

Chap 5. Knowledge-Based Recommender Systems

▼ Constraint-Based Recommender Systems

- Constraint-based recommender systems는 사용자가 제약조건을 걸 수 있게 한다.
- 고객의 조건 예시:
결혼 여부(카테고리), 가족 구성원 수(수치), 도시인지 아닌지(binary), 최소 방 수(수치), 최대 방 수(수치), 최대 예산(수치)
- 이러한 조건은 고객의 속성을 더 잘 나타냄.
- Constraint-based recommender system의 3가지 primary types
 - 첫번째 input class는 user의 inherent properties와 상품에 대한 구체적인 요구를 나타낸다.
다른 user가 같은 요구를 적용했을 경우 같은 결과가 반환된다. 이전 이력 데이터 기반으로 지속적으로 개인 맞춤을 해준다는 점에서 다른 추천시스템 타입과 다르다.
 - 두번째 input class는 고객의 attributes와 requirements를 상품의 attributes에 mapping 시킨다.
mapping은 직접적 또는 간접적 둘다 가능하다.

Directly: 고객의 요구사항을 상품의 엄격한 요구사항에 연결

$Suburban-or-rural=Suburban \Rightarrow Locality= \langle List\ of\ relevant\ localities \rangle$
 $Min-Bedrooms \geq 3 \Rightarrow Price \geq 100,000$

Indirectly: 고객의 요구사항을 일반적으로 예상되는 상품의 요구사항에 연결

$Family-Size \geq 5 \Rightarrow Min-Bedrooms \geq 3$
 $Family-Size \geq 5 \Rightarrow Min-Bathrooms \geq 2$

- 세번째로, 상품의 Catalog는 상품과 비슷한 속성의 모든 상품에 대한 리스트를 포함한다.

- Returning Relevant Results

- 관련 결과를 반환하는 문제는 카탈로그의 각 항목을 속성에 대한 제약으로 보고, 카탈로그를 **disjunctive Normal** 형식으로 표현함으로써 제약 만족 문제의 한 예라고 할 수 있다.

제약 조건들은 카탈로그의 데이터 필터링 작업으로 축소될 수 있다.

- 사용자 인터페이스에서 고객이 지정한 각 요구 사항(또는 개인 속성)에 대해 지식 기반의 규칙 선행 조건과 일치하는지 확인한다.
그러한 일치가 존재하는 경우 해당 규칙의 결과가 유효한 선택 조건으로 처리된다.
- 이러한 확장된 요구사항은 데이터베이스의 AND 쿼리로 구축할 수 있다.

$$(Bedrooms \geq 3) \wedge (Bathrooms \geq 2) \wedge (Price \geq 100,000) \wedge (ZIP\ Code = 10547)$$

- 이러한 Selection query는 user 요구사항과 관련있는 카탈로그의 인스턴스를 검색한다.

▼ Case-Based Recommenders

사례 기반 추천에서, 유사 행렬을 특정 타겟 (또는 사례)와 유사한 예시를 검색할 때 사용한다.

Constraint-based system과 달리 엄격한 조건 (no hard constraints)이 없다.

유사 함수는 특정 타겟과 가장 유사한 것을 검색하기 때문에 constraint-based와 달리 empty sets을 검색하는 문제는 사례 기반 추천에서 문제가 되지 않는다.

초기의 사례 기반 추천은 적절한 solution을 찾을 때까지 사용자의 요구사항을 반복적으로 수정하는 것을 허용했다.


후에 Critiquing이라는 방법이 개발되었는데 user들이 하나 이상의 검색 결과를 선택하고 다음과 같이 추가 형식을 지정할 수 있다.

“Give me more items like X, but they are different in attribute(s) Y according to guidance Z.”

Critiquing의 main goal은 아이템 공간의 대화형 브라우징을 지원하는 것이다. 사용자는 검색된 예시를 통해 사용할 수 있는 추가 옵션을 점점 알게 된다.

