DATA MINING 101

Somewhere in between Statistics and AI

Contents

1. K-Nearest Neighbors Algorithm

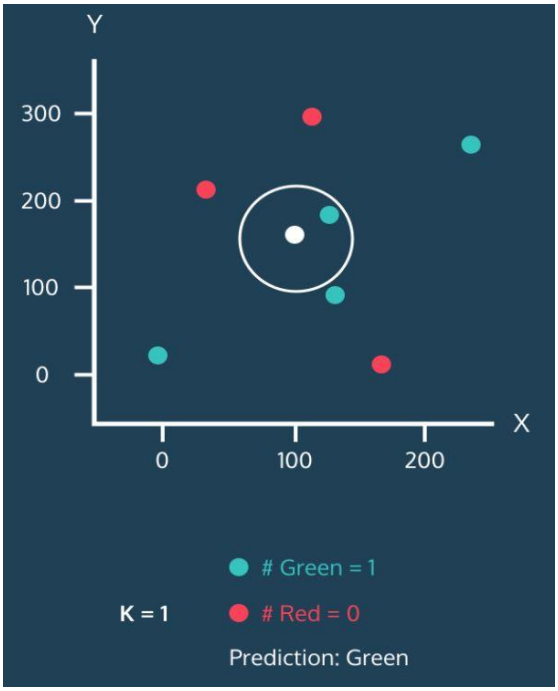
A. Algorithm Fundamentals

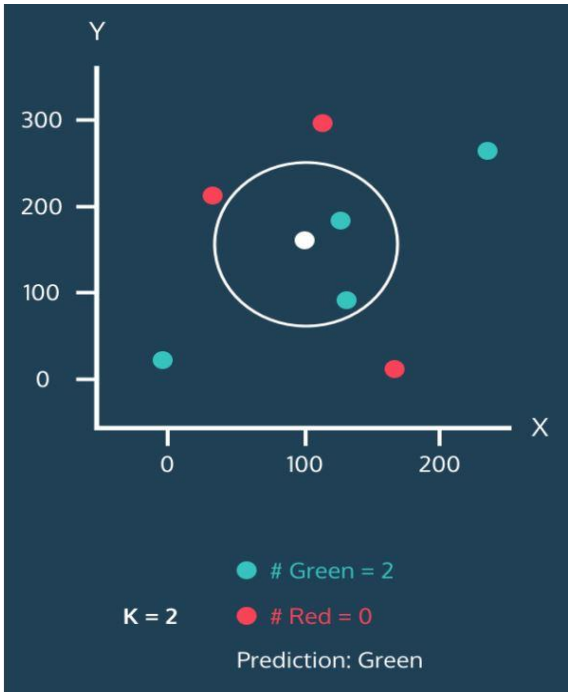
오늘 본격적으로 비지도학습을 다루기 전에 지도학습의 영역에 포함된, 요긴하게 쓸 수 있는 알고리즘을 하나 소개하려 한다.

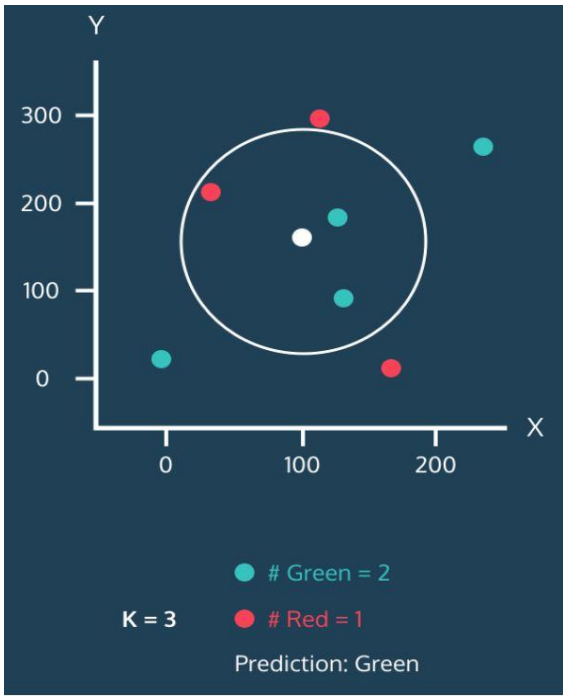
'Birds of the same feather flock together.' 라는 속담의 메시지를 우리의 목적에 맞게 해석한다면 '유사한 특성을 가진 데이터는 유사한 범주에 속하는 경향이 있다' 정도가 될 듯하다. 이때 그 '유사성'을 정의 내리기 위해 우리는 거리의 개념을 차용하고, 이 거리 계산의 구체적인 지표로는 우리에게 익숙하고 또 직관적으로 받아들이기 쉬운 유클리드 거리(Euclidean Distance)를 주로 사용한다.

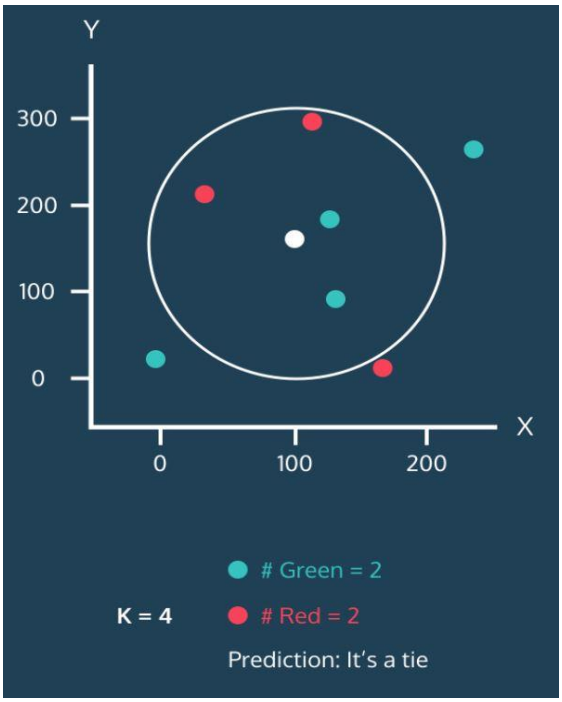
K-Nearest Neighbors Algorithm, 한국어로 K-최근접 이웃 알고리즘은 거리기반의 분류분석 모델이다. 지도학습의 영역에 속하기에 관측치들은 y 라벨을 지니고 있으며, 이 y 라벨을 모르는 새로운 데이터에 대해서 예측/추론을 실시할 수 있다. 가까운 거리에 위치해 있는 관측치들이 동일한 범주에 속한다고 판단하여, 새롭게 알아내고자 하는 관측치가 **기존의 관측치들과 얼마나 가까운지**에 대한 정보를 통해 범주를 결정하는 방법이라 할 수 있겠다. 이 방법은 간단하면서도 기발한 아이디어이기에 이미지 처리 분야(image classification, image segmentation 등), 추천 시스템과 같은 다양한 분야에 적용되어 사용된다.

자 그럼 거두절미하고, 예시를 통해 감을 잡아보자. 흰 색의 점을 새로운 데이터라 생각하면 되겠다.









위 일련의 그림들에서 볼 수 있듯, 새로운 데이터는 k개의 이웃하는 기존 관측치들의 범주 중 **최빈값(mode)**을 따른다. 거리를 기준으로 가까운 k개의 '비슷한' 관측치들을 묶고, 이들이 대표하는 범주를 그대로 따른다는 의미로 받아들이면 쉽겠다. 우리가 지난 2주 내내 이야기해 온 모델의 오버피팅/언더피팅 가능성과 연결 지어 이야기해볼까? k=1, 즉 주변의 한 가지 관측치 예시만을 사용한다면 데이터 하나하나를 동일한 중요도로 여기는 것과 다름없기 때문에 대체로 오버피팅할 가능성을 시사한다. 하나 덧붙여 말하자면 k=1의 경우 이상치 처리에 대해 융통성을 발휘하지 못한다. 위 그림 중 k=1 예시에서 흰색 점 옆에 자리하고 있는 데이터가 하필 빨강도, 초록도 아닌 검정이라고 해보자. 그렇게 되면, 이 새로운 데이터는 검정 class를 따르게 될 것이고 이 경우 적절히 분류되었다고 말할 수 없다. 반대쪽 스펙트럼에 있는 극단적인 예시를 하나 언급하자면, 만약 k=n(관측치의 개수)라면 이는 극심한 언더피팅 가능성을 가리키는데, 새로운 데이터가 모두 기존 데이터셋의 majority class를 따르기 때문이다. 그렇기 때문에 이 둘을 적절히 고려한 k를 찾는 과정이 수반되어야 하는 것이다.

또 한 가지 의논해 볼 점은, 관측치들의 특성들을 고르게 반영하고 이를 토대로 분류 작업이 이루어져야 하기 때문에 때로는 정규화(normalization) 작업을 취해주어야 한다는 것이다. 설명변수 간 scale을 고르게 정리해준다는 의미에서 scaling이라고도 부를 수 있으며, 가장 널리 사용되는 방법은 아래 두 가지이다.

