Journal of Korean
Data & Information Science Society
2004, Vol. 15, No. 4, pp. 803~816

CHAID Algorithm by Cube-based Proportional Sampling

Hee-Chang Park[1] · Kwang-Hyun Cho[2]

Abstract

The decision tree approach is most useful in classification problems and to divide the search space into rectangular regions. Decision tree algorithms are used extensively for data mining in many domains such as retail target marketing, fraud dection, data reduction and variable screening, category merging, etc. CHAID uses the chi-squired statistic to determine splitting and is an exploratory method used to study the relationship between a dependent variable and a series of predictor variables. In this paper we propose CHAID algorithm by cube-based proportional sampling and explore CHAID algorithm in view of accuracy and speed by the number of variables.

Keywords : CHAID, data mining, decision trees, proportional sampling

1. 서론

데이터 마이닝이란 대량의 데이터로부터 새롭고 의미 있는 정보를 추출해 의사결정에 활용하는 작업이라고 정의된다. 의사결정나무는 데이터 마이닝에서 사용하는 여러 기법들 중의 하나이며, 의사결정 규칙을 도표화하여 관심대상이 되는 집단을 몇 개의 소집단으로 분류하거나 예측을 수행하는 분석방법으로 다른 분석방법에 비해 연구자가 분석과정을 쉽게 이해하고 설명할 수 있다는 장점이 있다. 이러한 의사결정나무 알고리즘은 시장세분화, 고객세분화 등의 세분화 문제, 고객을 신용도에 따라 우량/불량으로 분류하는 분류 문제, 고객속성에 따라 대출한도액을 예측하는 예측, 차원축소 및 변수선택, 범주의 병합 또는 연속형 변수의 이산화 등의 분야에서 유용하게 활용되고 있다.

---

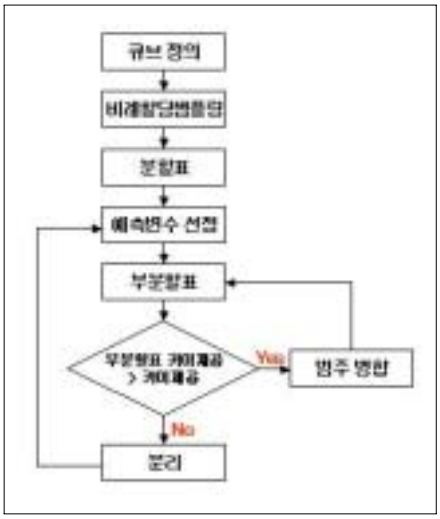
1) First Author : Professor, Department of Statistics, Changwon National University, Changwon, Gyeongnam, 641-773, Korea
E-mail : hcpark@sarim.changwon.ac.kr
2) Graduate Student, Department of Statistics, Changwon National University, Changwon, Gyeongnam, 641-773, Korea

그동안의 연구를 살펴보면 의사결정나무분석을 수행하기 위한 다양한 분리기준, 정지규칙, 가지치기 방법들이 제안되었으며, 이들을 어떻게 결합하느냐에 따라서 서로 다른 의사결정나무가 형성된다. 또한 정확하고 빠르게 의사결정나무를 형성하기 위해 다양한 알고리즘이 제안되고 있다. 의사결정나무 알고리즘에는 Hartigan(1975)이 제안한 CHAID, Breiman(1984)이 제안한 CART, Quinlan(1993)에 의해 제안된 C4.5, 그리고 Lon와 Shin(1997)이 제안한 QUEST 알고리즘 등이 있다. 이들 중에서 Hartigan에 의하여 처음 소개된 CHAID는 카이제곱-검정 또는 F-검정을 이용하여 다지분리를 수행함으로 데이터를 빠르고 효율적으로 탐색하는 의사결정나무의 기본적인 알고리즘이라고 할 수 있다. CHAID는 목표변수가 이산형일 경우에는 카이제곱검정을 이용하는데 독립변수가 목표변수와 유의한 차이가 있는지를 조사하여 목표변수에 가장 유의한 독립변수를 찾아 그 변수를 기준으로 나무를 형성해 나가며, 각 마디에서 정지규칙중의 하나가 만족될 때까지 이 과정을 반복한다. 연속형 목표변수에 대한 분리기준으로는 ANOVA의 F값을 사용한다.

