Journal of the Korean Institute of Industrial Engineers
Vol. 30, No. 4, pp. 267-276, December 2004.

교호효과를 고려한 향상된 의사결정나무 알고리듬에 관한 연구

권근섭 · 최경현†

한양대학교 산업공학과

Improved Decision Tree Algorithms by Considering Variables Interaction

Keunseob Kwon · Gyunghyun Choi

Department of Industrial Engineering, Hanyang University, Seoul, 133-79

Much of previous attention on researches of the decision tree focuses on the splitting criteria and optimization of tree size. Nowadays the quantity of the data increase and relation of variables becomes very complex. And hence, this comes to have plenty number of unnecessary node and leaf. Consequently the confidence of the explanation and forecasting of the decision tree falls off.

In this research report, we propose some decision tree algorithms considering the interaction of predictor variables. A generic algorithm, the k-1 Algorithm, dealing with the interaction with a combination of all predictor variable is presented. And then, the extended version k-k Algorithm which considers with the interaction every k-depth with a combination of some predictor variables. Also, we present an improved algorithm by introducing control parameter to the algorithms. The algorithms are tested by real field credit card data, census data, bank data, etc.

Keywords : decision tree, data mining, interaction

1. 서 론

데이터마이닝(Data Mining)에서의 의사결정나무(Decision Tree)는 탐색(Exploration)과 모형화(Modeling)라는 두 가지 특성을 모두 가지고 있다고 할 수 있다. 즉, 의사결정나무는 판별분석(Discrimination Analysis) 또는 회귀분석(Regression Analysis) 등과 같은 모수적(Parametric)모형을 분석하기 위해서 사용되기도 하고, 또한 사전에 이상치(Outlier)를 검색하거나 분석에 필요한 변수 또는 모형에 포함되어야 할 교호효과(Variable Interaction)를 찾아내기 위해서 사용될 수도 있고, 그 자체가 분류 또는 예측 모형으로 사용될 수도 있다(Fayyad, 1996; Quinlan, 1986).

인공지능, 기계학습 등 다양한 분야의 분석방법들이 데이터마이닝을 위해 사용되고 있지만 데이터마이닝에서는 대용량의 데이터를 다룰 뿐만 아니라 변수의 성격이나 변수 간의 관계가 잘 알려져 있지 않은 많은 데이터를 사용해 분석하기 때문에 분석결과를 해석하는 것이 쉽지 않은 경우가 대부분이다. 이 같은 이유로 정보를 쉽게 이해할수 있는 의사결정나무 기법이 유용하게 사용되고 있다.

특히 경제활동이 다양해지고 급속한 시장환경의 변화에 따른 의사결정변수의 다양화는 의사결정나무의 교호효과의 특성을 더욱 중요시 하게 되었다. 하지만 실제로 사용되고 있는 알고리듬은 각각의 독립된 변수 한 개로 나무를 구성해 나가는 방식을 취하고 있어, 교호효과를 해석하는 데 적절하지 못한 부분이 있다. 또한 자료의 양이 증가하고 변수들의 상호 이해관계가 복잡하게 얽힘에 따라 모형을 구성하는 내부 마디(node)와 잎(leaf)의 수는 불필요하게 많아지고 복잡하게 되기 때문에 모형의 예측력과 설명력도 낮아지게 된다(<Figure 1> 참조).

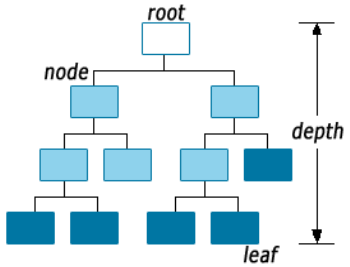

Figure 1. Decision tree.

† 본 논문은 2001년도 한양대학교 교내연구비에 의하여 수행되었음.
연락저자 : 최경현 교수, 133-791 서울시 성동구 행당동 17번지 한양대학교 산업공학과, Fax : 02-2295-8049, E-mail : ghchoi@hanyang.ac.kr
2004년 2월 접수; 2004년 6월 수정본 접수; 2004년 8월 게재 확정.

본 연구의 목적은 변수들의 상호 이해관계를 보다 명확히 분석하여 보다 간결한 의사결정나무 모형을 구성하고 분석의 효과를 높이는 동시에 모형의 예측력과 설명력을 높이는 알고리듬을 제안하는 데 있다.

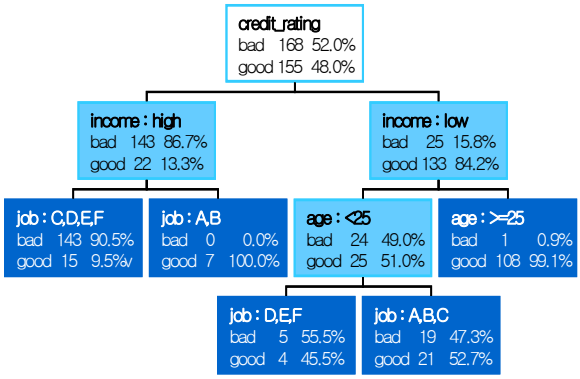

Figure 2. Decision tree for the credit evaluation.

본 논문의 구성은 다음과 같다. 2장에서는 일반적 의사결정나무 알고리듬을 제시하고 그 취약점을 살펴본다. 3장에서는 의사결정나무가 형성되는 과정을 살펴보고 이 논문에서 제안하는 **k-1** 알고리듬, **k-k** 알고리듬 그리고 향상된 교호효과 알고리듬을 설명한다. 4장에서는 실험결과를 소개한 뒤 마지막으로 5장에서 결론을 제시하였다.

2. 문제 정의

의사결정나무는 의사결정규칙을 <Figure 1>과 같은 나무구조로 도식화하여 분류와 예측을 수행하는 분석방법이다. 이 방법은 분류 또는 예측의 과정이 나무구조에 의한 추론규칙(Induction Rule)에 의해서 표현되기 때문에, 다른 방법들(예를 들면, 신경망, 판별분석, 회귀분석 등)에 비해서 사용자가 그 과정을 쉽게 이해하고 설명할 수 있다는 장점을 가지고 있다. 사용자는 <Figure 2>와 같은 의사결정나무 구조로부터의 어떤 변수가 신용상태의 분류에 영향을 많이 주는지 그리고 어떤 경우에 신용상태가 나쁜 것으로 분류되는지를 쉽게 파악할 수 있다. 이 때 전체의 나무를 구성할 경우, 발생 가능한 규칙이 모두 생성되므로 변수들의 순서는 별 의미가 없게 된다. 하지만 비즈니스 측면에서 의사결정나무 기법을 사용하고자 하는 것은 의사결정규칙을 통해 개개인의 성향을 분석하기 위함이 아니라 집단의 특성을 분석하고 예측하기 위한 것이므로 이런 경우 Full Tree를 구하게 되면 분류오류(Classification Error)를 크게 할 위험이 높거나 부적절한 추론규칙을 가지고 있는 가지(Branch)를 포함하게 되어 분석의 취지를 벗어나게 된다. 따라서 분석의 목적과 자료구조에 따라서 적절한 분리기준과 정지규칙을 지정하여 의사결정나무를 형성하게 된다(Quinlan, 1996; Shlien, 1990). 그러나 예를 들어, 만약 나무깊이를 2로 제한한 경우에는 의사결정규칙은 각 나무마다 서로 다른 형태를 가지게 된다(Smyth, 1992). 따라서 목표변수를 잘 설명하는 순서대로 예측변수를 선택하는 것이 나무의 정보품질을 좌우하게 된다. 즉, 효과적인 분석을 위해 주요변수를 찾아내는 것이 가장 중요한 문제가 되었고, 단계별 나무구성의 최적 분리변수를 찾고자 분리기준에 대한 많은 연구가 시행되어 왔다.

