Journal of the Korean Institute of Industrial Engineers
Vol. 27, No. 4, pp. 394-405, December 2001.

증분 의사결정 트리 구축을 위한 연속형 속성의 다구간 이산화

백준걸[1†] · 김창욱[2] · 김성식[3]

[1]고려대학교 정보통신기술공동연구소 / [2]명지대학교 산업시스템공학부/ [3]고려대학교 산업공학과

Multi-Interval Discretization of Continuous-Valued Attributes for Constructing Incremental Decision Tree

Jun-Geol Baek[1] · Chang-Ouk Kim[2] · Sung-Shick Kim[3]

[1]Research Institute for Information and Communication Technologies, Korea University, Seoul, 136-701

[2]Department of Industrial and Systems Engineering, Myongji University, Yongin, 449-728

[3]Department of Industrial Engineering, Korea University, Seoul, 136-701

Since most real-world application data involve continuous-valued attributes, properly addressing the discretization process for constructing a decision tree is an important problem. A continuous-valued attribute is typically discretized during decision tree generation by partitioning its range into two intervals recursively. In this paper, by removing the restriction to the binary discretization, we present a hybrid multi-interval discretization algorithm for discretizing the range of continuous-valued attribute into multiple intervals. On the basis of experiment using semiconductor etching machine, it has been verified that our discretization algorithm constructs a more efficient incremental decision tree compared to previously proposed discretization algorithms.

Keywords: continuous-valued attribute, multi-interval discretization, incremental decision tree

1. 서 론

기계 상태 데이터의 분석을 통해 기계 고장 원인에 관한 정보를 추론하는 것은 기계 고장 진단 시스템 구축에 있어서 매우 중요한 과제이다. 이러한 문제는 기존의 신경망 (neural network), 퍼지 이론 (fuzzy logic), 또는 사례기반 추론 (case- based reasoning) 등 여러 가지 방법을 이용하여 연구되어 왔으나, 의사결정 트리 (decision tree)는 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 제공한다.

의사결정 트리는 데이터를 구성하는 속성 (attribute)과 클래스(class)와의 연관 관계를 규명하기 위해 데이터 집합 (data set)을 부분 집합 (subset)으로 분할하고, 분할된 부분 집합의 특성을 규명하는 데 사용되는 방법론으로서 의사결정 트리 문제를 구체적으로 정의하면 다음과 같다.

데이터는 n개의 속성값 a_i, $i=1,2,\cdots, n$과 결과 클래스 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. 데이터를 구성하는 각각의 속성값은 이산형(discrete) 또는 연속형(continuous) 값을 가질 수 있으며, 결과 클래스 값은 이산형 집합 C에서 하나의 원소 값을 갖는다. 따라서 의사결정 트리 문제는 속성값과 결과 클래스 값의 직교 곱(cartesian product)으로 표현되는 데이터 공간에서 데이터 집합 D가 주어졌을 때, 이 데이터 집합을 각 속성값에 따라 부분 집합으로 분할하는 것이며, 이때 분할 기준은 각 부분 집합에 포함된 모든 데이터가 최대한 같은 결과 클래스 값을 갖도록 하는 것이다. 의사결정 트리는 <그림 1>에서 보듯이 그래프의 일종으로 단말 노드(leaf node)를 제외한 중간 노드(이하 결정 노드

본 연구는 고려대학교 특별연구비 지원으로 수행되었음.
† 연락저자 : 백준걸 연구교수, 136-701 서울시 성북구 안암동 고려대학교 정보통신기술공동연구소, Fax : 02-929-5888, e-mail : jgbaek@korea.ac.kr
2001년 9월 접수, 1회 수정 후 2001년 10월 게재 확정.

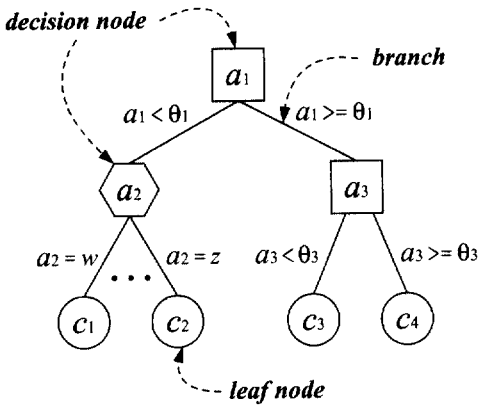

그림 1. 의사결정 트리.

(decision node)라 함)는 분할 기준이 되는 속성을 의미하고, 단말 노드는 상위 결정 노드 패스(path)에 의해 분류된 결과 클래스들의 집합을 의미한다. 가지(branch)는 결정 노드의 분할 기준이 되는 속성의 분할점(splitting point)이 된다.

이와 같은 문제는 탐욕 (greedy) 알고리듬을 이용한 최적화 문제와 거의 일치한다. 결정 변수는 의사결정 트리에서 확장될 다음 결정 노드의 속성 (이하 검사 속성이라 함)과 검사 속성의 분할점이며, 최적 검사 속성은 주어진 데이터들을 최대한 같은 결과 클래스를 갖는 부분 집합으로 분류할 수 있는 것으로 선택한다. 효율적인 의사결정 트리를 구축하기 위한 최적 검사 속성을 선택하기 위해서는 각각의 속성에 대한 분할점을 정의하는 일이 무엇보다 중요하다. 이산형 속성의 경우 분할점으로 사용되는 값은 해당 속성이 가질 수 있는 속성값이 되기 때문에 유한(finite)하고 정의하기 쉽다. 그러나 연속형 속성의 경우 분할점으로 사용될 수 있는 값이 무한히 많기 때문에 가능한 모든 속성값에 대해 분할점 사용 여부를 평가하는 것은 무의미하다. 다시 말하면, 분류 문제 (classification problem)를 해결하기 위해 사용되는 의사결정 트리는 원래 이산형 속성을 대상으로 설계된 학습 방법이므로 연속형 속성을 포함하는 데이터 집합을 대상으로 학습을 수행하기에는 많은 어려움이 따른다 (Quinlan, 1996). 따라서 연속형 속성을 포함하는 데이터를 기반으로 의사결정 트리를 구축하기 위해서는 연속형 속성이 가질 수 있는 값의 범위를 이산화 (discretization)시킴으로서 연속형 속성을 이산형 속성과 같은 형태로 변환하여 분할점을 쉽게 찾을 수 있도록 하는 일이 중요하다 (Ching and Wong, 1995; Loh and Shih, 1997).

연속형 속성을 이산화하기 위한 일반적인 방법으로는 연속형 속성의 값을 두 개의 부분 구간으로 분할하는 이진(binary) 이산화 방법이 있다. Quinlan (1993)은 고정된 (static) 데이터를 기반으로 의사결정 트리를 구축할 때, 결정 노드에서 분할 기준이 되는 최적 검사 속성을 선택하기 위한 방법으로 정보 획득량 (information gain)을 이용하는 방법을 제시하고 있다. Qunilan이 제시한 정보 획득량은 정보 이론 (information theory)의 엔트로피 (entropy) 개념을 사용한 것으로서 엔트로피 값은 데이터의 분할이 잘 될수록 작은 값을 갖는 특성을 지닌다. 따라서 검사 속성 후보 중에서 엔트로피 값이 최소가 되는 속성을 해당 결정 노드의 검사 속성으로 사용하게 되는데, 연속형 속성에 대한 정보 획득량을 계산하기 위한 방법으로 연속형 속성의 값을 두 개의 부분 구간으로 분할하였을 때 얻을 수 있는 정보 획득량을 계산하는 방법을 제시하고 있다. 그러나 Quinlan이 제시한 의사결정 트리 구축 방법은 고정된 데이터를 기반으로 의사결정 트리를 구축하는 비증분형 (non-incremental) 의사결정 트리 구축 방법으로서 본 연구에서와 같이 기계 상태 데이터가 실시간으로 수집되는 경우 추가로 수집되는 데이터를 반영한 의사결정 트리를 구축하기 위해서는 데이터가 추가될 때마다 기존의 트리를 버리고 처음부터 다시 새로운 트리를 구축해야 하는 문제점을 지니고 있다. 또한 연속형 속성에 대한 정보 획득량을 계산하기 위해서 연속형 속성의 구간을 두 개의 부분 구간으로 분할하는 이진 이산화 방법을 사용함으로써 완전한 이산화를 위해서는 이진 이산화를 반복적(recursive)으로 수행해야 하는 문제점을 지니고 있다.

