Item Dependency Map을 기반으로 한 개인화된 추천기법

염 선 희° , 조 동 섭
이화여자대학교 컴퓨터학과

Personalized Recommendation based on Item Dependency Map

Youm Sun Hee° , Cho Dong Sub
Dept. of Computer Sciendce and Engineering, Ehwa Womans University

Abstract - 데이터 마이닝을 통해 우리는 숨겨진 지식, 예상되지 않았던 경향 그리고 새로운 법칙들을 방대한 데이터에서 이끌어내고자 한다. 본 논문에서 우리는 사용자의 구매 패턴을 발견하여 사용자가 원하는 상품을 미리 예측하여 추천하는 알고리즘을 소개하고자 한다.

제안하고 있는 item dependency map은 구매된 상품간의 관계를 수식화 하여 행렬의 형태로 표현한 것이다. Item dependency map의 값은 사용자가 A라는 상품을 구매한 후 B상품을 살 확률이다. 이런 정보를 가지고 있는 item dependency map은 홉필드 네트웍(Hopfield network)에서 연상을 위한 패턴 값으로 적용된다. 홉필드 네트웍은 각 노드사이의 연결가중치에 기억하고자 하는 것들을 연상시킨 뒤 어떤 입력을 통해서 전체 네트워크가 어떤 평형상태에 도달하는 방식으로 작동되는 신경망 중의 하나이다. 홉필드 네트웍의 특징 중의 하나는 부분 정보로부터 전체 정보를 추출할 수 있는 것이다. 이러한 특징을 가지고 사용자들의 일반적인 구매패턴을 일부 정보만 가지고 예측할 수 있다. Item dependency map은 홉필드 네트웍에서 사용자들의 그룹별 패턴을 학습하는데 사용된다. 따라서 item dependency map이 얼마나 사용자 구매패턴에 대한 정보를 가지고 있는지에 따라 그 결과가 결정되는 것이다. 본 논문은 정확한 item dependency map을 계산해 내는 알고리즘을 주로 논의하겠다.

1. 서 론

많은 기업들은 방대한 양의 데이터를 수집하고 모인 데이터 들로부터 숨겨진 지식, 예견되지 않았던 패턴 그리고 새로운 법칙들을 이끌어내고자 한다. 데이터마이닝은 이러한 기업들의 욕구에 부합하는 가장 적절한 수단이 되어 왔다. 데이터마이닝이란 대량의 데이터로부터 쉽게 드러나지 않는 유용한 정보들을 추출하는 과정을 말한다. 여기서 정보는 묵시적이고 잘 알려져 있지 않지만 잠재적으로 활용가치가 있는 정보를 말한다.

특히 전자상거래에서 사용자들의 선호도, 관심, 구매경험과 같은 자료를 기초로 원하는 정보를 자동으로 제공하는 것은 중요한 문제이다. 이런 경우 사용자의 지속적인 이용을 이끌어 낼 수 있고 사용자 역시 유용한 정보를 쉽게 얻을 수 있다.

본 논문에서는 사용자의 아이템 구매패턴을 분석하여 관심이 있을 아이템을 예측하여 추천하는 시스템을 제안하고자 한다. 데이터마이닝 기법에는 연관규칙, 분류규칙, 클러스터링, 유사성 탐색, 순서 패턴, 신경망, 결정트리 등등이 있다. 그 중에서 본 논문은 item dependency map이라는 새로운 방법을 제안하고 이를 홉필드 네트웍이라는 신경망에 적용하여 사용자의 구매패턴을 예측하고 또한 예측된 결과를 이용하여 사용자에게 새로운 정보를 제공하고자 한다.

Item dependency map은 사용자가 구매한 상품에 대해서 사용자가 상품 A를 산 후 다시 B를 구매할 확률에 대하여 수식으로 표현한 행렬이다. 이는 나중에 사용자의 일부 구매 패턴만으로 사용자의 전체 구매패턴을 예측하는데 사용된다.

이 논문은 2001년도 두뇌한국21사업에 의하여 지원되었음.

2. 본 론

2.1 관련연구

2.1.1 홉필드 네트웍

홉필드 네트웍은 각 노드사이의 연결가중치에 기억하고자 하는 것들을 연상시킨 뒤 어떤 입력을 통해서 전체 네트워크가 어떤 평형상태에 도달하는 방식으로 작동되는 신경망 중의 하나이다.