현재 사용되어지는 대표적인 의사결정나무 알고리듬으로는 카이제곱검정(이산형 변수) 또는 F검정(연속형 변수)을 이용하는 CHAID 알고리듬, Gini 지수(이산형 변수) 또는 분산의 감소량(연속형 변수)을 이용하는 CART 알고리듬, 엔트로피(entropy) 지수를 분리기준으로 사용하는 C4.5 알고리듬, 예측변수의 측도에 따라서 서로 다른 분리규칙을 사용하는 QUEST 알고리듬 등이 있다.

이들 알고리듬들은 분리기준의 방법들이 서로 상이하지만 모두 나무를 단계적으로 구성하면서 각 단계별 최적분리변수를 선택하는 방법을 취하고 있다. 이렇게 나무가 구성되면 보통 두 개 이상의 변수가 결합하여 목표변수에 어떻게 영향을 주는지를 파악한다. 즉, 처음부터 교호작용의 영향은 배재하고 나무를 구성한 뒤 결과 해석을 할 때는 변수들의 상호이해관계로서 분석을 하는 모순된 모습을 보여주고 있다. 따라서 이 논문에서는 보다 효과적인 분석을 위해서 변수들의 교호효과를 고려한 의사결정나무를 제안한다. 본 연구에서 제안하는 알고리듬은 각기 교호효과를 고려하는 방법으로, 실데이터를 통한 실험결과를 제시함으로써 알고리듬의 유용성을 증명하였다.

3. 제안하는 알고리듬

CART(Brieman, 1984), CHAID(Kass, 1980), C4.5(Quinlan, 1993)의 의사결정나무 알고리듬들은 이런 영향을 계산하는 방법의 차이와 어떻게 가지치기를 멈추어 가장 적절한 크기의 나무를 최종모형으로 결정하는지에 대한 방법의 차이에서 붙여진 이름들이라 할 수 있겠다.

C4.5를 언급할 때 따라다니는 지수는 엔트로피이다. CART와 마찬가지로 C4.5도 마디의 순수함을 측정하는데, 이 때 비트(bit) 개념을 이용한다. C4.5 알고리듬에서는 p가 메시지(message)의 확률일 때 이 메시지로 전달되는 정보량(information)을 $-\log_2 p$ 로 표시하며 이는 작은 확률로 일어나는 메시지일수록이를 알기 위해 많은 정보가 필요함을 의미한다. 사례들의 집합인 S에서 C_j에 속하는 사례들의 개수를 $freq(C_j, S)$로 표현하면 S에서 임의로 한 사례를 선택할 때, 이 사례가 C_j에 속할 확률은 $\frac{freq(C_j, S)}{|S|}$로 표시되고, 이 사례가 전달하는 정보량은 $-\log_2\left(\frac{freq(C_j, S)}{|S|}\right)$와 같이 표시된다. 따라서 S에서 기대되는 정보량은

$$info(S) = -\sum_{j=1}^{k} \frac{freq(C_j, S)}{|S|} \times \log_2\left(\frac{freq(C_j, S)}{|S|}\right)$$

와 같이 계산된다. 이로써 훈련 데이터 집합 T가 테스트(또는 classifier) X에 의해 n개로 분할된 후의 기대되는 정보량은

$$E(X) = info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times info(T_i)$$

로 계산되고 X에 의한 분할로 얻어진 정보이익은

$$gain(X) = info(T) - info_X(T)$$

으로 정의한다. 정보이익을 최대로 하는 테스트의 선택을 gain criterion이라 하는데 범주 수가 많은 변수로의 심각한 치우침(bias)현상이 발생할 수 있는 문제점이 있다. 따라서 T에 있는 하나의 사례가 속하는 부분집합을 필요한 평균 정보의 양(split info)으로 정규화할 필요가 있다. 이때, split info는 T가 n개의 부분집합으로 분할됨에 따라 발생하는 정보의 양을 의미하며

$$split\ info(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \log_2\left(\frac{|T_i|}{|T|}\right)$$

와 같이 계산한다. split에 의해 생성된 유용한 정보의 비율(gain ratio)은

$$gain\ ratio(X) = \frac{gain(X)}{split\ info(X)}$$

로 정의하여 분석한다.

3.1 k-1 알고리듬

k-1 알고리듬은 C4.5를 기반으로 변수들의 교호효과를 고려하는 방법으로, 이때의 **k**는 교호효과를 고려하는 변수조합의 수를 의미하는 모수로 이때의 결과로 depth **k**의 의사결정나무가 구성된다. 여기까지의 알고리듬 반복 수(iteration)는 m개의 예측변수 중에서 순서를 고려하여 **k**개 선택하는 가지 수(즉, ${}_mP_k$)가 된다. m개의 변수 중 모수 **k**개의 변수를 뽑아 만들어진 j번째 조합을 $C_j(m,k)$로 표현할 때, 훈련 데이터집합 T가 $C_j(m,k)$에 의해 n개로 분할된 후의 기대되는 정보량은

$$info_{C_j(m,k)}(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times info(T_i)$$

와 같이 계산하고, 이때의 $C_j(m,k)$ 분할로 얻어진 정보이익은 $gain(C_j(m,k)) = info(T) - info_{C_j(m,k)}(T)$ 이다.

• k-1 알고리듬 Flow Chart

***Step 1.* (Initialization)**

분석자료의 변수 중 목표변수와 예측변수를 선택하고, 구성할 나무의 최대 깊이를 설정한다. 가지치기(pruning)를 위해 하나의 마디가 포함해야 할 최소 자료 수를 설정하고 목표변수 범주 값의 하한치(lower bound)와 상한치(upper bound)를 설정한다. 마지막으로 고려할 교호효과 정도에 따라 모수 **k**를 결정한다.

***Step 2.* (Root Node depth k 나무구성)**

m개의 예측변수 중에서 모수 **k**에 맞게 변수조합을 구해 각 경우의 엔트로피를 계산한다. 정보이익을 최대로 하는, 즉 최적분리를 갖는 조합을 선택하여 depth **k**인 나무를 해당 예측변수로 구성한다. 단, 이때 가지치기되는 마디는 생성되지 아니한다.

