데이터 마이닝을 위한 대용량 고차원 데이터의 셀-기반 분류방법

진두석° 장재우
전북대학교 컴퓨터공학과
{dsjin, jwchang}@dblab.chonbuk.ac.kr

Cell-based Classification of High-dimensional Large data for Data Mining Application

Du-Seok Jin° Jae-Woo Chang
Dept. of Computer Engineering, Chonbuk National University

요 약

최근 데이터 마이닝에서 대용량 데이터를 처리하는 응용이 많아짐에 따라, 클러스터링(Clustering) 및 분류(Classification)방법이 중요한 분야가 되고 있다. 특히 분류방법에 관한 기존 연구들은 단지 메모리 상주(memory-resident) 데이터에 대해 한정되며 고차원 데이터를 효율적으로 처리할 수 없다. 따라서 본 논문에서는 대용량 고차원 데이터를 효과적으로 처리할 수 있는 새로운 분류 알고리즘을 제안한다. 이는 데이터들을 차원 공간상의 셀(cell)로 표현함으로써 수치(numerical) 애트리뷰트 와 범주(categorical) 애트리뷰트 모두 처리할 수 있는 알고리즘을 제안한다. 아울러, 실험결과를 통해 제안한 알고리즘이 데이터의 양,차원 그리고 속성에 관계없이 분류를 효과적으로 수행함을 보인다.

1. 서론

데이터 마이닝은 대용량의 데이터베이스에서 숨겨진 지식, 패턴, 연관된 규칙 등을 발견하는 방법이다. 데이터들을 분석하여 서로 유사한 그룹으로 데이터를 분류하는 방법은 데이터 마이닝의 중요한 분야 중 하나이다. 입력 데이터, 트레이닝 셋(Training Set)은 다수의 애트리뷰트를 가진 대규모의 레코드 셋이다. 분류는 입력 데이터를 분석하여 데이터에 나타난 특성에 맞는 모델을 설정하는 일이다. 분류(Classification)모델을 설정하기 위하여 첫째, 입력 데이터의 클러스터링(Clustering) 과정이 필요하다. 클러스터링 기법은 클러스터들 사이에 계층구조를 가지고 있는 계층적(hierarchical) 클러스터링과 데이터 공간을 정해진 수의 클러스터에 따라 분할하고 클러스터링하는 공간분할(space partitioning) 클러스터링 방법이 있다. 그리고 클러스터를 결정하는 기준은 거리(distance)를 사용하는 방식과 밀도(density)를 사용하는 방식으로 구분된다. 기존의 클러스터링 알고리즘은 저차원 데이터의 클러스터링에는 적합하지만 차원이 높아질수록 급격한 성능 저하를 보인다.그러나 점점 고차원 형태의 데이터가 증가하고 있기 때문에 고차원 데이터를 효율적으로 처리할 수 있는 분류방법이 요구되고 있다. 둘째, 분류 모델을 설정하기위한 기존의 분류 알고리즘은 단지 메모리 상주(memory-resident) 데이터에 대해 한정되어 있어 대용량의 데이터를 효과적으로 처리하지 못하고 있다. 분류모델은 Decision Tree[1], 통계적모델[2], 유전학적모델[3]이 지금까지 연구 되고있다. Decision Tree 는 입력데이터를 하나의 그룹에 속할 때까지 재귀적으로 분할하여 트리를 구성한다. 이때 효율적인 분할 인덱스(Splitting Index)에 대한 연구가 분류기법의 성능을 결정하는 중요한 역할을 한다. 따라서 본 논문에서는 ***gini*** index[4]에 기반한 셀-기반 분류모델에 적합한 분할 인덱스를 정의하여 대용량의 데이터를 처리하기에 적합한 분류 알고리즘을 제안한다. 또한 데이터의 속성값이 수치(numerical) 애트리뷰트 또는 범주(categorical) 애트리뷰트에 따라 기존의 연구에서는 추가적인 작업이 필요하지만 제안하는 방법에서는 별도의 처리과정이 필요하지 않다. 아울러, 본 논문에서는 데이터 공간을 각 차원별 구간으로 분할하고 각 분할구간의 밀도를 구하여 저장하는 방식으로 고차원데이터를 효율적으로 처리할 수 있는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 본 논문에서 제안하는 셀-기반 분류방법의 공간분할 및 분류결과를 저장하고 검색하는 방법에 대해 설명하고 제 3 장에서는 셀-기반 분류방법의 성능을 제시한다. 마지막으로 4 장에서는 결론 및 향후 연구과제를 제시한다.

