진적 의사결정 트리의 성능을 검증하기 위한 방법으로 반도체 공정에서 에칭 작업을 수행하는 기계 P의 이력 데이터를 기반으로 C4.5와 ID5R에 의해 구축된 의사결정 트리와의 비교를 실시한다.

7.1 기계 이력 데이터

반도체 공정에서 에칭 작업을 수행하는 기계 P는 공정 변수에 따라 에칭율(Etching Rate)이 변화하는 특징을 갖는다. 기계 P는 에칭율이 너무 높거나 너무 낮아지는 경우 에칭에 문제가 생기기 때문에 적절한 에칭율을 유지할 수 있도록 하는 일이 중요하다. 따라서 이 연구에서는 에칭율에 영향을 미치는 공정변수를 속성으로 정의하고 에칭율에 따라 결과 클래스(LE : Low Etching Rate, NE : Normal Etching Rate, HE : High Etching Rate)를 정의한 이력 데이터를 기반으로 에칭율의 이상 상황을 예측할 수 있는 의사결정 트리를 작성한다. 기계 P의 이력 데이터는 <표 1>과 같은 형식을 갖는다.

<표 1>에서 설명한 기계 이력 데이터의 속성들은 <표 2>와 같은 속성 정의 형식으로 나타낼 수 있다.

표 1. 기계 이력 데이터 형식

속 성	PRESS	기계 내의 압력을 나타내는 속성. (μ Torr)
	TEMP	기계 내의 온도를 나타내는 속성. (°K)
	GAS_FLOW	에칭가스의 밀도를 나타내는 속성 (sccm)
	ION_VOLT	이온 전압을 나타내는 속성 (V)
	LOCATION	웨이퍼의 위치를 나타내는 속성
결 과 클래스	LE	에칭율이 너무 낮아 문제가 생기는 경우
	NE	적절한 에칭 작업이 수행되는 경우
	HE	에칭율이 너무 높아 문제가 생기는 경우

표 2. 기계 이력 데이터의 속성 정의 형식

PRESS, continuous, 1, 900
TEMP, continuous, 100, 500
GAS_FLOW, continuous, 1, 100
ION_VOLT, continuous, 25, 100
LOCATION, discrete, ground, powered

7.2 수행도 평가

위에서 기술한 기계 P의 이력 데이터를 기반으로 이 연구에서 제시하는 적응형 점진적 의사결정 트리와 C4.5, ID5R에 의해 구축된 의사결정 트리와의 성능을 비교한다. 의사결정 트리 구축의 기반이 되는 기계 이력 데이터는 30개의 초기 데이터가 수집된 상태에서 180개의 데이터가 추가로 수집되며 수집된 기계 이력 데이터는 <그림 14>와 같이 나타낼 수 있다. <그림 14>는 수집된 데이터에 대해 에칭율에 영향을 미치는 중요 속성인 PRESS, TEMP, GAS_FLOW를 축으로 결과 클래스를 표현한 것이다.

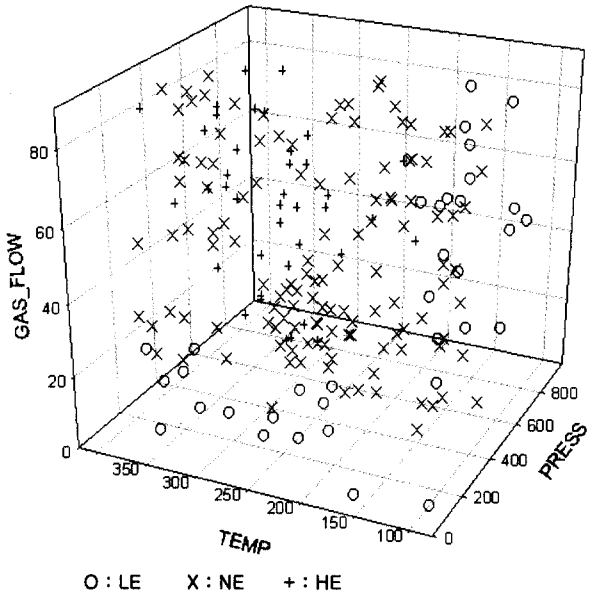


그림 14. 기계 이력 데이터.

이 연구에서 제안하는 적응형 의사결정 트리(ADT)와 C4.5, ID5R에 의해 구축된 트리의 성능을 비교하기 위한 척도로는 트리의 정확도(Correctness), 복잡도(Complexity), 강건성 (Robustness)을 사용한다.