- StandardScaler : 평균과 표준편차를 이용. z-점수 산출 방식.

$$X' = \frac{X - \mu}{\sigma}$$

- MinMaxScaler : 최솟값을 0으로, 최댓값을 1로 고정한 뒤 모든 값들을 0과 1 사이 값으로 변환.

$$X' = \frac{(X - X_{min})}{(X_{max} - X_{min})}$$

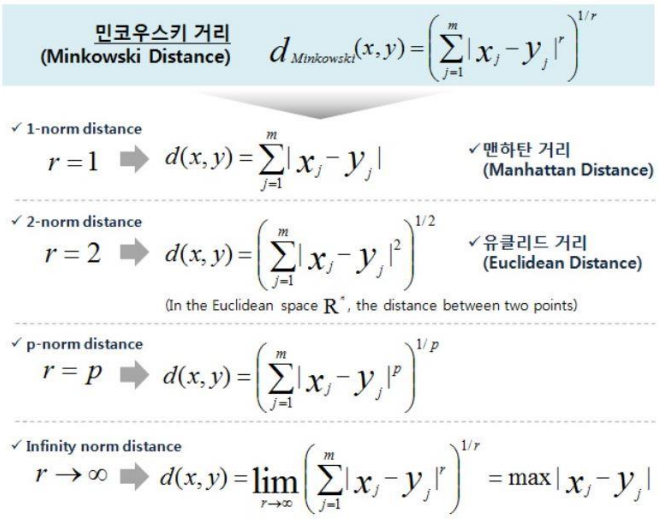
그럼, K-NN 알고리즘은 분류 문제에만 사용되는가? 그렇지 않다. 회귀 문제에도 사용할 수 있다. 예를 들어 새로운 영화에 대한 영화 평점을 예측하고 싶다면 이와 비슷한, 그러니까 가까운 거리의 영화 평점들의 평균을 취해주면 되는 것이다. 만약 k가 3이고. 이웃하는 3개 영화의 평점이 각각 5.0, 6.8, 9.2라면 이들의 평균인 7.0을 새로운 영화의 평점이라고 예측할 수 있다. 사실 더 스마트한 방식을 취해본다면 이웃 영화들 평점을 단순 평균 취하는 것이 아니라 각 이웃이 얼마나 가까이 위치해 있는지도 고려해볼 수 있는데, '가중 회귀(weighted regression)'라고 궁금하면 직접 찾아보세욤~

B. Distance Metric

관측치들이 서로 얼마나 떨어져 있는지에 대한 거리 지표로서 유클리드 거리를 자주 사용한다고 가볍게 언급했다. 사실 다른 metric들도 많이 있어 간단히 소개하려 한다. 파이썬에서 KNNClassifier, KNNRegressor를 불러올 때 그 metric을 무엇으로 지정할지 명시해줄 수 있다.

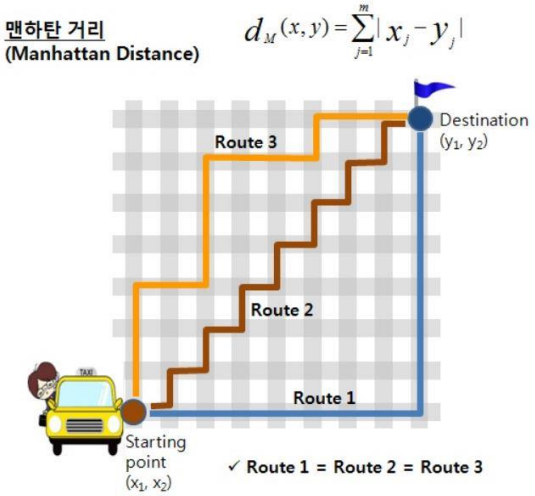
1) Minkowski Distance(민코우스키 거리)

회귀 문제에서의 릿지, 라쏘 그리고 오버피팅 방지를 위해 취해주는 정규화에서의 L1, L2 norm에서의 바로 그 개념이다. 쉽게 말하면 norm들의 일반화된 표현이라고 할 수 있는데 r=2일 때가 유클리드 거리, r=1인 경우가 맨하탄 거리(Manhattan Distance)로써 정의된다.



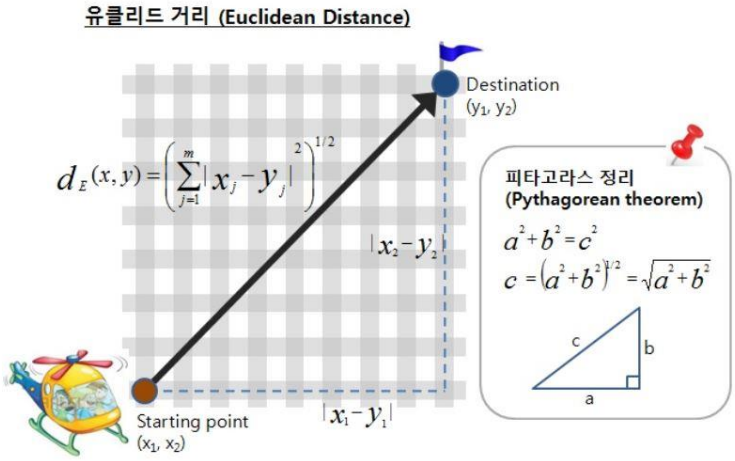
2) Manhattan Distance(맨하탄 거리)

뉴욕의 택시가 출발지에서 도착지로 갈 때, 도시의 도로가 격자 모양과 같이 형성되어 있다면 건물이라는 장애물을 뚫고 대각선으로 달릴 수는 없으니 그 도로를 거쳐서 목적지로 향한다는 원리로 설명된다. 그래프 이론(Graph Theory)에서 최단 거리(shortest path)를 찾는데 사용되는 개념이라 할 수 있겠다.



3) Euclidean Distance(유클리드 거리)

무슨 말이 더 필요할까...? 중학교 때 배운 피타고라스 공식을 떠올리면 되겠다(피타고라스…중학교 때 배우는 거 맞지?).



2. Cluster Analysis

A. Deciding the Number of Clusters

비지도학습 들어가자~! 비지도학습(Unsupervised learning)이 **비**지도학습이라 불리는 이유는… 무수히도 많이 말했지만 y 라벨이 없기 때문이라고 했다. 그렇기 때문에 분석자의 주관이 많이 들어갈 수밖에 없다는 이야기도 했다! 예측이나 추론과 같이 명확한 task가 정의되어 있지 않은 상황에서(또는 이와 같은 방법론을 고수하기보다 조금 더 '탐색적'인 분석을 하고 싶을 때) 사용 가능하며, 대표적으로 clustering(군집화)를 꼽을 수 있겠다. 예시로 서울시에 거주하는 외국인들의 웰빙 관련 지표들(외국인 학교의 개수, 영어 소통 가능한 병의원의 개수 등)이 데이터프레임에 열로서 주어졌을 때, 어느 지역구들이 외국인이 살기에 좋고 또 어떤 지역구들이 병의원 시설 등이 부족한지를 파악하는 방법으로 클러스터링을 사용할 수 있다. 이들 질문은 군집화 결과에서 어떠한 지역구들이 하나의 군집을 이루고 있는지, 또 이들 군집 각각의 주요한 특성은 무엇인지 파악하는 과정을 통해 답을 찾을 수 있다.

앞선 KNN 알고리즘에서도 거리의 개념을 사용해 이들을 바탕으로 관측치들이 서로 얼마나 가까운지에 대해 이야기할 수 있었듯, 클러스터링 알고리즘 역시 거리 기반이다. 클러스터링 작업에서 주요하게 제기되는 질문은: 몇 개의 클러스터를 생성할 것인가? 인데, 그 값을 구체적으로 결정하기 위한 방법으로 Silhouette Method와 Elbow Method를 소개하려 한다.

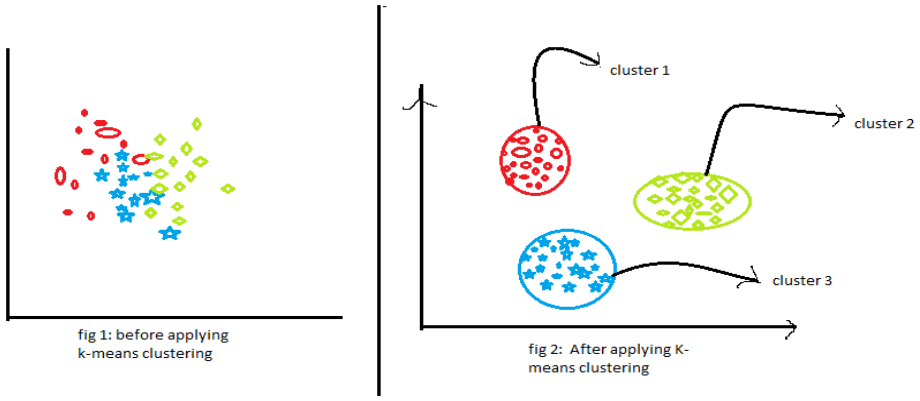
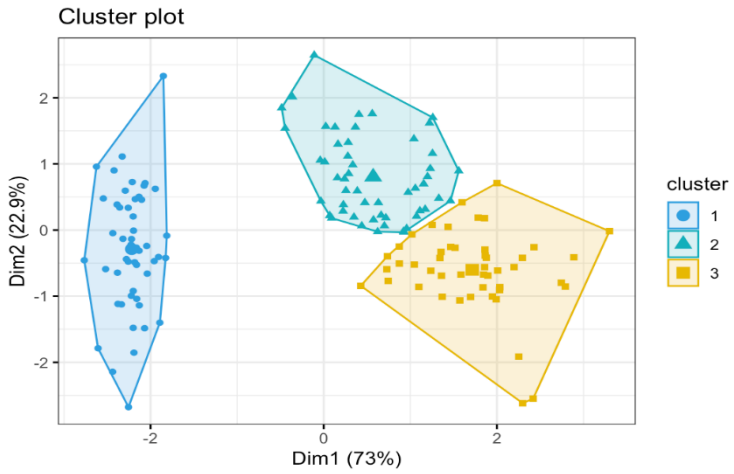


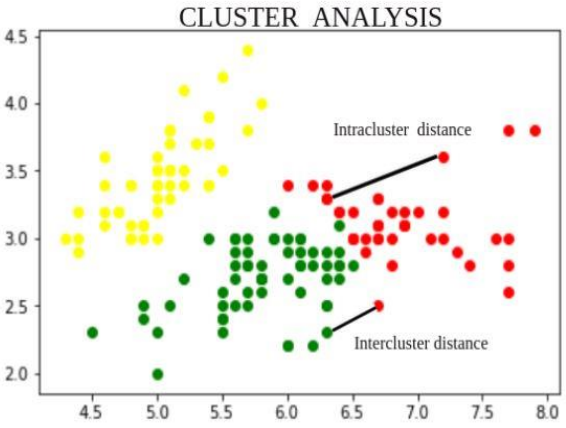
fig 1: before applying k-means clustering

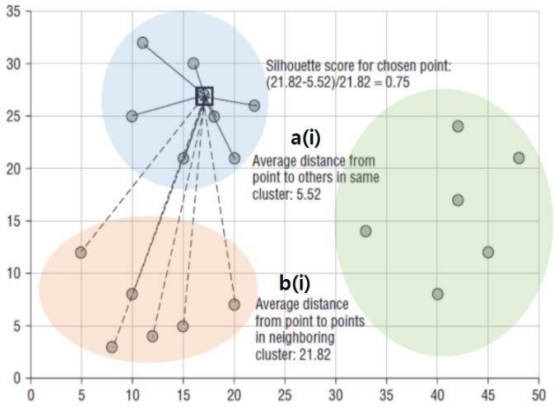
fig 2: After applying K-means clustering



1) Silhouette Method

지난 시간 트리 모델의 분기점 결정 및 input space 분할과 관련하여 불순도(impurity)에 대해 이야기했었다. 비슷한 애들끼리 '몰려 있는 형태'가 가장 이상적이고 그렇기 때문에 동일한 범주를 가진 관측치들은 서로 가까이 위치해 있어야 함을 살펴보았다. 클러스터링의 관점에서 보면 이는 곧 동일 군집 내의 관측치들은 서로 뭉쳐 있어야 하고(사이 거리가 작아야 하고), 각기 다른 군집 간 관측치들은 서로 멀리 떨어져 있어야 함을 의미한다. 다시 말해 최적의 클러스터링은 군집 간 분산(inter-cluster variance)을 최대로, 군집 내 분산(intra-cluster variance)를 최소로 하는 것이다.



