병렬확장을 활용한 규칙생성 기법

이 기 철† · 김 진 봉†

요 약

가공되지 않은 데이터에서 직접 규칙 형태의 지식을 추출하는 문제는 자료의 홍수 속에서 정보의 부족을 느끼는 모순을 해결하기 위한 데이터 마이닝 분야에서 매우 중요하다. 논리 최적화 도구는 주어진 ON 집합과 DC 집합을 이용하여 최적화된 형태의 지식을 추출하는 도구인데, 본 논문에서는 논리 최적화 기법 중 병렬 확장 기법을 이용하여 초기 지식을 생성한 후 정렬, 축소, 규칙 확장 등의 방법을 이용하여 실 세계 데이터에 적용할 수 있는 규칙이 생성될 수 있음을 보였다. 이와 같은 새로운 접근 방법이 종래의 C4.5 등의 결정 트리 기법에 손색없는 규칙을 생성할 수 있음을 실험을 통해 입증하였다.

A Rule Generation Technique Utilizing a Parallel Expansion Method

Kee-Cheol Lee† · Jin-Bong Kim†

ABSTRACT

Extraction of knowledge, especially in the form of rules, from raw data is very important in data mining, the aim of which is to help users who feel the lack of knowledge in spite of the abundance of data. Logic minimization tools are ones which derive optimized knowledge given ON set and DC set. First, the parallel expansion scheme of logic minimization is extracted and used to obtain initial knowledge. Then, sorting, reduction, and rule expansion schemes are designed and applied to the initial knowledge to get final rules, which are successfully applicable to real world data. The prototype system based on this new approach has been experimented with real world data to show that it is as practical as conventional long studied decision tree methods like C4.5 system.

1. 서 론

정보화 시대에서 밀려드는 자료를 대하고 있는 현대인은 넘치는 자료속에서 정보의 부족을 느끼는 어려움을 겪고 있다. 80년대에 각광을 받은 전문가 시스템은 전문지식의 수집이라는 병목 현상 때문에 어려움을 겪고 있다. 비전문가에 의한 전문지식의 획득은 성공적인 전문가 시스템 또는 상담 시스템을 위해 반드시 필요한데, 결국 가공되지 않은 데이터에서 직접 지식을 추출하는 작업이 중요하다. 자료의 홍수 속에서 필요한 정보의 부족이라는 모순을 해결하는 지식 추출의 연구가 최근 각광을 받기 시작한 데이터 마이닝이다[1-4]. 본 논문에서는 사용하기 용이한 규칙 형태의 지식을 추출하는 한가지 새로운 접근 방식을 제시하고자 한다.

가공되지 않은 데이터에서 지식을 추출하는 연구는 다양하다. 양의 예제 또는 음의 예제가 입력됨에 따라, 가장 일반적인 경계와 최소한의 경계를 유지하면서, 이 두 경계의 크기가 같을 때까지 점진적 학습을 계속하여 주어진 개념을 알아내는 버전 공간(Version Space)[5], 정보이론적인 기법으로 속성을 선택하여

※ 본 연구는 과학재단 특정기초연구 97-01-02-04-01-3(제1세부과제)로 수행되었음.
† 정 회 원 : 홍익대학교 컴퓨터공학과
논문접수 : 1997년 10월 23일, 심사완료 : 1998년 2월 17일

결정 트리를 구성해가는 ID3[6]와 C4.5[7] 등을 들 수 있다. 특히 결정 트리 기법을 기반으로 하는 C4.5의 출력에서 규칙을 생성하는 C4.5rules 시스템은 실세계 데이터(real world data)에서부터 규칙을 얻는 매우 실용적인 방법으로 볼 수 있다. 이외에도 출력단의 에러를 최소화하도록, 뉴론(신경세포)간의 연결 강도를 조정하는 역전파(backpropagation) 기법[8], 내장된 생성용 귀납 규칙들(Constructive induction rules)을 이용하여 입력 데이터의 일반화를 구하는 INDUCE 시스템[9] 등을 역사적인 의미를 갖는 시스템으로 볼 수 있으며, 기존의 기법이 갖는 장단점을 파악하기 위한 비교 연구도 진행되어 왔다[10]. 논리 최적화 기법도 주어진 양의 예제, 음의 예제, 및 DC(don't care) 예제들로 부터 가장 확장된 큐브(cube)를 생성한 다는 점에서 유사성이 있으나, 데이터에 잡음이 없고 테스트 데이터가 따로 존재하지 않는 등의 차이점이 있는 영역이다. 이 논리 최적화의 연구는 매우 수학적이고, 체계적인 연구가 수행되어 왔으나, 논리최적화와 규칙 추출이라는 영역의 차이점으로 인하여 데이터 마이닝을 위해 사용되기 어려웠다.

