한국지능정보시스템학회논문지 제7권 제1호 2001년 6월(pp. 27-45)

시계열 데이터로부터의 경향성 기반 순차패턴 탐색

오용생
포항공과대학교 정보통신학과 연구원
(albatros@postech.ac.kr)

이동하
포항공과대학교 정보통신연구소 연구원
(dongha@postech.ac.kr)

남도원
포항공과대학교 전자컴퓨터공학부 연구원
(irene@postech.ac.kr)

이전영
포항공과대학교 전자컴퓨터공학부 부교수
(jeon@postech.ac.kr)

데이터마이닝에서 시계열 데이터로부터 순차패턴을 발견하는 연구는 사건이나 아이템이 주로 연구되어왔지만, 최근에는 설비의 상태를 알 수 있는 센서와 같은 수치 값의 형태를 가지는 분야에 관심을 가지게 되었다. 그러나 수치 형태의 데이터는 패턴을 만드는 동안 동일한 값을 가지는 경우가 거의 없기 때문에 기존의 사건이나 아이템 등으로 변환될 수 있는 패턴요소의 특징을 만드는 것이 가장 중요하다. 이러한 패턴요소를 발견하는 지금까지 방법은 이동 윈도우와 클러스터링을 사용하는 방법을 적용하였는데, 이러한 방법은 다양한 윈도우의 크기와 클러스터 값을 적용하여 반복적으로 작업을 하며, 찾아진 결과를 해석하는데도 많은 문제가 있다.

본 연구는 수치 값을 가진 데이터를 벡터의 형태로 만들어 패턴요소를 만드는 방법을 제시한다. 이렇게 만들어진 패턴요소는 전체 데이터를 사용하는 것 보다 이해되기 쉽고 보다 빠르게 순차패턴을 찾을 수 있다. 벡터로 변환된 패턴요소는 각도와 크기를 가지는데 우리는 이들 벡터들의 상호 연관성을 정의하고, 이들 연관성을 이용하여 순차패턴을 찾는 방법을 제시한다.

1. 서 론

데이터마이닝 분야에서 시계열 데이터(time-series data)내의 데이터들이 가지는 순차패턴을 발견하는 연구는 상품판매점에서 고객의 구매 패턴을 발견하는 것부터 증권에서 유사한 종목의 주식가격의 시간적 패턴 분석, 엔지니어링 분야에서 설비의 고장분석에 이르기까지 다양한 적용범위를 가지고 있다. 예를 들어 서점에서 일정한 기간동안 판매된 정보를 조사해 보니 "삼국지를 구입한 사람은 몇 주일 안에 수호지를 구입한다"와 같은 순차패턴 정보를 발견하는 것으로 시간적 순서를 가진 사건이나 트랜젝션(transaction)의 발생을 분석하여 이들이 자주 발생되는 사건이나 트랜젝션이 어떤 순서로 발생되는 것인지를 찾는 것이다. 그러나 상품(Items)이나 어떤 사건(Event)과 같이 데이터의 특징이 명확한 대상들 간에 발생되는 순차패턴의 발견은 많이 연구되어 왔으나 엔지니어링 데이터와 같이 패턴을 구성하는 데이터가 다양한 수치 값을 가지는 분야에서의 순차패턴 발견은 최근[7]에서야 관심대상이 되었다. 수치 값과 같은 데이터로 이루어진 데이터 집합에서 어떤 패턴을 발견하기 위해서는 먼저 데이터의 특징을

* 본 연구는 전자·컴퓨터공학부를 통한 교육부 두뇌한국 21사업의 지원에 의해 수행되었음.

발견하여야 한다. 즉 기존에 상품이나 사건의 경우는 타임시리즈에 존재하는 데이터 집합이 $I=\{i_1, i_2, \ldots, i_m\}$과 같이 m개의 문자로 표현될 수 있는 데이터 집합이지만, 수치 값을 가진 데이터의 경우 $I=\{-\infty, 0, +\infty\}$의 값의 범위를 가지므로 단순히 이들 값의 순서에 따른 패턴을 발견하는 것은 불가능하고 무의미 한 것이다. 이러한 분야에서 순차패턴의 발견은 먼저 데이터 값 자체보다는 데이터가 가지는 시간적인 변화 특성을 기준으로 패턴을 발견하여야 한다. 예를 들어 어떤 상품의 할부금액의 변화 패턴을 분석한 결과 "2개월간 일정한 수준 이상의 할부금액 감소가 발생되면 향후 3개월 내에 할부금액의 급격한 증가가 나타난다"와 같은 패턴을 발견하기 위해서는 각 데이터가 발생되는 시점의 값이 중요하기 보다는 이들 값이 시간적으로 연속해서 만들어내는 특징에 의해 발생되는 패턴을 찾는 것이 보다 의미가 있을 것이다. 우리의 연구는 위와 같이 시계열 데이터 내의 데이터가 값의 범위가 정해지지 않은 데이터 집합일 경우 이들 값이 만들어 내는 특징을 발견하고 또한 이들 내부에 숨어있는 모든 순차패턴을 발견하는 것을 목적으로 한다.

본 논문의 구성은 제 2장에서 기존에 수행된 시계열 데이터 내에서 순차패턴 발견에 관한 연구를 알아보고 제 3장에서 시계열 데이터를 경향성 형태로 변환하여 패턴 발견을 위한 데이터 정의 방법과 제 4장에서 정의된 데이터에서 패턴을 발견하는 방법을 살펴본 후 제 5장에서 실험결과와 6장에서 결론 및 향후 연구방향을 제시한다.

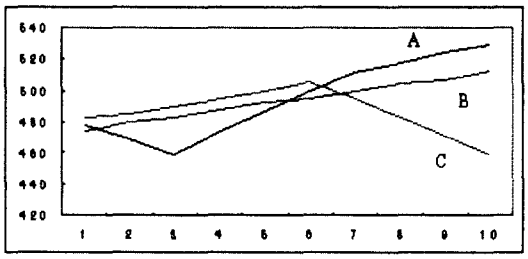
2. 관련 연구

수치 값을 가지는 시계열 데이터에서 순차패턴을 발견함에 있어 가장 중요한 것은 시계열 데이터 내에 있는 패턴 요소를 찾는 것이다. 이에 관련된 연구는 크게 4가지의 형태로 구분할 수 있다. 첫번째로는 시계열 데이터를 시간축에 따른 진폭의 형태로 보는 것이다.[3] 따라서 발생주가 일정하다면 시계열 데이터의 표현을 주파수 영역(Frequency Domain)으로 변환하여 표현을 하고, 주로 나타나는 주파수가 해당 시계열 데이터를 가장 대표하는 특징(Feature)으로 본다는 개념으로 다음의 식(1)로 변환하여 구한다.

$$X_f = 1/\sqrt{n} \sum_{t=0}^{n-1} x, \exp(-j2\pi ft/n) \quad \text{식(1)}$$

그러나 이것은 적용대상이 반드시 일정한 시간적 영역에서 항상 주기성을 가지는 데이터들만 적용이 가능하고 주기성이 불명확한 보다 일반적인 시계열 데이터에는 적용할 수 없다. 또한 주파수 영역만을 적용해서는 패턴 내부에 있는 순차적 연관성을 찾기가 곤란하다. 두 번째로는 데이터가 가지는 값을 일정한 범위로 나누어서 같은 범위의 값을 가질 경우 대표 값으로 변환하는 방법이다. 이러한 방법은 단지 값의 범위를 묶어 표현했을 뿐 전체 데이터를 다루는 점에선 별다른 차이가 없다. 즉 패턴요소의 값이 중간에 약간 다른 범위의 값으로 표현될 경우 전혀 다른 순차패턴으로 간주하게 된다. 또한 최근에 연구된 선형적 표현방법[10][11]의 경우 전체 데이터를 k개의 데이터로 표현하여 이들 값을 직선의 형태로 표현한 것