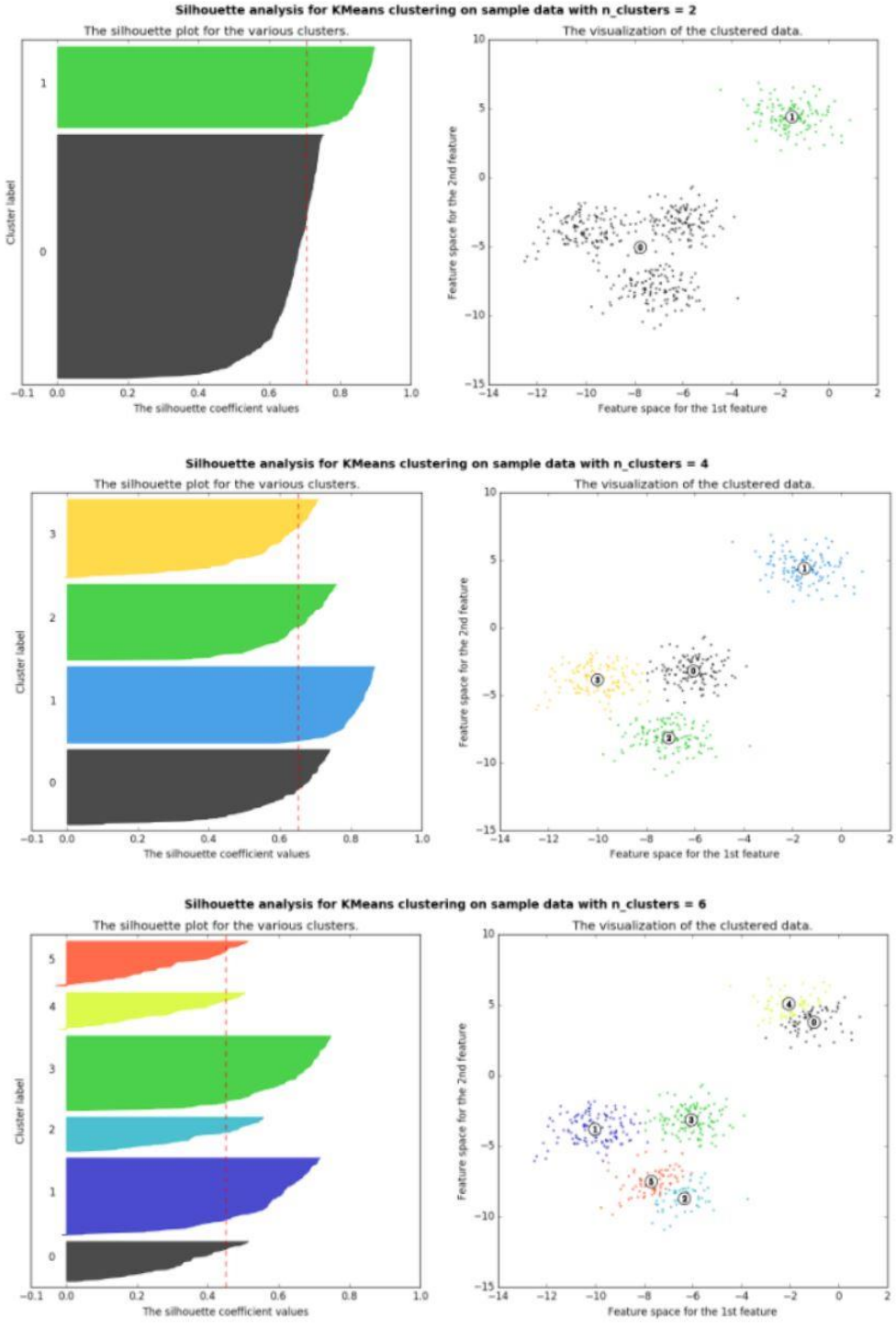


먼저, 실루엣 계수는 다음과 같이 표현될 수 있다. 이 값이 1에 가까울수록 군집 형성이 더욱 타당한 것이다.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$a(i)$는 i번째 개체와 같은 군집에 속한 요소들 간 거리의 평균, 다시 말해 클러스터 내의 응집도(cohesion)을 의미한다. $b(i)$는 클러스터간 분리도(separation)을 나타내는 값으로 i번째 개체와 가장 가까운 클러스터내의 모든 데이터들과의 평균거리이다. 이때 $a(i)$가 작게 형성되는 이상적인 상황이라면 실루엣 계수는 1에 가까워진다. $a(i)$가 0의 값을 가진다면 실루엣 값은 1이다. 한편, $b(i)$가 0에 가까울수록 최악의 클러스터링이라 할 수 있는 이유는 바로 실루엣 값이 -1을 지니기 때문이다. 실루엣 값이 어느 지점에서 형성되어야 최적의 클러스터링이 이루어질 것인가에 대한 절대적인 기준은 없으나, 대체적으로 사회과학 분야에는 0.5 이상의 실루엣 계수가 나왔을 때 잘 된 클러스터링이라 판단한다.

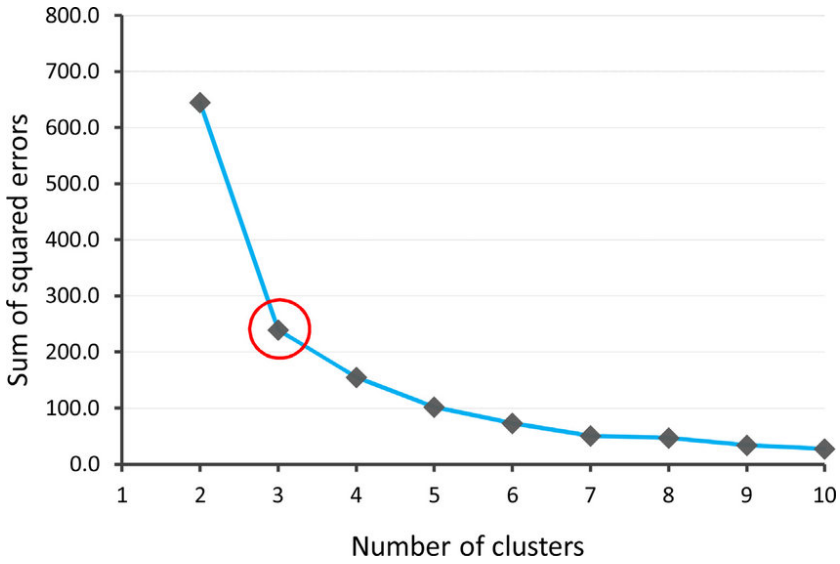
아래 예시에서 군집의 개수를 달리 설정했을 때의 실루엣 계수 변화를 살펴보자.



1990년 Silhouette Width의 개념으로 먼저 제시되었으며, 위 그림에서도 볼 수 있듯 클러스터링 결과에 따라 각 군집의 실루엣 계수에 대한 양상이 다르게 나타난다. 빨간색 점선은 이들 군집 전부의 평균 실루엣 값이라고 보면 된다.

2) Elbow Method

엘보우 기법은 클러스터 내 오차제곱합(RSS)이 최소가 되도록 클러스터의 중심을 결정해 나가는 방법으로, 쉽게 말하자면 클러스터 내의 분산(Within Cluster Sum of Square, WCSS) 이 최소가 되게끔 하는 방식이다. 클러스터의 개수가 많아지면 당연히 이 합은 줄어들기 마련인데, 이것이 점점 줄어들다가 급격히 줄어드는 지점의 이 elbow point를 클러스터의 개수로 사용하면 된다.



클러스터 내의 분산을 줄이는 원리를 수식으로 표현하면 마치 잔차제곱합을 구하는 것과 같은 형태인데, 다만 여기에서는 한 클러스터 내의 평균과 해당 클러스터에 속한 다른 점들 간의 거리의 제곱합을 최적화하는 방법을 취한다. 1979년 Hartigan과 Wang이 고안해냈으며 R에서의 kmeans() 함수는 이 알고리즘을 기본 디폴트값으로 취한다고 한다. 아래 수식과 그림을 통해 내용을 정리해보자.