본 논문에서는 디지털 논리최소화 기법을 규칙 생성 시스템으로 이용하는 방법을 연구하고, 그 실험 결과로 방법의 타당성을 입증하려고 한다. 본 접근 방법에서는 초기 규칙 생성을 위해 휴리스틱 기법에 의존하는 기법에서 벗어나[11,12], 검증된 시스템인 에스프레소(Espresso) 논리 최적화 시스템에서의 병렬 확장 루틴을 이용한다. 논리 최적화 기법은 주어진 양의 예제들의 집합(ON 집합)과 음의 예제 들의 집합(OFF 집합), 양의 예제거나 음의 예제 어느 것이나 가능한 집합(DC 집합)을 이용하여, 음의 예제들에 어긋나지 않는 최대 크기의 규칙(논리 최적화 기법에서는 큐브라고 함)을 생성하는 기법으로 가공되지 않은 데이터로 부터 규칙을 생성해내는 규칙 추출과 유사성을 가지고 있다. 그러나 논리 최적화와 규칙추출에는 다음과 같은 차이점이 있다.

(1) 규칙 추출 영역의 경우 ON 집합과 OFF 집합으로 주어지는 데이터가 문제의 영역의 크기에 비해 통상 엄청나게 작으므로, DC 집합이 다룰 수 없을 정도로 커질 수 있다.

(2) 논리 최적화의 경우는 주어지는 양의 데이터 및 음의 데이터가 완전 신뢰할 수 있는 데이터이나, 규칙 추출의 경우, 주어진 데이터에 잡음이 존재할 수 있다.

(3) 논리 최적화의 경우는 학습에 이용되는 데이터 자체를 완전 인식하는 규칙이 요구되지만, 규칙 추출의 경우는 학습용 데이터에 대한 인식률 보다는 학습에 참가하지 않은 별도의 테스트 데이터에 대한 인식률이 중요하다. 즉 학습용 데이터에 존재하는 잡음에 적응하는 규칙의 생성은 학습용 데이터의 인식률은 증가시키지만, 테스트 데이터의 인식률은 오히려 감소하는 과적응(Overfitting)의 현상이 발생할 수 있으며, 이를 피할 수 있어야 한다.

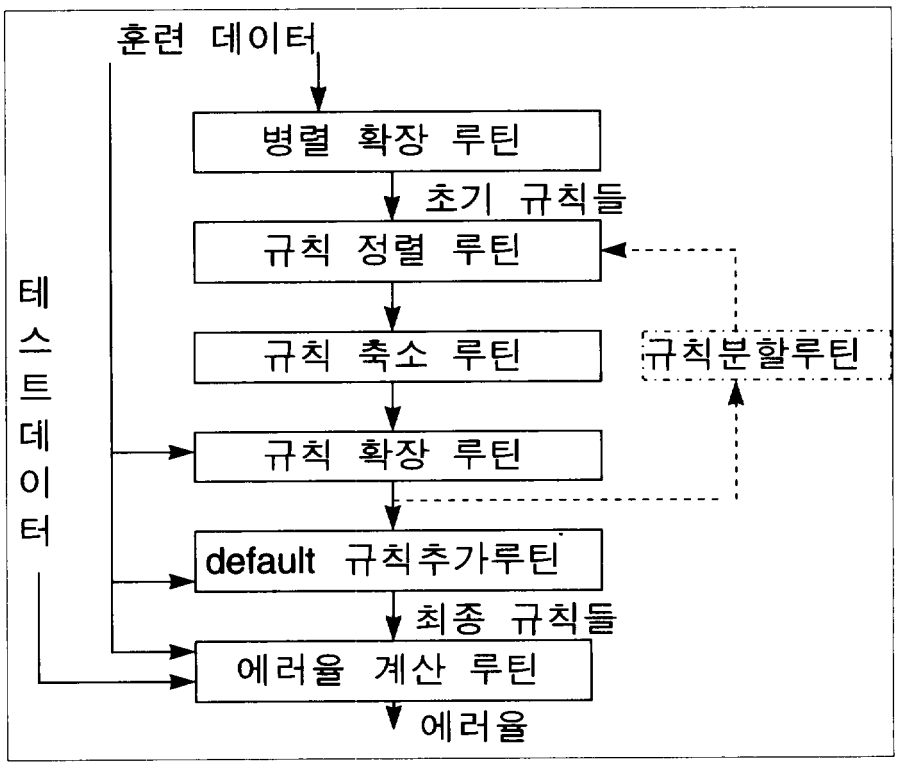
이상의 문제로, 상당한 수준에 도달해 있는 논리 최적화 도구를 규칙획득 도구로 사용할 수 없었다. 이러한 문제를 해결하기 위해, 기존의 에스프레소 시스템의 주요루틴 들을 규칙획득 도구로 사용하기 위한 연구를 진행하였다. 연구 방법으로는 주요 루틴들을 DC 집합을 사용하지 않는 유사 루틴으로 변환하는 방법으로 바꾸거나, 에스프레소 시스템의 가장 핵심 루틴이라 할 수 있는 병렬 확장 루틴의 출력을 추출하여 이를 사후 보완하는 방법이 가능하며, 이 후자의 접근방법을 본 논문에서 제시하고자 한다.