거대한 양의 데이터베이스에 대해 CHAID 알고리즘을 이용하여 모형을 구축하기 위해서는 시간과 노력이 많이 소비된다. 본 논문에서는 방대한 데이터베이스에 대하여 큐브 기반 비례할당 샘플링을 CHAID 알고리즘에 적용시켜 기존의 CHAID 알고리즘의 나무 모형과 동일하면서 모형구축 시간을 단축시키는 알고리즘을 제시하여 그 결과를 탐색하고자 한다. 2절에서는 큐브 기반 표본에 의한 CHAID 알고리즘을 제시하고, 3절에서는 예제 및 실험을 통하여 수행결과를 탐색하며, 4절에서 결론을 맺고자 한다.

2. 큐브 기반 비례할당 샘플링에 의한 CHAID 알고리즘

전형적인 방법의 CHAID 알고리즘은 거대한 데이터베이스를 대상으로 의사 결정나무를 형성해나가기 까지 매우 많은 계산과정을 거치게 되므로 상당한 처리시간이 요구된다. 이러한 문제점을 보완하기 위해 이 절에서는 큐브 기반 비례할당 샘플링를 고려한 CHAID 알고리즘을 제안하고자 한다. 큐브를 기반 비례할당 샘플링으로 추출된 표본을 사용한 CHAID 알고리즘은 다음과 같다.



<그림 1> 큐브기반 비례할당 샘플링에 의한 CHAID 수행단계

각 단계에 대한 설명을 요약하면 다음과 같다.

[제 1 단계] 큐브의 정의

큐브 기반 비례할당 샘플링을 수행하기 위하여 각 데이터 별로 큐브를 정의한다. 큐브는 예측변수와 그 범주의 수만큼 생성을 한다.

```
<cfquery name="serch" DataSource="#DB#">
    Select #Prediction_var#
    From decision
</cfquery>
<cfloop query="serch">
  <cfset num = #Cube_lavel#*#Prediction_var#>
  <cfquery name="insert" DataSource="#DB#">
     <cfinclude template = "Cube_create.cfm">
  </cfquery>
</cfloop>
```

[제 2 단계] 샘플링

비례할당 표본비율을 결정하고 각 큐브의 데이터 크기를 계산한다. 각 큐브별 크기에 따라 비례할당 표본비율을 적용하여 데이터를 샘플링한다. 비례할당 샘플링 된 데이터는 의사결정나무 형성에 필요한 데이터베이스로 재구성되며, 결측치나 불량 데이터는 이 단계에서 제거된다. 알고리즘은 다음과 같다.

```
<cfquery name="distinct" DataSource="#DB#">
  Select distinct(lavel) as number
  From #Data_Base#
</cfquery>
<cfloop query="distinct">
  <cfset proportion_num = Round(#proportion_number#)>
  <cfquery name="serch" DataSource="#DB#">
     Select #Prediction_var#
     From #Data_Base#
     Where lavel = #number#
     order by RAND() LIMIT #proportion_num#
  </cfquery>
  <cfif #Prediction_var# neq null or 0 or #Prediction_num#>
    <cfinclude template = "reject_data.cfm">
  </cfif>
</cfloop>
```

[제 3 단계] 분할표의 작성

재구축된 데이터베이스에서 각 예측변수에 대해 목표변수의 범주를 수준으로 하는 분할표를 만든다. 각 예측변수에 대해 카이제곱 통계량을 구하고, 이를 카이제곱 통계량 DB에 저장한다. 저장된 카이제곱 통계량은 예측변수의 선정에 사용된다. 이에 대한 알고리즘은 다음과 같다.

```
<cfinclude template = "delete_partition.cfm">
<cfloop index = i from = 1 to = #ArrayLen(prediction_var)#>
  <cfset partition = ArrayNew(3)>
  <cfloop query=#size#>
    <cfloop index = j from = 1 to = #target_size#>
      <cfinclude template = "create_partition.cfm">
    </cfloop>
  </cfloop>
  <cfset chi_value = Σ(n_ij − υ_ij)^2 / υ_ij >
   <cfset df = 2 * (#size.RecordCount#-1)>
  <cfif #chi_value# gt #Evaluate("chi_statistic[#df#]")#>
    <cfinclude template = "insert_Qvalue.cfm">
  </cfif>
</cfloop>
```