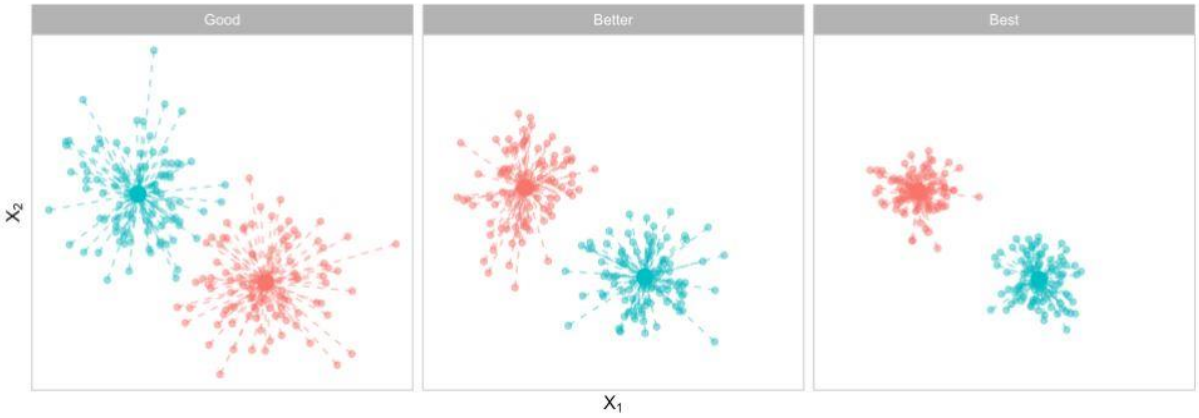
$$W(C_k) = \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

$$where\ x_i\ is\ an\ observation\ belonging\ to\ the\ cluster\ C_k$$

$$\mu_k\ is\ the\ mean\ value\ of\ the\ points\ assigned\ to\ the\ cluster\ C_k$$

위와 같은 작업을 모든 클러스터에 대해 실시해주는데,

$$SS_{within} = \sum_{k=1}^{k} W(C_k) = \sum_{k=1}^{k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$



클러스터 내 원소들의 compactness를 위의 산식을 이용하여 계산하면 되고, 이에 따라 가장 좋은 클러스터링 결과는 오른쪽의 경우라고 할 수 있는 것이다.

R에서의 kmeans() 함수는 다음과 같이 정의가 되어 있다.

```
kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace=FALSE)
# S3 method for kmeans
fitted(object, method = c("centers", "classes"), ...)
```

B. K-Means Clustering

실루엣 기법과 엘보우 기법을 통해 구해낸 군집의 개수 k에 대한 정보를 가지고, 클러스터링을 본격적으로 진행해보자! 클러스터링은 크게 비계층적 클러스터링(non-hierarchical clustering)과 계층적 클러스터링(hierarchical clustering)으로 나뉘는데, 이 중 우리는 더 자주 쓰는 비계층적 클러스터링에 집중해보려고 한다. 먼저 K-means!

K-means clustering은 이름에서 알 수 있듯 데이터들의 **평균값**을 이용하여 군집화를 진행한다는 의미이다. 클러스터 내의 요소들은 분산을 작게 하고, 클러스터 간의 분산을 최대화한다는 아이디어를 수식으로 표현하면 다음과 같다.

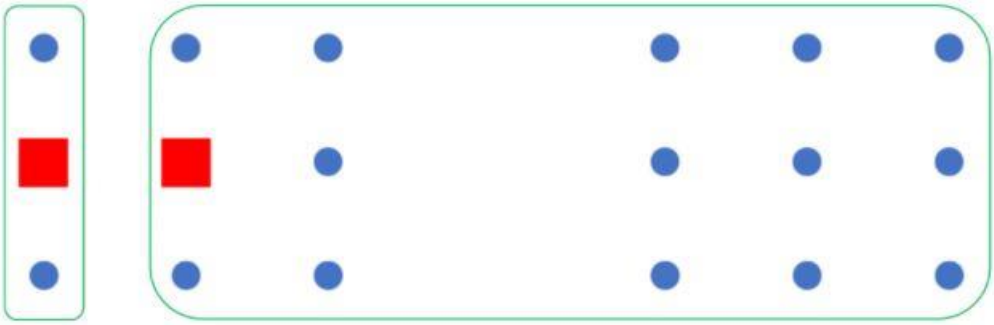
$$X = C_1 \cup C_2 \dots C_K, \qquad C_i \cap C_j = \emptyset$$

$$argmin_C \sum_{i=1}^{K} \sum_{x_j \in C_j} \|x_j - c_i\|^2$$

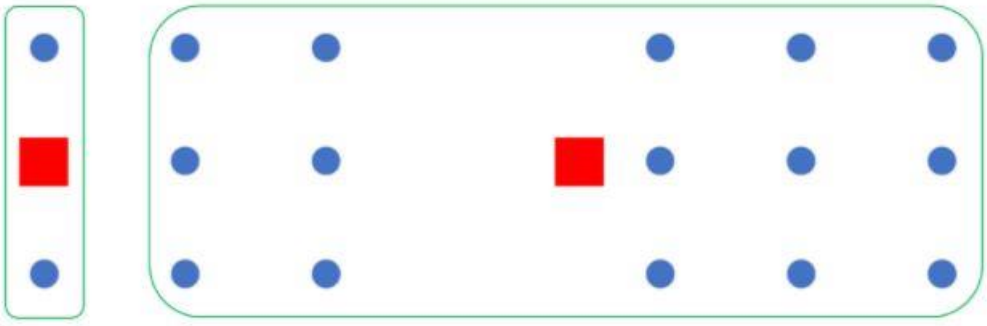
그럼, 그림을 통해 개념을 추상화해보자.

i) 군집의 개수 k를 2로 설정하고, 군집의 중심을 랜덤 초기화한다면(빨강색 점으로) 가장 먼저 다음과 같이 표현할 수 있다.

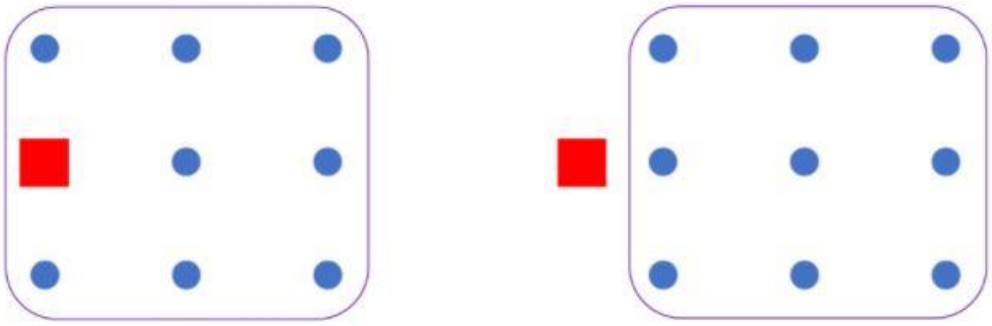
ii) 관측치들을 파란색 점으로 보았을 때, 이들을 가장 가까운 중심점과 함께 군집화한다면:



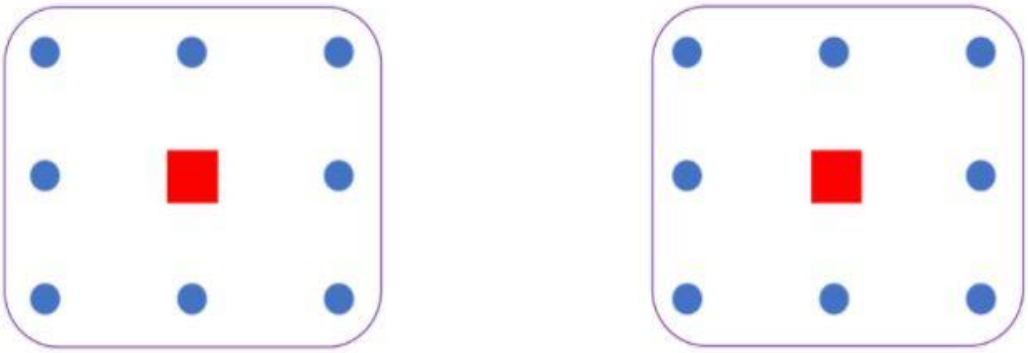
iii) K-means clustering은 각 군집의 '중심'을 찾는 방향으로 진행되므로, 각 군집(초록색 상자)의 중앙부에 그 중심점이 위치하도록 해준다.



iv) ii단계에서와 같이, 이들 중심점을 바탕으로 재군집화를 실시해준다.

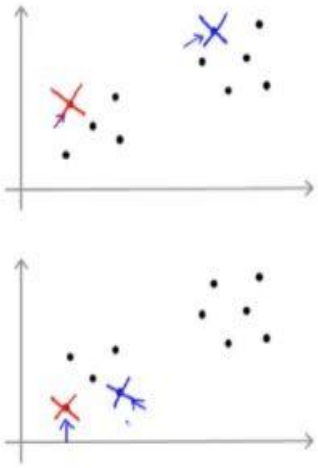


v) 그러고는 다시 iii에서와 같이 중심점을 업데이트해준다.



위 과정을 계속 반복하더라도 중심점이 변하지 않거나(수렴한다) 사용자가 정한 iteration 횟수를 채우면 학습을 마친다.

자주 쓰이는 K-means 방법이지만 설정한 반복 수를 얼마로 두냐에 따라 클러스터링의 결과가 다르게 양산된다는 점은 단점이라고 꼽을 수 있겠다. 또, 초반부 어느 지점을 중심점으로 둘지에 대한 랜덤성 역시 한계라고 말할 수 있다. 예를 들어, 한 쪽 방향으로 치우친 것과 같이 중심점을 두고 시작한다면 local optima에 빠질 수 있어 최적해를 구하기 어려워진다. 우측의 그림이 바로 이러한 예시가 될 수 있겠다.