2. 시스템의 설계

(그림 1)에는 본 시스템의 계통도가 명시되어 있다. 우선 논리 최적화 도구인 에스프레소 시스템의 주요 루틴 중에서 DC 집합을 사용하지 않는 병렬 확장 루틴을 이용하여 훈련 데이터에 충실한 규칙을 생성한다. 이 규칙들은 훈련 데이터에 대한 인식률은 거의 완벽에 가까운데, 대신 훈련 데이터에 있는 특별 경우(special case)들이나, 잡음 데이터까지도 적응하여, 훈련에 참가하지 않은 테스트 데이터에 적응하지 못하는 과적응(Overfitting) 현상이 발생할 수 있으며, 이 과정에서 규칙의 수도 너무 많아질 수 있다. 계통도 상의 다른 루틴들은 결국 과적응 극복과, 규칙 수의 축소를 위해 필요한 루틴으로 이를 통해 유용한 규칙 생성 시스템이 가능하다. 이제 주요 루틴 별로 그 기능을 알아 보자. 단 점선으로 연결된 규칙 분할 루틴은 본 논문에서는 제외된다.

2.1 속성, 속성값, 데이터와 규칙

규칙 생성은 가공되지 않은 주어진 데이터로 부터, 규칙을 추출하는 과정을 이른다. 문제는 입력 데이터가 속한 공간이 방대하여, 주어진 데이터는 가능한 공간의



(그림 1) 규칙 생성도구의 시스템 계통도
(Fig. 1) The system flow of the rule generation tool

극히 일부에 지나지 않는다는 점이다. 한정된 데이터로부터, 비교적 정확한 규칙을 생성한다면, 전문가의 도움을 받지 않거나, 적은 도움 만으로도 전문가 시스템을 제작할 수 있다.

크기	색깔	표면질	클래스
소	황색	매끄러움	A
중	적색	매끄러움	A
소	적색	매끄러움	A
대	적색	거침	A
중	황색	매끄러움	B
중	황색	거침	B

크기, 색깔 및 표면질의 3 가지 속성을 가진 간단한 영역을 생각해 보자. 크기 속성은 소, 중, 대의 3가지 속성값을 갖고, 색깔 속성은 황색,적색의 2가지 속성값을, 표면질은 매끄러움, 거침의 2가지 속성값을 갖는다고 가정하자. 클래스 A를 기준으로 보면, 앞의 네 개의 데이터는 양의 예제(positive instances)이고, 나머지 데이터는 음의 예제(negative instances)이다. 규칙의 추출은 이 같이 주어진 한정된 예제들로 부터 일반적인 규칙을 추출하는 과정이며, 예를 들면 다음의 규칙이 추출될 수 있다.

If (색깔=황색) and (크기=소), then 클래스 A에 소속된다.

If (색깔=적색), then 클래스 A에 소속된다.

If (색깔=황색) and (크기=중), then 클래스 B에 소속된다.

본 논문에서 제시하는 논리 최적화 기법을 이용한 방법에서는 다음과 같은 규칙들이 생성된다.

If (크기 in [소, 대]), then 클래스 A에 소속된다.

If (크기 = 중) and (색깔 = 적) and (표면질 = 매끄러움), then 클래스 A에 소속된다.

If (크기 = 중) and (색깔 = 황), then 클래스 B에 소속된다.

즉 본 논문에서 생성된 규칙의 좌변에 존재하는 조건은 (속성 in {속성값1, 속성값2,...})의 형태를 가질 수 있어, 결정트리기법 등에서 흔히 사용하는 (속성=속성값) 형태보다는 일반적이다. 즉 후자의 경우는 속성이 한 개의 속성값으로 구성된 집합에서 속성값을 선택하는 경우에 해당된다. 예를 들어 (크기 in {중}) 이란 조건은 (크기 = 중) 과 일치한다. 또 모든 속성값을 가질 수 있을 때는 그 조건은 생략된다. 예를 들어 (크기 in {소, 중, 대}) 란 조건은 생략될 수 있다.