[제 4 단계] 예측변수의 선정

카이제곱 통계량 데이터베이스에서 통계량이 가장 큰 예측변수를 선정한다. 이 때 트리의 분리가 된 예측변수는 카이제곱 통계량 데이터베이스에서 제거된다. 선정된 예측변수에 대하여 범주의 병합을 시도한다. 예측변수 범주의 각 쌍과 목표변수의 범주로 구성된 부분할표에서 유의한 정도가 가장 약한 쌍의 유의확률이 주어진 임계값보다 큰 경우 이들 두 쌍을 하나의 범주로 병합하는 과정을 반복한다. 이에 대한 알고리즘은 다음과 같다.

```
<cfquery name = "tree" DataSource="#DB#">
  select node
  from #partition#
  order by s_value desc
</cfquery>
<cfloop query = "tree">
  <cfinclude template = "sub_partition.cfm">
</cfloop>
```

[제 5 단계] 부분할표

선택된 예측변수에 대하여 예측변수의 범주 각 쌍과 목표변수의 범주로 구성된 새로운 부분할표를 작성한다. 새로 구성된 부분할표에서 구해진 통계량이 카이제곱 통계량보다 큰 경우 이들 두 쌍을 하나의 범주로 병합하는 과정을 반복한다. 알고리즘은 다음과 같다.

```
<cfloop index = "n" from = 1 to = #sub_size#>
  <cfset partition_sub = ArrayNew(2)>
  <cfset category = ArrayNew(1)>
  <cfinclude template = "create_node.cfm">
  <cfinclude template = "sub_partition.cfm">
  <cfloop query=#size#>
    <cfloop index = j from = 1 to = #target_size#>
      <cfinclude template = "create_sub_partition.cfm">
    </cfloop>
  </cfloop>
  <cfset chi_value = Σ(n_ij − υ_ij)^2 / υ_ij >
  <cfset df = 2 * (#size.RecordCount#-1)>
 <cfinclude template = "Annexation.cfm">
</cfloop>
```

[제 6단계] 범주 병합 및 분리

각 예측변수의 부분할표에서 얻어진 범주들의 병합여부를 판정한다. 부분할표에서 새로 구해진 통계량이 카이제곱 통계량보다 큰 경우 그 범주를 병합하고 [단계 5]로 돌아가 새로운 부분할표를 작성한다. 부분할표에서 새로 구해진 통계량이 카이제곱 통계량보다 작은 경우 예측변수를 범주에 따라 노드를 분리한다. 이때 분리가 된 예측변수는 카이제곱 통계량 데이터베이스에서 제거된다. 예측변수를 분할한 후 [단계 4]로 돌아가 아직 분리되지 않은 예측변수에 대하여 [단계 5], [단계 6]의 과정을 반복하여 노드를 분리한다. 이에 대한 알고리즘은 다음과 같다.

```
<cfif #chi_value# gt #Evaluate("chi_statistic[#df#]")#>
  <cfset index_2 = #index_1#-1>
  <cfloop index = "in" from = 1 to = #index_2#>
    <cfif #q_value_sub[1][in]# eq #ArrayMax(q_value_sub[1])#>
       <cfset array_list = "#q_value_sub[2][in]#">
    </cfif>
  </cfloop>
  <cfset category = ArrayNew(1)>
  <cfset new_category = ArrayNew(1)>
    <cfloop index = "id" from = 1 to = #listLen(array_list,'/')#>
       <cfset category[id] = #listGetAt(array_list,id,'/')#>
       <cfset new_category[id] = #listGetAt(array_list,id,'/')#>
    </cfloop>
<cfelse>
   <cfbreak>
</cfif>
```