이를 극복하기 위한 방법으로 기존의 random initialization을 반복적으로 수행하는 multiple random initialization이 제기될 수 있는데, 간단히 말하자면 for문을 통해 50~100번 정도 반복 수행을 실시함으로써 이 중 최적의 초기화 조건을 찾는 것이다. 아 물론 여기서 클러스터의 개수 k는 고정. 여기서는 랜덤 초기화로 인해 발생할 수 있는 문제에 초점을 맞추는 것이다. 100번 정도 반복 수행을 하면 대체로 최적의 결과를 산출해낼 수 있으나 클러스터가 10개 이상일 때는 다시 local

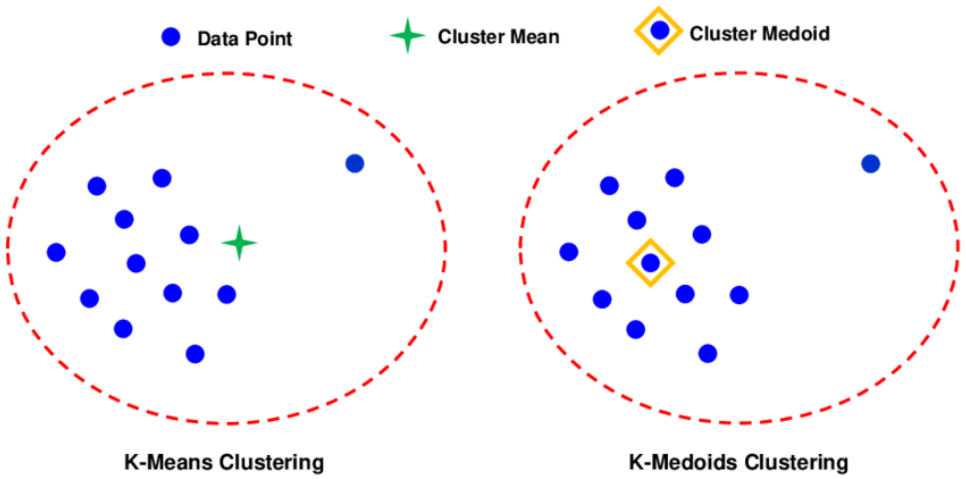
optima에 빠질 수 있다고 하니 자신의 데이터 domain에 specific한 방법을 고안해낼 수 있어야 할 것이다.

C. K-Medoids Clustering

앞서 살펴본 K-means가 각 관측치 설명변수들의 평균값에 근거하여 클러스터링을 실시한 것이라면, K-medoids는 **중앙값**을 그 기준으로 삼는다. Partitioning Around Medoids라고, 그 이름에서처럼 중앙값을 기준으로 파티셔닝을 실시하는 기법이기에 PAM이라고도 불린다.

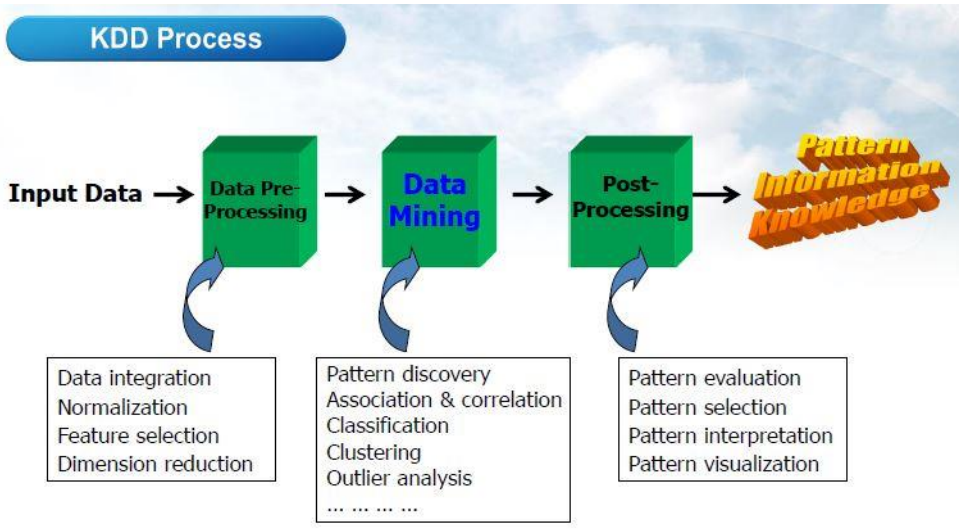
결과적으로 말하자면, K-medoids 방법이 K-means보다 더욱 강건한데 그 중앙값을 취할 경우 평균값을 취할 때와 비교하였을 때 이상치로부터 받는 영향이 적기 때문이다. 대체로 큰 값이 많음에도 엄청 작은 값이 몇 개 있을 경우 큰 숫자의 효과가 상쇄되는 개념을 떠올려보면 이 논의가 더욱 쉽게 다가올 수 있을 것이다. 또한, 각 클러스터에서 대표적인 객체를 찾는 것이 쉽다는 면에서 해석하는데 용이하다는 점이 장점으로 작용한다. 특정 클러스터 내의 중앙값을 뽑아낸다는 것은 곧 특정 관측치를 추출해내거나, (예를 들어 관측치 짝수개라면) 유사한 관측치를 추출해낼 수 있다는 것이므로 해당 클러스터의 특징을 기술하는데 있어서 예시로 삼을 수 있는 개체가 존재한다는 것이다.

한편, K-medoids 역시 random initialization에 의한 local optima 가능성을 지니고 있기 때문에 동일한 방식으로 multiple random initialization을 실시해주면 된다.



3. Data Mining Applications

본격적으로 추천시스템(은 아니고! 일단은 인트로)! 우리팀에서 처음으로 추천시스템 내용을 다룬다고 한다 ㅎㅎ 1주차때 언급했듯 데이터 마이닝이라는 분야는 '다양한 학문의 교집합에 자리'하고 있고 또 그렇기 때문에 그 정의를 내리는데 있어서도 broad하게 기술할 수 있다. 앞선 주차들에서 머신러닝 중 지도학습 부분에 초점을 맞춰 설명했고 이번주 교안 이전 챕터에서도 역시 머신러닝의 갈래 중 비지도학습을 다뤘기에 모델링의 관점을 넘어서 '데이터' 를 활용한다는 그 사실에 집중, 다양한 이야기를 해보고 싶어 이 내용을 추가하게 되었다.



데이터 마이닝이라는 개념이 데이터를 수집하고 이를 처리에 용이하도록 전처리/형식 변환/재범주화하고, 이를 통해 규칙을 찾거나 모델링을 실시하며 마지막으로 결과를 해석하기 위해 시각화 등을 시도해보는 일련의 과정을 총 망라한다는 관점을 취해본다면, 우리가 앞서 실시한 모델링은 이 중 일부분에 불과하다는 것을 알 수 있다. 최근 학계/산업계가 알고리즘적으로 더 light한, 성능을 잘 내는 모델링 기법을 연구하는데 많은 에너지를 쏟고 있다. 희소성의 원칙에 입각해보자면, 오히려 기본으로 회귀하여 우리가 직관적으로 이해할 수 있으면서도 기발하게 설계된 방법들에 대해 검토, 애초에 왜 그러한 방법론들이 제기되었는지를 살펴보는 것이 'breakthrough' 라고 불릴 만한 기법들을 고안하는 과정에서의 필요충분조건이라고 할 수 있을 것이다. Paypal의 창립자 Peter Thiel도 그의 저서 Zero to One에서 'breakthrough'라 불릴 만한 기술/아이디어의 개발은 0에서 1로의 수직적인 성장(zero to one)인 반면에 기존의 기술을 조금씩 고쳐 특정 기술의 주요한 아이디어를 그대로 따르는 것은 수평적인 성장에 불과, 눈에 띄는 발전을 기대하기 어렵다고 서술한 바 있다.

구구절절 써 놨는데 결론은:

i) 수리적인 접근이 장땡이 아니다, 모델링 성능만 올린다고 장땡은 아니다

이미 30% 정확도 -> 70% 정확도에 이르도록 하는 기술의 발전은 예~전에 이루어졌다.

지금은 일의 자릿수/소수점 단위의 성능 향상에 열을 올리고 있다.

KorQuad(https://korquad.github.io/)만 봐도… / 비슷한 맥락에서 무어의 법칙(Moore's Law)

ii) 넓게 넓게 보자.

모델링은 머신러닝의 베이스라인을 가져온 거라서 데이터마이닝과 동치로 두기에는 어려움이 있다.

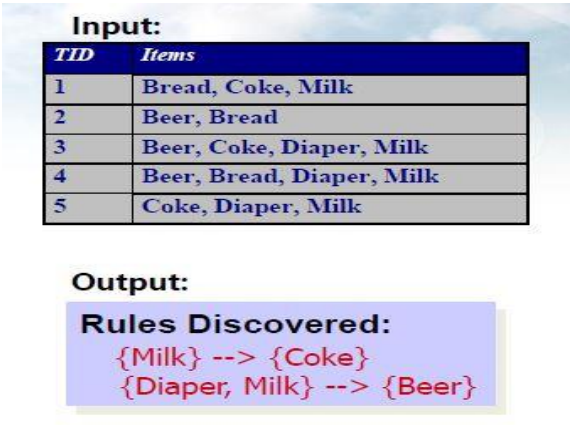
소셜 네트워크 분석, 텍스트 마이닝 등 어떠한 형태의 데이터를 어떤 방식을 활용하여 처리할 수 있을 지에 대한 옵션은 매우 다양하다. 진짜 재밌는 논문 많으니 궁금하면 제게 갠톡~

iii) 아무리 어려운 개념 같아 보여도 이해를 하면 쉬워진다. 모든 것은 이해하는 데서 출발한다.

이전의 방법들을 서로 연계하여 이해해보고 더 나아가 이러한 기존 방법들의 단점을 보완할 수 있다면 당신도 breakthrough라고 불릴 만한 방법론을 고안할 수 있다!