일반적으로 생성된 규칙들은 훈련에 사용된 데이터에 매우 충실하며, 훈련에 참가하지 않은 테스트 데이터에 대한 비교적 정확한 예측이 가능해야 한다. 따라서 테스트 데이터에 대한 인식률을 높이기 위해 다양한 학습기법이 존재할 수 있다. 생성된 규칙의 평가는 테스트 데이터에 대한 인식률, 규칙의 수, 생성된 규칙의 복잡성, 학습 시간 등을 고려해야 한다. 따라서, 최종적으로 생성된 규칙이 훈련 데이터에 대한 인식률은 낮아도, 미지의 테스트 데이터에 대한 인식률이 높을 수 있다. 본 기법에서는 추가로 default 규칙을 생성하는데 이 규칙은 미지의 데이터를 만족시키는 규칙이 없을 때에 적용하는 규칙으로 위의 예에서는 클래스 A 로 판정한다. 물론 이 default 규칙은 문제 영역에 따라서는 적용하지 않을 수도 있다. 위의 예는 3가지 속성, 속성당 평균 2.3개의 속성값을 갖는 2 클래스 문제의 훈련 데이터 8 개 만을 사용하여 단 3 개의 규칙 만이 생성된 경우이다. 본 논문에서는 수십개의 속성을 갖는 수천, 수만 개 수준의 데이터를 처리하는 새로운 기법을 제시하고 분석한다.

2.2 병렬 확장

병렬 확장 루틴은 초기의 규칙을 생성한다는 점에서 매우 중요하다. 기존의 결정 트리 기법과 같은 휴리스틱에 의한 기법보다는 좀 더 정리된 논리최적화 기법인 에스프레소 시스템[13]에서 확장 루틴을 추출 적용하여 훈련 데이터에 매우 충실한 초기 규칙을 사용하였다. (그림 2)의 EXPAND 루틴이 그 개요를 설명하고 있다. 에스프레소에는 다양한 루틴이 존재하는데, 대부분의 루틴과는 달리 본 병렬 확장 루틴은 DC 집합을 필요로 하지 않으므로, 초기 규칙 생성용으로 사용할 수 있다. 기타의 루틴을 DC 집합을 이용하지 않는 방식으로의 변환 등도 일부 연구되었으나 본 논문에서는 사용하지 않았다[14].

```
procedure EXPAND(F, R)
{
        F := DECREASING_ORDER(F);
        for each f in F {
          (W, f) := EXPAND1(f, F, R);
          F := (F ∪ {f}) - W;
        }
}
```

(그림 2) 병렬 확장 기법
(Fig. 2) Parallel expansion scheme

여기서, **F**에는 양의 예제 들의 집합 또는 동작이 진전되면서, 양의 예제들을 포함하는 규칙으로 바뀌어 저장된다. **R**은 음의 예제들의 집합이다. EXPAND는 F의 각 예제 (또는 규칙)를 크기가 큰 것부터 확장을 시도하여, 확장된 규칙에 포함되는 예제들(W로 표기되었음)을 제거하는 방식을 모든 남아 있는 **F**의 성분에 적용한다. 그 복잡성 때문에 여기에서 생략된 EXPAND1은 주어진 예제 또는 규칙인 f를 확장한 새로운 f와 그 f가 포함하는 **F** 상의 모든 성분들의 집합 W를 반환한다. EXPAND1은 주어진 규칙 f가 **F**와 **R**의 각 성분을 어떻게 포함하는 가의 여부를 저장하는 포함 행렬(covering matrix) **C**와 봉쇄 행렬(blocking matrix) **B**를 이용하여, 확장되어서는 안되는 속성(=변수)들을 다양한 방식으로 결정한 후, 나머지 속성들을 동시에 확장하는 매우 체계적인 방식이다. 자세한 내용은 본 논문의 범위를 벗어난다[13].

2.3 규칙 정렬 및 규칙 축소

병렬 확장 기법으로 생성된 초기 규칙들은 그 특성상 훈련데이터에는 매우 충실하나, 훈련데이터가 가진 잡음 등에 과적응하여, 훈련에 참가하지 않은 데이터에 대한 인식률이 감소할 수 있다. 이를 위해 초기 생성된 규칙은 축소되어 다른 확장을 시도하게 되는데, 이 규칙 축소를 위한 준비작업으로 규칙 정렬을 할 수 있다. (그림 3)에는 규칙 정렬 기법이 설명되어 있다. 단 여기서, 규칙의 크기는 각 규칙의 속성의 개수가 작은 순으로 하며, 속성의 수가 같으면 규칙이 가진 조건의 수

가 큰 순으로 한다.

