대한산업공학회/한국경영과학회 2000 춘계공동학술대회 논문집. 2000년4월21-22일. 경남대학교
Session OB5.2

기계 진단을 위한 적응형 의사결정 트리 알고리즘

Adaptive Decision Tree Algorithm for Machine Diagnosis

백준걸*[1], 김강호[1], 김창욱[2], 김성식[1]
[1]고려대학교 산업공학과 [2]명지대학교 산업시스템공학부

This article presents an adaptive decision tree algorithm for dynamically reasoning machine failure cause out of real-time, large-scale machine status database. 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

본 연구에서는 실시간으로 수집되는 기계 상태 데이터로부터 기계 고장 원인을 점진적으로 발견하는 데이터 마이닝 방법으로 적응형 의사결정 트리인 ADT(Adaptive Decision Tree) 알고리즘을 제시한다. 의사결정 트리의 구축에 관련된 대표적인 연구로는 C4.5(Quinlan[4])와 ID5R(Utgoff[5])을 들 수 있다. C4.5는 검사 속성 선택 척도로서 정보획득량(Information Gain)을 사용하여 고정된 데이터 집합을 기반으로 의사결정 트리를 구축하는 방법으로서 실시간으로 수집되는 기계 상태 데이터의 변화를 정확히 반영하지 못하는 문제점을 지닌다. ID5R은 데이터가 추가될 때 E-score를 척도로 사용하여 기존에 구축된 의사결정 트리를 확장할 수 있는 방법을 제시하지만 데이터가 수집되는 순서에 매우 민감하고 의사결정 트리의 잦은 재구성으로 인한 계산 부하가 증가하는 단점을 지닌다. 따라서 본 연구에서는 위에서 언급한 C4.5와 ID5R의 문제점을 해결할 수 있는 새로운 적응형 의사결정 트리의 구축 방법을 제시하고, 이를 통해 실시간으로 수집되는 기계 상태 데이터를 효율적으로 분석하여 기계 고장을 예측할 수 있도록 한다.

2. 의사결정 트리 정의

의사결정 트리 문제를 구체적으로 정의하면 다음과 같다. 데이터는 n개의 속성(Attribute) 값 a_i, $i=1,2,,,n$와 결과 클래스(Class) 값 c로 정의되며, 데이터 집합 D에 속한 i번째 데이터 d_i, $i=1,2,,,|D|$는 $(a_{i1}, a_{i2},,, a_{in}, c_i)$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.

의사결정 트리는 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.

3. 적응형 의사결정 트리

본 연구에서 제시하는 적응형 의사결정 트리인 ADT는 다음과 같은 절차에 의해 구축되어진다.

단계 1) 초기 의사결정 트리 구축
본 연구에서 제안하는 속성 선택 척도인 SDM(Sum of Dominance Metric)을 이용하여 이미 수집된 데이터 집합에 대한 초기 의사결정 트리를 구축한다.

단계 2) 군 데이터 구성
너무 잦은 트리의 갱신을 방지하고 오류 데이터의 검사가 용이하도록 수집되는 데이터를 일정 크기의 군으로 분할한다.

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 2로 가고(현재 트리를 보존), 그렇지 않으며 단계 3.2로 간다.
단계 3.2) 노이즈(Noise) 데이터 검사
데이터 군에 노이즈 데이터가 섞여있는지를 판단한다. 노이즈 데이터가 있으면 이 데이터를 제외하고 단계 4로 간다.

단계 4) 의사결정 트리의 갱신(변경 및 확장)
SDM 척도를 이용하여 트리를 변경하거나 확장한다.

3.1 군 데이터 구성 및 적합도 검사

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

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

where

$|B|$: 데이터 군 B의 크기

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

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

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 c에 의해 결정되는데 $|B|$와 c의 값은 계수형 품질 관리에서 사용하는 $(|B|, c)$ 방식을 응용한 것으로 데이터 군(B)의 오류 데이터 수가 c개를 초과하면 현재 트리에 문제가 있는 것으로 판별한다.

3.2 노이즈 데이터 검사

본 연구에서 제안하는 ADT는 노이즈 데이터의 입력으로 인한 비효율적인 트리의 재구성을 방지하기 위해 노이즈 데이터 검사를 수행한다. 노이즈 데이터는 연속형 검사 속성의 분할점에 의해 데이터가 부분 집합으로 나뉘어 질 때 사후(Posterior) 확률 분포 $P(c\,|a_1, a_2, , , a_n)$ 특성상 분할점 부근에 자연스럽게 발생하는 것으로써 노이즈를 확정적 함수인 트리를 사용해서 무리하게 맞추려고 하면 분할점이 무수히 많아지고 트리가 과도 적합(Overfitting)하게 되어 오히려 트리의 정확도가 떨어지는 문제점을 발생시킨다.

3.2.1 연속형 속성의 노이즈 처리

연속형 속성의 노이즈 데이터 처리를 위해서 본 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.

$$If\ ((a_{ij}-\theta_l \leq offset)\ and\ (\frac{E_c(\theta_l)}{E_c(\theta_l)+N_c(\theta_l)} < \zeta_1))$$

Then 노이즈로 처리

where

$offset = \lambda_l \cdot \omega$

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

$mean(a_{\cdot j})$: j번째 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 첫 번째 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.