으로 선택되는 k값에 따라 원래 형태와는 전혀 다른 형상을 만들 수 있다. 마지막으로 가장 많이 사용되는 방법으로는 윈도우를 이동시키며 시계열 데이터의 특징이 되는 패턴을 찾는 방법으로 전체 데이터를 윈도우 크기 w를 가지는 데이터 집합으로 계속 나누면서 유사한 패턴을 찾는 방법으로 이 방법은 기존의 연구 중 에피소드 방법[3]을 일반적인 수치 값을 가지는 시계열 데이터에 적용하는 시도를 하였다. 이들은 먼저 시계열 데이터를 사용자가 지정한 윈도우 w 크기를 가지는 데이터 집합으로 구분하여 시계열 데이터베이스 내의 데이터 특징을 구분하였다. 즉 시계열 데이터 집합을 S라 할 경우 $S=\{x_1, x_2, \ldots, x_n\}$이고 이들을 각각 크기 w인 부분적 시계열(Subsequence)로 나누므로 부분적 시계열 데이터를 S_i라 할 경우 $S_i=\{x_i, \ldots, x_{i+w+1}\}$가 된다. 따라서 최초의 시계열 데이터 집합은 윈도우 w로 구성된 $S=\{S_1, S_2, \ldots, S_{n-w+1}\}$으로 변환된다. 이들은 패턴탐색 시 보다 높은 지지도(Support) 값을 얻기 위해 유사한 윈도우들 간에 클러스터링을 한다. 예를 들어 *k-means* 방법을 사용한다면 위의 데이터 집합을 k개의 클러스터로 구분하고 이들 각각의 클러스터에 알파벳 이름을 붙인다. 따라서 클러스터의 개수 k만큼의 알파벳들로 구성된 시계열 데이터가 생성되는 것이다. 따라서 이렇게 변환된 시계열 데이터는 기존의 사건이나 트렌젝션 데이터를 가지는 시계열 데이터에서 순차패턴을 찾는 방법과 동일 방법인 AprioriAll 알고리즘을 이용하여 가장 빈번하게 발생되는 패턴 후보를 만들어 순차패턴의 길이를 확장하는 방법을 사용하였다. 그러나 윈도우 방식은 유사한 윈도우를 클러스터링 하는 과정에서 <그림 1>에서 볼 수 있듯이 전혀 다른 의미적 해석이 동일한 클러스터로 결합되는 현상이 자주 나타난다.

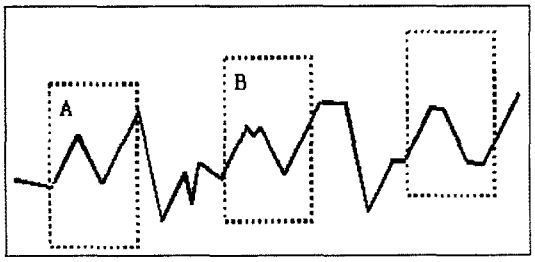


<그림 1> 같은 클러스터에 있는 데이터의 모양

즉 <그림 1>은 전체 데이터 520개의 시계열 데이터를 윈도우크기 w=10과 클러스터수 k=20으로 클러스터링 한 후 Cluster ID=5에 있는 데이터를 그래프로 나타낸 것으로 중심 값을 지나는 값의 변화 B를 기준으로 볼 때 선 "A"와 선 "C"는 서로 상반된 모습을 보인다. 이 것은 단지 윈도우 크기 내에 있는 모든 점들간에 떨어진 거리 만을 고려하여 가장 평균점과 가까운 윈도우를 같은 클러스터로 결합하기 때문에 발생되는 것으로 응용적인 측면에서 사용자가 의미적 해석을 할 때 많은 혼란을 야기시킨다. 즉 다시 말해서 윈도우의 클러스터링 작업은 클러스터의 개수 k에 따라 k값이 너무 작을 경우 전혀 의미 없는 결과가 만들어지고, 또한 너무 많은 k값을 적용할 경우는 패턴을 발견 시 충분한 지지도를 얻기 힘들어 패턴을 발견하기 힘들게 된다. 두 번째 문제점으로는 윈도우의 크기 w의 지정이다.

동일한 시계열 데이터를 서로 다른 윈도우 크기로 패턴을 찾을 경우 패턴이 발견될 때와 그렇지 않을 경우가 발생된다. 그것은 윈도우

크기보다 작은 패턴의 경우 윈도우를 클러스터링 하는 과정에서 다른 클러스터에 속하게 되기 때문이다. 이로 인해 시계열 데이터내의 모든 순차패턴을 발견하기 위해서는 가능한 모든 종류의 윈도우 크기를 다 조사해 봐야 하는 문제점이 있다.



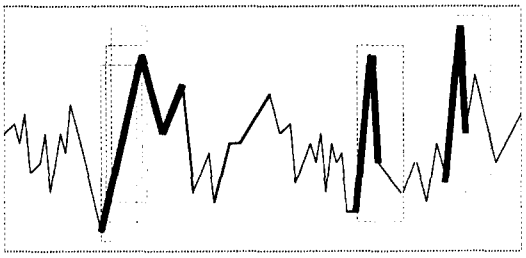
<그림 2> 유사한 패턴을 가지는 시계열 데이터

마지막으로 언급할 수 있는 윈도우 방식의 문제점으로는 같은 클러스터에 속하는 윈도우를 구분하기 위한 측정방법(Measure)에 있다. 일반적으로 시계열 데이터 내에 존재하는 패턴의 형태는 항상 일정한 모양의 패턴만을 만들지는 않는다. 대개는 모양은 유사하지만 이들 패턴이 만드는 시간적 길이가 다양하게 존재한다. <그림 2>에서 보면 패턴 A와 패턴 B는 거의 동일한 형태의 패턴을 나타내고 있지만 이들이 클러스터링 될 경우 다른 클러스터에 속하게 되는 경우가 종종 발생된다. 그것은 클러스터의 측정기준이 일반적으로 사용하는 Euclidean Distance 방법을 사용할 경우 데이터가 시간 축에 약간의 변형만 발생되어도 측정값의 변화가 크게 발생되기 때문이다.

이와 같이 윈도우 방식을 포함한 기존 연구는 시간적 흐름에 따른 데이터 값의 변화성에 따른 패턴 요소의 추출보다는 단지 고정되고 일정한 시간적 크기를 기준으로 데이터를 구분하여 보고 특징을 찾고 있기 때문에 순차패턴과 같이 앞에서 발생된 패턴요소로 인한 다음 패턴의 발생에 대한 연관성을 발견하는 분야에는 적용하기 곤란하고 단지 미리 알고 있는 일정한 길이의 시계열 데이터가 데이터베이스에서 어느 구간에 존재하는지를 찾는 유사패턴 탐색에 적합하다.

3. 제안된 방법

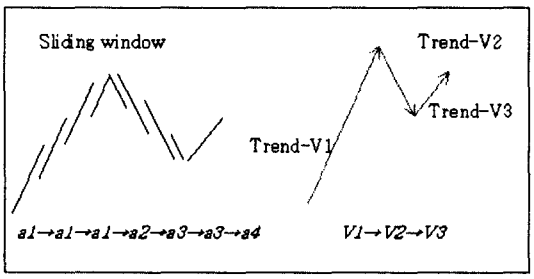
2장에서 언급했듯이 기존 연구의 대부분의 특징은 패턴요소를 축출하는 방법을 단지 데이터를 일정한 시간 간격으로 나누어 유사한 윈도우를 묶는 형태를 취하였다.



<그림 3> 윈도우와 트렌드

즉 모든 데이터를 일정한 윈도우 크기 w로 연속적으로 나누어 표현하기 때문에 위의 <그림 3>에 있는 굵은 선으로 표시된 "상승->하강->짧은 상승"을 나타내는 단순한 패턴의 모습을 정의하기 위해서는 많은 윈도우가 계속적으로 나타나면서 정보를 표현하여야 한다. 그러나 우리는 시계열 데이터내의 패턴요소를 시간의 흐름에 따른 데이터 값의 변화가 같은 경향성

을 가지는 대상으로 본다면 단지 일정한 크기 이상의 상승과 그리고 연이어 하강의 경향성이 나타나고 다시 짧은 상승을 나타내는 트렌드의 순서로 표현이 가능할 것이다. 즉 시간에 따른 같은 변화성향을 가지는 구간을 하나의 트렌드로 구분하며, 우리는 벡터(Vector)라고 부르는 패턴요소로 시계열 데이터의 특징을 구분한다. 따라서 이러한 연속되는 벡터들로 시계열 데이터를 표현한다면 보다 의미적으로 쉽게 패턴의 변화를 나타낼 수 있다.