EXAMPLE OF HYPOTHETICAL CASE-BASED RECOMMENDATION
INTERFACE FOR HOME BUYING (critique-example.com)

[ENTRY POINT]



I WOULD LIKE TO BUY A HOUSE SIMILAR TO ONE WITH THE FOLLOWING FEATURES:

NUMBER OF BR

NUMBER OF BATH

HOME STYLE

PRICE RANGE

ZIP CODE

SUBMIT SEARCH

I WOULD LIKE TO BUY AN HOUSE JUST LIKE THE ONE AT THE FOLLOWING ADDRESS:

812 SCENIC DRIVE

MOHEGAN LAKE

NY

SUBMIT SEARCH

위 이미지는 가상의 집을 구매할 때 고객이 구체화할 수 있는 두 가지 방법을 나타낸 것이다. 위쪽 부분은 타겟의 feature들을 구체적으로 나타낸 부분이고 아래 부분은 타겟의 주소를 나타낸다. 후자의 방식은 feature들을 구체적으로 고르기 힘들 때 사용하기 좋다.

이 시스템은 결과를 검색할 때 similarity 또는 utility function과 함께 conjunction 쿼리 (AND) 방식을 사용한다.

- 사례 기반 추천 시스템을 효과적으로 동작하기 위해 2가지 중요한 측면이 효과적으로 설계되어야 한다.
 - Similarity metrics : 효과적으로 설계된 similarity metrics은 사례 기반 시스템에서 관련 결과를 검색할 때 굉장히 중요하다. 시스템이 효과적으로 작동하려면 다양한 attributes의 중요성이 similarity 함수 내에 적절하게 통합되어 있어야 한다.
 - Critiquing methods: 다양한 critiquing 방식은 다양한 탐색 목표를 지원할 수 있게 한다.
- Similarity Metrics
 - Similarity metrics의 적절한 설계는 유의미한 검색을 위해 필수적이다. 일반적으로 parameter가 domain 전문가에 의해 설정되거나 학습 프로세스에 의해 조정될 수 있는 폐쇄형 similarity 함수를 개발하는 것이 바람직하다.

$$f(\overline{T}, \overline{X}) = \frac{\sum_{i \in S} w_i \cdot \text{Sim}(t_i, x_i)}{\sum_{i \in S} w_i}$$

similarity function f

d 개의 attributes가 있는 product가 있을 때, X 와 T 는 각각 부분적으로 지정될 수 있는 d 차원벡터

T : target

$\text{Sim}(t, x)$: t 와 x 의 유사도

w : weight

이러한 attribute 들은 양적 또는 범주적일 수 있고, 이는 그러한 시스템의 이질성과 복잡성을 가중시킨다. 또한 속성은 높다 낮다의 관점에서 대칭일 수도 비대칭일 수도 있다.

예를 들어 집 구매 예시에서의 가격 속성을 생각하면, product 가격이 target price보다 낮은 가격을 반환하면 높을 때보다 더 쉽게 접근 가능해진다. 정확한 비대칭 수준은 속성마다 다를 수 있다.

user가 원하는 값이 target 값과 정확히 일치할 경우 완전히 대칭을 이룰 수도 있다.

$$\text{Sim}(t_i, x_i) = 1 - \frac{|t_i - x_i|}{\max_i - \min_i}$$

\max_i : i 속성이 가질 수 있는 최댓값

\min_i : i 속성이 가질 수 있는 최솟값

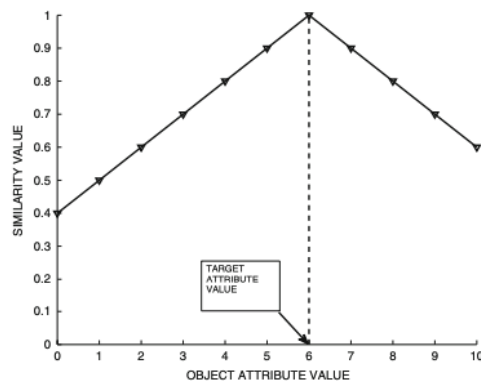
표준 편차 (과거 데이터에서의)를 사용한 similarity function:

$$\text{Sim}(t_i, x_i) = \max \left\{ 0, 1 - \frac{|t_i - x_i|}{3 \cdot \sigma_i} \right\}$$