***Step 3.* (Child Node의 나무구성)**

depth **k**이후의 나무는 기존의 C4.5 알고리듬을 이용해 모든 가지의 마디가 끝마디가 될 때까지 나무를 구성해 나간다.

k-1 알고리듬의 경우, 변수들 간의 상호작용을 보다 정확하고 효율적으로 이해할 수 있으므로 보다 정교한 분석 및 예측이 이루어짐으로써 모형의 실제적 유용성을 높일 수 있다. 하지만 모수 **k**나 예측변수의 크기가 커지면 계산량이 매우 커지게 되어 교호효과를 고려하는 것이 용이하지 않게 된다.

3.2 k-k 알고리듬

k-1 알고리듬을 이용하여 나무 전체에 교호효과를 적용하기 위해서는 k값이 커지게 되고 그 결과 많은 계산량을 필요하게 되는 단점이 있다. 이 문제를 해결하기 위해 **k-k** 알고리듬을 제안한다. 모수 **k**를 설정한 뒤 depth **k**씩 나무를 구성해 나간다. <Figure 3>에서는 **k**=2로 설정하여 depth 2씩 나무를 구성한 것이다. 우선 예측변수(A, B, C, D, E)의 2단계 조합을 고려하여 목표변수를 가장 잘 설명하는 분리조합(A-B, A-C)을 찾는다. 이때 생성된 네 개의 자식마디도 목표변수를 잘 분리하도록 각각 2단계씩 부모노드에서 사용한 변수를 제외한 변수의 조합으로 구성한다. 그룹2에서는 부모노드로 사용한 변수 A, B를 제외한 C, D, E중 최적분리조합(C-E, C-D)을 찾으면 <Figure 3>과 같이 나무를 구성한다. 이 같은 방식으로 원하는 깊이까지 나무를 형성한다. 이와 같이 나무를 구성하면 전체적으로 교호효과를 적용하면서도 훨씬 적은 계산량으로 전체 나무를 구할 수 있게 된다.

k-k 알고리듬 Flow Chart

***Step 1.* (Initialization)**

분석자료의 변수 중 목표변수와 예측변수를 선택하고, 구성할 나무의 최대 깊이를 설정한다. 가지치기를 위해 하나의 마디가 포함해야 할 최소 자료 수를 설정하고 목표변수 범주 값의 하한치와 상한치를 설정한다. 마지막으로 고려할 교호효과 정도에 따라 모수 **k**를 결정한다.

***Step 2.* (Root Node의 depth k 나무구성)**

m개의 예측변수 중에서 모수 **k**에 맞게 변수조합별 엔트로

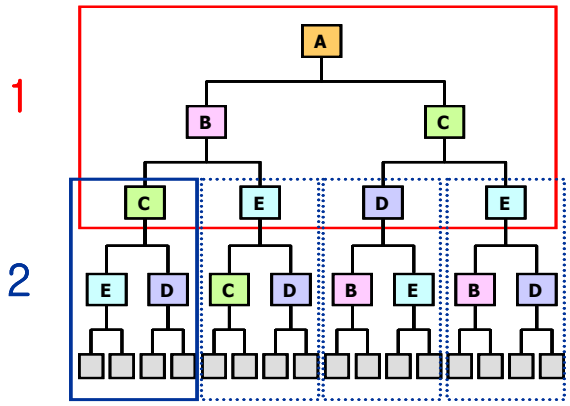

Figure 3. 2-2 Decision tree.

피를 계산한다. 최적분리를 갖는 조합을 선택하여 depth **k** 인 나무를 해당 변수로 구성한다. 단, 이때 가지치기되는 마디는 구성되지 아니한다.

Step 3. **(Child Node의 depth k 나무구성)**
depth **k** 이후의 생성된 자식마디마다 끝마디가 아닐 경우 Step2를 실시하여 depth **k** 나무를 생성한다. 이때 부모마디의 분리변수로 사용된 변수를 제외한다.

Step 4. **(Terminal Node 여부 확인)**
모든 가지의 마디가 끝마디가 되면 알고리듬을 끝낸다.

k-k 알고리듬의 경우 작은 **k**로도 전체적으로 교호효과를 고려한 나무를 구성할 수 있다. 뿐만 아니라 적은 계산량으로 나무를 구성할 수 있다는 큰 장점을 가지고 있다. 하지만 단계별로 그룹의 교호효과를 적용하는 방법이므로 교호효과의 효과가 미비할 수 있다는 단점을 가지고 있다. 또한 예측변수의 개수가 많아지거나 범주 수가 증가하면 안정적인 계산량을 유지하는데 어려움이 따른다.

3.3 향상된 교호효과 알고리듬

앞서 설명한 **k-1** 알고리듬은 예측변수의 수가 많아지거나 많은 범주 수에 대해서 뿐만 아니라 모수 **k**를 큰 값으로 선택할 경우 실효적인 계산이 불가능하게 된다. 또한 **k-k** 알고리듬은 계산량이 안정적이긴 하지만 적은 **k**로 단계별로 교호효과를 적용하므로 전체적인 교호효과 영향은 미비할 수 있다. 따라서 교호효과를 최대한 고려하면서도 안정적인 계산방법이 필요하다. 향상된 교호효과 알고리듬이란 **k-1** 알고리듬과 **k-k** 알고리듬을 적용함에 있어 depth **k**를 구성할 때 고려하는 변수를 모수 ϵ를 통해 후보변수를 추출함으로써 안정적인 분석률을 유지하면서 계산량은 획기적으로 줄이는 알고리듬이다. 제안하는 알고리듬은 변수개수 선택모수 $\epsilon > 0$을 사용하여 실제적인 교호효과에 영향을 줄 수 있는 변수를 추출하여 그들만을 대상으로 하기 때문에 **k**를 큰 값으로 선택하더라도 실제 계산은 선택된 변수만을 고려함으로써 **k-1** 알고리듬의 장점을 그대로 살리는 동시에 안정적인 계산량을 유지할 수 있도록 하였다. 즉, **k-1** 알고리듬과 **k-k** 알고리듬의 장점만을 살린 것이다.

앞서 제안하는 세 가지의 알고리듬(C4.5 알고리듬, **k-1** 알고리듬, **k-k** 알고리듬)의 교호효과 적용의 특징을 비교하면 $E(X)$는 해당 마디의 분리기준의 변수를 선택하기위해 계산하는 변수 X의 엔트로피를 의미하고 이때 최소의 엔트로피(분리를 가장 잘하는 경우)를 M으로 둔다.

$M = \min E(X),\ X = 1, 2, \ldots, N$일 때, C4.5 알고리듬은 M만 고려하고 교호효과는 고려하지 않으며, **k-1** 알고리듬, **k-k** 알고리듬은 모든 변수의 조합으로 교호효과를 고려하고 있으나 **k**의 크기에 따라 계산량이 기하급수적으로 늘어날 수 있으며, 마지막으로 향상된 교호효과 알고리듬은 주어진 $\epsilon > 0$에 대하여, $|M - E(X)| < \epsilon$를 만족하는 변수 X들만을 후보변수로 규정하고 후보변수의 조합으로 교호효과를 고려하고 있다.