<그림 4> 이동 윈도우와 트렌드의 패턴요소

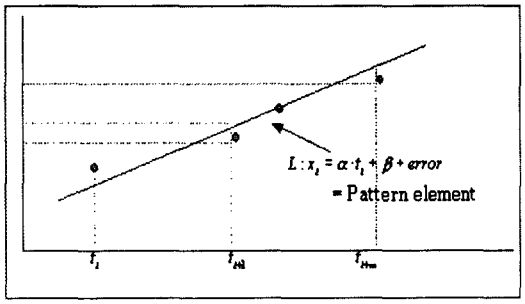
이렇게 시계열 데이터를 기존의 윈도우 방식 대비 벡터로 표현된 방식을 비교한 <그림 4>를 보면 기존방법의 경우 $a1 \rightarrow a1 \rightarrow a1 \rightarrow a2 \rightarrow a3 \rightarrow a3 \rightarrow a4$ 로 표현되는 상승과 하강의 연속적인 값의 변화 패턴이 벡터방식을 적용할 경우는 보다 간단히 $V1 \rightarrow V2 \rightarrow V3$ 와 같이 표현된다. 또한 벡터가 가지는 정보를 기울기 값과 시작위치 및 지속시간(Duration)의 정보로 표현할 경우 순차패턴을 찾을 때 사용자가 일정한 지속시간과 기울기 이상의 패턴 요소들이 만드는 순차패턴을 발견하거나 또는 시작 값이 사용자가 정의하는 값 이상인 벡터들이 만드는 순차패턴의 발견이 용이 할 것이다. 그러나 윈도우 방법에서는 단지 사전에 최소값 만을 정의 하여, 최소값 이상인 전체 데이터를 다루므로 패턴의 가치가 없는 불필요한 패턴 요소를 포함하는 순차패턴을 발견함으로 인해 발견된 패턴을 적용하기 곤란한 경우가 종종 발생되고, 시계열 데이터 전체를 다루므로 인해 많은 수행시간이 소요된다. 따라서 본 연구는 수치 값을 가지는 시계열 데이터에서 순차패턴을 발견하는데 윈도우 방법을 사용하지 않고 데이터 내부에 존재하는 값의 시간적 변화를 나타내는 트랜드를 기초로 한 순차패턴을 발견하고 보다 의미 있는 패턴 발견을 목적으로 한다.

4. 순차패턴 요소

4.1 패턴요소 정의

시계열 데이터는 시간의 변화에 따라 발생되는 데이터들의 집합이다. 이들은 데이터 특징에 따라 차이는 있지만 대분의 경우 일정한 시간적 길이를 가지면서 데이터 값이 변화하는 그래프 모양을 가진다. 따라서 시계열 데이터 집합을 D라 할 때 $D=\{x_1, t_1, x_2, t_2, \ldots, x_n t_n\}$으로 표현되며 만일 임의 시간구간 $[i,..,i+m]$ 에 있는 데이터가 시간의 변화에 따른 값의 변화 경향성이 상호 연관성이 있다면 $\{x_i, x_{i+1}, \ldots, x_{i+m}\}$에 이들 구간을 대표하는 직선의 방정식이 존재할 것이다. 따라서 시간과 값의 상호 연관성이 높은 구간에 대하여 이들 구간에서 각 시점에 나타나는 값과 거의 유사한 값을 만드는 직선의 방정식으로 구간을 대표하는 패턴요소를 만드는 것이다. 이와 같은 방법의 장점으로는 구간내의 모든 값을 하나의 패턴요소로 간단히 표현할 수 있다는 것이다. 따라서 순차패

턴을 찾을 때 전체 데이터를 다루는 것이 아니고 단지 패턴요소로 압축된 데이터만을 다루므로 보다 빠르게 패턴을 발견할 수 있다.



<그림 5> 시간과 데이터 값의 연관성

이렇게 데이터 값이 시간적 연관성을 만족하는 구간을 구분하기 위해 우리는 상관계수 값(correlation coefficient)을 측정하여 검사한다. 즉 임의 구간 $[i,..,i+m]$ 내에 있는 값을 시간에 대한 직선의 방정식을 만족하기 위해서는 직선의 방정식을 적용한 시간의 함수에 의해 발생되는 각각의 값과 실제 데이터가 가지는 값의 차이가 가능한 적어야 한다.

따라서 <그림 5>에서 $x_i = \alpha \cdot t_i + \beta + error$ 에서 오차(*error*)값을 최소로 만드는 형태를 취하여야 한다. 이러한 오차 값의 허용범위를 정하는 기준을 상관계수 값을 기준으로 하며, 본 연구에서는 사용자가 정한 상관계수 값을 λ 라 할 때, 값 λ 이상을 만족하는 직선의 방정식으로 데이터를 표현한다. 시계열 데이터 D 에서 상관계수 값을 만족하는 i 번째 패턴요소를 s_i 라 할 때 일반적인 정의 식으로 표현하면 다음과 같다.

$$s_i = \{x_i, x_{i+1}, \cdots, x_{i+m}\}, \{s_i \| r_i| \geq \lambda\} \quad \text{식(2)}$$

여기서 γ_i는 일반적으로 사용하는 Pearson의 상관계수 값으로 1 또는 -1의 값에 가까울수록 직선의 표현식을 만족하며 0 에 가까울수록 상관관계가 없다. Pearson의 상관계수를 구하는 방법은 식(3)과 같다.

$$\gamma = \frac{m\left(\sum_{i=1}^{m} t_i x_i\right) - \left(\sum_{i=1}^{m} t_i\right)\left(\sum_{i=1}^{m} x_i\right)}{\sqrt{\left[\left(\sum_{i=1}^{m} t_i^2\right) - \left(\sum_{i=1}^{m} t_i\right)^2\right] \times \left[m\left(\sum_{i=1}^{m} x_i^2\right) - \left(\sum_{i=1}^{m} x_i\right)^2\right]}} \quad \text{식(3)}$$

위의 식에서 $1 \leq t_i \leq m$ 인 범위를 가지며 x_i 는 타임시리즈 내의 값을 나타낸다.

예를 들어 D = {157, 169, 179, 188, 168, 154, 152, 162, 172, 182, 192, 200, 210, 220, 240} 이고 γ = 0.9일 경우 이들의 패턴요소는 다음과 같이 3개로 표현 된다. s_1 = {157, 169, 179, 188}, s_2 = {188, 168, 154}, s_3 = {152, 162, 172, 182, 192, 200, 210, 220, 240} 이들은 모든 상관계수 값이 모두 0.9 이상을 가지므로 조건을 만족하는 패턴요소가 된다. 이렇게 만들어진 데이터 집합을 보다 다루기 편리하게 하기 위해 벡터의 형태로 표현한다. 즉 기울기(α)와 시작위치 값(St) 및 기간(d) 값을 가지는 벡터적인 표현으로 표기한다. 따라서 하나의 트렌드는 다음 벡터와 같다.

$$v_i(s_i) = (\alpha_i, St_i, d_i)$$

여기서 α_i 는 기울기이며 기간 $d_i = x_{i+m}.t - x_i.t$, $St_i = x_i.t$를 나타낸다.