규칙은 (그림 4)의 방식으로 축소된다. 즉 현재의 각 규칙(f)에 대해, 원래의 주어진 데이터중 현재까지 다른 규칙이 포함하지 않고 f가 포함하는 데이터들(D)을 포함하는 최소의 규칙 f'으로 축소해 간다. 이미 사용한 규칙이 포함하는 데이터는 고려하지 않으므로, 규칙 f는 포함하는 데이터가 없어, 제거될 수 있으며, 이에 따라 규칙의 수효를 줄이는데 기여할 수 있다. 이를 위한 준비 작업이 규칙 정렬이며, 규칙의 수효를 많이 제거하도록, 규칙 정렬 루틴이 설계되었다. 규칙 정렬 루틴에서 큰 규칙을 중요시 여기는 이유는 큰 규칙이 먼저 사용되면, 잔여 데이터의 수가 감소하여, 규칙 축소에 기여할 수 있기 때문이다. 또한 규칙 정렬시에 처음 선택된 규칙과 가까운 규칙이 선택되므로 이미 삭제된 데이터를 고려한 더 작은 규칙 생성에 도움을 준다. 만약 규칙 축소를 위해 해당 규칙이 포함하는 모든 데이터를 이용하도록 한다면, 즉 스텝 R4를 생략한다면, 규칙 정렬은 필요없으며, 규칙 축소시에 규칙의 수의 감소는 발생하지 않을 것이다.

```
procedure SORT(F)
{
가장 큰 규칙을 F에서 선택하여 F'에 저장하자;
나머지 규칙들을 처음 선택된 규칙에서의 거리 순으로 정렬하여 F'에 추가한다. 단 정렬시 거리가 같은 경우 큰 규칙을 우선 선택한다 :
  return F';
}
```

(그림 3) 규칙 정렬
(Fig. 3) Rule sorting

```
procedure REDUCE(F)
{
  remFF := FF; /* FF는 원래의 데이터들의 모임 */
  F' := {}; /* 공집합으로 초기화 한다. */
  for f := each rule in F {
    D := f에 포함되는 모든 데이터 들의
         집합; /*R1*/
    D를 포함하는 최소의 규칙을 생성하여 f'
     이라 부르자; /*R2*/
    remFF := remFF - D; /*R3*/
    F := F - {f}; /*R4*/
    if f' is not empty,
    then F' := F' U {f'}; /*R5*/
  }
  return F';
}
```

(그림 4) 규칙 축소
(Fig. 4) Rule reduction

2.4 규칙 확장 루틴

규칙 축소 기법으로 축소된 규칙은 테스트 데이터에 대한 인식률을 제고하고, 규칙의 수를 줄이는 방향으로 확대되어야 한다. 그림 5에는 본 시스템에서 사용한 규칙 확장 루틴이 요약되어 있다. 단 실제 구현 시에는, 데이터는 파일에서 읽어 오며, 규칙은 주기억장치에 저장하여 사용하였다. 이는 방대한 데이터의 처리를 위해서 이며, 이러한 데이터 파일을 읽어들이는 횟수 즉 패스(pass)의 수는 규칙 확장 루틴의 속도에 가장 큰 요소이다. C0(f)는 규칙 f의 조건을 뿐 아니라, 클래스까지 만족하는 데이터의 수이고, E0(f)는 규칙 f의 조건은 만족하되 클래스가 틀린 데이터들의 수이다. 또, 규칙 f의 속성 v를 확장하면(즉 속성을 제거하면), 그 조건을 만족시킬 수 있는 데이터들 중 클래스가 규칙 f와 동일한 데이터의 수효를 C(f, v)라 하고, 클래스가 규칙 f와 다른 데이터의 수효를 E(f, v)라고 한다. 훈련 데이터만의 인식률을 고려한다면, 규칙 f의 속성 v만을 확장하여 인식률이 좋아질 경우는 다음과 같이 훈련에러율이 감소하는 경우이다.

$$\frac{E0(f)+E(f,v)}{C0(f)+C(f,v)+E0(f)+E(f,v)} < \frac{E0(f)}{C0(f)+E0(f)}$$

그러나 이는 테스트 데이터에 대한 인식률을 고려한다고 볼 수 없으므로, 본 논문에서는 C4.5에서 트리 절삭 용 등으로 사용한 염쇄적 에러율(pessimistic error rate) perror을 사용하였다. 염쇄적 에러율은 훈련 에러율보다 크며, 훈련에 참가하지 않은 테스트 데이터에 좀더 적응하는 성질을 가지고 있다[15,16]. 〈표 1〉에는 정확한 데이터와 오류 데이터의 수가 각각 c 개, e 개인 경우의 두 에러의 표가 예시되어 있다. 염쇄적 에러율은 결국 훈련에 참가한 데이터가 매우 많