3. 예제 및 실험

본 절에서는 본 논문에서 구현한 알고리즘을 바탕으로 큐브기반 비례할당 샘플링에 따른 수행속도와 정확도를 탐색하기 위하여 실험한 결과를 논의한다. 본 실험을 위해 CPU는 Intel Pentium4-1.8GHz Northwood을 사용하였으며, 운영체제는 Linux 7.1, 프로그래밍 언어는 coldfusion 5.0, 데이터베이스는 mysql 3.23.51을 이용하였다. 실험을 위해 사용된 데이터는 모 대학교 신입생 1383명을 대상으로 가정환경에 대해 조사한 실제 데이터로, 이들 중 본 연구에 사용된 변수는 다음과 같다.

목표변수	예측변수
1. 가정환경 : (1) 화목함 (2) 화목하지 않음	1. 성별 : (1) 남자 (2) 여자 2. 아버지학력 : (1) 대학이상 (2) 중, 고졸 (3) 초등학교 이하 3. 가정교육방식 : (1) 이해심 많음 (2) 보통 (3) 권위적 (4) 무관심 4. 생활부담 : (1) 부모 (2) 자신부담 (3) 기타

실제 데이터의 크기가 큐브기반 비례할당 샘플링에 의한 CHAID 알고리즘의 효율성 문제를 논의하기에는 부적절하다고 판단되어, 먼저 1383건의 데이터에 대하여 random으로 500개의 데이터를 20번 반복으로 추출하여 얻어진 100,000건의 데이터와 40회 반복 추출하여 얻어진 200,000건의 데이터를 이용하여 실험하였다. 이 실험에서 사용된 100,000건의 데이터에 대한 속성은 다음과 같다.

<표 1> 100,000건의 데이터에 대한 속성

1. 성별

	화목함	화목하지 않음
남	34602	17471
여	34706	13221

2.아버지학력

	화목함	화목하지 않음
대학이상	46932	23232
중,고졸	19384	5609
초등학교졸 이하	2992	1851

3. 가정교육방식

	화목함	화목하지 않음
이해심 많음	13621	14199
보통	45504	7984
권위적	403	442
무관심	9780	8067

4. 생활부담

	화목함	화목하지 않음
부모	60338	24980
자신부담	7397	4614
기타	1573	1098

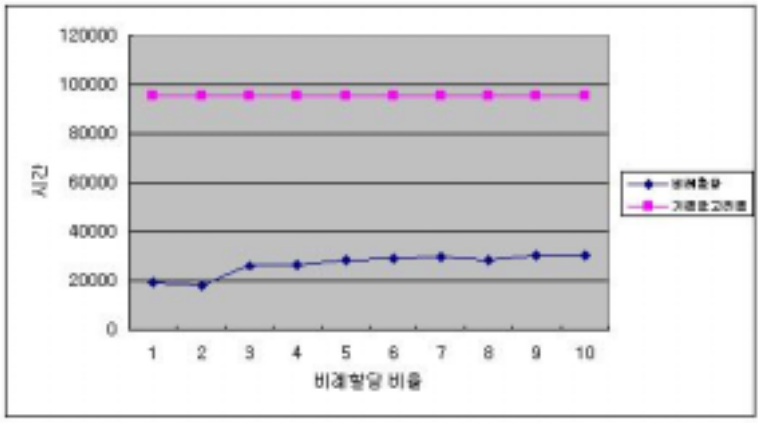
이들 데이터에 대해 큐브기반 비례할당 샘플링에 의한 CHAID 알고리즘을 수행한 결과는 <표 2>와 같다.