2. 대용량 고차원의 데이터를 위한 셀-기반 분류방법

본 장에서는 대용량 고차원의 데이터를 효율적으로 클러스터링 및 분류하기 위한 셀-기반 분류방법을 제안한다. 이를 위해 공간상에서 분할 인덱스(splitting index)를 통해 각 차원의 구간을 정하고 각 구간에 따라 셀을 구성한다. 얻어진 셀의 밀도가 셀 임계값 이상인 셀을 저장하

고 검색하는 방법을 제안한다.

2.1 공간분할 및 분류된 결과 저장

공간상에 표현된 입력데이터를 해당되는 셀로 맵핑하기위해서 먼저 각 차원에 대해서 N 개의 구간으로 나눈다. 이때, 차원 축을 계속해서 2N 개로 구간을 나눌지를 분할 인덱스를 통해 결정한다. 트레이닝 데이터에 대해서 차원 축을 N 개로 분할한 경우 각각의 분할영역에서 C 개의 클래스에 대하여 상대적인 밀도 P_{ij}를 구하여 모두 합한 결과를 가지고 결정한다. 분할 인덱스는 식 1 로 구한다. 그림 1 은 입력 데이터가 2 차원인 경우의 분할과정을 설명하며, 그림 1 에서 실선으로 나타난 부분이 분할구간이 된다.

$$SplittingIndex = 1 - \sum_{i=1}^{N} \sum_{j=1}^{c} p_{ij}^2 \quad (식\ 1)$$

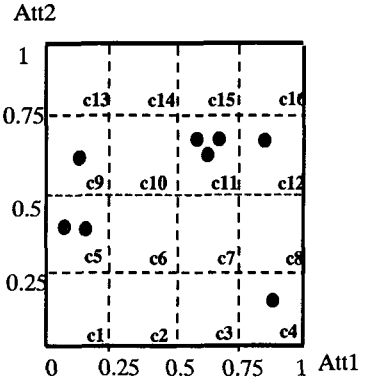

그림 1. 분할인덱스를 통한 공간분할

트레이닝 셋을 통해 분류된 결과를 저장하는 방법은 다음과 같다. 첫째, 모든 셀의 빈도수를 구하여 분류정보파일(Classification Information file)에 분류 결과 및 빈도수를 저장한다. 이 방법은 셀의 번호를 분류정보파일의 오프셋으로 사용하여 질의 처리 시 한번의 I/O 로 결과를 바로 얻을 수 있으나 셀의 수가 많은 경우 분류정보파일의 크기가 매우 큰 단점이 있다. 둘째, 모든 차원에 대한 각각의 구간별 빈도수를 구하여 구간 임계값 이상인 구간과 미만인 구간을 구분하여 근사정보파일(Approximation Information file)에 0 과 1 로서 저장하고, 분류정보파일에는 각 셀에 대해 빈도수 구하여 빈도수가 셀 임계값 이상인 셀의 번호 및 빈도수를 저장한다. 셀 임계값과 구간 임계값은 식 2 와 같다.

$$\text{간임계값 } (\lambda) = \begin{cases} \lambda = \dfrac{NR}{NI} \times F \\ NI : \text{입력데이터의수} \\ NR : \text{각차원별구간수} \\ F : \text{'1'로 받아들이는 구간내 최소빈도수} \end{cases} \quad (식\ 2)$$

셀임계값(τ) : 양의정수값

이 방법의 경우 분류정보파일 크기는 셀 전체를 저장하는 경우에 비해 평균 (1/10 ~ 1/100)정도에 해당한다. 그러나 질의 처리 시 근사정보파일과 분류정보파일을 순차 검색해야 하는 단점이 있다. 따라서 비교적 셀의 수가 적은 데이터는 셀을 모두 저장하는 방법이 우수하며, 셀의 수가 많은 데이터에는 각 셀에 대해 빈도수가 셀 임계값 이상인 셀의 번호를 저장하는 방법이 효율적이다.