비증분형 의사결정 트리의 문제점을 해결하기 위한 방법으로 데이터가 증가하더라도 기존에 구축된 의사결정 트리를 버리지 않고 데이터가 추가될 때마다 기존의 트리를 수정해 나가는 증분 (incremental) 의사결정 트리 구축에 대한 대표적인 연구로는 ID5R (Utgoff, 1989; Utgoff, 1997)을 들 수 있다. ID5R은 새로운 데이터가 추가될 때마다 엔트로피를 나타내는 E-score를 척도 (metric)로 사용하여 기존의 트리를 재구성하는 방법을 제시하고 있으며, 풀-업 (pull-up) 알고리듬을 이용하여 데이터가 추가될 때마다 속성별 E-score를 갱신하고 이를 기준으로 기존에 구축된 의사결정 트리를 수정함으로서 추가로 입력되는 데이터의 특성을 반영한 의사결정 트리를 구축할 수 있도록 하였다. 그러나 ID5R 또한 연속형 속성에 대한 이산화를 위해 이진 이산화 방법을 사용함으로서 완전한 이산화를 위해 이진 이산화를 반복적으로 수행해야 하는 문제점을 지니고 있다. 이러한 반복적 이진 이산화를 통한 의사결정 트리의 구축은 연속형 속성에 대해 반복적인 분할 작업을 수행함으로서 구축되는 트리의 깊이 (depth)가 깊어지는 문제점을 유발한다 (Catlett, 1991).

Baek *et al.* (2000)은 기계 상태 데이터가 실시간으로 수집되고, 수집되는 데이터에 노이즈 데이터가 포함되어 있는 경우 효율적인 증분 의사결정 트리를 구축하기 위한 방법으로 적응

형 의사결정 트리(adaptive decision tree) 구축 방법을 제시하고 있다. 이 연구에서 제안하는 적응형 의사결정 트리는 데이터 전처리 단계를 통해 노이즈 데이터를 효율적으로 제거하고, SDV(Sum of Dominant Values)라는 새로운 척도를 이용하여 효율적인 증분 의사결정 트리를 구축하는 방법을 제시하고 있다. 그러나 Baek et al.이 제안한 적응형 의사결정 트리 구축 방법도 ID5R과 마찬가지로 연속형 속성의 이산화를 위해 이진 이산화를 실시함으로서 완전한 이산화를 위해서는 반복적인 이진 이산화를 실시해야 하는 문제점을 여전히 지니게 된다.

Fayyad and Irani (1993)는 위에서 언급한 이진 이산화의 문제점을 해결하기 위한 방법으로 엔트로피를 이용한 다구간(multi-interval) 이산화 방법을 제시함으로서 반복적인 이진 이산화로 인해 의사결정 트리의 깊이가 깊어지는 문제점을 해결할 수 있도록 하였다. 그러나 Fayyad and Irani가 제시한 다구간 이산화 방법은 고정된 데이터를 기반으로 하향식(top-down) 이산화를 실시하는 방법으로서, 본 연구에서와 같이 실시간으로 수집되는 기계 상태 데이터를 기반으로 의사결정 트리를 구축하기 위해서는 데이터가 추가로 수집될 때마다 연속형 속성에 대한 이산화 작업과 의사결정 트리 구축 작업을 처음부터 다시 실시해야 하는 문제점을 지닌다.

따라서 본 연구에서는 실시간으로 수집되는 기계 상태 데이터를 기반으로 효율적인 증분 의사결정 트리를 구축하기 위한 연속형 속성의 다구간 이산화 방법을 제시하고자 한다. 본 연구에서 제시하는 연속형 속성의 다구간 이산화 방법은 하향식 이산화와 상향식(bottom-up) 이산화가 혼용된 혼합형(hybrid) 이산화 방법으로서 새로운 데이터가 추가로 입력되는 경우 입력된 데이터와 관련된 부분 구간에 대해서만 이산화 작업을 추가로 실시하고, 이를 기반으로 효율적인 증분 의사결정 트리를 구축하는 방법을 제시하고 있다.

2. 연속형 속성의 이산화

연속형 속성을 이산화하기 위한 일반적인 방법으로는 연속형 속성의 값을 두 개의 부분 구간으로 분할하는 이진 이산화 방법이 있다. 그러나 이진 이산화는 구축되는 의사결정 트리의 깊이가 깊어지는 문제점을 유발하기 때문에 이러한 문제를 해결하기 위한 방법으로 다구간 이산화 방법이 제안되었다. 다구간 이산화를 이용하여 구축한 의사결정 트리는 이진 이산화를 이용하여 구축한 의사결정 트리에 비해 트리의 깊이가 적은 간단한(simple) 트리를 구축할 수 있다는 장점을 지닌다 (Fayyad and Irani, 1993).

이진 이산화와 다구간 이산화의 특징은 다음과 같은 예제를 통해 설명할 수 있다. 연속형 속성 a가 <그림 2>와 같은 값을 갖고 이들이 오름차순으로 정렬되어 있다고 가정하자. <그림 2>에서 표기된 수치는 속성 a를 이산화할 때 사용될 수 있는

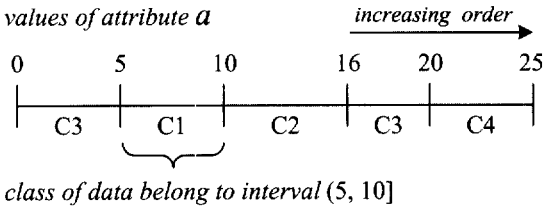

그림 2. 연속형 속성 a의 속성값.

분할점을 나타낸 것이고, 각 구간별로는 해당 구간별 결과 클래스를 표기하였다.

<그림 2>와 같은 값을 갖는 연속형 속성은 반복적 이진 이산화를 통해 <그림 3>과 같은 부분 집합으로 분할될 수 있으며 이를 통해 구축된 의사결정 트리는 <그림 4>와 같다. <그림 3>에서 반복적 이진 이산화는 우선적으로 속성 a의 값을 분할점 10을 기준으로 2개의 부분 구간으로 분할한 후 이들을 각각 분할점 5와 16에 의해 다시 2개의 부분 구간으로 분할하는 것을 나타낸다. 또한 16에 의해 분할된 부분 구간은 다시 분할점 20에 의해 2개의 부분 구간으로 분할되어 모두 5개의 부분 구

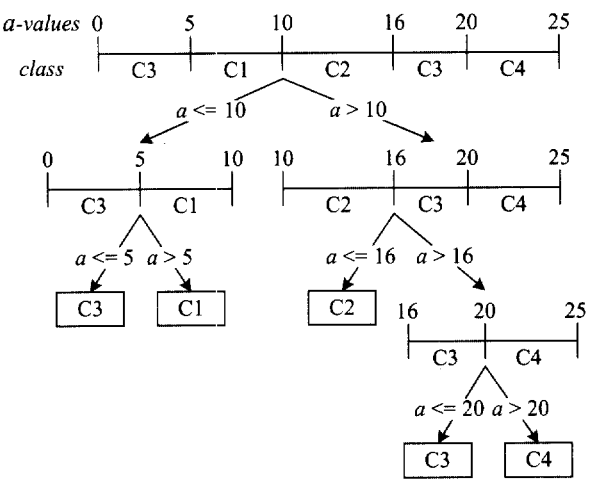

그림 3. 반복적 이진 이산화.

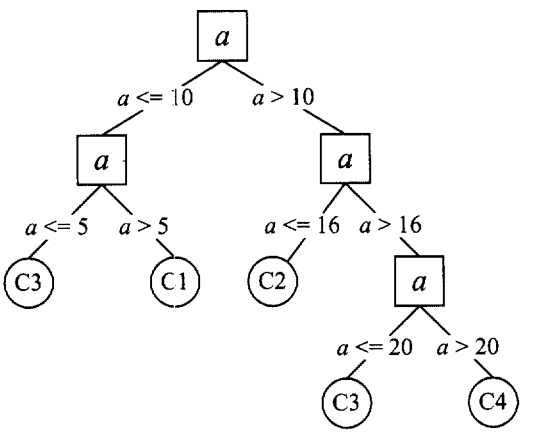

그림 4. 이진 이산화를 이용한 의사결정 트리.

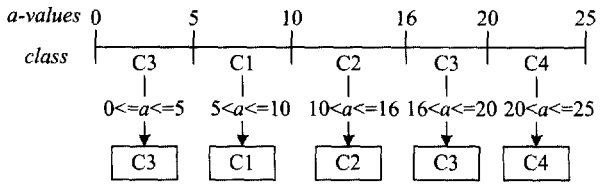

그림 5. 다구간 이산화.

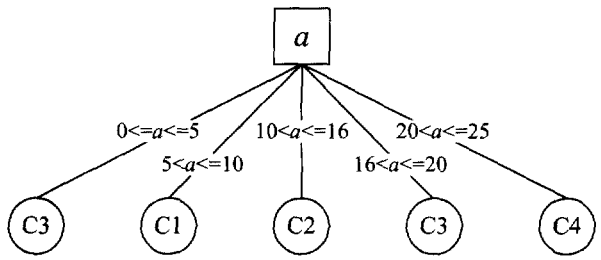

그림 6. 다구간 이산화를 이용한 의사결정 트리.