<표 2> 100,000건의 데이터에 대한 알고리즘 수행표(단위 : 밀리초)

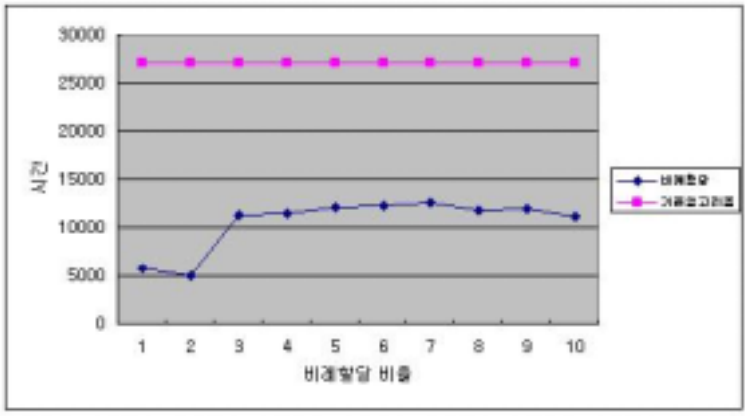
구분	표본의 크기	트리	전체시간	샘플링제외
기존알고리즘	100000	가정교육방식-아버지학력-생활부담-성별	95365	27052
비례할당:10%	10004	기존 알고리즘과 동일	30458	11161
비례할당: 9%	9001	중간노드 약간다름:3번째노드	30727	11983
비례할당: 8%	7999	중간노드 약간다름:2,3번째노드	28846	11771
비례할당: 7%	7000	중간노드 약간다름:3번째노드	30124	12605
비례할당: 6%	6001	중간노드 약간다름:3번째노드	28933	12205
비례할당: 5%	5001	중간노드 약간다름:3번째노드	28346	12070
비례할당: 4%	4001	중간노드 약간다름:2,3번째노드	26707	11353
비례할당: 3%	2996	가정교육방식-아버지학력-성별-생활부담	25810	11219
비례할당: 2%	2003	가정교육방식-아버지학력-성별	17612	4854
비례할당: 1%	1000	가정교육방식-아버지학력-생활부담	19331	5736

이 표에서 보는 바와 같이 기존의 CHAID 알고리즘 수행시간보다 큐브기반 비례할당 샘플링 알고리즘 수행시간이 현격히 줄어드는 것을 알 수 있다. 기존의 CHAID 알고리즘의 수행시간은 95.365초이고 비례할당 10%일 때 기존의 CHAID 알고리즘의 트리구조와 동일하면서 30,458초로 알고리즘 수행시간이 1/3로 줄어드는 것을 볼 수 있다. 그리고 비례할당 9%에서 비례할당 4%까지는 기존의 CHAID 알고리즘의 트리구조와 별 차이가 없으면서도 수행시간이 1/4 정도로 줄어드는 것을 볼 수 있다. 큐브기반 비례할당 샘플링을 이용한 CHAID 알고리즘이 기존의 CHAID 알고리즘 보다 수행시간을 단축시킬 수 있다고 볼 수 있다.

기존의 알고리즘과 100,000건 데이터에 대한 큐브기반 비례할당 샘플링에 의한 알고리즘의 수행시간을 그림으로 비교하면 <그림 2> 및 <그림 3>와 같다.



<그림 2> 전체수행시간 비교 그래프(100.000)



<그림 3> 데이터 구축을 제외한 수행시간 비교 그래프(100,000)

실험에서 사용된 200,000건의 데이터에 대한 속성은 다음과 같다.