이와 같은 표현방법은 기존의 윈도우 방식보다는 데이터의 특징을 보다 잘 반영할 수 있고 전체 데이터를 다루는 것보다는 보다 압축된

데이터를 다루게 된다. 그러나 위와 같이 벡터의 표현으로 만들어진 순차패턴의 패턴요소는 아주 다양한 값의 범위를 가진다. 즉 기울기(α) 값이 가지는 다양성과, 기간(d)도 같은 기울기 값에서도 아주 많은 종류가 발견될 것이다. 따라서 이들 값 자체만으로는 자주 발생되는 공통적인 패턴요소를 발견하기 힘들므로 충분한 지지도를 얻는 패턴요소를 발견하기 위해 위에서 만들어진 벡터리스트 전체에 대해 그룹화 작업을 해야 한다. 이것은 일반적으로 생각할 수 있는 간단한 방법으로 처리가 가능하다. 즉 기울기를 기준으로 0부터 +90 까지 및 -90까지의 각도에 대하여 사용자가 일정한 간격으로 나누고 이 범위에 포함되는 모든 기울기는 같은 값을 가지도록 변환하는 것이다. 우리는 여기서 기간을 고려하지 않았는데 그것은 다음에 설명되는 패턴요소간의 연관성 정의에 따라 고려된다. 따라서 모든 벡터들은 소속되는 그룹의 기울기(α) 값을 대표 값으로 하는 벡터로 이루어진 집합이 된다.

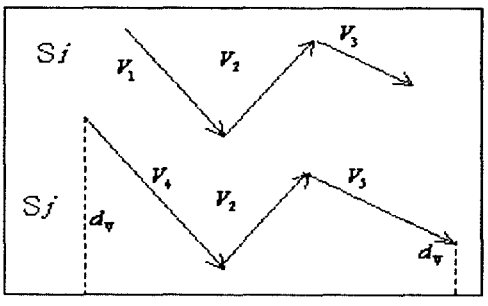
4.2 패턴 요소들 간의 연관성

우리는 시계열데이터베이스 내의 각 데이터 값을 일련의 연속되는 값들이 사용자가 정의한 상관계수 λ값 이상을 가지는 벡터로 표현하였고 또한 이렇게 표현된 벡터들은 패턴발견을 위해 충분한 지지도를 얻기 위해 그룹화을 통해 패턴이 되는 기준벡터를 발견하였다. 우리는 이들 통해 순차 패턴을 발견한다. 그런데 이렇게 만들어진 m개의 패턴요소 벡터의 집합 $I_V = \{v_1, v_2, \ldots, v_m\}$에 있는 기준 벡터는 $v_i = (\alpha, *, d_i)$로 시작점을 가지지 않는 벡터들 이다. 그리고 이들 표현된 기준 벡터들은 기울기가 같은 두개의 벡터 v_i, v_j간에 다음과 같은 정의 4.1이 성립한다.

[정의4.1]:

$$\exists v_i, v_j, \{v_i.\alpha \wedge d_i \leq d_j \mid v_i \in v_j\}$$

즉 $v_i = (\alpha_i, *, d_i), v_j = (\alpha_j, *, d_j)$ 두 벡터가 기울기(α)가 같고 $d_i \leq d_j$일 경우 벡터의 합에 따라 기간(d_j)이 큰 벡터는 $v_j = (\alpha_i, *, d_i) + (\alpha_i, *, d_j - d_i)$와 같다. 따라서 서로 다른 패턴요소 일지라도 기울기가 같으면 그 내부에는 기간의 크기가 작은 벡터가 포함된 것으로 볼 수 있다. 즉 $v_j = v_i.d_i + v_\nabla.d_\nabla$ 가 되며 이것은 벡터 v_j가 발생된 기간에 발생된 것이므로 $v_j = \{(v_i v_\nabla), (v_\nabla v_i)\}$ 의 집합으로 볼 수 있다. v_∇ 벡터가 먼저 발생되고 다음에 v_i가 이후에 발생된 $v_j = \{(v_i v_\nabla)\}$ 또는 반대의 경우 $v_j = \{(v_\nabla v_i)\}$가 동시에 존재한다. 예를 들어 다음 <그림 6>과 같은 두개의 트렌드가 있고 이들 트렌드를 만드는 각 벡터는



<그림 6> 벡터의 포함관계

$V_1.\alpha = V_4.\alpha \wedge d_r = (d_1 + d_\nabla)$,
$V_3.\alpha = V_5.\alpha \wedge d_5 = (d_3 + d_\nabla)$ 의 관계를 가질 경우 두개의 시계열 데이터 $\{S_i = V_1, V_2, V_3\}$ 와 $\{S_j = V_4, V_2, V_5\}$는 앞의 정의 4.1에 따라 $V_4 = \{(V_\nabla V_1), (V_1 V_\nabla)\}$ 의 정보를 가지고 있으므로 $V_1 \in V_4$ 를 만족하고 또한 $V_5 = \{(V_\nabla V_3), (V_3 V_\nabla)\}$ 이므로 $V_3 \in V_5$ 를 만족한다. 따라서 시계열 데이터 s_j 는 이들의 포함관계의 정보에 따라 다음과 같이 다시 나타낼 수 있다.

$$s_j = \{(V_\nabla V_1)(V_2)(V_3 V_\nabla)\} = \{V_\nabla V_1 V_2 V_3 V_\nabla\}$$

따라서 $s_i \in s_j$ 를 만족한다.

또한 위의 정의 4.1은 다음과 같은 정의 4.2로 확장이 가능하다.

[정의 4.2]: $\exists v_i, v_j, n>0$ 일경우

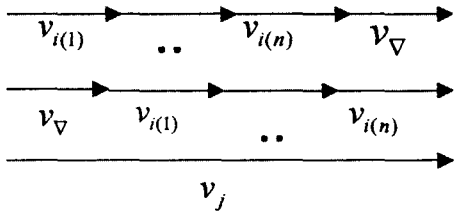
$$\{v_i.\alpha = v_j.\alpha \wedge d_j = n \cdot d_i \mid v_j = Y_n v_i\}$$

위의 정의는 벡터의 곱의 기본적인 성질에 따라 만족하는 것으로 $v_j = v_{i(1)}.d_i + v_{i(2)}.d_i + \ldots + v_{i(n)}.d_i$ 와 같다. 따라서 우리는 $v_j = \{(v_{i(1)} v_{i(2)} \ldots v_{i(n)})\}$ 으로 벡터 v_j 를 표현한다. 이와 같은 정의 4.2는 정의4.1과 결합되어 다음 정의 4.3을 만든다.

[정의 4.3]:

$$\exists v_i, v_j, n>0, \{v_i.\alpha = v_j.\alpha \wedge (n \cdot d_i + \nabla) = d_j \mid v_j = Y_n v_i + v_\nabla\}$$

위의 정의는 v_i, v_j 가 같은 기울기를 가질 경우 $v_j = n \cdot v_i + v_\nabla$ 와 같으므로 다음과 같은 벡터의 트렌젝션이 발생된 것으로 볼 수 있다.



<그림 7> 벡터 트렌젝션

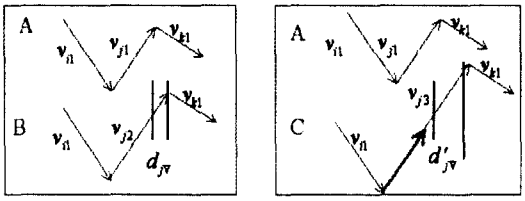
따라서 벡터 $v_j = \{(v_{i(1)} v_{i(2)} \ldots v_{i(n)} v_\nabla), (v_\nabla v_{i(1)} v_{i(2)} \ldots v_{i(n)})\}$ 로 다시 표현될 수 있다.

이와 같은 벡터의 포함관계를 이용할 경우의 장점으로는 기존 윈도우방법에 발견하기 곤란한 형태의 그래프 패턴을 보다 손쉽게 발견 할 수 있고 데이터의 처리도 간단한 장점이 있다.

4.3 패턴의 유사성

우리는 앞에서 벡터의 상호 포함관계를 정의하여 기울기 값은 같으나 서로 다른 기간을 가지는 벡터로 이루어진 패턴에서 공통된 패턴을 발견하였다. 그러나 서로 다른 벡터의 경우 v_∇ 의 영향에 대해서 언급이 없이 공통 패턴을 다루므로 인해 벡터의 기간 차이가 심할 경우 v_∇ 으로 인해 전혀 다른 패턴이 같은 패턴으로 만들어지는 오류를 범하게 된다.