2.2 질의 처리

새로운 데이터의 입력 시 모든 차원의 해당 구간이 근사정보파일의'1'인 구간으로 나타나는 경우에만 분류정보파일을 검색하고 그렇지 않은 경우 검색대상에서 제외함으로서 검색성능이 향상된다. 차원이 증가할수록 적어도 하나의 차원에 대하여 근사정보파일의 값이 '0'이 나올 확률이 커지므로 고차원의 데이터에서 크기가 큰 분류정보파일을 검색하는 횟수를 줄일 수 있다 그림 2 는 그림 1 에서 셀 임계값과 구간 임계값이 '1'인 경우 구성된 근사정보파일과 분류정보파일에서 질의 레코드를 처리하는 과정을 보여준다.

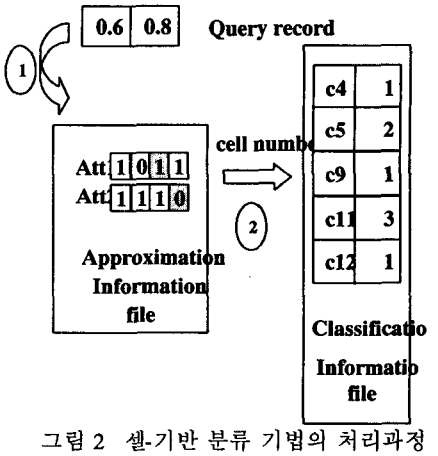

그림 2 셀-기반 분류 기법의 처리과정

전체적인 처리과정을 보면 먼저 질의레코드의 각 차원에 대해서 해당구간을 구한다. 해당구간의 빈도수가 구간 임계값 미만인 경우, 즉 근사정보파일의 값이 '0'인 경우에는 분류정보파일을 검색하지않고 질의레코드의 모든 차원의 구간이 '1'인 경우에만 분류정보파일 검색한다. 이때 구간 값들의 조합으로 셀 번호를 구한다. 구해진 셀 번호를 가지고 분류정보파일을 순차적으로 검색하여 일치하는 셀 번호가 있으면 분류정보파일에서 필요한 정보를 얻는다.만약 분류정보파일에 일치하는 셀 번호가 없는 경우는 질의레코드는 트레이닝 셋의 빈도수가 셀 임계값 이하의 구간에 속하므로 의미 없는 데이터로 간주한다.

3. 실험 및 성능평가

본 장에서는 제안된 셀-기반 분류방법을 구현하여 성능 평가를 수행한다.실험을 통하여 제안된 셀-기반 분류방법이 대용량의 데이터에 대한 처리가 가능함을 보이고, 각 차원의 구간별 밀도에 대한 구간 임계값을 변화하면서 성능을 측정한다. 실험에 사용된 시스템 환경은 CPU

650 MHz dual, 메모리 512MB 의 리눅스 서버에서 수행하였다. 사용된 데이터는 IBM Quest Data mining project[5]에서 사용한 Synthetic Data Generation Code for Classification을 이용하여 각각 4 차원, 8 차원의 10 만 건의 데이터를 사용하였다. 성능평가는 10 만 건의 데이터를 분류하여 근사정보파일과 분류정보파일을 생성하는 시간과 1000건의 질의레코드를 처리할 때 구간 임계값에 따라서 분류정보파일 검색하지 않고 근사정보파일에서 걸러지는 비율과 정확율을 측정한다. 분류시간은 4 차원의 경우 약 6 초 정도 소요되며 8 차원의 경우 약 300 초 정도 소요된다

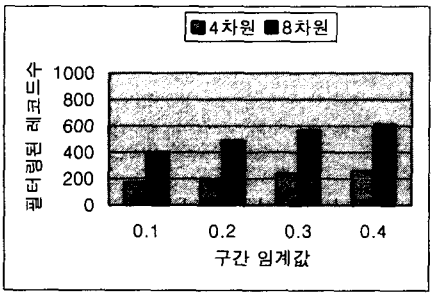

그림 3 필터링 비율

그림 3 은 각 구간 임계값에 따라 근사정보파일에서 여과되는 비율을 나타낸다. 구간 임계값을 크게하면 여과되는 비율은 점점 커진다. 그러나 정확성이 떨어지는 단점이 있다. 반대로 구간 임계값을 적게하면 정확성은 높으나 여과되는 비율이 적어 평균검색 시간이 느려지는 단점이 있다.