즉, 교호효과를 고려하는 변수(후보변수)는 해당 변수의 엔트로피와 M과의 차이가 ϵ 이하일 경우를 대상으로 한다. 이것은 확률적으로 교호효과를 고려할 가치가 높은 변수들의 교호효과를 고려하자는 것으로, 대부분의 경우 변수들이 분리율이 낮은 경우는 다른 변수와의 상호연관으로도 그 효과가 미비하기 때문이다. 이로써 불필요한 계산을 줄이고도 교호효과의 영향도 극대화할 수 있게 된다. 따라서 앞서 제안한 **k-1** 알고리듬과 **k-k** 알고리듬의 높은 계산량 문제점을 해결할 수 있다.

위에서 언급한 알고리듬의 효율향상과 안정적인 수행을 위해서는 모수 $\epsilon > 0$의 선택방법이 무엇보다 중요하다. 본 연구에서는 고정수량법, 비율법, 그리고 변수의 분산을 활용하는 분산법을 고려한다. 먼저 고정수량법은 교호효과를 고려할 변수의 수를 일정한 개수로 정하는 것으로, 간단하다는 장점은 있지만 예측변수의 개수가 많고 적음에 관계하지 않기 때문에 예측변수의 개수가 적을 때는 불필요한 계산을 하게 되고, 예측변수의 개수가 많을 때는 극소수 변수들의 교호효과만을 고려하기 때문에 전체적인 정보품질의 향상을 기대하기 어렵다는 단점이 있다.

둘째로 비율법은 일정비율을 정해 두고 예측변수의 개수에 따라 교호효과를 고려할 대상변수의 수를 정하는 방법으로, 수량표시법에 비해 분석 데이터에 의해 유동적인 계산량을 가지고 있지만 분석 데이터의 성격은 반영하지 못한다. 데이터에 따라 변수별 분리율이 좋은 경우도 있고 그렇지 못한 경우도 있는데, 일반적으로 분리율이 좋을 경우 교호효과를 고려할 대상변수를 그렇지 않은 경우에 비해 더 많이 고려해야 하기 때문이다. 변수의 분리율은 변수의 엔트로피를 통해 나타나게 되는데 이 값은 0,1 사이의 실수값을 가지게 된다. 즉, 0으로 갈수록 분리율이 좋은 것이며 1의 가까운 값을 가지게 될 경우 분리율이 낮아지게 된다. 예를 들어

Data1. 예측변수={A, B, C, D, E},
변수별 엔트로피=
$\{E_A = 0.05, E_B = 0.07, E_C = 0.08,$
$E_D = 0.26, E_E = 0.32\}$

Data2. 예측변수={A, B, C, D, E},
변수별 엔트로피=
$\{E_A = 0.05, E_B = 0.57, E_C = 0.68,$
$E_D = 0.86, E_E = 0.95\}$

와 같은 경우 비율표시법으로 전체 예측변수 개수의20%에 대해 교호효과를 고려하기로 하였다면 Data1, Data2 모두 A, B 2개 변수만의 교호효과를 고려하게 된다. 하지만 위의 엔트로피를 살펴보면 Data1의 경우엔 A, B, C 변수의 교호효과를, Data2의 경우엔 A 변수만을 고려하는 것이 효과적이라는 것을 알 수 있다. 이처럼 비율법에는 데이터의 성격을 반영하지 못한다는 단점이 있다. 마지막으로, 분산법은 변수들의 엔트로피의 퍼짐 정도를 고려하여 ϵ을 결정하는 방법으로 엔트로피들의 표준편차를 계산하여 선정한 ϵ값 이하의 엔트로피를 가지는 변수만으로 교호효과를 고려함으로써 분석 데이터의 크기뿐만 아니라 성격까지 반영할 수 있다는 장점을 가진다.

예를 들어, 변수별 엔트로피=
$\{E_A = 0.05, E_B = 0.07, E_C = 0.08,$
$E_D = 0.26, E_E = 0.32\}$ 이고
$\epsilon = \sigma(\sigma = 0.11)$라고 한다면
$|M - E(X)| < \epsilon$ $(M = 0.05)$ 조건을 만족하는 변수 A, B, C에 대해서만 조합을 고려하여 교호효과를 반영하는 것이다. 이 방법으로 분석을 할 경우 모수 ϵ의 값을 적게 잡을수록 계산량이 적어지는 효과가 있으며, 모수 ϵ의 값을 크게 잡을수록 Tree의 분석률을 높일 수 있다는 장점을 가진다.

• 향상된 k-1 알고리듬 Flow Chart

***Step 1.* (Initialization)**
분석자료의 변수 중 목표변수와 예측변수를 선택하고, 구성할 나무의 최대 깊이를 설정한다. 가지치기(pruning)를 위해 하나의 마디가 포함해야 할 최소 자료 수를 설정하고 목표변수 범주 값의 하한치(lower bound)와 상한치(upper bound)를 설정한다. 고려할 교호효과 정도에 따라 모수 **k**를 결정하고 마지막으로 모수 ϵ의 값을 결정한다.

***Step 2.* (변수추출)**
m개의 예측변수 중에서 $|M - E(X)| < \epsilon$를 만족하는 변수를 추출한다.

***Step 3.* (Root Node depth k 나무구성)**
추출된 변수 중에서 모수 **k**에 맞게 변수조합을 구해 각 경우의 엔트로피를 계산한다. 정보이익을 최대로 하는, 즉 최적분리를 갖는 조합을 선택하여 depth **k**인 나무를 해당 예측변수로 구성한다. 단, 이때 가지치기되는 마디는 생성되지 아니한다.

***Step 4.* (Child Node의 나무구성)**
depth **k** 이후의 나무는 기존의 C4.5 알고리듬을 이용하여 모든 가지의 마디가 끝마디가 될 때까지 나무를 구성해 나간다.

• 향상된 k-k 알고리듬 Flow Chart

***Step 1.* (Initialization)**
분석자료의 변수 중 목표변수와 예측변수를 선택하고, 구성할 나무의 최대 깊이를 설정한다. 가지치기를 위해 하나의 마디가 포함해야 할 최소 자료 수를 설정하고 목표변수 범주 값의 하한치와 상한치를 설정한다. 고려할 교호효과 정도에 따라 모수 **k**를 결정하고 모수 ϵ의 값을 결정한다.

***Step 2.* (변수추출)**
m개의 예측변수 중에서 $|M - E(X)| < \epsilon$를 만족하는 변수를 추출한다.

***Step 3.* (Root Node의 depth k 나무구성)**
추출된 변수 중에서 모수 **k**에 맞게 변수조합별 엔트로피를 계산한다. 최적분리를 갖는 조합을 선택하여 depth **k**인 나무를 해당변수로 구성한다. 단, 이때 가지치기되는 마디는 구성되지 아니한다.