A. Association Rule Discovery

여러분이 자주 시간을 보내곤 하는 넷플릭스와 유튜브의 추천 시스템 알고리즘의 시초라 할 수 있는, 연관 법칙 발견에 대해 이야기해보려 한다. 이후 더 자세히 다룰 컨텐츠 기반 추천(content-based filtering)의 기본이 되는 방법론이다. 슈퍼마켓에서 어떤 물건과 어떤 물건이 주로 같이 많이 구입되는지 궁금한 것과 같은 원리이기에, 'Market-basket analysis'라고도 불린다. 벌써 마케팅과 접목시킬 수 있을 것 같다는 삘이 팍팍 오지 않는가… 실제로 미국의 Amazon은 이런 식으로 소비자의 구매 패턴을 파악한다고 한다. 가장 classic한 예시로는 우유와 기저귀를 같이 구입하는 고객들이 맥주도 함께 같이 구입하곤 한다는 것이다. (육아가 많이 힘든 가 보오…)



연관 법칙을 발견에 대표적으로 적용되는 알고리즘으로는 **A Priori 알고리즘**이 있다. 이를 잠시 철학적으로 접근해보자면, 칸트의 <순수 이성 비판>이라는 책에서 A posteriori 라는 라틴어 구와 함께 등장해 대중화되었다. A priori의 뜻을 검색해보면 '선천적','선험적' 이라는 결과가 나온다. 몰라~ 구글 선생님이 알려줌. 단어의 이름에서부터 알 수 있듯 인과 관계와 관련되어 있으며 이를 우리의 맥락에서 풀어보자면 아이템들의 등장 순서, 즉 if '아이템1이 구매되었다면 아이템2도 구매될 것이다'와 같은 조건문을 작성해보는 것이라고 이해할 수 있겠다. A priori 알고리즘에서 주요하게 제기되는 두 가지의 질문은 다음과 같다.

i) 어떤 아이템들끼리 같이 많이 구입되는가 ~ Frequent Itemset Generation

ii) i을 바탕으로 어떤 규칙을 추론해낼 수 있는가 ~ Rule Generation

***예시* 다섯 개의 물품 Milk, Coke, Pepsi, Beer, Juice가 있다고 해보자.**

편의상 이들 물품에 대해 m,c,p,b,j라는 축약어로 이들을 가리키려고 한다.

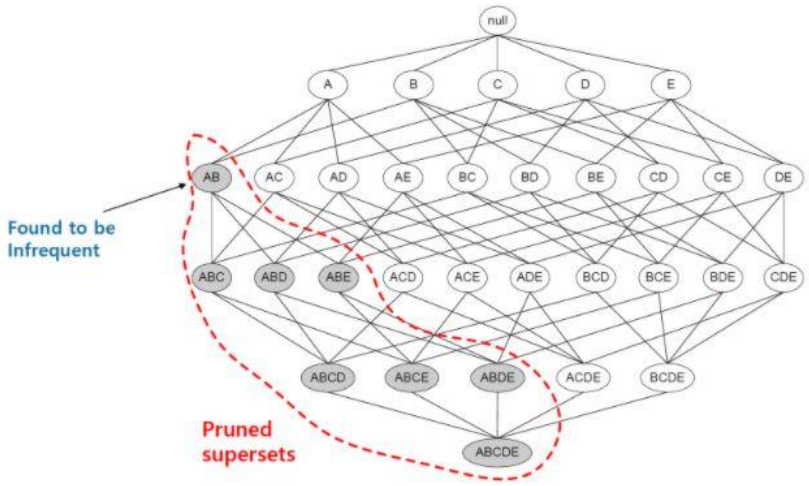
거래 내역(transaction)은 다음과 같다.

거래 순번 (transaction number)	구매 물품
T1	m,c,b
T2	m,p,j
T3	m,c,b,n
T4	c,j
T5	m,p,b
T6	m,c,b,j
T7	c,b,j
T8	b,c

그럼 먼저, 'frequent'함의 정의를 내려 frequent itemset에는 어떤 것들이 있는지 파악해보자. Support는 특정 물품의 등장 횟수인데, 이 값이 사용자가 지정한 최소 지지도 조건(support threshold) 이상의 값을 취할 때 우리는 해당 물품이 frequent하게 등장한다고 말할 수 있다. 가장 먼저 최소 지지도 조건을 만족하는 2개짜리 아이템 집합을 생성하고, 하나씩 아이템을 더해가며 더 이상 조건을 만족하는 아이템 집합쌍이 없을 때까지 support 계산을 실시한다. 만약 이 경우 threshold가 3이라면 frequent itemset은 **{b, m}, {b, c}, {c, m}, {c, j}, {m ,c, b}**가 나오겠다.

이때, transaction 리스트에 올라와 있는 제품의 개수가 매우 많을 경우 이들 itemset의 경우의 수를 전부 계산하는 것이 많은 컴퓨팅 파워를 요구하지는 않을지에 대해 의문을 품을 수 있다. 초기 setting에서처럼 두 개의 아이템 집합을 구성하기 위해 완전 탐색(exhaustive search, 하나 하나씩 다 들여다보면서 특정 조건을 만족하는 요소만 취하는 방법)을 실시한다면 이 연산만으로도 벌써 $O(N^2)$의 복잡도를 지닌다. (여기에서 설명하

는 복잡도의 개념과 우리가 1주차에서 다룬 모델의 복잡도가 완전히 일치하는 개념은 아니다. 이 맥락에서의 복잡도는 알고리즘의 수행 시간과 관련한 측면으로 대략적인 감만 잡으면 충분할 듯하다) A Priori 알고리즘은 frequent itemset을 찾아내는 데 있어 특정한 조건 두가지를 만족했을 때의 발생확률이, 단일한 조건 만족으로 인한 사건 발생확률보다 늘 작거나 같다는 아이디어에 기초한다. 예를 들어, 단일한 물품으로 구성된 아이템 집합 {A}의 발생 확률이 0.1이라면 아이템 집합 {A,B}의 지지도는 아무리 높아도 0.1을 넘지 못한다. 바로 여기서 왜 최소지지도의 개념을 도입하는지 알 수 있는 것이다. 만약 임의의 아이템 집합의 지지도가 일정 기준(최소지지도 조건)을 넘지 못한다면, 해당 아이템 집합의 초월집합(이 경우 {A,B})의 지지도는 그 기준보다 당연히 작을 것이므로 유용한 규칙으로 인정받을 수가 없다는 이야기이다. 그러므로 이러한 경우들은 아예 계산을 해볼 필요가 없는 것이다. 이러한 방법으로 A Priori 알고리즘은 계산 효율성을 달성하는데, 아래 그림을 통해 그 원리를 정리해보자.



Frequent한 itemset들을 찾았으니 이제 rule을 유도해보자. 이 때 쓰이는 지표인 confidence(신뢰도, c)는 조건부 확률의 개념을 차용한 것으로, A->B의 연결 관계에서 A가 등장했을 때 A와 B가 모두 등장하는 횟수를 의미한다. 지지도와 관련하여서도 최소 지지도 조건이 있었듯, confidence 역시 threshold 값을 둘 수 있으며 구한 frequent itemset들 전부에 대해 confidence 계산, threshold보다 낮은 경우 폐기처분(?)하는 방식이다.

Support threshold가 3일 때 살아남은 아이템 집합 쌍들은 아래와 같았다.

➔ {b, m}, {b, c}, {c, m}, {c, j}, {m, c, b}

만약 여기서 confidence threshold를 0.75$(=\frac{3}{4})$로 두었을 때 몇몇 집합들은 사라지게 된다.

➔ * b->m : c $= \frac{\#\ of\ itemsets\ that\ contain\ b\ \boldsymbol{and}\ m}{\#\ of\ itemsets\ that\ contain\ b} = \frac{4}{6}$ -> 폐기

***m->b : c $= \frac{4}{5}$** *b,c->m: c $= \frac{3}{5}$ -> 폐기 *** b,m->c: c $= \frac{3}{4}$**

support threshold와 confidence threshold를 모두 넘은 itemset들에 대해서만 남겨놓고 다시 결과를 보자. 모든 high-confidence 조건들이 항상 interesting하다고 말하기는 어렵다. 예를 들어 X->milk와 같은 관계는 자주 발생하는 것이 사람들이 대체적으로 우유를 많이 구입(X와는 독립적으로 시행)하기 때문에 자연스럽게 그 confidence가 높아지기 때문이다.

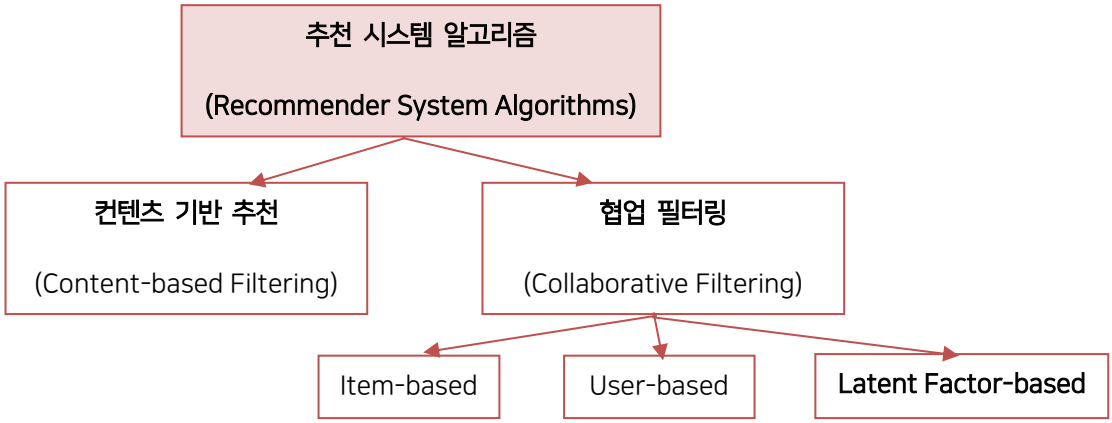
'Interesting'한 정보가 무엇인지를 파악하는 방법은 아래 수식과 같이 표현되며 대체로 0.5 이상의 값이 산출될 때 'interesting한 rule이구나~'라고 말한다.