간으로 분할되고 있음을 나타낸다. 따라서 반복적 이진 이산화에 의해 구축된 의사결정 트리는 <그림 4>와 같고 트리의 깊이는 3이 됨을 알 수 있다.

<그림 2>와 같이 주어진 연속형 속성에 대해 다구간 이산화 방법을 적용하는 경우, 부분 집합의 분할은 <그림 5>와 같이 수행되어지고 이를 통해 구축된 의사결정 트리는 <그림 6>과 같다. <그림 6>에서 보듯이 다구간 이산화에 의해 구축된 의사결정 트리는 깊이가 1로서 이진 이산화에 의해 구축된 의사결정 트리보다 간결한 구조를 지닌다. 즉, 다구간 이산화에 의해 구축된 의사결정 트리는 데이터의 분할을 위해 단 한번의 속성 검사를 실시하기 때문에 이진 이산화를 통해 구축된 의사결정 트리에 비해 빠른 시간에 분류 작업을 수행할 수 있다는 장점을 지닌다.

2.1 이산화 방법론의 분류

연속형 속성에 대한 이산화 작업은 방법론의 특성에 따라 다음과 같은 5가지 범주로 구분해 볼 수 있다(Dougherty *et al.*, 1995; Cerquides and Mantaras, 1997).

2.1.1 속성의 수에 따른 분류

연속형 속성에 대한 이산화 작업을 수행하는 데 있어서 대상이 되는 속성의 수에 따라 지역적(local) 방법론과 전역적(global) 방법론으로 구분할 수 있다. 지역적 방법론은 여러 개의 연속형 속성 중에서 부분적인 속성에 대한 이산화 작업을 수행하는 것을 의미하고 전역적 방법론은 모든 연속형 속성에 대한 이산화 작업을 한꺼번에 수행하는 것을 의미한다.

2.1.2 결과 클래스 정보의 이용에 따른 분류

이산화 작업을 수행할 때 데이터가 지니고 있는 결과 클래스 정보를 이용하는지 여부에 따라 교사(supervised) 방법과 비교사(unsupervised) 방법으로 분류할 수 있다. 교사 방법은 데이터가 지니고 있는 결과 클래스의 정보를 이용하여 이산화 작업을 수행하는 것을 의미하고, 비교사 방법은 결과 클래스 정보를 이용하지 않고 이산화 작업을 수행하는 것을 의미한다.

2.1.3 구간의 변동성에 따른 분류

이산화 작업에 의해 분할될 구간(interval)의 수가 고정적인지 아닌지에 따라 정적(static) 방법과 동적(dynamic) 방법으로 분류할 수 있다. 정적 방법에 의한 이산화 작업은 분할될 구간의 수를 주어진 값 (k)으로 고정하여 분할을 수행하는 것을 의미하고 동적 방법에 의한 이산화 작업은 상황에 따라 변화하는 구간의 수에 의해 분할을 수행하는 것을 의미한다.

2.1.4 분할점의 수에 따른 분류

이산화 작업을 수행하는 데 있어서 가장 중요한 일은 구간의 분할을 위해 사용되는 분할점을 어떻게 결정할 것인가에 대한 문제를 해결하는 것이다. 분할점을 결정할 때 한번 결정된 분할점을 영구적으로 사용하는 방법을 직접적인(direct) 방법이라 정의하고, 데이터가 증가함에 따라 변화하는 분할점을 찾아내서 사용하는 방법을 증분형(incremental) 방법이라고 정의한다.

2.1.5 분할점을 정의하는 방법에 따른 분류

증분형 방법을 이용하여 이산화 작업을 수행하기 위해서는 필요한 분할점을 찾아내는 일이 중요하다. 분할점을 찾아내기 위한 방법으로는 새로운 분할점을 계속적으로 추가해 나가는 하향식 (top-down) 방법과 모든 가능한 분할점을 찾아낸 후 필요 없는 분할점을 제거해 나가는 상향식(bottom-up) 방법이 있다.

본 연구에서는 실시간으로 수집되는 기계 상태 데이터를 기반으로 효율적인 증분 의사결정 트리를 구축하기 위한 연속형 속성의 다구간 이산화 방법으로서 지역적, 교사, 동적, 증분형, 그리고 혼합형 (hybrid) 이산화 방법을 제시하고자 한다. 여기서 혼합형 방법론이란 하향식 방법론과 상향식 방법론을 절충한 방법론을 의미한다.

2.2 다구간 이산화 알고리듬

본 연구에서 제시하는 다구간 이산화 알고리듬은 구간 분할을 위해 사용되는 분할점을 선택하기 위한 척도로서 *Kullback-Leibler* 거리(Cover, 1991)를 사용하였고, <그림 7>에 나타난 바와 같이 상향식 방법과 하향식 방법을 혼합한 형태의 구조를 지닌다.

2.2.1 *Kullback-Leibler* 거리

본 연구에서 제시하는 다구간 이산화 방법론은 연속형 속성

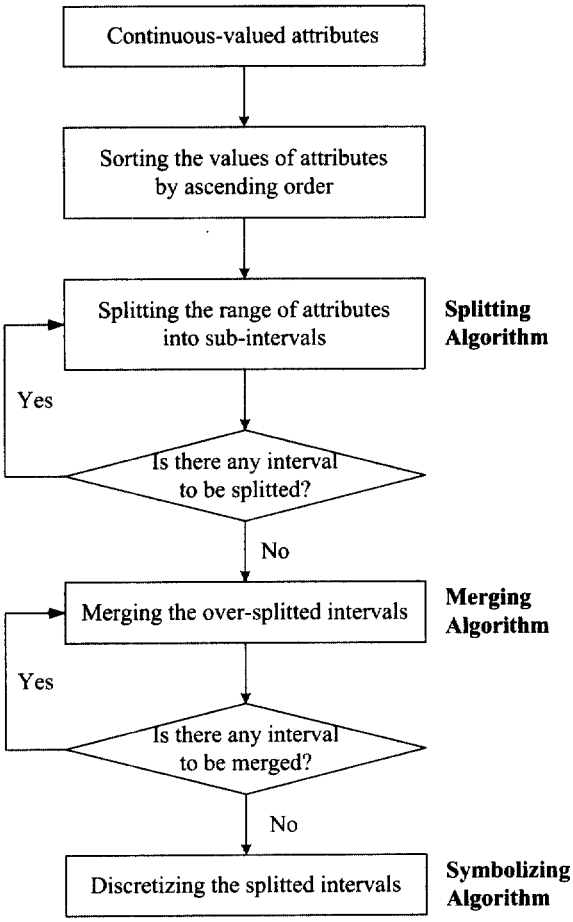

그림 7. 다구간 이산화 알고리듬.

이 가질 수 있는 값을 다수의 부분 구간으로 분할할 때 *Kullback-Leibler* 거리를 기준으로 이산화 작업을 수행하는 방법이다.

Kullback-Leibler 거리는 통계학에서 두 개의 샘플 집합(sample set) 사이의 유사성을 평가하기 위해 사용되는 척도로서 본 연구에서는 이산화를 위해 분할될 두 개의 부분 구간 사이의 유사성을 평가하기 위한 척도로 사용한다.

Kullback-Leibler 거리를 이용한 부분 구간의 분할은 각 구간에 속한 데이터에 대해 클래스별 확률 값을 계산한 후 이를 통해 두 개의 부분 구간 사이의 유사성을 평가하는 방법으로서 *Kullback-Leibler* 거리가 0이 되는 경우 두 구간의 클래스별 확률 값이 일치하는 것을 의미하고 두 구간의 특성은 같다고 평가할 수 있다. 만약 *Kullback-Leibler* 거리가 매우 큰 경우는 두 구간의 클래스별 확률 값의 차이가 크다는 것을 의미하고 이를 근거로 두 구간의 특성이 전혀 다르다고 평가할 수 있다.

본 연구에서 구간 분할의 기준으로 사용하는 *Kullback-Leibler* 거리는 식 (1)과 같이 정의할 수 있다. 식 (1)에서 $KD(L\|R)$은 연속형 속성의 구간을 두 개의 부분 구간 $\{L,R\}$로 분할하였을 때 두 구간 사이의 *Kullback-Leibler* 거리를 나타내는 값이다.

$$KD(L\|R) = \sum_{c=1}^{|C|} P_L'(c)\log_2\frac{P_L'(c)}{P_R'(c)} = \sum_{c=1}^{|C|} P_L'(c)\{\log_2 P_L'(c) - \log_2 P_R'(c)\} \quad (1)$$

where

$$P_L'(c) = \frac{N_L(c)}{\sum_{c=1}^{|C|} N_L(c)},\ P_R'(c) = \frac{N_R(c)}{\sum_{c=1}^{|C|} N_R(c)}$$

$N_L(c)$: 구간 L에서 클래스가 c 인 데이터 수

$N_R(c)$: 구간 R 에서 클래스가 c 인 데이터 수

예를 들어 <그림 8>과 같은 두 개의 구간이 존재할 경우, *Kullback-Leibler* 거리 계산에 의해 그림 (i)은 두 구간이 비교적 유사한 특성을 지니고 있음을 알 수 있고, 그림 (ii)의 경우는 두 구간의 특성이 차이가 남을 알 수 있다.