을 경우에만 훈련에러율에 접근하며, 데이터의 수가 적은 경우에는 매우 염세적으로 미지의 테스트 데이터에 대한 에러율을 예측하는 기법으로, 결국 훈련 에러율이 아닌 염세적 에러율을 이용하여, 규칙의 한 속성을 확장할 것인가를 결정하는 방법이 R-EXPAND 기법이다. 본 논문에서 사용된 염세적 에러율은 이항분포의 누적확률을 25%의 신뢰도로 이용하는 방법이 적용되었으며, 신뢰도의 변화는 비교적 큰 변화를 주지 못하는 것이 알려져 있다. 중요한 점은 추가적인 데이터 없이 훈련 데이터 만을 의존하여, 미지의 테스트 데이터의 에러율을 예측하는데 이 염세적 에러율을 사용할 수 있다는 점이다. 훈련 데이터의 수가 적을수록 훈련 에러율은 신뢰할 수 없으며, 이를 반영한 정도로 염세적 에러율이 증가하게 된다. 이와 같은 새로운 에러율을 적용한 속성 제거(즉 규칙의 확장)를 이용하여 사용된 속성의 감소 또는 규칙의 감소 등의 바람직한 결과가 발생하게된다. 본 방법의 신뢰성에 대한 정확한 검증, 즉 데이터의 수가 충분하지 않을 경우 증가되는 에러율이 신뢰도의 감소를 정확하게 반영하는가의 문제는 매우 어려운 문제이며, 본 논문의 범위를 벗어나는 통계학적인 연구 대상이다.

```
procedure R-EXPAND(F)
{
 repeat
  C0, E0, C, E의 각 성분을 0으로 초기화 ;
  for each d in FF  /* FF는 원래의 데이터
                       들의 집합 */
   for each f in F  { /* 각 규칙 f에 대해 */
    na : = d가 f의 조건 중 만족하지 못하는
           속성의 수 :
    if na = 0 then
      if d와 f가 동일 클래스에 소속되면,
      then C0(f)++ :
      else E0(f)++
    else if na = 1 then {
     v : = d와 f의 좌변에서 다른 유일한 속성:
     if d와 f가 동일한 클래스에 소속하면,
     then C(f, v)++ :
     else E(f, v)++ :
    }
  flag : = false: /* false로 초기화 */
  } /* end of for each f ... */
  for each f in F { /* 각 규칙 f에 대해 확장
                       할 속성 v를 찾는다 */
   choose a variable v such that
      perror(C0(f)+C(f, v), E0(f)+E(f, v)) is
      maximum over all variables, and greater
      than perror(C0(f), E0(f)):
   if 위의 조건을 만족하는 속성 v가 존재,
   then { 규칙 f 의 속성 v를 확장한다:
          flag : = true:}
 until flag = false: /* 어떤 규칙도 확장할 속
      성을 발견하지 못할 때까지 계속함 */
}
```

(그림 5) 규칙 확장
(Fig. 5) Rule Expansion

〈표 1〉 염세적 에러율과 훈련 에러율의 비교
〈Table 1〉 Comparisons of Pessimistic and training error rates

정확한 데이터 수	1	10	100	1000	8	80	800
오류 데이터 수	0	0	0	0	2	20	200
훈련 에러율	0	0	0	0	0.2	0.2	0.2
염세적 에러율	0.75	0.129	0.014	0.001	0.355	0.234	0.209

2.5 default 규칙의 생성

규칙은 축소와 확장을 거치므로 최종 규칙들이 모든 훈련 데이터 또는 테스트 데이터를 포함한다는 보장은 없다. 이러한 데이터들에 대한 클래스 판단을 위해 default 규칙이 필요하다. 여기서는 포함되지 않은 훈련 데이터가 있는 경우 그 들 중 가장 많은 데이터가 소속된 클래스를 default로 하며, 만약 그와 같은 데이터가 없으면, 원래의 훈련 데이터 중에서 가장 많은 데이터가 소속된 클래스를 default로 한다. 실제, 이 규칙의 적용여부는 상황에 따라 달라질 수 있는데, 예를 들어 독버섯 여부가 확실치 않은 경우는 일단 독버섯으로 판단하는 것이 안전할 수도 있기 때문이다. 이러한 영역에 따른 문제는 여기서 고려하지 않았다.

3. 실험 및 비교분석

본 실험을 위하여 데이터의 개수가 수 천에서 수 만에 이르는 4 종의 실 세계 데이터(real world data)를 준비하였다. 〈표 2〉에는 각 영역 별 클래스 수, 속성 당 속성값의 평균수, 훈련 데이터 및 테스트 데이터

의 수가 명시되어 있다. connect-4 는 6 x 7 크기의 판에서의 합법적인 위치여부를 결정하는 문제이고, 두 가지의 체스-종료 게임과, 버섯 영역이 실험 대상인데, 데이터의 수가 충분한 객관적인 학습 기법의 확인을 위해 사용되었다.