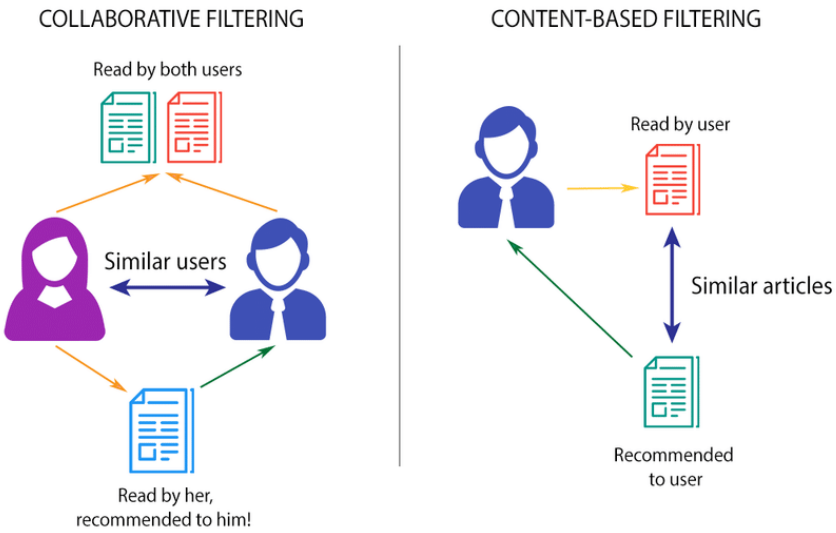
$$Interest(I \rightarrow J) = conf(I \rightarrow J) - \Pr[j]$$

B. Recommender Systems

추천 시스템이라고 해서 어려운 게 아니다. 흔한 유튜브 알고리즘을 생각해보자. 왠지 계속해서 비슷한 컨텐츠들이 자신의 피드에 뜨고 있다는 생각이 들지 않는가? 예를 들어, 유나가 한참 NCT에 빠졌을 지난 10월의 피드 상태는 '마크가 체리밤 춤추는 영상', '태용이가 Baby don't like it 콘서트 때 공연하는 영상', 'NCT 미국 투어 브이로그'로 가득했는데… 그런 영상들은 대체로 내가 시청한 동일한 영상을 시청한 다른 사용자들이 시청한(말장난이가) 영상들도 일부 포함되어 있는 것인데 이는 자신과 비슷한 취향을 가진 사람들이 소비한 컨텐츠를 그 사람에게도 추천해준다면 서비스 만족도가 올라갈 것이라는 리즈닝을 전제로 한다. (그리고 먹혔어!! 탈덕은 있을 수 없는 일이야!!)

그럼, 본격적인 논의를 시작하기 전 추천 시스템 알고리즘의 대략적인 갈래를 살펴보자.





i) Content-based Filtering

컨텐츠 기반의 추천은 말 그대로 해당 컨텐츠에 대한 정보만을 이용하여 추천을 실시하는 방법이다. 사용자가 과거에 소비한 컨텐츠의 특성을 분석하여 이와 유사한 특성을 지닌 컨텐츠를 사용자에게 추천해주는 방식이라고 이해하면 되겠다. 즉, 사용자의 성향을 파악하여 해당 사용자가 좋아할 만한 컨텐츠를 추천해주는 방식이다. 컨텐츠 기반 추천시스템의 구조는 아래와 같이 요약될 수 있다.

예를 들어, 유나가 지난 석 달 동안 넷플릭스에서 시청한 영상들을 토대로 새로운 컨텐츠를 추천한다고 해보자. 컨텐츠의 특성을 분석하여 이를 토대로 추천이 진행된다고 하였으므로 그에 대한 분석 방법론을 구체적으로 정립해봐야 하는데, 해당 컨텐츠에 대한 **메타데이터**(감독, 출연진, 장르 등)가 쓰일 수 있겠다. <슬기로운 의사생활>을 그것도 세 번씩이나(대사를 아주 외우겠다 외우겠어) 재밌게 본 유나에게 동일한 제작진이 연출한 드라마 - <슬기로운 감빵생활>, <응답하라 1997>을 추천해줄 수 있을 것이고, 동일한 배우가 출연한 드라마 - <미스터 션샤인>, 영화 <건축학개론>을 추천해줄 수 있을 것이다.

결국, **유사도**가 키포인트라 할 수 있겠다. 그렇기에 컨텐츠간 유사도를 어떤 방식으로 계산할 수 있는 지의 문제를 제기할 수 있는데… 이 유사도를 계산한다는 것은 곧 각 컨텐츠의 주요한 특징을 추출해내는 것으로부터 출발한다. 유나가 최근에 본 영화 <작은 아씨들>의 줄거리를 통해 유의미한 요소를 추출, 비슷한 줄거리/장르를 지닌 영화를 추천해줄 수는 없을까? 텍스트 데이터에서 feature를 뽑아내는 방법으로서 TF-IDF(Term Frequency-Inverse Doc Frequency)라는 방법을 사용하면 어느 문서이든지 간에 많이 나오곤 하는 단어들에

대해 효과적으로 페널티를 부여함과 동시에, 특정 문서를 대표적으로 설명할 수 있는 주요 단어들을 추출해낼 수 있다.

이 원리를 수식적으로 표현하면 다음과 같다.

$$TF - IDF = TF \cdot \log\frac{n_D}{1+n_t}$$

$$n_D : \text{전체 문서 수}, \qquad n_t : \text{단어 t가 나온 문서 수}$$

TF-IDF 방법에서는 자주 나오곤 하는 단어들에 대해서 이들의 출현 횟수를 그대로 사용하는 것이 아니라 이들이 등장하는 총 문서 수를 사용, 특정 단어가 하나의 문서에서 1번이 나오든 100번이 나오든 해당 문서는 한 개로 카운트한다. log를 사용하는 이유도 비교적 직관적인데 이는 총 문서의 수가 어마어마하게 클 경우 위 연산식의 IDF 부분이 기하급수적으로 커지게 때문에 이를 효과적으로 제약하는 역할을 수행한다고 볼 수 있다.

아이템의 특성과 사용자의 지난 기록을 분석하는 과정(유저 프로필 분석)을 수행했다면 이제 아이템들을 추천해줄 차례인데 아이템들을 크게 몇 가지의 범주로 나누는 과정에서 우리가 오늘 배운 클러스터링 기법이 활용되며, 유사도를 '거리'의 개념으로 보았던 원리를 다시 적용하자면 이를 '코사인 유사도(cosine similarity)'를 통해서 서술할 수 있다. TF-IDF 등을 통해 문서에 출현한 단어와 그 빈도 수를 매핑, 더 나아가 이를 컴퓨터가 이해할 수 있게끔 수치형으로 표현하는 '인코딩(encoding)' 과정을 거치고 나면 이들의 feature는 벡터의 형태를 취하게 된다. 다음의 산식을 활용해, 처리한 벡터들의 유사도를 구할 수 있다.

$$cosine\ similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

코사인 유사도 : -1　　코사인 유사도 : 0　　코사인 유사도 : 1

컨텐츠 기반 필터링의 장점은 cold-start로 인한 데이터 부족 현상이 발생하지 않는다는 것이다. 다음에서 따라 나올 방법 중 다른 유저의 정보를 함께 참조하는 협업 필터링을 사용할 경우, 처음 소비자들에게 선을 보이는 시작 단계의 플랫폼에서는 데이터가 부족하여 정확한 추천을 해주지 못한다는 점에서 한계를 지닐 수 있다. 또한, 이렇게 해당 유저가 과거에 소비한 컨텐츠를 기반으로 추천 시스템을 구현할 경우 대중적이지 않은 취향을 지닌 소비자들의 취향 저격도 가능하다. 대체로 추천 시스템에서는 호불호가 안 갈리는, 모두에게 반응이 좋은 인기 컨텐츠를 대부분 소비자의 feed에 올리고는 하는데, 이 방법을 사용하게 된다면 더욱 personalized한 추천을

수행해낼 수 있다.

ii) Collaborative Filtering

앞선 방법에서는 특정 사용자를 고정시켜 놓고 컨텐츠의 특성을 주요한 요소로 활용하여 추천을 진행했다면, 협업 필터링에서는 말 그대로 사용자들간, 아이템들 간 협업이 이루어진다.

아이템 기반 협업 필터링의 경우 컨텐츠 기반 필터링 방법과 유사한 듯하지만 사실은 다른 것이, 전자의 경우 사용자의 특정 아이템에 대한 구입내역/선호도/만족도를 기반으로 추천을 진행하는 것이라면, 후자는 아이템에 대한 정보를 드러내는 아이템 프로필과 유저의 성향을 기술하는 유저 프로필을 기반으로 추천을 진행하는 것이라고 이해하면 좋을 것 같다. 아이템 기반 협업 필터링에서의 특정 아이템에 대한 선호도(rating)는 다음의 수식으로 요약될 수 있다.

$$r_{xi} = \frac{\sum_{j \in N(i;x)} S_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} S_{ij}}$$

$$S_{ij} : similarity\ of\ items\ i\ and\ j \quad r_{xj} : rating\ of\ user\ x\ on\ item\ j$$

$$N(i;x): set\ items\ rated\ by\ x\ similar\ to\ i$$

지금부터는 사용자들 간 협업을 이루어 추천을 진행하는 user-based collaborative filtering에 초점을 맞춰 설명을 진행하려 한다.

사용자 기반 협업 필터링은 내가 평점을 준 영화들에 대해 전반적으로 비슷한 양상의 점수를 부여한 사용자들을 나와 '유사'하다고 판단, 그들의 정보를 통해 새로운 영화에 대한 나의 평점을 예측해내 이를 통해 추천을 구현하는 방식이라고 할 수 있겠다. 우리에게 익숙한 피어슨 상관계수(Pearson correlation coefficient)를 사용할 수도 있다고 한다.

어떤 영화를 보기 전, 우리는 주변 친구들에게 내가 보려는 영화를 본 적 있는지. 그리고 또 이들 영화에 대해서 어떻게 생각하는지를 묻곤 한다. 과거의 대화 등을 통해 나와 비슷한 영화 취향을 지닌 친구의 평이라면 그것이 내게도 비슷하게 적용 가능할 것이라는 사실을 전제로 하기 때문이다. 이 원리를 그대로 아래의 예시에 적용해보자.

2627 데마 팀원들+학회장이 각 영화에 대해 다음과 같이 평점(0~4)을 매겼다고 해보자.

	라라랜드	어벤져스	기생충	끝까지 간다	인터스텔라
유나	5	4	4	3	
서영	1	0	1		4
지현	4	4		5	3
진모		2	1	4	3
재성	4		4	4	2
남택	4	2	3		1

예를 들어, 재성 오빠가 <어벤져스>에 대해 평점을 몇 점으로 줄지 예측해본다고 하자. 가장 먼저, 코사인 유사도 계산을 통해 각 사용자가 서로 얼마나 비슷한 지에 대해 기술해야 한다.