<표 3> 200,000건의 데이터에 대한 속성

1. 성별

	화목함	화목하지않음
남	66732	28568
여	69714	34986

2. 아버지학력

	화목함	화목하지않음
대학이상	36188	11912
중,고졸	94991	47109
초등학교졸 이하	5337	4463

3. 가정교육방식

	화목함	화목하지않음
이해심 많음	19902	16498
보통	30625	28275
권위적	84960	18340
무관심	92	48

4. 생활부담

	화목함	화목하지않음
부모	117421	50779
자신부담	16008	10792
기타	3099	1901

<표 4>에서는 200,000건 데이터에 대한 기존 알고리즘과 큐브기반 비례할당 샘플링

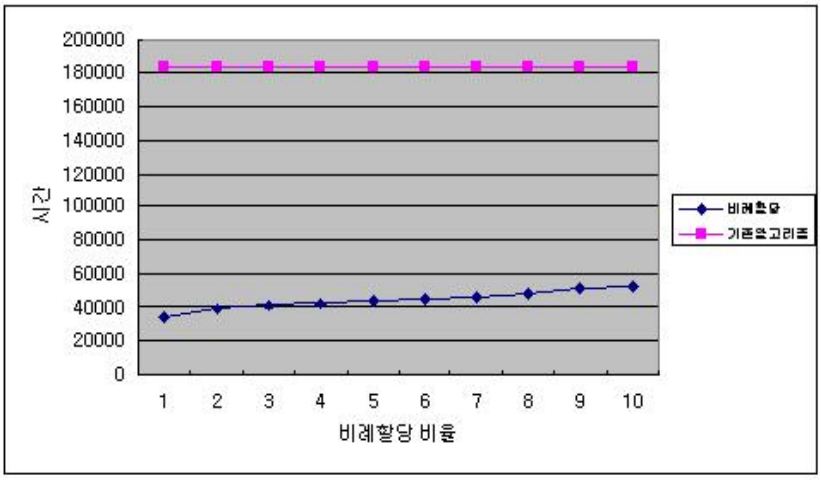
에 의한 CHAID 알고리즘에 대해 수행결과를 나타내었다.

<표 4> 200,000건의 데이터에 대한 알고리즘 수행표(단위 : 밀리초)

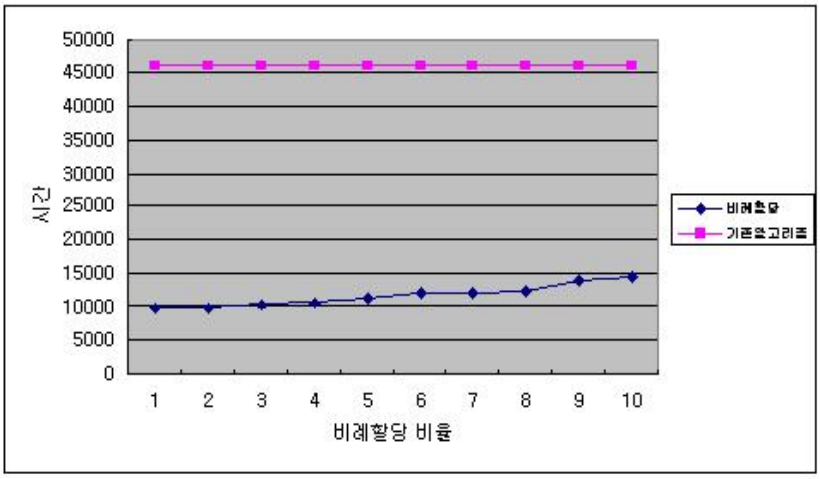
구분	표본의 크기	트리	전체시간	샘플링제외
기존 알고리즘	200000	가정교육방식-아버지학력-생활부담-성별	183398	45922
비례할당:10%	20003	기존 알고리즘과 동일	52573	14422
비례할당: 9%	18000	기존 알고리즘과 전체데이터와 동일	52299	13951
비례할당: 8%	16001	기존 알고리즘과 전체데이터와 동일	48335	12509
비례할당: 7%	13999	중간노드 약간다름:3번째노드	46847	12272
비례할당: 6%	11999	중간노드 약간다름:3번째노드	45694	12053
비례할당: 5%	9999	중간노드 약간다름:3번째노드	44654	11482
비례할당: 4%	8004	중간노드 약간다름:3번째노드	42684	10571
비례할당: 3%	6000	중간노드 약간다름:3번째노드	41015	10442
비례할당: 2%	3998	중간노드 약간다름:3번째노드	39578	9853
비례할당: 1%	2003	중간노드 약간다름:2,3번째노드	33658	9688

<표 4>에서 보는 바와 같이 100,000건의 데이터에 대한 큐브기반 비례할당 샘플링 알고리즘보다 트리의 정확도가 증가한 것을 알 수 있다. <표 2>에서는 비례할당 10%일 때가 기존 CHAID 알고리즘과 트리의 형태가 동일하였는데, <표 4>에서는 비례할당 8%일 때까지 기존 CHAID 알고리즘과 트리의 형태가 동일하고 수행시간도 약 1/4 정도로 줄어든 것을 알 수 있다. 이것은 데이터의 크기가 클수록 비례할당 샘플 크기가 적어도 트리의 정확도는 높다는 것을 의미한다.