〈표 2〉 실험에 사용된 영역의 특성
〈Table 2〉 Characteristics of experimental domains

영역	클래스 수	속성 수	속성당 속성값의 수	훈련 데이터의 수	테스트 데이터의 수
connect-4	3	42	3.0	13511	54046
kr-vs-kp	2	36	2.03	2338	584
krkopt	18	6	8.0	22445	5611
mushroom	2	22	5.68	2822	2822

비교대상으로는 최근 10여년 동안 많은 연구가 진행되어 매우 좋은 시스템으로 자리 잡은 결정 트리 기법을 기반으로 하는 C4.5 시스템으로 결정 트리를 생성한 후 이를 규칙으로 변환하는 C4.5rules 시스템 [Quinlan 93]을 사용하였다. 본 논문에서 제안한 접근 방식으로도, 기존의 오랜 연구를 거친 결정 트리 기반 시스템에 못지 않은 결과를 얻을 수 있음을 보였다. 〈표 3〉과 〈표 4〉에는 훈련 데이터와 테스트 데이터에 대한 인식률이 비교되어 있다. 표에서 random으로 표기된 방법은 영역당 클래스의 수를 고려하여, 임의의 클래스를 답으로하는 경우 즉 straw man 시스템의 최저 인식률을 보여주기 위해 참고로 사용하였다. 훈련 데이터의 인식률 보다는 〈표 4〉의 테스트 데이터에 대한 인식률이 매우 중요하며, 중간 단계인 C4.5의 출력보다는 같은 규칙의 형태를 출력하는 C4.5rules 시스템과의 비교가 의미를 갖는다. 표를 보면, 본 시스템의 인식률이 손색 없음을 알 수 있으며, C4.5rules가 결국 실패한 krkopt의 경우도 매우 높은 인식률의 규칙을 생성하였다.

〈표 3〉 훈련 데이터에 대한 인식률
〈Table 3〉 Recognition rates for training data

영역 \ 기법	random	C4.5 (절삭후 트리)	C4.5rules	본 시스템
connect-4	33.3%	85.0%	81.9%	95.16%
kr-vs-kp	50.0%	99.7%	99.8%	99.36%
krkopt	5.6%	66.6%	fail	94.15%
mushroom	50.0%	100.0%	100.0%	100.0%

〈표 4〉 테스트 데이터에 대한 인식률
〈Table 4〉 Recognition rates for test data

영역 \ 기법	random	C4.5 (절삭후 트리)	C4.5rules	본 시스템
connect-4	33.3%	76.3%	75.5%	75.32%
kr-vs-kp	50.0%	99.1%	99.1%	98.29%
krkopt	5.6%	54.5%	fail	78.45%
mushroom	50.0%	100.0%	100.0%	99.93%

〈표 5〉 생성된 규칙의 수효
〈Table 5〉 Number of generated rules

영역 \ 규칙수 (노드수)	C4.5 (절삭후 트리)	C4.5rules	본 시스템
connect-4	1040(1564)	307	963
kr-vs-kp	25 (47)	22	23
krkopt	4321(4945)	fail	2485
mushroom	24 (30)	12	4

〈표 6〉 학습 시간 [유닉스 서버 Axil 420에서 측정]
〈Table 6〉 Training time [measured on Unix server Axil 420]

영역 \ 사용자 시간 (초)	C4.5+C4.5rules	본 시스템
connect-4	19.4 + 13196.8	56098.6
kr-vs-kp	0.6 + 5.7	332.4
krkopt	fail after 1420	3175.0
mushroom	0.6 + 4.9	387.4

〈표 5〉와 〈표 6〉에는 생성된 규칙의 수효와 학습 시간이 참고로 비교되어 있다. 버섯 데이터의 경우 규칙의 수가 현격히 감소되었는데, 그 이유로는 병렬확장을 통해 생성하는 규칙은

If (v1 in {v11, v12,...}) and (v2 in {v21, v22,...}) and .... then class i

의 형태로, C4.5rules의 규칙 형태인

If (v1 = v11) and (v2 = v21) and ..., then class i

보다 일반적인 형태를 취할 수 있기 때문으로 풀이 된다. 즉 규칙의 조건에 사용된 각 속성값이 수 개의 속성값 중 선택할 수 있는데, 이는 C4.5rules의 경우와

같이 속성당 모든 속성값이 허용되거나(즉 속성이 생략되거나), 1 개의 속성값 만을 사용하는 경우보다 일반적인 규칙이 가능하기 때문이다. 따라서, 실험에 이용된 버섯 영역에서와 같이 인식에 필요한 규칙의 수가 많지 않을 때에는 큰 효과를 볼 수 있음을 이해할 수 있다. 다만 영역이 매우 복잡하여 필요한 규칙이 매우 많을 경우, 영역의 분할을 통한 규칙 생성 등을 추후 연구에서 검토해 볼 수 있다. 잡음이 많은 환경에 대한 본 시스템의 성능 평가는 [17]을 참고하면 된다.