(i) 두 구간의 특성이 유사한 경우

L (left region)	R (right region)
OXOOXXO	OXXOXXO

$P_L'(O) = 4/7 = 0.57$ $P_R'(O) = 3/7 = 0.43$
$P_L'(X) = 3/7 = 0.43$ $P_R'(X) = 4/7 = 0.57$

$KD(L\|R) = P_L'(O)*\{\log_2 P_L'(O) - \log_2 P_R'(O)\}$
$+ P_L'(X)*\{\log_2 P_L'(X) - \log_2 P_R'(X)\}$
$= 0.57*(\log_2 0.57 - \log_2 0.43) + 0.43*(\log_2 0.43 - \log_2 0.57)$
$= 0.057$

(ii) 두 구간의 특성이 상이한 경우

L (left region)	R (right region)
XXOXXXX	OOOOXOO

$P_L'(O) = 1/7 = 0.14$ $P_R'(O) = 6/7 = 0.86$
$P_L'(X) = 6/7 = 0.86$ $P_R'(X) = 1/7 = 0.14$

$KD(L\|R) = P_L'(O)*\{\log_2 P_L'(O) - \log_2 P_R'(O)\}$
$+ P_L'(X)*\{\log_2 P_L'(X) - \log_2 P_R'(X)\}$
$= 0.14*(\log_2 0.14 - \log_2 0.86) + 0.86*(\log_2 0.86 - \log_2 0.14)$
$= 1.885$

그림 8. *Kullback-Leibler* 거리 계산.

2.2.2 구간 분할 알고리듬

임의의 연속형 속성 a 가 가질 수 있는 값의 구간을 다구간으로 이산화하기 위한 방법으로 본 연구에서는 우선적으로 서로 상이한 특성을 갖는 구간을 분할하는 방법을 사용한다. 구간 분할을 위한 알고리듬은 앞에서 제시한 *Kullback- Leibler* 거리를 이용하여 두 구간의 특성이 서로 상이하다고 판단되는 경우 분할을 실시하게 된다. 본 연구에서 제시하는 구간 분할 알고리듬은 <표 1>과 같이 정리할 수 있다.

구간 분할 알고리듬인 MultiSplitting(*a, D, Splitting_List*)은

부분 구간의 분할을 위해 경계점(boundary point) T_i 별로 *Kullback-Leibler* 거리 $KD_{T_i}(L\|R)$를 계산한 후, 이 중 가장 큰 값을 갖는 경계점 T^*를 선택하여 이를 통해 전체 구간을 두 개의 부분 구간으로 분할하는 방법을 사용한다. 따라서 다구간 이산화를 수행하기 위해서는 속성값의 전체 구간이 최적의 부분 구간으로 분할될 때까지 반복적으로 분할 알고리듬을 적용해야 한다. <표 1>의 ①과 ②는 다구간 이산화를 위해 MultiSplitting(*a, D, Splitting_List*)이 반복적으로 수행되고 있음을 보여준다.

표 1. 구간 분할 알고리듬

MultiSplitting (*a, D, Splitting_List*)

a : continuous-valued attribute
D : a set of data on which *a* is defined
T : a set of boundary points
Splitting_List : pointer to a list of splitting points

Begin Algorithm
  $D \leftarrow$ **SORT** (*D, a*)
  $T \leftarrow$ **SelectBoundaryPoint** (*D*)
  FOR ($t=1$; $t<=|T|$; t^{++})
    /* *split range into 2 intervals* */
    $L \leftarrow \{d \in D | a(d) < T_t\}$
    $R \leftarrow D \sim L$
    /* *calculate* $KD(L\|R)$ for each T_t */
    $T^* = argmax_{T_t \in T}\ KD_{T_t}(L\|R)$
  IF ($KD_{T^*}(L\|R) \geq \beta$) **THEN**
    Splitting_List $\leftarrow T^*$
    MultiSplitting (*a, L, Splitting_List*) ····· ①
    MultiSplitting (*a, R, Splitting_List*) ····· ②
  Return *Splitting_List*
End

본 연구에서 제시하는 구간 분할 알고리듬인 MultiSplitting (*a, D, Splitting_List*)은 속성 *a*의 구간 분할을 위한 분할점 집합 (*Splitting_List*)을 정의할 때 분할점의 후보가 되는 값으로 경계점을 사용하였다. 경계점이란 <그림 9>에서와 같이 속성 *a*의 값을 기준으로 데이터를 오름차순으로 정렬하였을 때 결과 클래스가 변화하는 점을 의미한다.

즉, 속성 *a*의 값을 기준으로 오름차순으로 정렬되어 있는 데이터 집합 (*D*)에서 결과 클래스 값이 서로 다른 ($Class(d_1) \neq Class(d_2)$) 두 개의 데이터 $d_1, d_2 \in D$가 존재하고, 두 개의 데이터 사이에는 어떠한 데이터도 존재하지 않는다고 가정할 때, 경계점 T는 $\{a(d_1)+a(d_2)\}/2$로 정의된다. 경계점을 이용한 분할점의 결정 방식은 연속형 속성의 특성상 무수히 많은 값이 분할점의 후보로 사용됨으로서 발생되는 계산상의 비효율을 해결하여 빠른 시간에 분할점을 결정할 수 있도록 하는 장점을 제공한다. 분할점의 후보 값으로 경계점을 사용할 수 있는 이유는 구간 분할의 기준이 되는 *Kullback-Leibler* 거리 $KD_T(L\|R)$를 최대화시키는 값 T는 항상 경계점이라는 다음과 같은 정리를 통해 가능하다.

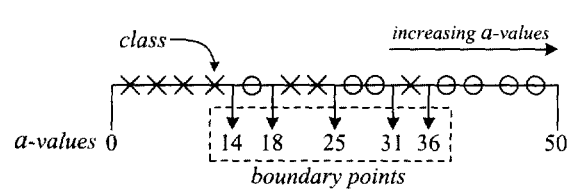

그림 9. 경계점의 정의.

정리. 임의의 값 T가 *Kullback-Leibler* 거리 $KD_T(L\|R)$를 최대화시키는 속성 a의 분할점이라면 T는 속성 a의 경계점이다.

(증명) <그림 10>은 연속형 속성 a의 값에 따라 오름차순으로 정렬한 데이터를 나타낸 것으로서 만약 T_1과 T_2가 경계점이라면 T_1과 T_2의 사이에는 동일한 클래스(C_k)를 갖는 $n \geq 1$개의 데이터가 위치하게 되고 T_1의 왼쪽으로는 L개의 데이터가 T_2의 오른쪽으로는 R개의 데이터가 위치한다고 가정할 수 있다.

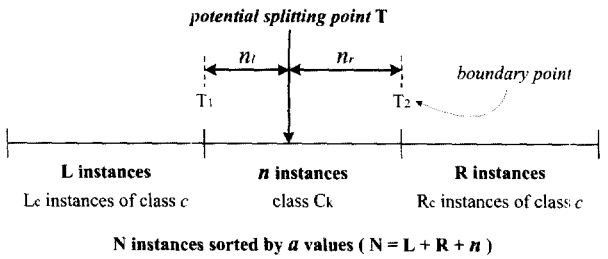

그림 10. 잠재적 분할점 T.