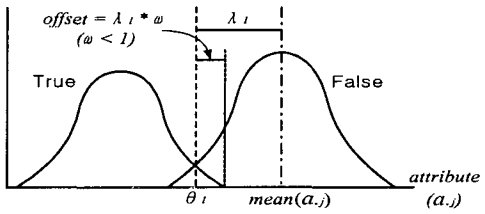

그림 1. *offset* 정의

3.2.2 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 본 연구에서는 이산형 검사 속성에 대한 노이즈 데이터를 검사하기 위해 순수 베이지안 분류식(Naive Bayesian Classifier)을 이용한다.

순수 베이지안 분류식(Mitchell[3])은 결과 클래스($c_j \in C$)가 주어졌을 때 결합 우도 확률(Joint Likelihood Probability) $P(a_1, a_2, \ldots, a_n|c_j)$가 조건부 독립(Conditional Independence)을 보장한다는 가정 하에 다음과 같은 베이지안 공식을 사용하여 사후 확률 $P(c_j|a_1, a_2, \ldots, a_n)$을 계산한다.

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

순수 베이지안 분류식은 각 결과 클래스마다 사후 확률을 계산한 후 확률 값이 가장 큰 클래스를 최대 사후 결과 클래스 c_{MAP}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 $P(a_1, a_2, \ldots, a_n|c_j)$가 c_j에 대해 조건부 독립이라는 가정에도 불구하고 상당히 정확한 결과를 보인다.(Domingos and Pazzani[1])

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 본 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 결과클래스 $c_j \in C-\{c_{MAP}\}$로 구분하고 이들의 사후 확률의 차이를 다음 식과 같이 계산하여 그 차이가 기준값 ξ_2보다 작을 경우 트리의 갱신을 실시한다.

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

3.3 의사결정 트리의 갱신

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

4. 검사 속성 선택 척도

초기 트리 구성 및 기존 트리 갱신에는 검사 속성 선택 기준이 있어야 한다. 본 연구에서는 SDM(Sum of Dominance Metric)이라는 속성 선택 척도를 제시한다. SDM은 기존의 엔트로피 방식이나 베이지안 방식에 비해서 계산 속도가 빠르기 때문에 적응형 의사결정 트리의 구축에 적합하다. 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.

4.1 DM_Array

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

4.1.1 이산형 검사 속성의 DM_Array 구성

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

$$DM\_Array(i) = (n_{i11}, n_{i12}, \ldots, n_{i1|C|}, n_{i21}, n_{i22}, \ldots, n_{ijc}, \ldots, n_{ia_i|C|})$$

where

i : i번째 속성

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

$|a_i|$: a_i가 가질 수 있는 속성 값의 개수

4.1.2 연속형 검사 속성의 DM_Array 구성

연속형 검사 속성의 경우 이산형 속성과 달리 가질 수 있는 값이 무한히 많기 때문에 연속형 속성이 가질 수 있는 값의 범위를 구간으로 나누고 각 구간에 속한 결과 클래스 수를 통해서 DM_Array를 구성한다. 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[4])으로 속성이 갖는 값들을 오름차순으로 정렬한 후 중간점들을 분할점 집합 Θ로 선택한다. 둘째는 중간점 방식의 단점을 해결하기 위해 제시된 경계점(Boundary Point) 방식(Fayyad and Irani[2])으로 결과 클래스의 값이 바뀌는 중간점들 만을 분할점 집합 Θ로 선택한다. 본 연구에서는 연속형 속성이 가질 수 있는 값을 오름차순으로 정렬한 AV_List를 정의한 후 경계점 방식을 이용하여 분할점을 선택한다.

경계점 방식을 이용하여 연속형 검사 속성에 대한 DM_Array를 구성하는 경우 새로운 데이터가 추가될 때 AV_List가 변하게 되고 이에 따라 DM_Array가 갱신되어야 하는데, 본 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

Case3.2 $AV\_List[2][i-1] = AV\_List[2][i+1]$

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

4.2 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 노드의 엔트로피가 낮음을 의미한다. 본 연구에서는 데이터를 구성하는 각각의 속성에 대해서 SDM 값을 계산한 후 가장 큰 SDM 값을 갖는 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.

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

where

i : i번째 속성

$|a_i|$: 속성 i가 가질 수 있는 속성 값의 수

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

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

위 식에서 절대값 안의 식은 우세(Dominant) 결과 클래스(c^*)를 갖는 데이터의 수와 나머지 클래스들을 갖는 데이터의 수를 합한 것의 차이를 의미하며 가중치 w_j는 같은 차이라 하더라도 동질성이 큰 속성에 선택 비중을 두기 위함이다. 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.

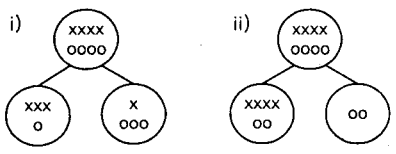

그림 2. 가중치의 의미

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

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

where

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

$L(\theta_l)$: 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.

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

5. 결 론

본 연구에서는 실시간으로 수집되는 기계 이력 데이터의 분석을 통해 현재의 기계 상태를 정확히 반영한 기계고장 정보를 추출해 낼 수 있는 적응형 의사결정 트리(ADT)의 구축 방법을 제시하였다.

본 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

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

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

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

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

5.Utgoff, P. E., "Incremental induction of decision trees", *Machine Learning*, Vol.4, pp. 161-186, 1989.