4. 결 론

본 논문에서는, 규칙 생성과는 다른 논리 최적화 기법의 병렬 확장 기법을 이용하여, 그 차이점에 의한 제약을 극복하고 규칙 생성 시스템으로 활용하기 위한 방식을 제시하고, 그 방법을 구현하고 기존의 시스템과 비교하였다. 실험 대상으로는 데이터의 수가 수 천에서 수 만에 이르는 실세계 영역들을 이용하였다. 실험 결과 십수년 간에 걸쳐 연구된 결정트리 기법에 기반한 규칙 생성 기법 들에 못지 않은 성능을 보였다. 앞으로 기본 병렬 확장 기법 외의 규칙 축소나 확장 기법 등을 변경 또는 보강하거나, 주어진 입력 데이터의 분할을 통한 규칙 생성 후 이를 효율적 합성하는 등의 기법으로 학습 시간을 단축하고, 인식률이 높은 규칙 생성 기법으로 발전시킬 수 있을 것으로 기대된다. 또한 연속 값을 갖는 속성 등의 처리를 위한 처리 기법 등과, 유용한 속성의 선택 기법 등을 보강하여 좀더 실용적인 시스템으로 발전될 수 있다.

참 고 문 헌

[1] M.S. Chen, J. Han, and P.S. Yu, "Data Mining : An Overview from Database Perspective," IEEE Trans. Knowledge and Data Engineering, 8(6), pp.866-883, Dec. 1996.

[2] U. Fayyad, G.P. Shapiro, and P. Smyth, "The KDD Process for Extracting Useful Knowledge from Volumes of Data," Communications of the ACM, 39(11), pp.27-34, Nov. 1996.

[3] T. Imielinski and H. Mannila, "A Database Perspective on Knowledge Discovery," Communications of the ACM, 39(11), pp.58-64, Nov. 1996.

[4] W.H. Inmon, "The Data Warehouse and Data Mining," Communications of the ACM, 39(11), pp.49-50, Nov. 1996.

[5] T.M. Mitchell, Version Spaces: An Approach to Concept Learning, Ph.D thesis, Stanford University, 1978.

[6] J.R. Quinlan, "Learning efficient classificat -ion procedures," In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell(eds.), Machine Learning : An Artificial Intelligence Appro -ach, Tioga Press, 1983.

[7] J.R. Quinlan, C4.5 : Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.

[8] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representation by error propagation," In Parallel Distributed Processing : Explorations in the Microstruct -ures of Cognition, Vol. 1, MIT Press, Cam -bridge, MA, 1986.

[9] R.S. Michalski, "Pattern recognition as rul -eguided inductive inference," IEEE Trans- actions on Pattern Analysis and Machine Intelligence, PAMI-2(4), pp.349-361, 1980.

[10] 김동완, 최재환, 이기철, 변영태, "귀납적 기계 학습 기법들의 구현 및 비교," 인공지능 연구회 춘계 학술발표 논문집, pp.13-18, 정보과학회, April 1992.

[11] 김동완, 수정된 R-MINI에 근거한 적응성 있는 학습 시스템, 석사학위 논문, 홍익대, Feb. 1993.

[12] 박용국, 실용적인 규칙 획득 시스템 HI-ARM, 석사학위 논문, 홍익대, Feb. 1994.

[13] Brayton, R.K., Hachtel, G.D., McMullen, C.T., and Sangiovanni-Vincentelli, A.L., Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Publishers, 1984.

[14] 김진봉, 이기철, 변환된 ESPRESSO에 기반을 둔 규칙 획득 도구, 정보과학회 학술 발표 논문집,

22(2), pp.417-420, 1995.
[15] J.R. Quinlan, "Simplifying Decision Trees," International Journal of Man-Machine Studies, 27, pp.221-234, 1987.
[16] Documentica Geigy Scientific Tables, 6th edition, p.185.
[17] K.C. Lee, "잡음섞인 다용량 데이터에서의 정보추출 기법," 과학기술연구 논문집, 홍익대학교, Dec. 1997.

이 기 철

1977년 서울대학교 공과대학 전자공학과 졸업
1979년 한국과학원 전산학과 석사학위 취득
1987년 미국 위스콘신대(매디슨 소재) 전기 및 컴퓨터공학과 박사학위 취득
1996년 1월~1997년 1월 미국 위스콘신대(매디슨 소재)방문교수
1989년 3월~현재 홍익대학교 공과대학 컴퓨터공학과 부교수
관심분야 : 기계학습, 데이터 마이닝, 신호처리, 시스템 프로그래밍

김 진 봉

1990년 홍익대학교 컴퓨터공학과 졸업
1992년 홍익대학교 컴퓨터공학과 석사학위 취득
현재 홍익대학교 컴퓨터공학과 박사과정
1998년 3월~현재 안산공전 전임강사
관심분야 : 기계학습 및 인공지능