T_1의 왼쪽에 위치하는 L개의 데이터와 T_2의 오른쪽에 위치하는 R개의 데이터는 데이터가 갖는 클래스 값에 따라 식 (2), (3)과 같은 조건을 만족시킨다. 단, L_c는 데이터 집합 L 중에서 클래스 $c(c \in C)$를 갖는 데이터의 수를 의미하고 R_c는 데이터 집합 R 중에서 클래스 c를 갖는 데이터 수를 의미한다.

$$\sum_{c=1}^{|C|} L_c = L,\ 0 \leq L_c \leq L \tag{2}$$

$$\sum_{c=1}^{|C|} R_c = R,\ 0 \leq R_c \leq R \tag{3}$$

만약 T_1과 T_2 사이의 구간을 임의의 분할점 T에 의해 두 구간으로 나눈다면 n개의 데이터는 <그림 10>에서와 같이 n_l과 n_r로 나뉠 수 있고 이는 다음과 같은 특성을 만족시키게 된다.

$$n = n_l + n_r \text{ and } N = L + R + n$$

$KD_T(L\|R)$를 최대화시키는 속성 a의 분할점 T는 항상

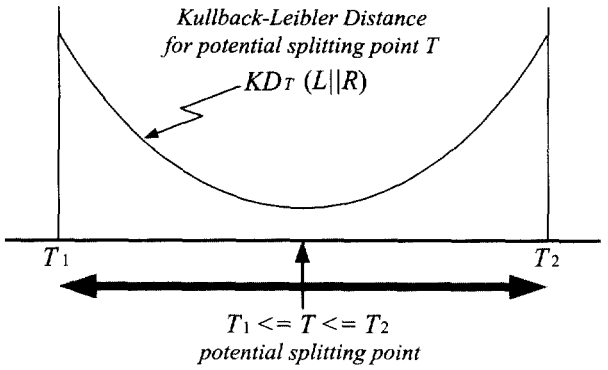

그림 11. $KD_T(L\|R)$.

경계점이라는 정리를 증명하기 위해서는 <그림 10>의 T가 반드시 T_1 또는 T_2 중의 하나여야 한다는 것을 증명해야 하며, 이는 $KD_T(L\|R)$ 함수가 <그림 11>에서와 같이 T_1과 T_2 사이에서 볼록 함수(convex function)임을 증명하는 것과 같다.

임의의 분할점 T에 대한 *Kullback-Leibler* 거리를 나타내는 $KD_T(L\|R)$ 함수가 T_1과 T_2 사이에서 볼록 함수임을 증명하기 위해서는 <그림 12>에서 표시된 잠재적 분할점 T_p, T_q에 대해 다음과 같은 식이 만족됨을 증명해야 한다.

$$KD_{\lambda T_p+(1-\lambda)T_q}(L\|R) \le \lambda KD_{T_p}(L\|R)+(1-\lambda)KD_{T_q}(L\|R)$$
$$\forall\ 0 \le \lambda \le 1 \qquad (4)$$

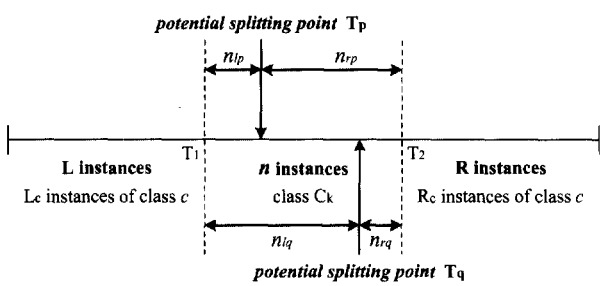

그림 12. 잠재적 분할점 T_p, T_q.

식 (4)에서 $KD_{T_p}(L\|R)$와 $KD_{T_q}(L\|R)$는 잠재적 분할점 T_p, T_q에 대한 *Kullback-Leibler* 거리를 나타내는 값으로 다음과 같이 정의된다.

$$KD_{T_p}(L\|R)=\sum_{c=1}^{|C|}P_L^p(c)\log_2\frac{P_L^p(c)}{P_R^p(c)} \qquad (5)$$

$$KD_{T_q}(L\|R)=\sum_{c=1}^{|C|}P_L^k(c)\log_2\frac{P_L^q(c)}{P_R^q(c)} \qquad (6)$$

식 (5)와 (6)에서 잠재적 분할점 T_p, T_q에 의해 분할된 두 구간에서의 임의의 클래스 c에 대한 확률 값은 다음과 같이 정의될 수 있다.

$$(P_L^p(c),P_R^p(c))=\left(\frac{L_c+n_{lp}I(c)}{L+n_{lp}},\frac{R_c+n_{rp}I(c)}{R+n_{rp}}\right)$$

$$(P_L^q(c),P_R^q(c))=\left(\frac{L_c+n_{lq}I(c)}{L+n_{lq}},\frac{R_c+n_{rq}I(c)}{R+n_{rq}}\right)$$

단, $I(c)=\begin{cases}1 & c=k\\ 0 & c\neq k\end{cases}$

위의 식에서 정의한 잠재적 분할점 T_p, T_q에 의해 분할되는 두 구간에서의 임의의 클래스 c에 대한 확률 값은 log sum inequality (Cover, 1991) 정의에 의해 다음과 같이 나타낼 수 있다.

$$\lambda P_L^p(c)+(1-\lambda)P_L^q(c)\ \log\frac{\lambda P_L^p(c)+(1-\lambda)P_L^q(c)}{\lambda P^p{}_R(c)+(1-\lambda)P_R^q(c)}$$
$$\le\lambda P_L^p(c)\log\frac{\lambda P_L^p(c)}{\lambda P_R^p(c)}+(1-\lambda)P_L^q(c)\log\frac{(1-\lambda)P_L^q(c)}{(1-\lambda)P_R^q(c)}$$

위 식에서 양변에 대해 모든 클래스($c\in C$)에 대한 합(summation)을 취하면 다음과 같은 식이 성립된다.

$$\sum_{c=1}^{|C|}\left(\lambda P_L^p(c)+(1-\lambda)P_L^q(c)\ \log\frac{\lambda P_L^p(c)+(1-\lambda)P_L^q(c)}{\lambda P_R^p(c)+(1-\lambda)P_R^q(c)}\right)$$
$$\le\lambda\sum_{c=1}^{|C|}\left(P_L^p(c)\log\frac{P_L^p(c)}{P_R^p(c)}\right)+(1-\lambda)\sum_{c=1}^{|C|}\left(P_L^q(c)\log\frac{P_L^q(c)}{P_R^q(c)}\right)$$

위 식은 *Kullback-Leibler* 거리의 정의에 의해 다음과 같이 정리될 수 있고, 이는 식 (4)와 같다는 것을 알 수 있다.

$$KD_{\lambda T_p+(1-\lambda)T_q}(L\|R)\le\lambda KD_{T_p}(L\|R)+(1-\lambda)KD_{T_q}(L\|R)\ \blacksquare$$

2.2.3 구간 병합 알고리듬

본 연구에서 제시한 구간 분할 알고리듬은 연속형 속성의 구간을 다수의 부분 구간으로 분할함으로서 이산화 작업이 이루어지도록 하였다. 그러나 구간 분할 알고리듬만을 이용하는 경우 연속형 속성의 구간이 너무나 많은 부분 구간으로 분할되는 과도 분할(over splitting) 문제가 발생될 수 있다. 특히 본 연구에서와 같이 실시간으로 수집되는 데이터를 기반으로 증분 의사결정 트리를 구축하는 경우 연속형 속성의 과도 분할로 인해 구축되는 의사결정 트리가 너무 복잡해지는 문제가 발생될 수 있다. 따라서 본 연구에서는 구간 분할 알고리듬에 의해 분할된 부분 구간을 대상으로 서로 유사한 특성을 지닌 부분 구간을 찾아내어 이를 병합함으로서 부분 구간의 수를 감소시킬 수 있는 구간 병합 알고리듬인 Merging (a, D, *Splitting_List*)을 제안한다.

<표 2>에서 Splitting_List는 구간 분할 알고리듬에 의해 정의된 분할점 집합을 의미하고, *Splitting_List*에 속한 각각의 분할점에 의해 형성되는 두 개의 부분 구간에 대해 *Kullback-Leibler* 거리를 계산하여 두 부분 구간의 특성이 유사하다고 판별된 경우 구간 병합을 수행하게 된다.

표 2. 구간 병합 알고리듬

```
Merging (a, D, Splitting_List)
a : continuous-valued attribute
D : a set of data on which a is defined
Splitting_List : pointer to a list of splitting points
Begin Algorithm
  FOR(t = 1; t<= |Splitting_List|; t++)
    /* generate 2 intervals by T_{t-1}, T_t, T_{t+1} */
    /* T_0 = Min(a), T_{|Splitting_List|+1} = Max(a) */
    L ← {d∈D | T_{t-1} < a(d) ≤ T_t}
    R ← {d∈D | T_t < a(d) ≤ T_{t+1}}
  /* calculate KD(L||R) for each T_t */
    IF ( KD_{T_t}(L||R) ≤ α) THEN
      DeleteSplittingPoint (Splitting_List, T_t)
  Return Splitting_List
End
```

2.2.4 기호화 알고리듬

구간 분할 알고리듬과 구간 병합 알고리듬에 의해 연속형 속성 a의 구간이 부분 구간으로 완전하게 분할되었을 경우 연속형 속성의 값을 이산형 속성으로 변환하기 위해서는 분할된 부분 구간에 대한 기호 (symbol)를 정의하여야 한다. <표 3>의 ①은 분할된 연속형 속성의 부분 구간 (Intervals)에 대해 이산형 값인 기호를 할당함으로서 연속형 속성을 이산형 속성으로 변환하는 것을 의미한다.