***Step 4.* (Child Node의 depth k 나무구성)**
depth **k** 이후의 생성된 자식마디마다 끝마디가 아닐 경우 Step2, 3을 실시하여 depth **k** 나무를 생성한다. 이때 부모마디의 분리변수로 사용된 변수를 제외한다.

Table 1. k-1 Training data

var	ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8	ID9	ID10	ID11	ID12	ID13	ID14	ID15	ID16	ID17	ID18	ID19	ID20
A	1	2	3	1	2	3	1	3	3	1	3	1	3	2	3	1	3	1	2	1
B	1	2	1	1	3	3	2	1	2	2	1	1	3	3	1	2	2	1	2	2
C	2	1	2	2	2	2	3	1	1	3	2	2	2	2	1	3	1	2	1	3
D	1	2	1	1	2	2	1	2	2	1	1	1	2	2	2	1	2	1	2	1

***Step 5.* (Terminal Node 여부 확인)**
모든 가지의 마디가 끝마디가 되면 알고리듬을 끝낸다.

4. 알고리듬 테스트

4.1 k-1 알고리듬

<Table 1>은 4개 변수의 20case를 나타내고 있는 훈련데이터이다. 이를 가지고 변수 D를 목표변수로 하고 변수 A, B, C를 예측변수로 한 뒤 기존 C4.5 알고리듬과 **k-1**알고리듬(**k**=2)을 사용하여 각각 의사결정나무를 구성해 보았다.

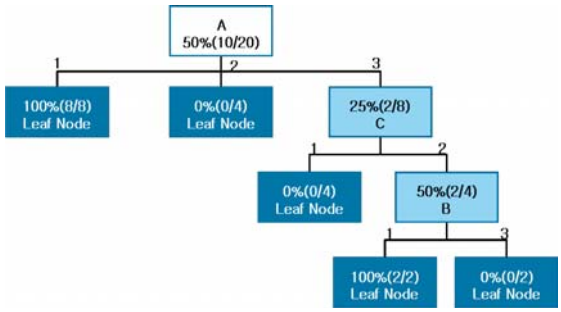

Figure 4. Decision tree for C4.5.

<Figure 4>에서 보는 바와 같이 C4.5 알고리듬을 사용한 경우 5개의 잎을 포함하여 총 8개의 마디를 가지고 깊이가 3인 의사결정나무가 구성되었다. 동일 자료를 가지고 k-1 알고리듬이 보다 간결해짐을 살펴볼 수 있다. 이는 변수들의 교호효과를 적용함으로써 보다 중요한 의사결정규칙이 발견되었음을 의미하는 것이다.

4.2 k-k 알고리듬

k-k 알고리듬을 적용하여 10만 데이터의 실 데이터를 가지고 분석을 실시하였다. 1절에서는 6변수의 5범주를 가지는 은행 데이터를 분석하였고 2절에서는 9변수의 2범주를 가지는 일본 신용카드 회사 자료의 분석을 통하여 **k-k** 알고리듬이 기존

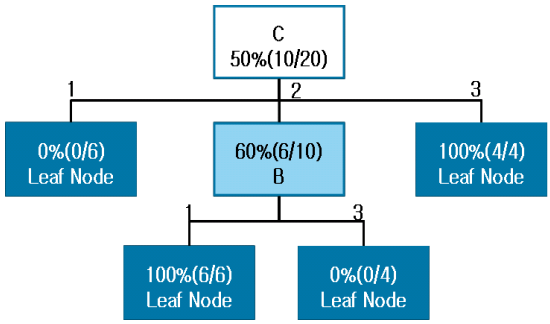

Figure 5. Decision tree for the level 2 variable Interaction.

알고리듬과 비교하여 어떠한 분석결과를 보이는지 알아보도록 한다.

4.2.1 국내 H은행 Database

<Table 2>는 분석을 위한 데이터 내용을 정리한 것이다. 이때의 훈련데이터는 나무를 구성하는 데 사용되는 데이터이고 검증 데이터는 나무의 설명력과 예측력을 평가, 검증하는 데 사용한다.

<Table 3>은 80% 신뢰도를 가지는 의사결정나무와 100% 신뢰도를 가지는 의사결정나무를 구해 C4.5 결과를 대조군으로 **k-k**를 실험군으로 하여 결과를 정리해 놓은 것이다. 각 경우를 나무의 깊이로 제한하여 가지치기에 따른 결과를 살펴보았으며 평가항목으로는 나무의 구조를 단적으로 말해주는

Table 2. Data of the Bank H

predictor variable	• 6 attribute • 5 element
target variable	• terminate/renewal
training data	• 80,000 cases
testing data	• 20,000 cases
estimate	• Node - total number of node • Ratio - misclassification ratio • Risk - misclassification risk

Table 3. Analysis results of Bank H

Algorithm	analysis	confidence : 80%, Minimum number of cases:1000			confidence : 100%, Minimum number of cases:0		
		depth 2	depth 4	full tree	depth 2	depth 4	full tree
C4.5	Node	18	58	60	23	182	278
	Ratio	0.2997	0.2085	0.2063	0.3250	0.1538	0.0908
	Risk	0.2845	0.2842	0.3052	0.2845	0.3319	0.2947
k-k (k=2)	Node	19	59	59	25	179	251
	Ratio	0.2675	0.1668	0.1668	0.2611	0.1108	0.0730
	Risk	0.2500	0.2680	0.2680	0.2500	0.3106	0.2731

총 마디 수와 나무의 분류율을 말해주는 오분류비율(Ratio), 예측률을 나타내는 오분류확률(Risk)로 나누어 분석하였다. <Table 3>에서 보는 바와 같이 신뢰도와 나무의 깊이에 따라 정도의 차이는 있으나 나무의 설명력을 말하는 오분류비율과 나무의 예측력을 나타내는 오분류확률이 기존의 C4.5로 분석한 결과보다 **k-k** 알고리듬을 적용하여 분석하였을 때 20% 정도 낮아짐을 볼 수 있다. 또한 분석의 깊이를 더할수록 **k-k**에 의한 분석결과가 나무의 마디수가 급격히 줄어들어 기존 C4.5로 분석한 나무구조가 보다 간단해 짐을 볼 수 있는데, 이것은 보다 강한 의사결정규칙으로 나무가 구성되어지고, 이로써 분류율과 예측률이 좋아지는 것을 나타낸다.

4.2.2 일본신용카드 Database를 이용한 실험

자료를 달리하여 1절의 국내 은행사의 자료를 분석한 것과 동일한 상황하에서 C4.5와 **k-k** 알고리듬으로 나무를 구성하여 결과를 분석해 보았다. <Table 4>는 분석자료의 내용과 평가항목을 정리한 것이다.