홉필드 네트웍의 기본구조는 입력과 연결 가중치를 곱한 값들을 모두 더해서 적당한 임계함수를 통해 출력하는 노드들이 여러 개 있고 이들이 상호 연결되어 있는 구조이다. 다른 신경망과의 차이는 출력 값이 다시 입력되는 구조로 이루어진다. 즉 일부 다른 신경망들이 단일방향으로 작동하는 정적인 반면에, 홉필드 네트웍은 시간에 따라서 내부상태가 동적으로 변화된다.

각 노드사이의 연결 가중치에 기억하고자 하는 것들을 연상시킨 후에 어떠한 입력에 대하여 출력을 반복적으로 구하면서 그 값이 더 이상 변화되지 않을 때 중단하게 된다.

2.1.2 Sequential Pattern Mining

바코드의 출현으로 소매 조직들은 방대한 판매 데이터들을 데이터베이스에 저장할 수 있게 되었고 이 데이터들은 트랜잭션(Transaction) 날짜와 판매된 아이템으로 구성되어 있다. 소매 조직들은 바코드 등을 통해 받은 정보를 시간의 순서에 따라 저장하게 되었다. 여러 기업들에서는 이러한 데이터베이스를 분석하여 의사결정의 중요한 요소로 활용하고 있다.

Sequential pattern mining은 Agrawal[2]에 의해 처음 제기되었다. 이 문제는 데이터를 시간적으로 분석한다는 의미에서 연관 규칙(Association rule)과는 차이가 있다. 각각 연속 데이터는 항목들을 포함하는 트랜재션들로 이루어지고 각 트랜재션은 트랜재션 시간들로 구성되어 있다. 결론적으로 사용자가 최소 지지도(Minimum support)를 만족하는 연속 패턴을 찾아내는 문제라고 볼 수 있다. 여기에서 최소 지지도는 패턴을 포함하는 연속 데이터의 백분율로 정의한다.

Sequential pattern mining은 소매산업과 우편을 이용한 마케팅, 부가 세일, 고객 만족 등에서 시작되었지만 다른 과학분야나 경영 분야에 적용되고 있다.

2.2 Item Dependency Map

2.2.1 문제 기술

Item dependency map은 앞에서도 잠시 설명했듯이 사용자가 구매한 상품들 간의 연관성을 행렬로 표현한 것이다. 이 장에서는 item dependency map을 결

정하는 방법에 대해서 설명하겠는데 사용자의 트랜재션을 기본 데이터로 한다.

고객 아이템 구매 트랜재션이 있다고 하자. 트랜재션은 사용자 아이디, 사용자 그룹 아이디, 트랜재션 시간, 구매 된 아이템으로 구성되어 있다. 사용자 그룹 아이디는 직업이나 연령 대와 같이 비슷한 성향을 가진 사람들을 나타내는 것이다. 예를 들어 그룹이 나이로 나누어진 경우 10대, 20대, 30대 등등으로 나누어질 수 있다. 이렇게 하는 이유는 각 그룹의 item dependency map이 하나의 패턴이 되어서 나중에 사용자의 일부 패턴이 들어갔을 때 어느 그룹에 속하는지를 예측할 수 있게 한다. 아이템셋(Itemset)은 각 사용자의 구매된 구매 된 아이템들의 집합이고 시간순서대로 되어 있다.

이제는 item dependency map을 만드는 과정을 보도록 하자.

Item dependency map을 만드는 과정은 chain rule과 mapping rule 두 가지 단계로 나누어진다. Chain rule은 pre item dependency map을 완성하는 과정이고 mapping rule은 chain rule에서 만들어진 pre item dependency map을 이용하여 최종적인 item dependency map을 만드는 과정이다.

Chain rule에서는 제품과 제품간의 판매 연관성의 값을 계산하는 과정이다. 즉 상품 A를 사고 상품 B를 살 확률이 다른 상품간의 연관성에 비해 어느 정도인가를 계산하는 것이다. Pre item dependency map은 이러한 값들로 이루어진 행렬이다. Mapping rule에서는 pre item dependency map의 값을 보고 이 값이 support보다 큰 경우 가치있는 값으로 결정하게 된다. 본 논문에서 사용하는 support는 값들의 평균과 값들의 순위 두 가지로 정의된다. 이러한 값들에 대해서는 앞으로 자세히 설명된다.

2.2.2 전처리

예를 보면서 item dependency map을 만들기 전에 아이템셋을 만드는 과정을 보자.

표1이 다음과 같이 주어져 있다고 하자.