따라서 우리는 서로 다른 길이로 구성된 두개의 시계열 데이터에서 이들 내부에 존재하는 벡터가 같은 벡터로 처리가 될 수 있는 제한조건을 주어 이 문제를 해결하였다. 즉 순차패턴을 이루는 두개의 패턴리스트에서 각각의 패



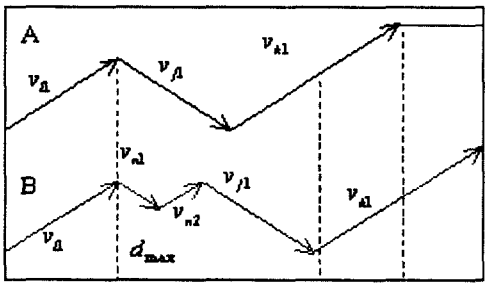
<그림 8> 두 벡터 사이의 v_{∇} 영향

턴에 있는 벡터가 유사하기 위해서는 $d_{\min}$ 이내의 기간의 차가 있을 경우로 제한된다. 즉 위의 <그림 8>을 보면 순차패턴 A와 B는 v_{j1} 과 v_{j2} 에서 $d_{j\nabla}$ 의 기간의 차가 발생되나 순차패턴 A와 C는 v_{j1} 과 v_{j3} 에서 $d'_{j\nabla}$ 의 기간의 차가 너무 크므로 인해 순차패턴 A와는 다른 모양을 그린다. 따라서 이러한 것을 구분하는 방법으로 두 벡터의 유사성을 비교할 경우 사용자가 정의하는 유사정도의 척도인 $d_{\min}$ 의 크기 이내인 벡터는 같은 벡터로 포함시키나 $d_{\min}$ 값 이상의 값을 가지는 경우는 별개의 벡터로 구분하는 것이다. 이와 같은 구분은 정의 4.4 와 같다.

[정의 4.4] : 두개의 순차패턴이 같기 위해서는 같은 순서의 모든 순차패턴 요소들이 서로 같은 기울기 값을 가져야 하며, 순차패턴을 이루는 가장 처음과 마지막 부분의 벡터를 제외한 모든 순차패턴 사이에 있는 벡터들간의 기간(Duration)의 편차 값은 $d_{\min}$ 값 이내의 값을 가져야 한다.

4.4 패턴의 확장

대부분의 순차패턴을 발견하는 과정은 단순히 다음 순차패턴 요소가 앞의 순차패턴 요소가 발생한 직후 바로 발견되는 경우는 많지 않다. 즉 일정한 시간이내에 발견되는 되는 것으로 단지 발견되는 순서가 일정한 것을 말한다. 따라서 이와 같은 패턴을 발견하기 위해서는 다음 순차패턴 요소가 발견되는 기간의 범위를 정할 필요가 있다. 이것은 기존의 GSP알고리즘[2]에서 적용된 것으로 패턴요소가 발생 되야 하는 시간적 범위를 사용자가 정의하게 함으로서 보다 의미적인 정보를 찾는 것이다. GSP알고리즘에서 적용한 예와 같이 어떤 서점에서 판매한 책의 종류와 이 책을 구매한 사람이 1년 뒤에 다른 종류의 책을 구매 했다고 할 경우 이러한 정보를 가지고 발견된 패턴은 의미 없는 패턴이 될 것이다. 따라서 최근 3개월 이내에 구매한 패턴으로 제한을 둔다면 보다 정보의 가치가 있다. 우리는 이러한 GSP알고리즘을 이용하여 트렌드의 발생이 시간적으로 연속되지는 않지만 사용자가 지정한 $d_{\max}$ 이내에 발생된 패턴까지 확장 시키면서 패턴을 찾는다.



<그림 9> 순차패턴의 확장

<그림 9>에서 두개의 순차패턴 A와 B를 비교할 $A = v_{i1} \alpha v_{j1} \alpha v_{k1}$ 경우 의 순서로 발생되고 있다. 그러나 $B = v_{i1} \alpha v_{n1} \alpha v_{n2} \alpha v_{j1} \alpha v_{k1}$ 의 순서를 만들면서 순차패턴 A와 비교할 때

순차패턴 B는 v_{j1}과 v_{j1}이 발생하는 시간 사이에 다른 벡터인 v_{n1}과 v_{n2}가 발생한다. 이 두 벡터가 만드는 전체 기간이 사용자가 정의한 d_{max} 범위 이내일 경우 다음에 발생되는 트렌젝션인 벡터 v_{j1}까지 확장하여 검색을 하여 패턴을 검색한다면 두 순차패턴 A,B는 같은 패턴을 만들고 있음을 알 수 있다. 따라서 순차패턴 B의 패턴 요소는 다음 3가지로 구분된다. $\{(v_{j1})\} \alpha \{(v_{n1}), (v_{n2}), (v_{j1})\} \alpha \{(v_{k1})\}$ 순차패턴 A와 비교시 v_{j1}의 발생에 대한 비교는 $\{(v_{n1}), (v_{n2}), (v_{j1})\}$의 발생이 d_{max} 범위이내인 경우 $v_{j1} \in \{(v_{n1}), (v_{n2}), (v_{j1})\}$이므로 같은 순차패턴으로 발견된다.

4.5 순차패턴의 탐색

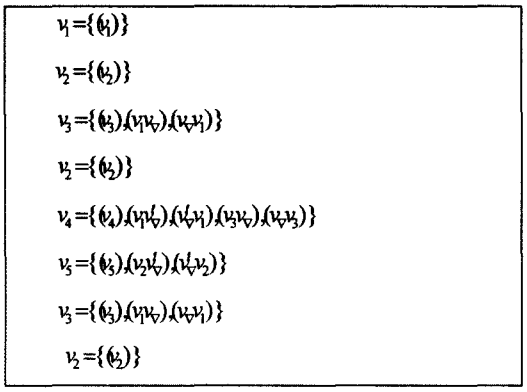
순차패턴을 찾기 위해 AprioriAll 알고리즘을 적용하여 패턴후보를 만드는 과정은 기존의 상품의 트랜젝션에서 순차패턴를 찾거나 사건에서 패턴을 찾는 과정의 방법과 동일하다. 그러나 패턴후보가 데이터베이스에 있는 지를 검색하고 지지도 값을 계산하는 방법은 기존의 방법과 다르다. 우리는 앞에서 유사한 순차패턴을 정의 하였다. 즉 각 벡터의 길이가 다르더라도 사용자가 사전에 지정한 d_{min}, d_{max}조건을 기초로 포함관계가 성립하는 경우는 같은 패턴으로 처리한다고 정의한 개념이 패턴후보와 시계열 데이터에서 동일한 패턴이 존재하는지 여부를 판단하는 기초가 된다. 따라서 충분한 지지도 값을 가지는 패턴후보를 찾는 과정은 2가지 과정으로 구분된다. 먼저 데이터베이스를 읽으면서 d_{max}의 범위를 가지는 데이터를 읽는다. 또한 각 패턴후보에 있는 각 벡터가 시계열데이터 벡터 각각에 대해 d_{min}을 만족하면서 포함할 수 있는지를 찾는 것이다. 이때 각 패턴후보의 처음과 마지막 벡터는 단지 기울기 값만을 비교하고 그 외의 벡터에 대해서는 d_{min}을 확인해야 한다. 이러한 과정은 <그림 10>과 같은 알고리즘으로 나타내었다.

```
/* Input : SP: Support, Dmin, Dmax
        L : Length of sequential Pattern
/* Generate Candidate & Find Sequential Patterns
L1 = {large 1-Vector Sequence}
For (k=2; L[k-1] ; k++) Do
  Ck = New Candidate Generated from L[k-1]
Do While Not EOF (Time-Series Vector Data)
  Foreach Candidate Ck do
    If (Candidate sequence contains Time-Series Vector List) then
      Check Relationship between vectors with Dmin
      Increment the count of all Candidates;
    Else Extend Current Time unit to Dmax Duration;
      If Candidate sequence contains then
        increment the count of all candidates;
Loop
End
```

<그림 10> 패턴검색 알고리즘

예를 들어 시계열데이타를 벡터로 변환한 결과가 다음과 같은 벡터의 순서를 가지고 있고 $S = \{V_1 \alpha V_2 \alpha V_3 \alpha V_2 \alpha V_4 \alpha V_5 \alpha V_3 \alpha V_2\}$ 또한 이들 벡터들 간에는 다음 <그림 11>과 같은 포함관계를 가지고 있으며, 최소 지지도 값은 2이며 $V_{\nabla} \le d_{min} < V'_{\nabla}$의 관계를 가진다고 할 때, 단, 기울기 값은 $V_1.\alpha = V_3.\alpha = V_4.\alpha$로 같고 또한 $V_2.\alpha = V_5.\alpha$일 경우, <그림 11>과 같이 된다.