7.2.1 트리의 정확도 비교

구축된 의사결정 트리의 정확도는 주어진 기계 이력 데이터에 대해 얼마만큼 정확히 결과 클래스를 예측할 수 있는지를

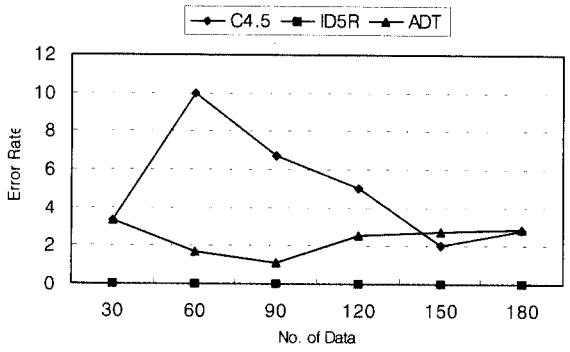


그림 15. 의사결정 트리의 정확도 비교.

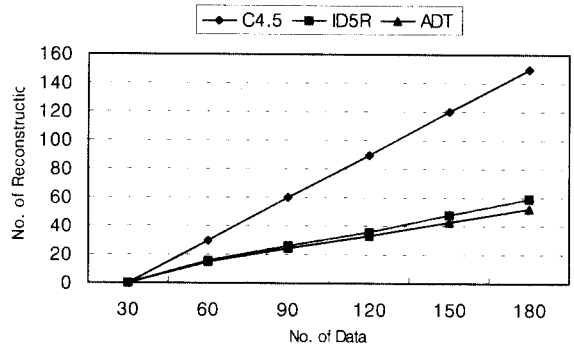


그림 17. 의사결정 트리의 재구성 빈도 비교.

통해 평가한다. 트리의 정확도 평가는 이 연구에서 제시한 적응형 의사결정 트리(ADT)와 C4.5, ID5R에 의해 구축된 의사결정 트리가 데이터의 수가 증가함에 따라 정확도가 어떻게 변하는지를 평가하였다. <그림 15>는 데이터의 수에 따른 의사결정 트리의 정확도를 나타낸 그래프로서 전반적으로 ID5R의 정확도가 좋다는 것을 알 수 있다. 이는 ID5R의 트리 구축이 기본적으로 현재의 트리와 일치하지 않는 데이터가 들어왔을 경우 트리를 확장해 나가는 방법을 취하기 때문에 입력된 데이터에 대해 100%의 정확도를 보장할 수 있다. 그러나 이러한 방법은 트리의 지나친 확장으로 인한 과도 적합 문제를 야기하게 되는데 ID5R의 과도 적합에 대한 문제는 아래의 트리 복잡도에서 설명한다.

7.2.2 트리의 복잡도 비교

트리의 복잡도는 노드의 수나 트리의 깊이를 통해 평가할 수 있는데 복잡한 트리는 의미를 해석하기 힘들 뿐 아니라 트리 구축 및 재구성 비용이 많이 드는 문제점을 지닌다. 이 연구에서는 트리의 복잡도 평가를 위해 ADT와 C4.5, ID5R에 의해 구축된 의사결정 트리의 노드 수가 데이터의 증가에 따라 어떻게 변하는지를 평가하였다. <그림 16>은 데이터의 수에 따른 의사결정 트리의 노드 수를 나타낸 그래프로서 이 연구에서 제안하는 ADT가 가장 간결한 트리를 구축함을 알 수 있다. <그림 16>에서 보는 바와 같이 ID5R은 지나치게 세분화된 트리를 생성함으로서 과도 적합 문제를 야기하는데 특히 입력되는 데이터가 오류 데이터일 경우 비효율적인 트리 확장을 수행하는 문제가 발생한다.

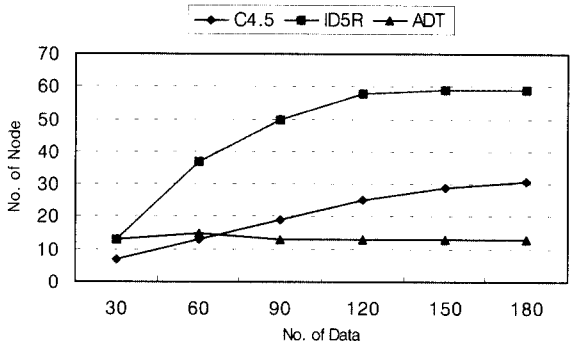


그림 16. 의사결정 트리의 노드 수 비교.