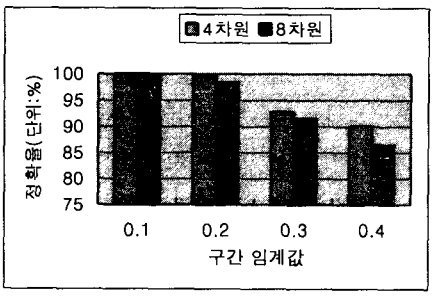

그림 4 정확율

그림 4 는 구간 임계값에 따른 1000 건의 질의에 대한 정확율을 나타내며, 그림 5 는 구간 임계값에 따른 1000 건의 질의에 대한 평균 검색시간을 나타낸다 구간 임계값이 커질수록 검색 시간이 빠르나 정확율이 낮아진다. 따라서, 정확율과 평균 검색시간의 상호관계를 고려한 본 논문에서 제안한 방법의 성능은 식 3 으로 표현할 수 있다.

$$E(\lambda) = P(\lambda) * \mathrm{Wp} + T(\lambda) * \mathrm{Wt} \quad (식\ 3)$$

식 3 에서 Wp 는 정확율의 가중치이며, Wt 는 검색시간의 가중치이다. P(λ)는 구간 임계값이 λ일때의 정확율을 나타내며, T(λ)는 구간 임계값이 λ일때의 평균검색시간을 나타낸다. P(λ)와 T(λ)는 0~1 사이의 값으로 정규화한 값이다. 그림 6 은 구간 임계값에 따른 성능 $E(\lambda)$를 나타낸다. Wp 와 Wt 각각 1 의 가중치를 주었을 때 λ=0.2 일때 가장 효율적인 성능을 보인다.

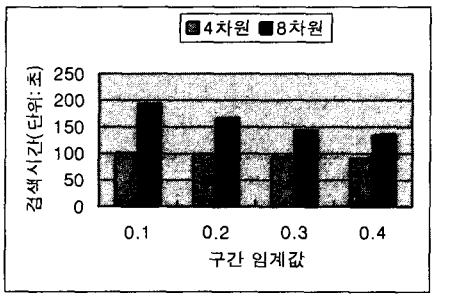

그림 5 평균 검색시간

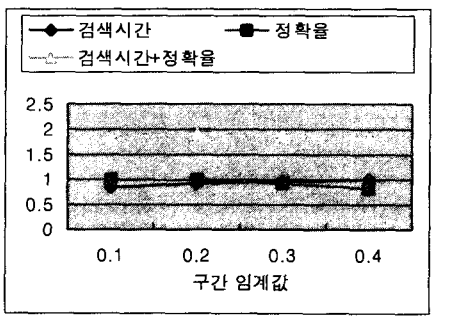

그림 6 가중치를 적용한 성능비교

4. 결론 및 향후연구

본 논문에서 제안한 셀-기반 분류방법은 첫째, 메모리 상주 데이터에 대해 한정되어 있는 문제점을 해결하였고, 고차원의 데이터를 효율적으로 처리할 수 없는 문제점을 각 차원별 구간 임계값을 사용하여 크기가 큰 분류정보파일을 검색하는 횟수를 줄임으로서 성능을 향상시켰다. 또한, 데이터의 속성에 관계없이 적용 가능한 새로운 분류 알고리즘을 제안하였다.

향후연구로는 보다 다양한 데이터를 가지고 실제 응용에 적용하여 본 논문에서 제안한 방법의 성능을 측정하는 연구이다.

참고 문헌

[1] J. Ross Quinlan. C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
[2] M. James. Classification Algorithms, Wiley, 1985.
[3].D.E.Goldberg, Genetic Algorithms in Search., Optimization and Machine Learning, Morgan Kaufmann, 1989.
[4].L.Breiman et.al. Classification and Regression Trees, Wadsworth, Belmont, 1984.
[5] http://www.almaden.ibm.com/cs/quest.