<그림 11>의 포함관계 예제는 패턴후보를



<그림 11> 각 벡터의 포함관계 정보 예제

만들기 위해 최초 작업인 L_1을 찾는 과정에서도 고려가 되야 하는 작업이다. 그러나 우리는 벡터를 표현하는 값의 형태가 $v_i = (a_i, *, d_i)$와 같이 기울기와 기간 값을 가지고 있으므로 이와 같은 정보를 별도의 데이터베이스에 저장할 필요는 없이 벡터간의 유사성을 비교하는 순간에 간단히 각 벡터의 정보를 이용하여 계산이 가능하다. 따라서 L_1이 빈번히 발생하는 벡터를 찾는 경우 위의 예제에서는 벡터 V_1은 V_3, V_4에서도 같은 기울기 이면서 기간만 다른 경우이므로 L_1을 발견할 때는 <그림 11>의 포함관계에 따라 발견된 횟수(Count Value) 증가를 가져온다. 따라서 V_1의 지지도 값은 4가 되

Seq'	count
v_1	4
v_2	4
v_3	3
v_4	1
v_5	1

<그림 12> 최소 지지도 값을 만족하는 벡터

며 나머지 모든 벡터에 대해서도 벡터의 수가 1인 L_1을 발견할 때 지지도 값은 <그림 12>와 같다.

다음 과정으로는 빈번하게 발생된 벡터 즉 사용자가 지정한 최소 지지도 값을 만족하는 벡터 V_1, V_2, V_3만을 이용하여 이들을 상호 결합하는 과정을 통해 보다 긴 패턴을 발견하기 위해 패턴후보를 만든다. 다음 <그림 13>은 <그림 12>에서 발견된 벡터를 이용하여 벡터가 2개로 확장된 패턴후보와 각 패턴후보의 지지도 값을 계산한 결과를 보여준다.

패턴후보 V_1V_2의 경우 시계열 데이터에서 4번 발견되었는데 앞의 정의에 따라 벡터들 사이에 새로운 벡터가 존재하지 않기 때문에 기울기 값과 패턴 후보 보다 크거나 같은 기간을 가지는 모든 시계열 데이터는 패턴 후보를 포함하는 정보를 가지므로 지지도 값 증가를 가져온다.

2-sequence

Seq'	count
v_1v_2	4
v_1v_3	0
v_2v_1	3
v_2v_3	3
v_3v_1	0
v_3v_2	3

<그림 13> 패턴의 길이가 2인 패턴 후보와 지지도 값

위의 <그림 13>에서는 지지도 값 2를 만족하는 순차패턴은 4개로 다시 이들을 이용하여 패턴의 길이가 3인 패턴후보와 이들의 지지도

값을 전체 데이터베이스를 읽으면서 검색하는 과정을 계속한다.

3-sequence

Seq'	count
$v_1v_2v_1$	2
$v_1v_2v_3$	2
$v_2v_1v_2$	2
$v_2v_3v_2$	3
$v_3v_2v_1$	1
$v_3v_2v_3$	1

<그림 14> 패턴 길이 3인 패턴후보와 지지도 값

<그림 14>는 패턴길이 3인 패턴후보에 대한 지지도 값을 나타낸 것으로 $V_1V_2V_1$의 경우 다음 같은 과정으로 지지도 값을 계산한다.

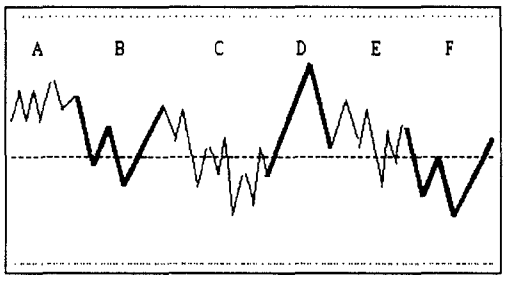
A. 시계열 데이터에서 처음 3개의 벡터를 읽는다. : $V_1V_2V_3$
B. 패턴후보 $V_1V_2V_1$의 정보를 가지고 있는 벡터로 다시 표현하면 다음과 같다. $V_1V_2V_3 = \{(V_1)(V_2)(V_1V_\nabla, V_\nabla V_1)\}$ 포함관계를 가진다. 따라서 $V_1V_2V_3$는 $V_1V_2V_1$의 정보를 가지고 있으므로 $V_1V_2V_1$의 발견 횟수를 1 증가 한다.
C. 다음 패턴후보 $V_1V_2V_3$를 시계열데이터 $V_1V_2V_3$와 비교한다. 같은 벡터이므로 마차가지로 패턴후보 $V_1V_2V_3$의 발견횟수를 1 증가한다.
D. 나머지 모든 패턴후보에 대해서도 위의 과정을 반복한다.
E. 마지막 패턴후보까지 패턴후보의 지지도계산을 완료한 후 시계열 데이터의 읽는 위치를 다음 벡터인 $V_2V_3V_2$가 되도록 이동한 후 위의 과정을 A 부터 다시 시작한다.

패턴후보 $V_1V_2V_1$의 검색 시 시계열데이터 $V_4V_5V_3$에서는 패턴후보 $V_1V_2V_1$를 만드는 각각의 벡터들은 발견되지만 연속된 순차패턴 $V_1V_2V_1$는 발견되지 않는다. 그것은 정의 4.4에 따른 것으로 벡터 $V_5 = \{(V'_\nabla V_2), (V_2V'_\nabla)\}$으로 표현되므로 앞의 벡터 V_4에 있는 것과 뒤의 벡터 V_3과 결합되는 가능한 방법의 패턴은 $V_\nabla V_1V'_\nabla V_2V_1V_\nabla$를 만들거나, $V_\nabla V_1V_2V'_\nabla V_1V_\nabla$를 만드는 순차패턴이 된다. 그런데 $V'_\nabla > d_{\min}$이므로 정의 4.4에 따라 $V_1V_2V'_\nabla V_1$는 패턴후보 $V_1V_2V_1$과 같지 않다. 따라서 마지막 과정으로 위의 길이 3인 패턴을 이용하여 패턴후보를 만들어 지지도 값을 계산하면 최종적으로 $V_1V_2V_3V_2$가 되는 길이 4인 순차패턴만이 발견된다.

4.6 패턴탐색 조건 적용

우리가 적용한 시계열데이터에서 순차패턴을 발견하는 트렌드 기반의 결과는 데이터베이스 내에서 발생 가능한 모든 순차패턴을 발견한다. 이로 인해 가장 빈번하게 발생되는 벡터를 찾는 과정에서 많은 벡터가 발견되므로 초기 패턴후보를 만드는 과정에 너무 많은 패턴후보 들이 발견된다. 이런 현상은 문제점이라기 보다는 모든 패턴을 발견하는 과정에 나타나는 현상일 뿐이다. 그러나 이러한 문제점도 순차패턴 요소인 벡터의 특징을 충분히 고려한다면 보다 빠르게 중요하고 의미 있는 패턴만을 발견할 수 있다. 즉 우리는 벡터가 가지는 정보를 시작점 값, 기울기 값, 벡터의 길이의 정보로 표현한다고 앞에서 언급하였

다. 따라서 사용자가 적용하는 응용분야에 따라 시작점의 최저 위치 값을 지정하거나 또는 기울기 값을 일정한 간격의 크기로 묶을 수 있고, 벡터의 길이가 가지는 최소값을 지정하여 보다 적은 패턴후보를 이용하여 순차패턴을 발견할 수 있다. 이와 같은 것은 기존의 윈도우 방식에서는 전체 데이터 값을 다루거나 또는 정해진 값 이상의 값만을 지정하는 방법 외에는 별도의 방법은 취할 수 없다.