표 3. 기호화 알고리듬

```
Symbolizing (a, D, Splitting_List)
a : continuous-valued attribute
D : a set of data on which a is defined
Splitting_List : pointer to a list of splitting points
Intervals : pointer to a list of intervals
Begin Algorithm
  FOR(t = 1; t<= |Splitting_List|+1; t++)
    /* generate intervals by T_t */
    /* T_0 = Min(a), T_{|Splitting_List|+1} = Max(a) */
    Interval[t] ← {d∈D | T_{t-1} < a(d) ≤ T_t}
    CreateIntervals (Intervals, Interval[t])
    Symbolize (Intervals) ..... ①
  Return Intervals
End
```

2.3 증분 의사결정 트리

본 연구에서는 연속형 속성을 포함하는 데이터를 기반으로 효율적인 증분 의사결정 트리를 구축하기 위한 혼합형 다구간 이산화 알고리듬을 제시하였다. 실시간으로 수집되는 기계 상태 데이터를 기반으로 증분 의사결정 트리를 구축하기 위한 절차는 Baek, *et al.* (2000)이 제안한 적응형 의사결정 트리 구축 절차를 따른다. 그러나 Baek, *et al.*이 제안한 적응형 의사결정 트리 구축 절차는 연속형 속성에 대해 반복적인 이진 이산화를 수행함으로서 구축되는 트리의 깊이가 깊어지는 문제점이 발생하기 때문에 <그림 13>에서와 같이 본 연구에서 제안한 혼합형 다구간 이산화 알고리듬을 이용하여 연속형 속성에 대한 이산화 작업을 수행한 후 이를 기반으로 증분 의사결정 트리를 구축하는 절차를 따른다.

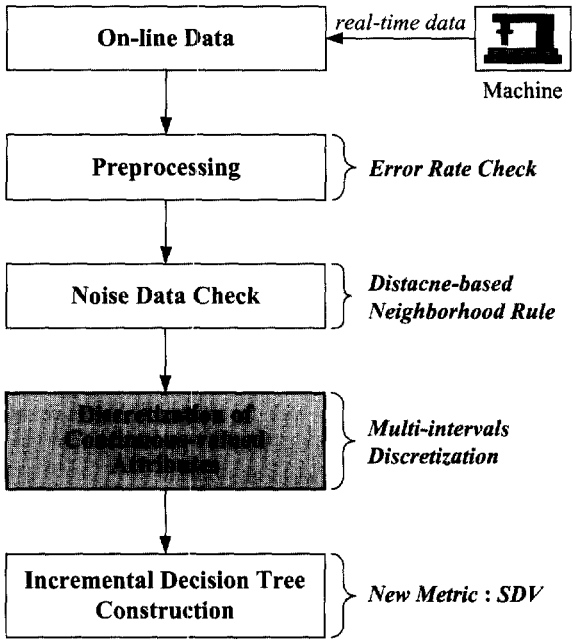

그림 13. 증분 의사결정 트리 구축 절차.

따라서 <그림 13>과 같은 절차를 통해 증분 의사결정 트리를 구축하는 경우, 의사결정 트리 구축 단계에서는 모든 속성이 이산형 값을 갖는 속성으로 변환되기 때문에 검사 속성 선택 척도 SDV는 식 (7)과 같이 정의될 수 있고, 의사결정 트리 구축 및 갱신 절차는 <표 4>와 같이 정리될 수 있다. <표 4>에서 사용되는 DM_array는 SDV를 계산하기 위해 사용되는 자료 구조로서 Baek, *et al.*이 제안한 자료 구조를 의미한다.

$$SDV(a_i) = \sum_{j=1}^{|A_i|} \{ \omega_j \times | n_{ijc^*} - \sum_{c \in C-\{c^*\}} n_{ijc} | \} \qquad (7)$$

where

a_i : i번째 속성

$|A_i|$: 속성 a_i가 가질 수 있는 속성값의 수

$$\omega_j = \frac{n_{ijc^*}}{\sum_{c \in C} n_{ijc}} \qquad j = 1, 2, \cdots |A_i|$$

n_{ijc} : 속성 a_i의 값이 j번째 속성값이고 클래스 값이 c인 데이터의 수

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

표 4. 의사결정 트리 구축 (갱신) 절차

Step 1. If the tree is empty, then define it as the leaf node setting the class name to the class of the instance.

Step 2. Otherwise, If the tree is not empty and actual instance of data is consistent with the predicted class of the current decision tree, update the DM_array of the class to which the instance belongs. Otherwise go to Step 3.

Step 3. ***Decision tree update.***

Step 3.1 Update the DM_arrays of attributes that are elements of the decision path of the instance.

Step 3.2 Expand the leaf node of the decision path using an attribute with the largest SDV value as a test attribute.

Step 3.3 *Decision tree reconstructing*
From the root node of the decision path, calculate SDV values of all attributes and replace the current test attribute of the node with one having the largest SDV value. Repeat this procedure for the sub-tree of the node.

본 연구에서 검사 속성 선택 척도로 사용되는 SDV는 결정 노드에 의해 분할된 데이터의 부분 집합이 갖는 결과 클래스의 동질성 (homogeneity)을 평가한 척도로서 SDV가 클수록 데이터의 동질성이 높음을 의미한다. 따라서 데이터를 구성하는 각각의 속성에 대해서 SDV 값을 계산한 후 가장 큰 SDV 값을 갖는 속성을 확장될 결정 노드의 검사 속성으로 사용하게 된다. 식 (7)에서 절대값 안의 식은 우세 (dominant) 결과 클래스 (c^*)를 갖는 데이터의 수와 나머지 클래스들을 갖는 데이터 수의 차이를 의미하며 가중치 ω_j는 같은 차이라 하더라도 동질성이 큰 속성에 선택 비중을 두기 위해 사용한 값이다.

예를 들어 <그림 14>의 트리 (i)과 (ii)의 경우 SDV 값을 계산할 때 절대값 부분은 4로 같지만 트리 (ii)가 트리 (i)보다는 동질성이 좋기 때문에 가중치 ω_j를 이용하여 트리 (ii)의 SDV 값이 높아지도록 한다.

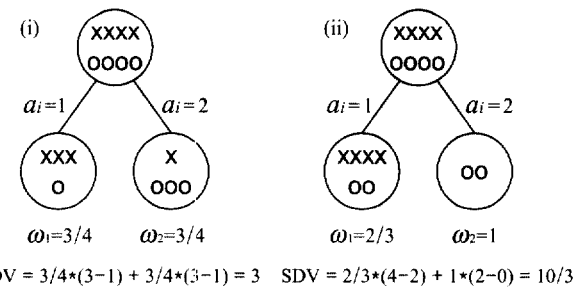

그림 14. SDV 계산 예제.

3. 다구간 이산화 알고리듬 성능 비교

본 연구에서 제시한 혼합형 다구간 이산화 알고리듬의 성능을 평가하기 위해 반도체 공정에서 에칭 작업을 수행하는 기계 P의 이력 데이터를 기반으로 구축된 의사결정 트리의 성능 비교를 실시하였다. 반도체 공정에서 에칭 작업을 수행하는 기계 P는 공정 변수에 따라 에칭률 (etching rate)이 변화하는 특징을 갖는다. 기계 P는 에칭률이 너무 높거나 너무 낮아지는 경우 에칭에 문제가 생기기 때문에 적절한 에칭률을 유지할 수 있도록 하는 일이 중요하다. 따라서 본 연구에서는 에칭률에 영향을 미치는 공정변수를 속성으로 정의하고 에칭률에 따라 결과 클래스 (LE : Low Etching Rate, NE : Normal Etching Rate, HE : High Etching Rate)를 정의한 이력 데이터를 기반으로 에칭률의 이상 상황을 추론할 수 있는 의사결정 트리를 구축한다. 기계 P의 이력 데이터는 <표 5>와 같은 형식을 갖는다.

<그림 15>는 Baek *et al.* (2000)이 제시한 이진 이산화를 이용한 적응형 의사결정 트리 구축 절차를 통해 기계 P의 이력 데이터를 기반으로 구축한 의사결정 트리를 나타낸 것이고, <그림 16>은 본 연구에서 제시한 혼합형 다구간 이산화 알고리듬을 이용하여 구축한 의사결정 트리를 나타낸 것이다. <그림 16>에서 보듯이 본 연구에서 제시한 혼합형 다구간 이산화를 이용하여 구축한 의사결정 트리가 이진 이산화를 이용하여 구축한 트리보다 깊이가 얕고 노드 수가 적은 간단한 트리임을 알 수 있다.