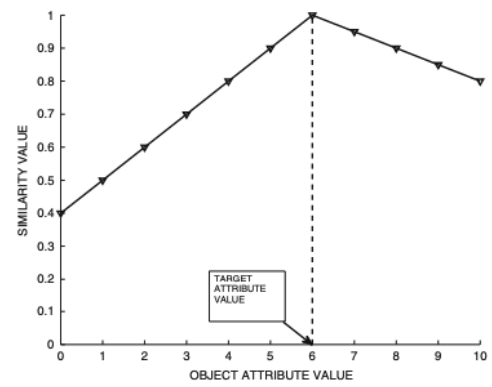
Symmetric metric 사례에서는 similarity가 두 개의 속성 간의 차이로 정의된다. 비대칭 속성의 경우, 대상 속성 값이 작은지 큰지에 따라 추가적인 비대칭 reward를 추가할 수 있다.

$$Sim(t_i, x_i) = 1 - \frac{|t_i - x_i|}{max_i - min_i} + \underbrace{\alpha_i \cdot I(x_i > t_i) \cdot \frac{|t_i - x_i|}{max_i - min_i}}_{\text{Asymmetric reward}}$$

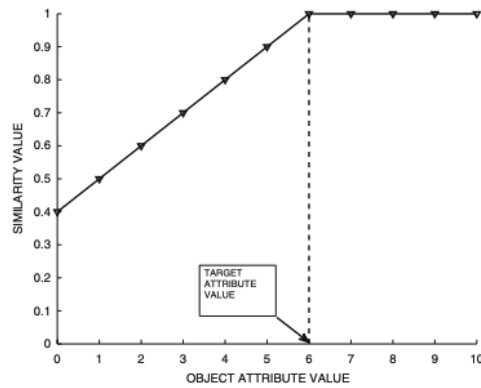
$$Sim(t_i, x_i) = 1 - \frac{|t_i - x_i|}{max_i - min_i} + \underbrace{\alpha_i \cdot I(x_i < t_i) \cdot \frac{|t_i - x_i|}{max_i - min_i}}_{\text{Asymmetric reward}}$$



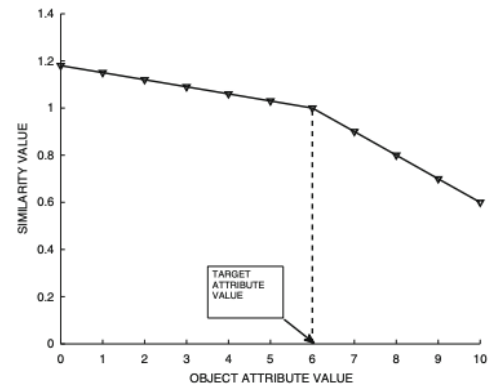
(a) Symmetric ($\alpha_i = 0$)
(penalty by absolute distance)



(b) Asymmetric ($\alpha_i = 0.5$)
(milder penalty for overshooting)



(a) Asymmetric ($\alpha_i = 1.0$)
(no penalty for overshooting)



(b) Asymmetric ($\alpha_i = 1.3$)
(less is always better)

Figure 5.6: Examples of different types of symmetric and asymmetric similarity

◦ Incorporating Diversity in Similarity Computation

사례 기반 시스템에서는 대부분 비슷한 결과가 반환되는데 이러한 다양성의 부족은 사용자가 얻은 결과가 마음에 들지 않을 때 문제가 된다.