Table 4. Data of the Credit Card Company

predictor variable	• 9 attribute • 2 element
target variable	• approval/disapproval
training data	• 80,000 case
testing data	• 20,000 case
estimate	• Node - total number of node • Ratio - misclassification ratio • Risk - misclassification risk

이진 변수 9개를 예측변수로 하여 승인/비승인 여부를 분석하는 의사결정나무를 구성하였고 <Table 5>를 통해서 알 수 있듯이 전체적으로 분류율과 예측률이 앞의 은행사 자료보다는 좋음을 알 수 있다. 이것은 분석에 사용한 예측변수와 목표변수가 상호연관성이 높음을 나타내고 있는 것으로, 이 같은 경우에도 앞 절보다는 적은 효과이지만 **k-k**에 의한 의사결정나무의 정보품질이 더 우수함을 알 수 있다.

k-1 알고리듬에서 **k**를 크게 할 경우 교호효과의 최적화를 반영할 수 있지만, 문제의 크기가 커질 경우 알고리듬 수행시간이 기하급수적으로 증가하는 단점이 있다. 이에 비해 **k-k** 알고리듬은 한번에 고려하는 교호효과의 적용범위가 그룹별로 나누어져 교호효과의 최적화는 보장할 수 없지만 나무 전반에 걸쳐서 교호효과를 고려하면서 예측변수의 조합으로 나무를 구성함으로써 수행시간을 줄이면서도 보다 중요한 의사결정규칙을 발견함을 알 수 있다. 이는 기존 알고리듬인 C4.5의 단계별 최적 분리로 해결하지 못한 부분의 의사결정나무의 설명력과 예측력을 개선하면서도 문제의 크기가 커질 경우 수행시간에서 단점을 보이는 **k-1** 알고리듬의 단점을 보안하는 것이다.

4.3 향상된 교호효과 알고리듬

향상된 교호효과 알고리듬은 **k-1** 기법과 **k-k** 기법의 계산량 문제를 해결하고자 ϵ을 설정하여 교호효과를 고려하는 기법이다. 본 테스트 자료로는 Credit Screening Database(출처:UCI Machine Learning Repository), Census Income Database(출처:UCI Machine Learning Repository), 그리고 국내 A보험사 고객자료를 사용하였으며, 이 중 임의적으로 2~5 범주를 가지는 6~10개의 예측변수로 구성되어진 총 20개 데이터 set을 구성하여 분석하였다(<Table 6> 참조). 분석에는 ϵ의 값에 따른 나무의정보품질과 계산시간(sec)을 기존 알고리듬인 C4.5의 결과 값과 비교하였다.

이 실험은 공통적으로 8만 훈련 데이터를 가지고 나무를 구성하여 분석률(Ratio)을 체크하였고, 2만 검증 데이터를 가지고 예측

Table 5. Analysis results of the credit card company

Algorithm	analysis	confidence : 80%, Minimum number of cases:1000			confidence : 100%, Minimum number of cases:0		
		depth 2	depth 4	full tree	depth 2	depth 4	full tree
C4.5	Node	5	11	21	7	29	167
	Ratio	0.1688	0.1475	0.1413	0.1687	0.1466	0.1047
	Risk	0.0650	0.0900	0.0950	0.0650	0.0900	0.0612
k-k (k=2)	Node	5	9	25	7	31	147
	Ratio	0.1688	0.1452	0.1346	0.1687	0.1450	0.0923
	Risk	0.0650	0.0750	0.0850	0.0650	0.0750	0.0575

Table 6. Test results of the proposed methods

list	(variable, category)	analysis	C4.5	k=2(2-2)			k=3(3-3)			k=m(m-1)		
				0.5σ	σ	2σ	0.5σ	σ	2σ	0.5σ	σ	2σ
1	(6, 2)	Node	16	15	17	17	15	17	17	15	17	17
		Ratio	0.1123	0.1058	0.0758	0.0758	0.1058	0.0758	0.0758	0.1058	0.0758	0.0758
		Risk	0.0987	0.0902	0.0814	0.0814	0.0902	0.0814	0.0814	0.0902	0.0814	0.0814
		CPU Time	1	3	3	3	3	3	3	21	22	24
2	(6, 3)	Node	55	51	51	58	51	51	58	51	58	58
		Ratio	0.3451	0.3268	0.3268	0.2416	0.3268	0.3268	0.2416	0.3268	0.2416	0.2416
		Risk	0.4568	0.4351	0.4351	0.2984	0.4351	0.4351	0.2984	0.4351	0.2984	0.2984
		CPU Time	5	14	15	18	15	16	19	66	68	77
3	(6, 4)	Node	48	46	45	45	46	45	45	45	45	45
		Ratio	0.2645	0.2451	0.2215	0.2215	0.2451	0.2215	0.2215	0.2215	0.2215	0.2215
		Risk	0.2984	0.2754	0.2465	0.2465	0.2754	0.2465	0.2465	0.2465	0.2465	0.2465
		CPU Time	5	12	12	13	12	12	12	51	55	62
4	(6, 5)	Node	60	59	59	59	59	59	59	59	59	59
		Ratio	0.2063	0.1668	0.1668	0.1668	0.1668	0.1668	0.1668	0.1668	0.1668	0.1668
		Risk	0.3052	0.2680	0.2680	0.2680	0.2680	0.2680	0.2680	0.2680	0.2680	0.2680
		CPU Time	6	12	13	15	11	11	13	59	62	70
5	(7, 2)	Node	9	10	11	13	10	11	13	12	11	13
		Ratio	0.0571	0.0543	0.0535	0.0511	0.0543	0.0543	0.0511	0.0537	0.0522	0.0511
		Risk	0.0978	0.0968	0.1023	0.1045	0.0968	0.1023	0.1045	0.1012	0.1009	0.1134
		CPU Time	1	2	3	3	2	3	3	15	16	19
6	(7, 3)	Node	26	31	25	25	30	25	25	30	27	27
		Ratio	0.1975	0.1842	0.1659	0.1659	0.1876	0.1659	0.1659	0.1876	0.1597	0.1597
		Risk	0.2254	0.2165	0.1965	0.1965	0.2203	0.1965	0.1965	0.2203	0.2412	0.2412
		CPU Time	2	7	7	7	9	9	11	27	30	35
7	(7, 4)	Node	38	42	42	42	42	42	42	42	42	42
		Ratio	0.2761	0.2416	0.2416	0.2416	0.2416	0.2416	0.2416	0.2416	0.2416	0.2416
		Risk	0.3212	0.3068	0.3068	0.3068	0.3068	0.3068	0.3068	0.3068	0.3068	0.3068
		CPU Time	5	16	17	19	19	22	25	59	66	76
8	(7, 5)	Node	64	61	54	54	61	54	54	63	54	54
		Ratio	0.2231	0.2101	0.1849	0.1849	0.2101	0.1849	0.1849	0.2031	0.1849	0.1849
		Risk	0.1965	0.1869	0.1648	0.1648	0.1869	0.1648	0.1648	0.2154	0.1648	0.1648
		CPU Time	7	22	22	26	26	29	33	75	81	88
9	(8, 2)	Node	24	24	24	24	24	24	24	24	24	24
		Ratio	0.1642	0.1642	0.1642	0.1642	0.1642	0.1642	0.1642	0.1642	0.1642	0.1642
		Risk	0.1609	0.1609	0.1609	0.1609	0.1609	0.1609	0.1609	0.1609	0.1609	0.1609
		CPU Time	2	6	6	6	7	7	7	22	24	27
10	(8, 3)	Node	31	29	36	36	29	36	36	30	36	36
		Ratio	0.1124	0.1067	0.0843	0.0843	0.1067	0.0843	0.0843	0.1006	0.0843	0.0843
		Risk	0.2341	0.2234	0.1675	0.1675	0.2234	0.1675	0.1675	0.2351	0.1675	0.1675
		CPU Time	3	10	12	13	12	15	19	41	48	53
11	(8, 4)	Node	68	61	57	57	61	57	57	63	57	57
		Ratio	0.2988	0.2765	0.2389	0.2389	0.2765	0.2389	0.2389	0.2591	0.2389	0.2389
		Risk	0.3512	0.3426	0.2954	0.2954	0.3426	0.2954	0.2954	0.3234	0.2954	0.2954
		CPU Time	8	30	33	39	41	42	47	84	90	95
12	(8, 5)	Node	89	83	77	77	80	77	77	80	77	77
		Ratio	0.2213	0.1984	0.1799	0.1799	0.1945	0.1799	0.1799	0.1945	0.1799	0.1799
		Risk	0.3154	0.2957	0.2684	0.2684	0.3056	0.2684	0.2684	0.3056	0.2684	0.2684
		CPU Time	10	35	38	42	45	50	55	110	119	132
13	(9, 2)	Node	21	21	25	25	21	25	25	21	25	25
		Ratio	0.1413	0.1413	0.1346	0.1346	0.1413	0.1346	0.1346	0.1413	0.1346	0.1346
		Risk	0.0950	0.0950	0.0850	0.0850	0.0950	0.0850	0.0850	0.0950	0.0850	0.850
		CPU Time	3	5	5	6	5	5	6	19	21	25
14	(9, 3)	Node	101	95	87	87	90	87	87	90	87	87
		Ratio	0.1667	0.1762	0.1416	0.1416	0.1695	0.1416	0.1416	0.1695	0.1416	0.1416
		Risk	0.2203	0.2135	0.1897	0.2135	0.2004	0.1897	0.2135	0.2004	0.2135	0.2135
		CPU Time	12	40	44	55	50	58	65	132	146	163
15	(9, 4)	Node	125	106	95	95	106	95	95	104	87	87
		Ratio	0.2351	0.2159	0.1986	0.1986	0.2159	0.1986	0.1986	0.2131	0.1853	0.1853
		Risk	0.2651	0.2461	0.2168	0.2168	0.2461	0.2168	0.2168	0.2418	0.1987	0.1987
		CPU Time	15	51	58	71	65	75	89	170	189	213