표 5. 기계 이력 데이터 형식

	Name	Description	Type	Domain
Attributes	PRESS	Pressure	Continuous	1 ~ 900 (μ Torr)
	TEMP	Temperature	Continuous	100 ~ 500 (°K)
	GAS_FLOW	Density of etching gas	Continuous	1 ~ 90 (sccm)
	ION_VOLT	Voltage of ion	Continuous	25 ~ 100 (V)
	LOCATION	Location of wafer	Discrete	Ground, Powered
Class	ER	Etching rate	Discrete	LE, NE, HE

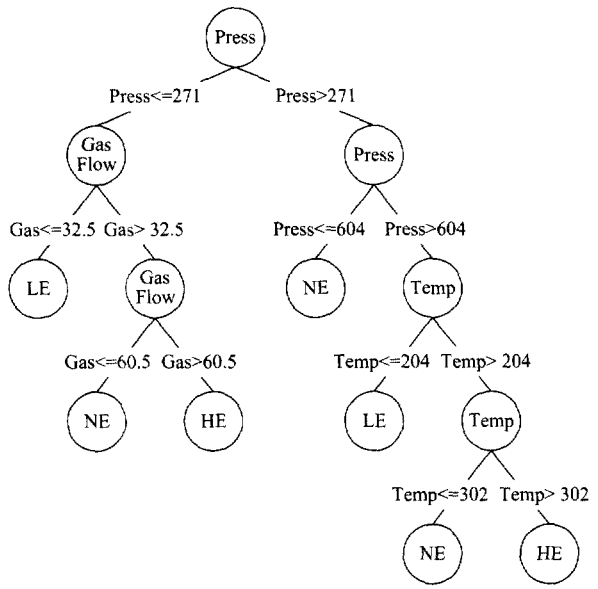

그림 15. 이진 이산화를 이용한 의사결정 트리.

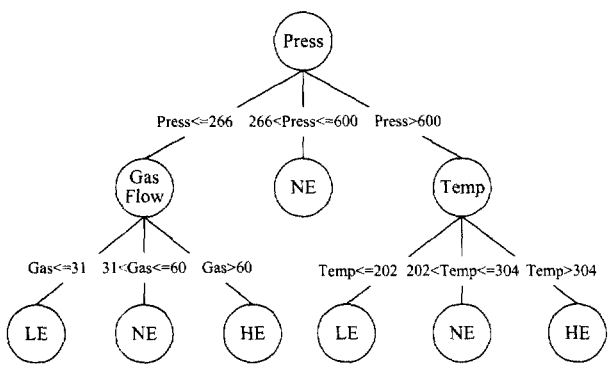

그림 16. 다구간 이산화를 이용한 의사결정 트리.

본 연구에서 제시한 혼합형 다구간 이산화 알고리듬의 성능을 자세히 비교하기 위해 데이터의 수가 증가함에 따른 의사결정 트리의 정확도, 복잡성, 그리고 이산화 작업을 수행하기 위해 필요로 하는 계산 횟수를 비교한 그래프는 <그림 17>, <그림 18>, <그림 19>와 같다. 그림에서 HMD로 표기된 것은 본 연구에서 제시한 혼합형 다구간 이산화 방법론(hybrid multi-interval discretization method)을 의미하고 BDM으로 표기된 것은 Baek *et al.*이 제시한 이진 이산화 방법론(binary discretization method)을 의미한다. 또한 본 연구에서 제시한 혼합형 다구간 이산화 방법론의 성능을 좀더 자세히 비교하기 위해 ESD와 TMD로 표기되는 두 가지 다구간 이산화 방법론과의 비교를 실시하였다. ESD는 다구간 이산화 방법론 중에서 가장 단순한 방법으로서 연속형 속성의 값을 오름차순으로 정렬한 후 k개의 동일한 크기의 구간으로 분할하는 동일 크기 (equal-sized) 다구간 이산화 방법론을 의미한다. 여기서 k는 사용자가 정의하는 모수로서 분할되어지는 연속형 속성의 구간 수를 나타내며, 각 구간의 크기는 $\delta=(a_{max}-a_{min})/k$

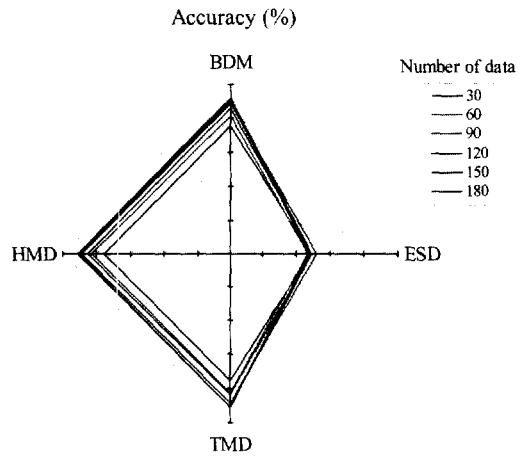

그림 17. 이산화 방법론별 트리의 정확도 비교.

와 같이 정의된다. 단, a_{max}는 연속형 속성 a가 갖는 최대 값을 의미하고 a_{min}은 연속형 속성 a가 갖는 최소 값을 의미하며 구간 분할을 수행하였을 때 분할점은 $a_{min}+i\delta\ (i=1, \cdots, k-1)$가 된다. TMD는 하향식 다구간 이산화 방법론 (top-down multi-interval discretization method)을 의미하는 것으로서 Fayyad and Irani (1993)가 제안한 엔트로피 기반 다구간 이산화 방법론을 비교 대상으로 선정하였다. <그림 17>은 데이터의 수가 증가할 때 각각의 이산화 방법론에 의해 구축된 의사결정 트리의 정확도 (accuracy)를 비교한 그래프이다. 그림에서 보는 바와 같이 BDM, TMD, HMD의 정확도가 거의 비슷하다는 것을 알 수 있고 ESD의 정확도는 현저히 떨어짐을 알 수 있다. 이는 ESD가 데이터가 갖는 결과 클래스 정보를 전혀 이용하지 않고 단순히 일정한 크기의 구간 분할을 실시하는 이산화 작업을 수행하였기 때문에 구축된 의사결정 트리의 정확도가 떨어진다는 것을 알 수 있다.

<그림 18>은 각각의 이산화 방법론에 의해 구축된 의사결정 트리의 노드 수가 데이터가 증가함에 따라 어떻게 변하는

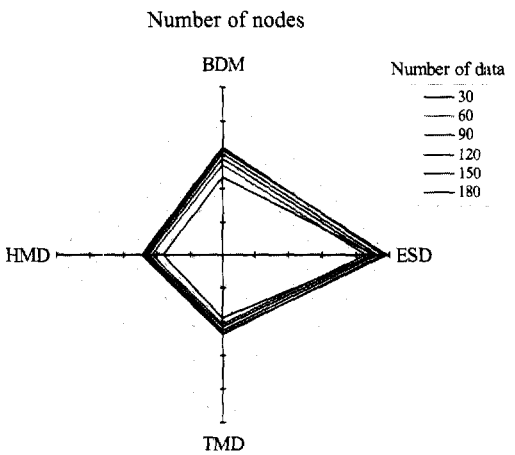

그림 18. 이산화 방법론별 트리의 노드 수 비교.

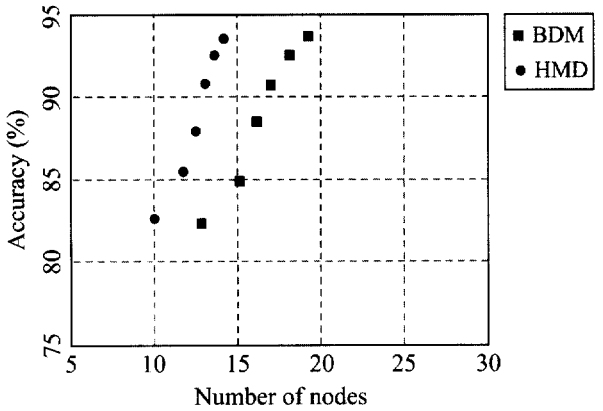

그림 19. 의사결정 트리의 구조적 효율 비교.

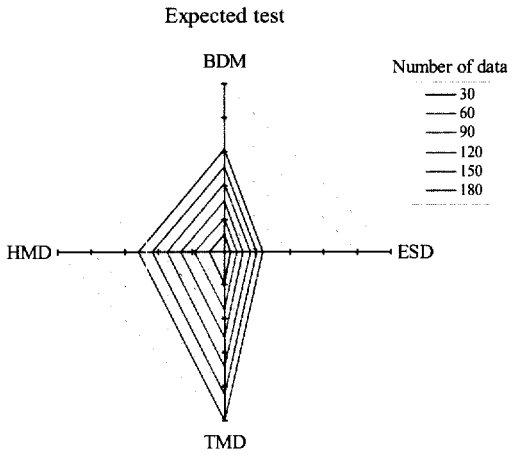

그림 20. 이산화 방법론별 계산 횟수 비교.