Bounded greedy selection strategy라는 효과적인 접근 방식이 있다. 이 방식은 대상과 유사한 사례로 시작하여 다양한 인스턴스를 점진적으로 구축한다.

similarity 함수는 항상 (0,1) 사이의 값으로 결정되기 때문에 diversity 함수 $D(X,Y)$ 는 X와 Y의 거리로 볼 수 있다.

$$D(\bar{X}, \bar{Y}) = 1 - f(\bar{X}, \bar{Y})$$

후보 X와 현재 선택된 집합 R 간의 평균 Diversity는 X와 R의 평균 다양성으로 정의된다.

$$D^{avg}(\bar{X}, R) = \frac{\sum_{\bar{Y} \in R} D(\bar{X}, \bar{Y})}{|R|}$$

Target T에 대해, 전반적인 quality 함수 $Q(T, X, R)$ 은 다음과 같이 계산된다.

$$Q(\bar{T}, \bar{X}, R) = f(\bar{T}, \bar{X}) \cdot D^{avg}(\bar{X}, R)$$

가장 좋은 quality의 경우 X는 집합 R의 카디널리티가 k (diverse 수)가 될때 까지 추가된다.

- Critiquing Methods

Critique는 사용자가 초기 쿼리에서 요구 사항을 정확하게 명시할 수 없다는 사실에서 시작한다.

- Simple Critiques

- simple critique에서, 사용자는 features 중 하나를 수정한다.

- Directional critique: 특정 속성을 늘리거나 줄임

장점: 사용자가 속성의 정확한 값을 알지 못할 때 사용된다.

단순한 대화 스타일이어서 사용자에게 더 직관적이다.

단점: product에 수정되어야 하는 속성이 많을 때 긴 chain으로 이어질 수 있다.

하나의 속성이 수정되면, 자동적으로 다른 속성들이 수정될 필요가 생긴다.

수정 전의 상태로 돌아갈 방법이 없다.

- Compound Critiques

- 추천 cycle의 길이를 줄이기 위해 개발되었다.
- 한 cycle에서 여러 속성을 수정할 수 있다.
- 사용자가 새 쿼리를 실행하거나 이전 쿼리에서 검색 결과를 제거하기 위해 여러 속성을 수정할 수 있다는 장점이 있다.

- Dynamic Critiques

- main goal: 검색된 결과에 대해 데이터 마이닝을 사용하여 가장 유익한 탐색 방법을 결정하고 사용자에게 제시
- 현재 검색된 결과를 기반으로 가장 관련성 높은 가능성의 하위 집합만 표시된다. 따라서 dynamic critique는 사용자에게 검색 과정에서의 더 나은 가이드를 제공하도록 설계되었다.
- 중요한 측면은 제품 기능 수정의 빈번한 조합을 찾는 것

▼ Persistent Personalization in Knowledge-Based Systems

- 지식 기반 시스템에서 사용자 session 데이터를 사용할 수 있을 때 개인화가 가능하다.
- 다양한 속성에 대한 utility, similarity 함수의 학습은 constraint-based 와 case-based 시스템 모두 개인화시킬 수 있다.
- 제약 조건 제안과정은 사용자의 session을 많이 사용할 수 있는 경우 개인화시킬 수 있다.
- Dynamic critique는 사용자로부터 관련 패턴을 결정하기 위해 충분한 데이터를 얻을 수 있는 경우 개인화시킬 수 있다.
사용자의 session과 비슷한 session을 mining 과정에 포함시켜서 추천의 collaborative power를 증대시킬 수 있다.
- 개인화를 함에 있어서 가장 큰 challenge는 특정 사용자로부터 충분히 많은 session data를 얻지 못한다는 것이다.
- 지식 기반 시스템은 본질적으로 복잡한 도메인 공간에서 고도로 맞춤화된 항목을 위해 설계되었고 이는 개인화의 수준이 일반적으로 제한되는 이유이다.