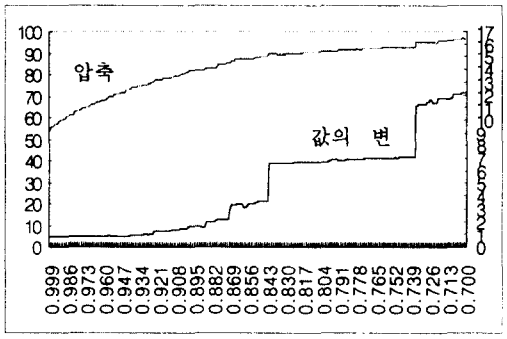
<그림 15> 패턴요소의 조건

위의 <그림 15>의 예제에서 시계열 데이터를 벡터로 변환하기 전에 점선이하의 값을 제거한다면 B와 F 구역에 있는 패턴은 원래 형상을 잘리어 표현되므로 인해 패턴발견이 곤란하나, 트렌드 방식의 경우 벡터의 기간 값을 보다 큰 것으로 제한하면 A,C,E 구역의 대부분의 패턴요소는 제거되어 패턴요소가 간단해진다.

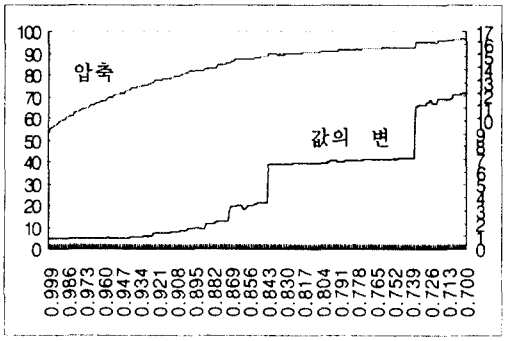
5. 실험 결과

5.1 상관계수와 데이터 값의 변화

우리는 Pearson의 상관계수 값을 변화시키면서 시계열데이터가 가지는 각각의 값이 벡터로 변화시킬 때 얼마나 많은 영향을 주는지를 분석하였다. <그림 16>과 <그림 17>은 약 100년간의 미국주식 데이터를 이용하여 분석한 결과를 그래프로 나타낸 것이다. 우리는 실험결과 상관계수 값은 적용대상 데이터의 종류에 차이는 있으나 데이터 압축률 값이 80%가 넘지 않는 범위의 상관계수를 적용하는 것이 데이터 값을 가장 잘 유지함을 알 수 있었다. 따라서 원시 데이터가 80% 압축이 될 수 있는 상관계수 값으로 0.95 이상의 값을 선택한다면 가장 적당할 것이다. 그러나 이것은 일반적인 사항일 뿐 응용분야의 분석하고자 하는 데이터 성격에



<그림 16> 상관계수와 데이터 값의 변화율과 압축률

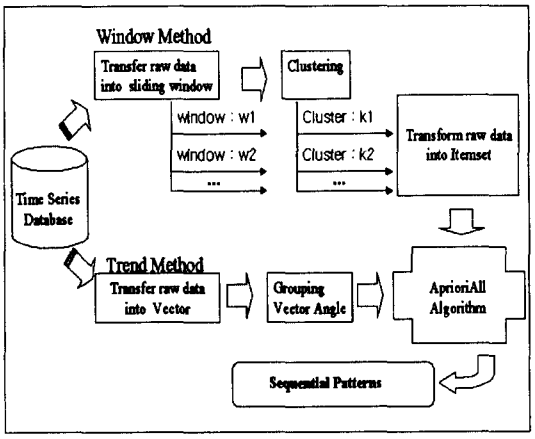


<그림 17> 그림 16을 벡터로 변환 후 결과 비교

따라 각기 달라진다. 만일 짧은 시간에서의 순차패턴을 분석한다면 단지 상관계수 값을 1로 설정하여도 무방하다. 이렇게 하면 벡터가 가지는 길이의 대부분은 2~10 범위 내에 발생될 확률이 높다. 그리고 보다 긴 경향성을 가지는 순차패턴을 분석한다면 0.90 까지 상관계수를 확장하면 된다.

5.2 성능 평가

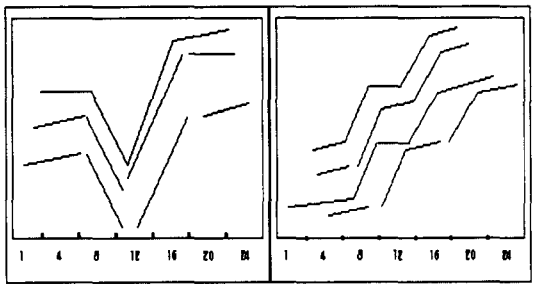
성능 평가를 위해 기존의 윈도우 방법과 트렌드 방법을 상호 단순 비교는 순차패턴 최종 결과물의 형태가 다름으로 곤란하다. 그것은 패턴대상을 만드는 과정의 차이점때문으로 기존 방법에서 각 데이터를 윈도우 크기로 패턴요소를 만들어 클러스터링을 하므로 인해 윈도우의 크기와 클러스터의 개수에 따라 전혀 다른 형태의 결과가 나타나기 때문이다. 따라서 성능평가를 할 수 있는 방법으로는 각 알고리즘이 가지는 특징을 구분하여 살펴볼 수 있다. <그림 18>은 상호알고리즘을 비교하고 있는데 기존방법은 윈도우 크기나 또는 클러스터의 종류가 바뀔 때 마다 반복한다.



<그림 18> Sliding Window와 Trend Method의 알고리즘 비교

순차패턴을 찾는 과정에서 가장 많은 시간적 비용이 소요되는 것은 AprioriAll 알고리즘을 수행하는 과정이다. 이 과정은 전체 시간의 거의 90% 이상을 차지한다. 따라서 기존의 방법과 같이 가능한 모든 순차패턴을 찾기 위해 다양한 크기의 윈도우와 클러스터를 적용하여 패턴요소를 만들고 이렇게 만들어진 패턴요소를 이용하여 다시 AprioriAll 알고리즘을 반복적으로 수행하므로 수행시간을 예측하기가 곤란하다. 그러나 우리 방법은 단지 상관계수나 벡터가 만드는 각도의 범위를 정하는 정도만 변수가 될 수 있으나, 상관계수 값의 변경은 순차패턴의 결과에 영향을 주지는 않고 단지 벡터의 수를 줄여주어 여러 개의 벡터가 만드는 결과를 단순화하여 표현할 뿐이다. 또한 각도를 묶는 범위의 지정도 순차패턴이 가지는 상승 또는 하강의 조합 형태를 변경시키지는 않는다.

우리는 기존의 윈도우방법과 트렌드 방법의 성능을 비교하기 위해 미국 다우존스주식 데이터(1896~1980)를 이용하여 패턴을 검색하였다.

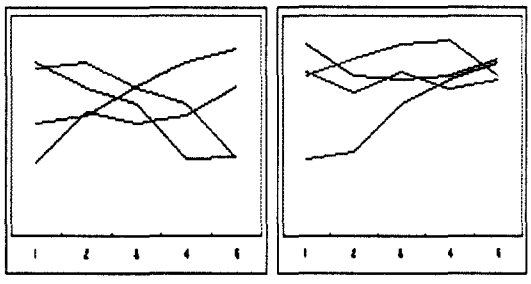


<그림 19> 트렌드 방법을 이용해 발견된 순차패턴

주어진 조건은 트렌드 방법에서는 벡터의 길

이가 3 이상인 벡터만을 고려하였고, 최대 기간이 10일 이내로 제하여 발생되는 모든 순차패턴을 발견하였다.