list	(variable, category)	analysis	C4.5	k=2(2-2)			k=3(3-3)			k=m(m-1)		
				0.5σ	σ	2σ	0.5σ	σ	2σ	0.5σ	σ	2σ
16	(9, 5)	Node	187	169	154	154	171	154	154	162	143	143
		Ratio	0.2580	0.2234	0.2037	0.2037	0.2198	0.2037	0.2037	0.2085	0.1862	0.1862
		Risk	0.3261	0.3008	0.2891	0.2891	0.3084	0.2891	0.2891	0.2856	0.2357	0.2357
		CPU Time	21	73	89	101	89	108	134	239	267	346
17	(10, 2)	Node	72	72	61	55	72	55	55	63	53	53
		Ratio	0.1309	0.1309	0.1132	0.0965	0.1309	0.0965	0.0965	0.1024	0.0891	0.0891
		Risk	0.1754	0.1754	0.1498	0.1561	0.1754	0.1561	0.1561	0.1481	0.1384	0.1384
		CPU Time	6	31	42	58	39	55	61	89	100	151
18	(10, 3)	Node	145	131	116	116	133	116	116	128	109	109
		Ratio	0.1987	0.1884	0.1690	0.1690	0.1813	0.1690	0.1690	0.1810	0.1408	0.1408
		Risk	0.2860	0.2710	0.2449	0.2449	0.2765	0.2449	0.2449	0.2214	0.2381	0.2381
		CPU Time	15	52	61	72	69	89	99	168	208	287
19	(10, 4)	Node	224	201	201	201	193	193	193	193	178	178
		Ratio	0.2197	0.2036	0.2036	0.2036	0.1984	0.1984	0.1984	0.1984	0.1761	0.1761
		Risk	0.3687	0.3587	0.3587	0.3587	0.3516	0.3516	0.3516	0.3516	0.3205	0.3205
		CPU Time	34	98	120	145	132	157	189	342	403	541
20	(10, 5)	Node	354	326	306	306	317	301	301	291	278	278
		Ratio	0.3160	0.3021	0.2491	0.2491	0.2987	0.2441	0.2441	0.2361	0.2184	0.2184
		Risk	0.2784	0.2813	0.2641	0.2641	0.2680	0.2591	0.2591	0.2443	0.2394	0.2394
		CPU Time	46	131	152	186	161	184	221	457	599	876

률(Risk)을 측정한 것이다. 이때 가지치기를 위한 정보의 신뢰도는 80%, 마디 최소 자료 수는 1000개로 제한하였다. 위의 체크 사항으로 Node는 나무를 구성할 때 생성된 총 마디의 개수를 의미하고, Ratio는 나무를 분류(Classification)기능으로 사용할 경우 오분류 비율을 나타내는 수치이고, Risk는 나무를 예측 (Prediction)기능으로 사용할 경우 오분류 확률을 나타내며, CPU Time(단위:초)은 알고리듬 수행시간이다. 이를 모수 **k**와 ϵ을 분산표시법에 의거하여 그 정도를 달리하며 기존의 C4.5 알고리듬과 비교분석한 것이다.

분석결과를 살펴보면 기존의 C4.5 알고리듬과 비교하여 ϵ을 사용할 경우 총 생성된 마디 수가 감소하여 나무가 보다 간결해짐을 알 수 있고 그 효과는 Training Set이 클수록 크게 나타났다. 또한 정보의 품질면에서도 우수해짐을 살펴볼 수 있다. 더욱이 그 효과는 교호효과의 적용 범위가 넓어질수록 증가되는데, 데이터마다 경우가 조금씩은 다르지만 일반적으로 계산시간 대비 정보품질을 살펴볼 때, $\epsilon = \sigma$일 때 가장 효율적임을 알 수 있다. <Figure 6>, <Figure 7>은 C4.5와 $\epsilon = \sigma$일 때의 분석결과를 도식화한 것이다.

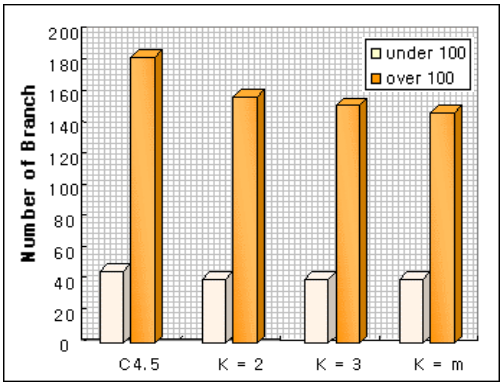

Figure 6. Average number of branches.