표 1 사용자 그룹 아이디, 사용자 아이디, 트랜잭션 시간으로 정렬된 데이터 베이스

사용자그룹 아이디	사용자 아이디	트랜잭션 시간	구매된 아이템
1	1	March 3	111
1	1	March 15	112
1	2	March 1	114
1	2	April 4	112
1	2	May 15	116
2	3	February 27	119
2	3	April 13	120
2	3	April 24	111
2	4	March 5	123
2	4	March 19	121

그룹별로 서로 다른 item dependency map을 만들어야 하므로 우선 그룹별로 나눈 후 사용자의 아이템셋을 시간을 기준으로 완성한다.

결과는 표 2, 3과 같다.

표 2 그룹 1의 아이템시퀀스

사용자 ID	Itemsets
1	(111,112)
2	(114, 112, 116)

표 3 그룹 2의 아이템 시퀀스

사용자 ID	Itemsets
3	(119, 120, 111)
4	(123, 121)

2.2.3 실행 단계

만들어진 각 그룹의 아이템 시퀀스를 각각 item dependency map을 만들기 위해 우선 pre item dependency map으로 변환해야 한다. 이 과정을 chain rule이라고 한다.

Item의 수를 m 이라고 가정하자. 모든 전처리 과정이 끝나면 m*m의 행렬이 만들어질 것이다. 각 아이템에 1에서 m까지 일련번호를 준다.

$$\begin{bmatrix} (1,1) & . & . & . \\ . & . & . & . \\ . & . & (i,j) & . \\ . & . & . & (m,m) \end{bmatrix}$$

K번째 그룹의 pre item dependency map이 위와 같을 때 (i, j)를 결정하는 방법을 두 가지로 제안하고자 한다.

Chain Rule 1:

K그룹의 각 아이템셋을 따라 가면서 아이템 i를 산 후에 j를 산 경우 (i, j)의 값을 1 증가시킨다. 그리고 나서 k그룹 사용자가 아이템 i를 구매한 전체 개수로 나누어 준다. 이를 모든 아이템셋을 모두 다 할 때까지 한다. 각각 그룹의 pre item dependency map 을 완성한다.

알고리즘은 다음과 같다.

```
for(i=0; i<# of itemset in each sequence item;
i++){
  item = 1;
  while(until all items are traversed in each
  itemset){
      temp_pt1 = transaction[i][item];
                        //the item of ith itemset
      temp_pt2 = transaction[i][item+1];
      pre_Pattern[group_id][temp_pt1][temp_pt2]
     =pre_Pattern[group_id][temp_pt1][temp_pt2]+1;
    }
item ++;
}
```

Ex) 아이템셋이 각각 (111, 113, 115, 112), (111, 115, 112)이고 111,112,113,115의 일련번호가 각각 1,2,3,5 일 때(1,3) = 1,(3,5) = 1,(5,2) = 2,(1,5) = 1이 된다.

Chain Rule 2:

Chain rule 1과 달리 여러 단계를 거친 후에 구매된 아이템의 연관성을 고려하여 값을 증가시킨다. 거쳐진 단계가 여러 번일 경우에 가중치($0<a\leq1$)를 낮추어서 그 연관성이 비교적 작다고 판단한다.

알고리즘은 다음과 같다.

```
for(i=0; i<# of itemset in each sequence item; I++){
   item = 1;
   while(until all items are traversed in each itemset){
      temp_pt1 = itemset[i][item];
      temp_pt2 = itemset[i][item+1];
      pre_Pattern[group_id][temp_pt1][temp_pt2]
     =pre_Pattern[group_id][temp_pt1][temp_pt2]+1;
   }
   for(j=item-2; j>0; j--){
      value = value * a;
      temp_pt1 = itemset[i][j];
      temp_pt2 = itemset[i][item];
      pre_Pattern[group_id][temp_pt1][temp_pt2]
=pre_Pattern[group_id][temp_pt1][temp_pt2]+value;
   }
item ++;
}
```

예를 보면서 보자.

Ex) 아이템셋은 위의 예와 같다고 하고 1-1의 경우에서 계산되지 않았던 (3, 2)와 (1,2)가 계산되고 (1,5)는 값이 변하게 된다.

(1,2)의 경우 첫 번째 아이템셋에서 두 번의 아이템을 거쳐서 구매되었으므로 a*a의 값이 더해지고, 두 번째 아이템셋에서는 한 번의 아이템을 거쳐서 구매되었으므로 a의 값이 더해지게 된다. 다른 경우도 마찬가지이다. 따라서 각각 다음과 같은 값을 가지게 된다.

$$(3,2) = a,\ (1,2) = a^*a + a,\ (1,5) = a + 1$$

이 후 계산은 1과 같다.