<그림 19>는 위의 조건에 따라 발견된 순차패턴 중 대표적인 두 가지를 보이고 있는데, 처음 앞의 그림은 3일 이상 주식 값의 변화가 없으면 4일 이상 하락을 한 후 다시 5일 이상 상승하는 패턴이다. 우리는 이 실험을 통해 693개의 패턴을 발견하였다. 발견된 패턴의 길이도 최대 5까지 확장된 벡터가 발견되었으며 수행시간은 33분이었다. 그러나 이동 윈도우를 이용한 실험에서는 <그림 20>과 같이 그 결과를 해독하는데 어려움이 있음을 발견하였다. 즉, 순차패턴으로 얻어진 결과의 질이 떨어지는 것이다. 이러한 결과가 얻어지는 이유는 기존의 클러스터링 기법에서 사용하는 패턴들간의 가까움의 정도(distance metric)가 단순한 거리에 기반한 것이기 때문이다.



<그림 20> 이동 윈도우를 이용하여 발견된 순차패턴

위의 <그림 20>은 클러스터의 수를 60으로 하고 윈도우 크기를 10으로 한 결과로 첫번째 그림을 보면 상승과 하강의 두 가지 전혀 다른 형태의 그래프가 같은 클러스터에 속해있어 실제 응용분야에서 사용자가 발견된 클러스터를 데이터와 비교할 때 명확하지 않다. 또한 실행시간에 있어서 윈도우 10과 클러스터 60만을 적용하여도 약 43분 소요되었고 윈도우를 확장하면서 계속 순차패턴을 찾기 때문에 변경된 모든 조건의 시간을 포함하여야 전체 수행시간이 되므로 벡터를 이용한 방법보다는 일반적으로 많은 시간이 소요됨을 알 수 있었다.

<그림 19>와 <그림 20>의 실험 결과로 우리가 제안하는 트렌드 방법이 기존의 이동 윈도우 기법보다 일반적으로 우수함을 주장할 수는 없지만, 트렌드 방법이 윈도우 기법보다 빠르게 동작하고, 좋은 순차패턴을 찾을 수 있는 경우가 있음을 보일 수 있었다. 찾아진 순차패턴의 질(quality)에 대한 정량적인 논의는 보다 연구가 필요할 것으로 보인다.

6. 결론 및 향후 연구방향

본 연구는 데이터베이스에서 수치 값의 형태로 구성된 순차패턴을 발견하는 방법을 제시하였다. 기존의 윈도우 방법에서 사용된 패턴요소를 찾는 방법은 패턴요소가 되는 각 윈도우에 대한 클러스터링 과정에서 클러스터의 수와 윈도우의 크기 등에 따라 다른 결과를 보이며 또한 만들어진 클러스터 내에 속하는 데이터를 분석하면 의미적인 해석이 곤란하여 실제로 응용분야에 적용하기에는 많은 문제점을 가지고 있다. 그러나 우리는 데이터가 시간에 따라 연관성이 있는 구간을 패턴요소로 보고 이들 트렌드라 부르는 패턴요소들 간에 순차패턴을 찾으므로 인해 보다 이해되기 쉽고 빠르게 순차패턴을 발견할 수 있는 방법을 제시하였다.

본 연구의 장점으로는 특히 응용분야 전문가

가 직접 패턴을 발견할 경우, 발견하고자 하는 패턴의 패턴요소인 벡터에 대하여 기울기나 벡터의 길이(시간)를 제한 조건으로 적용하므로 불필요한 패턴의 발견을 제거할 수 있고 보다 의미 있는 패턴만을 발견할 수 있다. 이와 같은 수치형태의 순차패턴 발견에 관한 연구는 제조설비에 설비의 고장관리나 조업현황을 감시하기위해 설치하는 많은 센서(Sensor)들이 만들어내는 패턴을 분석하여 정확한 고장의 위치를 진단하거나 또는 향후 발생되는 고장이나 조업상태의 분석 등 다양한 분야에 쓰일 수 있을 것으로 판단된다.

이후 연구로는 순차패턴이 만들어내는 변화의 경향성을 찾거나 또는 순차패턴의 주기적 규칙성을 찾는 것도 흥미로운 작업이 될 것이다. 또한 찾아진 순차패턴의 질을 판단하는 척도(measure)에 대한 연구가 필요할 것이다.

참고문헌

[1] 이원하, 최종국. 시계열 데이터의 성격과 예측모델의 예측력에 관한 연구, 한국지능정보시스템학회논문지, 제 4권 1호, 1998년 6월, 133-147.

[2] 박상현, 김상욱, 노웅기. 시퀀스 데이터베이스에서 시간왜곡 변환에 기반하는 서브시퀀스 탐색, 한국정보과학회 데이터베이스연구회 가을 학술발표논문집, 2001년 6월, 114-120.

[3] R. Agrawal, C. Faloutsos, A. Swami. Efficient Similarity Search In Sequence *Databases. Foundations of Data Organization and Algorithms, 4th International Conference,* (FODO'93)

[4] R. Agrawal, K.-P. Lin, H.S. Sawhney K. Sim. Fast Similarity Search in the Presence of Noise, Scaling , and Translation in Time-Series Databases. *In proceedings of the 21ST International Conference on Very Large Data Bases* (VLDB"95)

[5] R. Agrawal, G. Psaila, E.L. Wimmers. *Querying Shapes of Histories. In proceedings of the 21st International Conference on Very Large Data Bases* (VLDB"95)

[6] R. Agrawal, R.Srikant. *Mining Sequential Patterns. In 11TH International Conference on Data Engineering* (ICDE"95)

[7] G. Das, K.-I, Lin, H. Mannila . Rule Discovery from Time Series . In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (KDD"98)

[8] J. Eamonn , E. Keogh, M. Pazzani. Scaling up Dynamic Time Warping to Massive Datasets. *Principles and Practice of Knowledge Discovery in Databases* (PKDD99)

[9] C. Faloutsos, M. Ranganathan, Y. Manolopoulos. Fast Subsequence Matching in Time-Series Databases. *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data.*

[10] E. Keogh, M.Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proceedings of the 4rd International Conference on Knowledge Discovery and Data Mining* (KDD"98)

[11] E. Keogh, M. Pazzani. An Indexing Scheme for Fast Similarity Search in Large Time Series Database. *In Proceeding of the 11th International Conference on Scientific and Statistical Database Management* 1999

[12] H. Mannila, H. Toivonen. Discovering

generalized episodes using minimal occurrences. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (KDD"96)

[13] H. Mannila, H. Tovonen, A. Verkamo, Discovering frequent episodes in sequence. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining* (KDD"95)

[14] Y. S. Moon, K. Y. Whang, W. K. Loh, Duality-Based Subsequence Matching in Time-Series Databases. *In Proceedings of the Seventeenth International Conference on Data Engineering(ICDE2001)*, 263-272.

[15] R. Srikant, R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. *In 5th International Conference Extending Database Technology, Mar 1996*

Abstract

Trend-based Sequential Pattern Discovery from Time-Series Data

Young-Saeng Oh*, Dong-Ha Lee***
Do-Won Nam**, Jeon-Young Lee**

Sequential discovery from time series data has mainly concerned about events or item sets. Recently, the research has stated to applied to the numerical data. An example is sensor information generated by checking a machine state. The numerical data hardly have the same values while making patterns. So, it is important to extract suitable number of pattern features, which can be transformed to events or item sets and be applied to sequential pattern mining tasks. The popular methods to extract the patterns are sliding window and clustering. The results of these methods are sensitive to window size or clustering parameters; that makes users to apply data mining task repeatedly and to interpret the results.

This paper suggests the method to retrieve pattern features making numerical data into vector of an angle and a magnitude. The retrieved pattern features using this method make the result easy to understand and sequential patterns finding fast. We define an inclusion relation among pattern features using angles and magnitudes of vectors. Using this relation, we can find sequential patterns faster than other methods, which use all data by reducing the data size.

Key words: Time-series Data, Data Mining, Sequential Pattern, AprioriAll Algorithm, Numerical Data

* Dept. of Computer and Communications Engineering, POSTECH
** Division of Electrical and Computer Engineering, POSTECH
*** POSTECH Information Research Laboratories, POSTECH