7.2.3 트리의 강건성 비교

정적 의사결정 트리인 C4.5는 새로운 데이터가 입력될 때마다 기존의 트리를 버리고 새로운 트리를 구축해야 하는 문제점을 지닌다.

<그림 17>은 데이터가 증가함에 따라 구축된 의사결정 트리의 재구성이 얼마나 빈번히 발생했는지를 나타내는 그래프로서 정적 의사결정 트리인 C4.5는 입력된 데이터의 수만큼 트리의 재구성이 발생함을 알 수 있고 ID5R은 오류 데이터가 입력될 때에도 트리를 재구성하기 때문에 오류 데이터에 대한 처리를 수행하는 ADT에 비해 트리의 재구성 빈도가 높음을 알 수 있다.

8. 결 론

이 연구에서는 실시간으로 수집되는 기계 이력 데이터의 분석을 통해 현재의 기계 상태를 정확히 반영한 기계고장 정보를 추출해 낼 수 있는 적응형 점진적 의사결정 트리(ADT)의 구축 방법을 제시하였다. 이 연구에서 제안한 적응형 점진적 의사결정 트리의 구축 방법은 다음과 같은 특징을 갖는 의사결정 트리를 작성할 수 있도록 하였다.

- 실시간으로 수집되는 기계 이력 데이터에 따라 트리의 확장 및 변경이 가능한 점진적 의사결정 트리
- 오류 데이터의 감지 및 처리를 통해 불필요한 트리의 변경을 방지할 수 있는 적응형 의사결정 트리

이 연구에서 제시한 적응형 의사결정 트리는 정적 의사결정 트리인 C4.5가 새로운 데이터가 수집될 때마다 기존에 구축된 트리를 버리고 새로운 트리를 작성해야 하는 문제점을 해결하였고 점진적 의사결정 트리인 ID5R에 비해 트리의 복잡도와 강건성이 향상된 의사결정 트리가 구축될 수 있도록 하였다.

참고문헌

Adriaans, P. and Zantinge, D. (1996), *Data Mining*, Addison-Wesley.

Barr, D. and Mani, G. (1994), Using neural nets to manage investments, *AI Expert*, February, 16-21.

Bernhard, S., Burges, C. J. C., Smola, A. J. (1998), *Advances in Kernel Methods: Support Vector Learning*, MIT Press.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996), *Neuro Dynamic Programming*, Athena Scientific.

Bertsimas, D., Gamarnik, D., and Tsitsiklis, J. N. (1999), Estimation of time-varying parameters in statistical models: An optimization approach, *Machine Learning*, 35(3), 225-245.

Bishop, C. M. (1995), *Neural Network for Pattern Recognition*, Oxford Press.

Domingos, P. and Pazzani, M. (1997), Beyond independence: Conditions for the optimality of the simple bayesian classifier, *Machine Learning*, 29, 103-130.

Duda, R. O. and Hart, P. (1973), *Pattern Classification and Scene Analysis*, John Wiley and Sons.

Eubank, R. L. (1999), *Nonparametric Regression and Spline Smoothing*, Marcel Dekker.

Fayyad, U. M. and Irani, K. B. (1993), Multi-interval discretization of continuous-valued attributes for classification learning, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1022-1027, Morgan-Kaufmann.

Fayyad, U. M., Djorgovski, S. G., and Weir, N. (1996), *Automating the analysis and Cataloging of Sky Surveys, Advances in Knowledge Discovery and Data Mining(Eds. Fayyad, U. M. et al.)*, 471-494, AAAI Press.

Goldberg, D. (1989), *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison-Wesley.

Heckerman D. (1996), Bayesian networks for knowledge discovery, *Advances in Knowledge Discovery and Data Mining (Eds. Fayyad, U. M. et al.)*, 273-306, AAAI Press.

Koller, D. and Russell, S. J. (1995), Stochastic simulation algorithms for dynamic probabilistic networks, *Proceedings of the 11th Annual Conference on Uncertainty in AI*, Montreal, Canada, 346-351.

Mangasarian, O. L., Street, W. N., and Wolberg, W. H. (1995), Breast cancer diagnosis and prognosis via linear programming, *Operations Research*, 43(4), 570-577.

Mitchell, T. M. (1997), *Machine Learning*, McGraw -Hill.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers.

Rabiner, L. (1989), A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of IEEE*, 77, 257-285.

Sasisekharan R., Seshadri V., and Weiss, S. M. (1996), Data mining and Forecasting in large-scale telecommunication Network, *IEEE Expert*, February, 37-43.

Utgoff, P. E. (1989), Incremental induction of decision trees, *Machine Learning*, 4, 161-186.