지를 나타낸 그래프로써 TMD와 HMD에 의해 구축된 의사결정 트리가 다른 방법론에 비해 상대적으로 적은 노드 수를 지니게 됨을 알 수 있다. ESD는 앞에서 설명한 바와 같이 데이터가 갖는 결과 클래스 정보를 전혀 이용하지 않고 단순히 일정한 크기의 구간 분할을 실시하는 이산화 작업을 수행하였기 때문에 의사결정 트리가 무분별하게 과도 확장되어 트리의 정확도가 떨어질 뿐만 아니라 노드 수도 현저히 증가함을 알 수 있다. 또한 <그림 18>에서 HMD를 이용하여 구축한 의사결정 트리의 노드 수가 BDM을 이용하여 구축한 의사결정 트리의 노드 수에 비해 작다는 것을 알 수 있는데 이는 다구간 이산화를 이용하여 구축한 의사결정 트리가 이진 이산화를 이용하여 구축한 의사결정 트리에 비해 깊이가 적고 노드 수가 적은 간단한 트리임을 보여준다.

<그림 17>과 <그림 18>을 바탕으로 본 연구에서 제시한 혼합형 다구간 이산화 방법론(HMD)과 이진 이산화 방법론(BDM)의 성능을 비교하면 결론적으로 HMD를 이용하여 구축한 의사결정 트리가 BDM을 이용하여 구축한 의사결정 트리에 비해 트리의 정확도를 비슷하게 유지하면서 노드 수가 작은 간결한 트리임을 알 수 있고 이는 <그림 19>와 같은 구조적 효율 비교를 통해 설명할 수 있다.

<그림 20>은 각각의 이산화 방법론에 의해 이산화 작업을 수행할 때 필요로 하는 계산 횟수를 데이터의 수가 증가함에 따라 나타낸 그래프이다. 이산화 작업을 위해 필요로 하는 계산 횟수가 의미하는 것은 이산화 작업을 수행하기 위해 필요로 하는 계산 시간을 의미하는 것으로서 계산 횟수가 적다는 것은 이산화 작업을 수행하는 시간이 단축됨을 의미한다.

<그림 20>에서 보듯이 HMD를 이용한 이산화 작업이 TMD를 이용한 이산화 작업에 비해 상대적으로 적은 계산 횟수를 필요로 함을 알 수 있다. 이는 새로운 데이터가 추가되었을 때 HMD는 새로운 데이터 추가된 부분 구간에 대해서만 분할 작업과 병합 작업을 수행함으로서 이산화 작업을 완료하는 것에 비해 TMD는 전체 구간에 대해 처음부터 다시 분할 작업을 수행하기 때문에 계산 횟수가 증가한다는 것을 알 수 있다. 또한 <그림 20>에서 ESD가 가장 적은 계산 횟수를 필요로 하는 것으로 나타나는데 이는 ESD가 단순히 동일 크기의 구간으로 구간 분할을 실시하기 때문에 데이터가 증가하더라도 구간의 크기와 수가 변화하지 않고 구간에 속한 데이터의 수만 갱신하는 단순 계산을 수행하기 때문이다. 그러나 ESD는 <그림 17>과 <그림 18>에서 보듯이 무분별한 트리의 확장으로 인해 노드 수가 급격히 증가하고 트리의 정확도도 현격히 떨어지는 무의미한 의사결정 트리를 구축하게 된다.

<그림 21>은 데이터가 증가함에 따른 다구간 이산화 방법론의 계산 효율을 비교한 그래프로서 다구간 이산화 작업을 수행하기 위해 필요로 하는 계산 횟수와 구축된 의사결정 트리의 정확도를 비교한 그래프이다. <그림 21>에서 보듯이 본 연구에서 제시한 다구간 이산화 방법론인 HMD가 TMD에 비해 계산 효율이 높다는 것을 알 수 있는데 이는 HMD가 TMD에 비해 이산화 작업을 위해 필요로 하는 계산 횟수를 줄이면서도 트리의 정확도를 비슷하게 유지할 수 있다는 것을 의미한

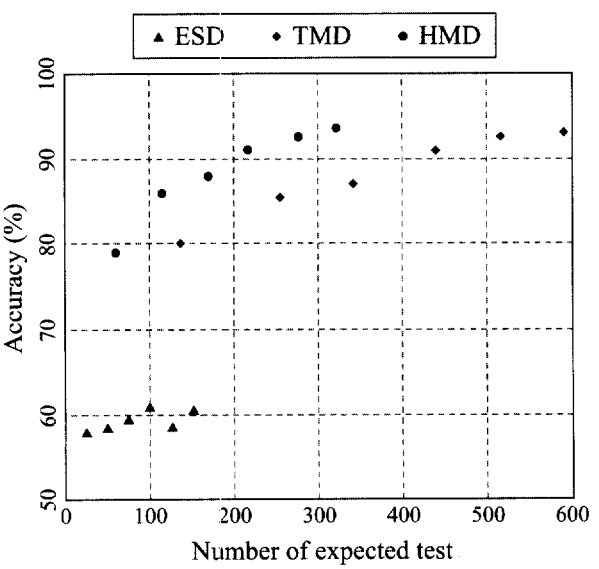

그림 21. 다구간 이산화 방법론의 계산 효율 비교.

다. <그림 21>에서 ESD는 다른 방법론에 비해 상대적으로 적은 계산 횟수를 필요로 하지만 구축되는 의사결정 트리의 정확도가 현저히 떨어지기 때문에 이산화 방법론으로서 가치가 없다는 것을 알 수 있다.

4. 결 론

본 연구에서는 실시간으로 수집되는 기계 상태 데이터를 기반으로 효율적인 증분 의사결정 트리를 구축하기 위한 연속형 속성의 다구간 이산화 방법을 제시하였다. 연속형 속성을 포함하는 데이터를 기반으로 효율적인 의사결정 트리를 구축하기 위해서는 연속형 속성의 값을 이산화함으로써 각각의 속성에 대한 분할점을 쉽게 정의할 수 있도록 하는 일이 무엇보다 중요하다. 따라서 본 연구에서는 실시간으로 수집되는 기계 상태 데이터를 기반으로 연속형 속성에 대한 효율적인 이산화 작업을 수행하기 위해 *Kullback-Leibler* 거리를 이용한 혼합형 다구간 이산화 방법을 제시하였다. 본 연구에서 제시한 다구간 이산화 방법은 하향식 이산화와 상향식 이산화가 혼용된 혼합형 이산화 방법으로서 새로운 데이터가 추가로 입력되는 경우 입력된 데이터와 관련된 부분 구간에 대해서만 이산화 작업을 실시함으로써 기존의 다구간 이산화 방법에 비해 계산 효율이 높은 효율적인 증분 의사결정 트리를 구축할 수 있다는 장점을 지니고 있으며, 이진 이산화 방법에 비해 트리의 정확도는 비슷하게 유지하면서 트리의 깊이와 노드의 수가 적은 효율적인 의사결정 트리를 구축할 수 있다는 장점을 제공한다.

참고문헌

Baek, J., Kim, K., Kim, S. and Kim, C. (2000), Adaptive Decision Tree Algorithm for Data Mining in Real-Time Machine Status Database, *Journal of the Korean Institute of Industrial Engineers*, **26**(2), 171-182.

Catlett, J. (1991), On changing continuous attributes into ordered discrete attributes, *Proceedings of the European Working Session on Learning: Berlin*, 164-178.

Cerquides, J. and Mantaras, R. L. (1997), Proposal and Empirical Comparison of a Parallelizable Distance-Based Discretization Method, *Third International Conference on Knowledge Discovery and Data Mining*, Newport Beach, California, USA, 139-142.

Ching, J. Y. and Wong, A. K. (1995), Class-Dependent Discretization for Inductive Learning from Continuous and Mixed-Mode Data, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(2), 641-651.

Cover, T. M. (1991), *Elements of Information Theory*, New York: Wiley.

Dougherty, J., Kohavi, R. and Sahami, M. (1995), Supervised and Unsupervised Discretization of Continuous Features, *Machine Learning: Proceedings of the Twelfth International Conferences*, Morgan Kaufman Publishers, San Francisco, CA.

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

Loh, W. and Shih, Y. (1997), Split Selection Methods for classification Trees, *Statistica Sinica*, **7**, 815-840.

Quinlan, J. R. (1993), *C4.5: Programs for machine learning*, Morgan Kaufmann.

Quinlan, J. R. (1996), Improved Use of Continuous Attributes in C4.5, *Journal of Artificial Intelligence Research*, **4**, 77-90.

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

Utgoff, P. E. (1997), Decision Tree Induction Based on Efficient Tree Restructuring, *Machine Learning*, **29**, 5-44.