기존의 알고리즘과 200,000건 데이터에 대한 큐브기반 비례할당 샘플링에 의한 알고리즘의 수행시간을 그림으로 비교하면 <그림 4> 및 <그림 5>와 같다.



<그림 4> 전체수행시간 비교 그래프(200.000)



<그림 5> 데이터 구축을 제외한 수행시간 비교 그래프(200,000)

지금부터는 1383명을 대상으로 가정환경에 대하여 조사한 실제 데이터를 이용하여 얻어진 결과를 설명하고자 한다. 이들 데이터에 대한 속성은 다음과 같다.

<표 5> 실제 데이터에 대한 속성

1. 성별

	화목함	화목하지않음
남	476	238
여	484	185

2. 아버지학력

	화목함	화목하지않음
대학이상	270	77
중,고졸	41	25
초등학교졸 이하	649	321

3. 가정교육방식

	화목함	화목하지않음
이해심 많음	133	111
보통	634	111
권위적	188	195
무관심	5	6

4. 생활부담

	화목함	화목하지않음
부모	836	342
자신부담	103	64
기타	21	17

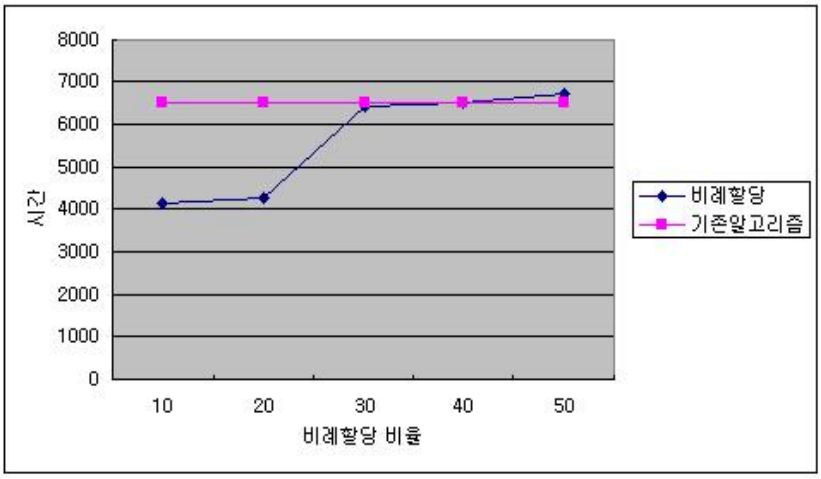
다음의 <표 6>은 실제 데이터를 이용하여 기존의 CHAID 알고리즘과 샘플링 기법을 사용한 알고리즘과의 수행속도를 비교한 표이다.

<표 6> CHAID 알고리즘 수행표(단위 : 밀리초)