위의 히스토그램을 통해서 주요변수의 교호효과만을 고려하는 것만으로도 약 20%의 정보품질(분류율, 예측률) 상승이 나타남을 알 수 있다. 또한 이러한 상승효과는 문제가 보다 복잡

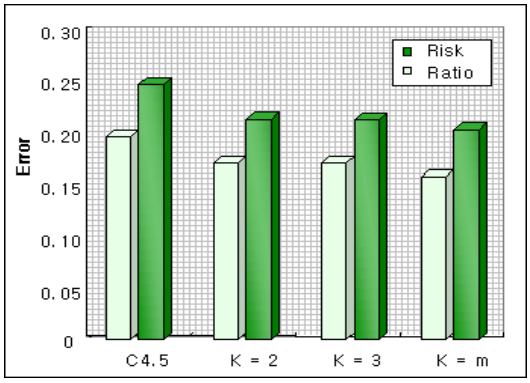

Figure 7. Comparison of ratio and risk.

해지고 커질수록 효과가 크게 나타남을 알 수 있다. 모수 **k**를 전체 예측변수 크기로 정하여 분석하는 것이 가장 큰 효과를 나타내지만 빠른 시간 내의 분석을 위해서는 적은 값으로 하여 **k-k** 기법을 적용해도 무방할 만큼의 효과를 볼 수 있었다. 앞서도 언급한 내용이지만 대규모 문제일수록 교호효과를 고려하는 것으로 보다 강력한 의사결정규칙을 얻을 수 있었다. 이러한 규칙의 발견은 작은 규모의 문제에서는 총 마디 수를 증가시키기도 하지만 대부분의 문제에서는 의사결정나무가 보다 간결하게 만드는 원인이 되어 보다 용이한 분석이 될 수 있게 한다.

4.4 알고리듬별 장・단점 비교

지금까지 제시한 알고리듬의 장점 및 단점을 살펴보면

Table 7. The Merits and demerits of the proposed algorithm

algorithm	strength	weakness	remark
k-1	• consideration of variable interaction • maximize information quality	• computational burden • less efficient for large size	• perfect decision tree
k-k	• consideration of variable interaction • efficient computation	• less efficient for variable interaction	• grouping strategy • Step by Step
new algorithm	• robustness • consideration of variable interaction • efficient computation	• algorithm performance depends on the choice of ϵ	

<Table 7>과 같다. 소규모 문제에는 **k-1** 알고리듬을 통해서 분석하는 것이 가장 이상적인 결과를 제시할 수 있다 하지만 실제적인 자료를 분석하는 데는 실효적인 계산이 불가능하므로 이런 경우는 그룹별 교호효과에 대한 고려만으로 분석이 용이할 경우에는 **k-k** 기법을 적용하고 보다 빠른 시간 내에 분석이 요구될 경우에는 ϵ을 고려해서 향상된 **k-1** 기법으로 분석하는 것이 용이하다. 보다 큰 대규모 문제를 해결해야 하는 경우에는 적절한 ϵ을 선택하여 향상된 **k-k** 기법으로 교호효과를 고려해서 분석하는 것이 효과적임을 알 수 있다.

5. 결 론

본 연구에서는 기존의 의사결정나무 기법이 독립된 변수의 단계별 구성에서 탈피하여 교호효과를 고려하여 변수들의 상호연관관계에 의한 나무구성 알고리듬을 제안하였다. **k-1** 기법과 **k-k** 기법으로 단계별 최적분리의 기존 알고리듬의 한계를 해결할 수 있음을 알 수 있었다. **k-1** 알고리듬은 가장 이상적인 결과를 제시하지만 계산량에 있어서 그 실효성이 떨어질 수 있었고 **k-k** 알고리듬은 계산량 문제를 어느 정도 해결할 수 있었지만 정보품질 면에서 그 효과가 미비할 수 있다는 문제점이 있었다. 이들의 문제를 해결하기 위해 분석에 있어서의 주요변수를 파악함으로써 그들의 교호효과를 고려하는 향상된 교호효과 알고리듬을 제안하였고, 실험을 통하여 변수들 간의 상호작용을 보다 정확하고 효과적으로 이해할 수 있으며 나무의 분석률과 예측률을 높이면서도 신뢰성을 확보할 수 있었다. 뿐만 아니라 변수들의 교호효과를 살림으로써 불필요한 마디와 가지의 생성을 막아, 보다 간결한 나무를 생성하여 강력한 의사결정규칙을 발견할 수 있음을 실험을 통해 증명하였다. 또한 보다 정확한 분석 및 예측이 이루어지게 됨으로써 모형의 실제적 유용성을 높일 수 있었다. 이러한 방법은 Over-Fitting 현상을 줄이기 위해 가지치기를 할 경우 그 효과가 더 증폭되며, 이는 향후 데이터 수집과 관리 작업을 향상시킬 수 있다. 실험을 통해 $\epsilon = \sigma(\sigma = E_i$의 표준편차, $i =$예측변수)로 하는 것이 정보의 품질(분석률, 예측률)을 높이면서도 실효성 있는 계산량을 가짐으로써 가장 효과적인 것으로 판명되었다. 이로써 그동안의 계산량 문제점을 해결함은 물론 변수들의 교호효과를 고려한 의사결정나무를 생성할 수 있었다. 이로써 앞으로의 연구는 최적분리변수를 찾는 방법이 아니라 주요 분리변수를 찾아 그들의 교호효과를 고려하는 데 초점이 맞춰져야 함을 나타낸다. 이는 기존의 의사결정나무를 새로운 시각으로 조명한 결과이며, 앞으로 분석 및 예측의 도구로서 의사결정나무기법의 활용도는 더욱 증가될 것이다.

참고문헌

Brieman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J.(1984), *Classification and Regression Trees*, Wadsorth.

Fayyad, U. M., Piatesky, G. Shapiro and Smith, P.(1996.), From Data Mining to Knowledge Discovery: an overview. In Fayyad, U.M. *et al*. (Eds) *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1-34.

Kass, G.(1980), An exploratory technique for investigating large quantities of categorical data, *Applied Statistics*.

Quinlan, J. R.(1986), Induction of Decision Trees. *Machine Learning*, **1**(1).

Quinlan, J. R(1993), *C4.5: Programs for Machine Learning*, San Mateo, Calif., Morgan Kanfmann.

Quinlan, J. R(1996)., Bagging, Boosting, and C4.5, *Proceedings of 13th National Conference on Artificial Intelligence*, Portland, Oregon, August 4-8.

Shlien, S(1990)., Multiple Binary Decision Tree Classifiers, *Pattern Recognition*, **23**(7), 757-763.

Smyth, P., and Goodman, R.M.(1992), An Information Theoretic approach to Rule Induction From Database. *IEEE Transaction on Knowledge and Data Engineering*, **4**(4), 301-316.