Chain Rule 1 또는 Chain Rule 2에서 만들어진 pre item dependency map이 흡필드 네트웍에서 패턴이 될 수 있기 위해서는 -1/1 또는 0/1의 값으로 바뀌어야 한다. 앞 단계에서 나온 (i, j)의 값을 두 가지 (-1/1 or 0/1) 중에서 한 가지로 결정하기 위해서는 기준이 필요하다. 그 기준값을 앞에서 얘기한 support라고 하자. 그래서 support 값보다 크거나 같으면 1, 아니면 -1/0으로 한다. 여기에서도 support 값 정하는 방법을 두 가지 제안한다.

Mapping Rule 1:

그룹에 상관없이 전체 아이템에 대한 평균값을 기준으로 삼는 것이다. 전체 사용자가 아이템 i를 산 후 아이템 j를 산 경우 (i, j)의 값을 1 증가시킨 후 전체 사용자가 아이템 i를 산 회수로 나누어준 하나의 total item dependency map을 계산한다. 각 (i,j)의 기준값은 이 map의 (i,j)값이 된다.

Mapping Rule 2:

각 그룹의 (i, j)의 값에 각각 순위를 매긴다. 그리고 나서 상위 R등수 이상의 값에 대하여는 1, 그렇지 않은 경우는 -1/0으로 한다. 이 방법은 3의 경우 그룹이 작을 때 평균값이 전체 그룹에 대한 기준 값이 되기에는 무리가 있음을 감안한 방법이다.

2.3 성능 평가

본 논문에서 제안하고 있는 알고리즘의 성능을 평가하기 위해 가상의 사용자 트랜잭션을 생성했다. 그리고 제안한 각각의 rule을 조합하여 모두 4가지 경우에 대하여 실험을 해 보았다.

가상의 트랜잭션에서 그룹별 아이템 시퀀스를 만들고 이를 이용하여 item dependency map을 만들었다. 그리고 나서 흡필드 네트웍의 연상을 위한 패턴으로 사용하였다. 성능을 평가하기 위해 사용자의 아이템셋 일부를 다시 사용했다. 그리하여 A 그룹 사용자의 아이템셋이 입력되었을 때 결과가 A그룹으로 나온다면 이 알고리즘은 정확한 예측을 하는 것이라고 평가 할 수 있다.

실험 결과는 각 단계의 두 번째 방법에 대한 조합에서 가장 좋은 성능을 보였다. 고객이 구매한 아이템 간의 시간에 가중치(a)를 주고 나중에 1/-1을 결정할 때 평균값보다는 순위를 매긴 것이 좀 더 성능이 좋았다.

이는 그룹의 수가 적기 때문에 평균이 그 대표 값이 될 수 없기 때문이고 또 시간을 고려한 방법 역시 관련 있는 여러 가지 아이템이 한꺼번에 구매되는 것에 대해서 고려해서 추천하는 것이 좀더 효과적임을 보여주고 있다.

3. 결 론

본 논문에서는 구매된 아이템간의 관계를 표현하는 item dependency map을 소개했다. 이 알고리즘은 사용자가 아직 구매하지는 않았지만 관심이 있거나 구매할 가능성이 있는 아이템을 예측하여 추천하는 것이다. Item dependency map은 흡필드 네트웍에서 연상 작용을 위한 패턴으로 사용되어 나중에 사용자 정보가 입력되었을 때 사용자가 가지는 패턴을 추측하여 사용자에게 원하는 정보를 정확하게 줄 수 있도록 해 준다.그러나 본 논문에서는 좀더 많은 성능 평가를 통해 a와 R값의 최적치를 결정하고자 한다. 또한 대량의 데이터에 대해서도 알고리즘을 적용해 보고 그 성능을 확인할 것이다.앞으로 지속적인 성능평가를 통해 알고리즘을 발전시켜서 최적의 방법을 제시할 것이다. 또한 실제 데이터를 적용해서 결과를 확인하고 실제 웹사이트에 적용해 보고자 한다.

[참 고 문 헌]

[1] Tarun Khanna, "Foundations of Neural Networks," Addison-Wesley Publishing Company, 1990.

[2] Rakesh Agrawal, Ramakrishnan Srikant, "Mining Sequential Patterns," In Proc. of the 11th Int'l Conference on Data Engineering, Taipei, Taiwan, March 1995.

[3] Mehmet M. Dalkilic, Edward L. Roberston, "Information dependencies," Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 245 ? 253 2000.

[4] Rosa Meo, "Theory of dependence values," ACM Trans. Database Syst. Pp. 380 ? 406, Sep. 2000.