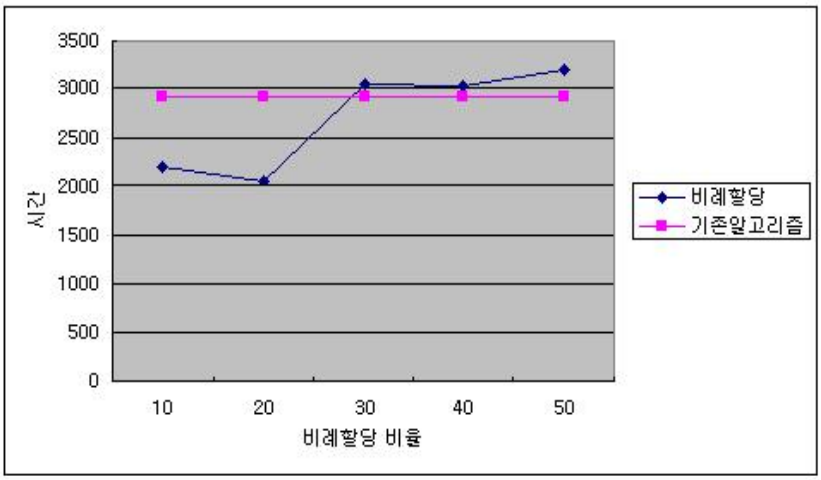
구분	·표본의 크기	트리	전체시간	샘플링제외
기존 알고리즘	1383	가정교육방식-아버지학력	6492	2925
비례할당:50%	760	전체데이터와 동일	6693	3186
비례할당:40%	608	전체데이터와 동일	6493	3021
비례할당:30%	417	중간노드 약간다름:2번째노드	6407	3038
비례할당:20%	377	중간노드 약간다름:2번째노드	4257	2061
비례할당:10%	277	가정교육방식	4117	2212

2절에서 제시한 큐브기반 비례할당 샘플링에 의한 CHAID 수행시간이 기존의 CHAID 알고리즘 수행시간보다 현격히 줄어들어야 한다. 하지만 <표6>에서 보는 바와 같이 비율이 30%이상인 경우에는 기존의 CHAID 알고리즘 수행시간과 큐브기반 비례할당 샘플링에 의한 CHAID 알고리즘 수행시간에는 별 차이가 없다. 이 같은 문제점이 발생한 이유는 예제 데이터의 크기가 매우 작기 때문이라고 생각한다. 예제 데이터의 크기가 작아서 알고리즘 수행 시간이 실제 알고리즘 계산시간 보다 하드웨어적인 요소에 더욱 민감하게 되었다.

기존의 알고리즘과 예제 데이터에 대한 큐브기반 비례할당 샘플링에 의한 알고리즘의 수행시간을 그림으로 비교하면 <그림 6> 및 <그림 7>와 같다.



<그림 6> 전체수행시간 비교 그래프



<그림 7> 데이터 구축을 제외한 수행시간 비교 그래프

4. 결론

본 논문에서는 의사결정나무 알고리즘 중에서 다지 분리를 수행함으로써 데이터베이스 효율적으로 탐색하는 CHAID 기법에 큐브기반 비례할당 샘플링을 이용한 알고리즘을 제시하고, 이에 대한 정확도와 수행속도를 탐색하였다. 대용량 데이터베이스에 대하여 CHAID 알고리즘은 알고리즘 수행에 있어 많이 시간을 필요로 한다. CHAID 알고리즘에 일반적인 샘플링 알고리즘을 사용하면 수행시간을 단축시킬 수 있다. 하지만, 일반적인 샘플링으로 생성된 트리구조가 기존의 트리구조와 달라지는 문제점이 발생한다. 본 논문에서 제시한 큐브기반 비례할당 샘플링에 의한 CHAID 알고리즘은 기존의 알고리즘보다 수행속도 면에서 효과적인 것을 알 수 있었으며, 트리 구조면에서도 비례할당 10%이상이면 기존의 트리구조와 큰 차이를 보이지 않고 있다. 향후 CHAID 알고리즘뿐만 아니라 CART, C4.5, QUEST 등의 의사결정나무 알고리즘에 대하여 본 논문에서 제시한 큐브기반 비례할당 샘플링에 의한 기법을 적용한 알고리즘의 연구가 필요하다.

참고문헌

1. Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and regression trees*, Wadsworth, Belmont.
2. Hartigan, J.A. (1975), *Clustering Algorithms*, New York: John Wiley & Sons, Inc.
3. Loh, W.Y and Shin, Y,S(1997). *Split Selection Methods for Classification Tree*, Statistica Sinica. 7, 815-840.

4. Quinlan, J.R. (1993), *C4.5 Programs for Machine Learning*. San Mateo, Morgan Kaufmann.

[2004년 8월 접수, 2004년 10월 채택]