유사도	유나	서영	지현	진모	재성	남택
유나	1	0.84	0.96	0.82	0.98	0.98
서영	0.84	1	0.61	0.84	0.63	0.47
지현	0.96	0.61	1	0.97	0.99	0.92
진모	0.82	0.84	0.97	1	0.85	0.71
재성	0.98	0.63	0.99	0.85	1	0.98
남택	0.98	0.47	0.92	0.71	0.98	1

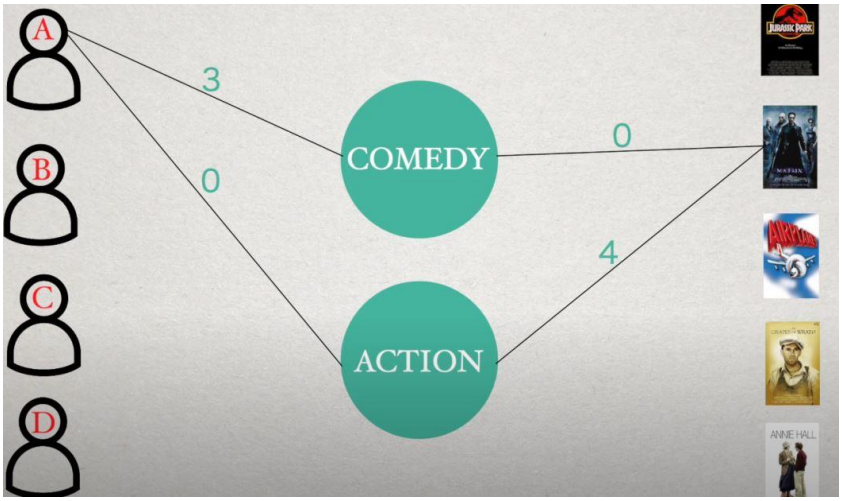
재성 오빠의 <어벤져스> 영화에 대한 평점을 예측하는 데 있어서 오빠의 평점 부여 기록에 기반하여 가장 유사한 몇 명의 점수만을 사용할 수도 있고 전체를 대상으로 하여 유사도 기반의 weighted sum값을 예측 점수로 사용할 수도 있다. 후자의 경우를 적용하여 평점을 예측해보자.

$$rating\ prediction = \frac{0.98 \times 4 + 0.63 \times 0 + 0.99 \times 4 + 0.85 \times 2 + 0.98 \times 2}{0.98 + 0.63 + 0.99 + 0.85 + 0.98} = 2.60$$

위와 같은 방법으로 계산되었을 때 평점은 2.60점으로 예측되는 것을 확인할 수 있다.

여기서 하나 질문. 사용자별로 각자 선호하는 특정 장르와 같이, 호불호 판단을 내리는 데 있어 주요한 요인(factor)을 매개로 rating matrix를 작성해볼 수는 없을까? 이와 같이 각 요인에 대한 사용자 각각의 가중치를 부여, 이를 토대로 컨텐츠 추천 방식을 운용하는 것은 꽤나 직관적인 접근법 같다. 한번 살펴보자.

예를 들어, 아래 그림에서와 같이 A라는 사용자가 코미디 장르를 3 정도로 선호하는 반면 액션 영화는 전혀 선호하지 않았을 때, 코미디적 요소를 전혀 지니지 않은 반면 액션 강도가 4로 매겨진 영화 <매트릭스>에 대해 이 사용자는 몇 점의 평점을 매길 것인지가 궁금하다.



위의 예시를 바탕으로 사용자 A가 영화 <매트릭스>에 대해 매기는 평점은 :

(특정 장르에 대한 선호도) × (해당 영화가 함유하고 있는 특정 장르의 정도)

로 요약될 수 있으며, 각 장르에 대해서 합산하면 $3 \times 0 + 0 \times 4 = \mathbf{0}$ 이라는 평점을 매길 것이라 예측할 수 있는 것이다.

모든 사용자가 모든 요인에 대해 가중치를 매기고 각 영화를 동일한 방식으로 평가한다고 생각, 문제를 확장해 보면 다음과 같이 표현할 수 있다. 마치 matrix multiplication을 실시한 것과 동일한 과정이다. 결국 사용자와 factor를 매핑한 결과를 하나의 matrix로 표현, 영화와 factor를 매핑한 정보를 또 다른 matrix에 표현하여 이들 matrix끼리의 multiplication을 시행하여 총 평점을 유추할 수 있다. 아래 구조에서와 같이 말이다.

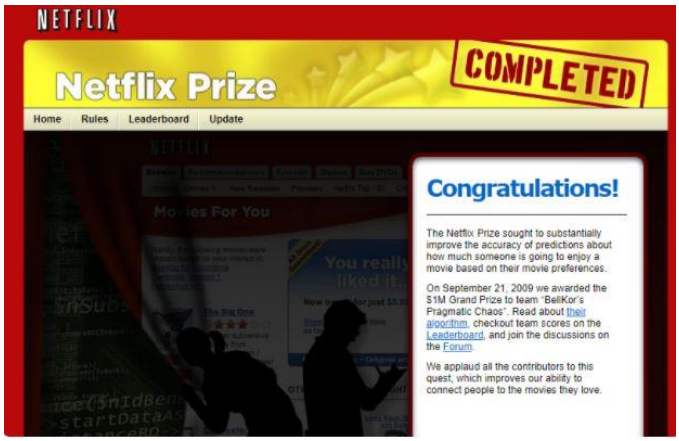
	Comedy	Action
A	3	0
B	2	2
C	4	4
D	0	4

	쥬라기공원	매트릭스	에어플레인	분노의 포도	제인 홀
Comedy	1	0	4	1	4
Action	3	4	3	2	0

	쥬라기공원	매트릭스	에어플레인	분노의 포도	제인 홀
A	3	0	12	3	12
B	8	8	14	6	8
C	16	16	28	12	16
D	12	16	12	8	0

위의 예시처럼 각 요인을 일일이 열거할 수 있다면 수동으로 처리해주어야 함에 다소 수고스러울 수는 있으나 어떤 요인들이 반영이 되는지 직관적으로 파악할 수 있다는 장점이 있다. 하지만, 겉으로 인식할 수 있는 요인들이 아닌 잠재적 요인들(latent factor)을 바탕으로 추천 평점 예측 작업을 수행하는 것이 더 정확한 결과를 산출하곤 한다. PCA 등의 기법을 통해 차원을 축소, 주요한 요인들만을 남겨놓은 채 모델링을 실시해 더욱 강건한 결과를 얻고자 하는 flow와 맥락을 같이 한다. 실제로, 잠재 요인을 이용한 협업 필터링을 구현했을 때 계산에 필요한 파라미터의 수가 확연하게 줄어든다.

잠재 요인 협업 필터링을 구현하는 가장 좋은 방법인 **Matrix Factorization(MF)**을 소개한다. 코미디나 액션과 같이 요인들을 직접 명시하는 것이 아니라, 평점 패턴으로부터 반대로 요인 벡터들을 추론하는 과정이라고 이해하면 되겠다. 이 방법을 취할 경우 높은 정확도를 보이는 것과 함께 확장성, 유연성 측면에서도 장점을 지닌다. 엄밀히 말하자면 추천 시스템에 국한해서 이 Matrix Factorization이라는 용어를 사용하는 것이 아니라 '행렬 분해'는 SVD에서와 같이 행렬을 분해하는 모든 과정을 포함하는 용어이다. 2009년 Netflix Prize 대회를 위한 알고리즘 개발 단계에서 이 표현을 언급, 대표적인 방법론으로서 제시되고 있다. (<Matrix Factorization Techniques for Recommender Systems> 논문 참고)



MF 방법은 사용자의 선호도를 명확하게 알 수 없는 implicit feedback이 주어진 상황에서 가장 효과적으로 운용된다. 'feedback이 implicit하다', 또는 'dataset이 implicit하다'는 것은 평점 matrix에 평점이 있는 아이템들보다 그 값이 0으로 매겨져 있는 데이터가 훨씬 많이 존재한다는 것을 의미한다. 아래 그림과 같이 전체 매트릭스에서 일부 cell에만 값이 있고 나머지는 0으로서 띄엄띄엄한 형태를 보이는 것을 'sparse'하다고 일컫는다.

$$\begin{pmatrix} 1.0 & 0 & 5.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.0 & 0 & 0 & 0 & 0 & 11.0 & 0 \\ 0 & 0 & 0 & 0 & 9.0 & 0 & 0 & 0 \\ 0 & 0 & 6.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.0 & 0 & 0 & 0 & 0 \\ 2.0 & 0 & 0 & 0 & 0 & 10.0 & 0 & 0 \\ 0 & 0 & 0 & 8.0 & 0 & 0 & 0 & 0 \\ 0 & 4.0 & 0 & 0 & 0 & 0 & 0 & 12.0 \end{pmatrix}$$

MF 방법은 값이 주어진, 오직 관측된 평점만을 이용하여 모델링을 실시하면서도 사용자-아이템 사이의 상호작용을 적절히 포착하여 패턴을 찾아낸다. 사용자의 평점 예측값을 구하는 산식을 아래와 같이 표현할 수 있다(여기서 전치행렬의 형태로 적어준 것을 확인하자).

$$\widehat{r_{ui}} = q_i^T p_u$$

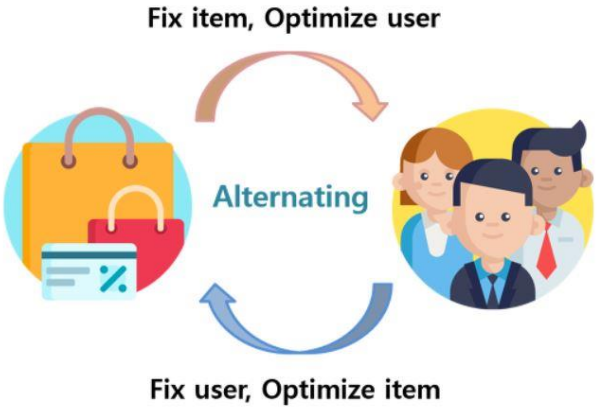
$$where\ item\ q_i, user\ \ p_u$$

이 때 잔차제곱합을 줄이는 방식에 overfitting 방지를 위한 regularization term을 더해 손실 함수를 작성해볼 수 있다.

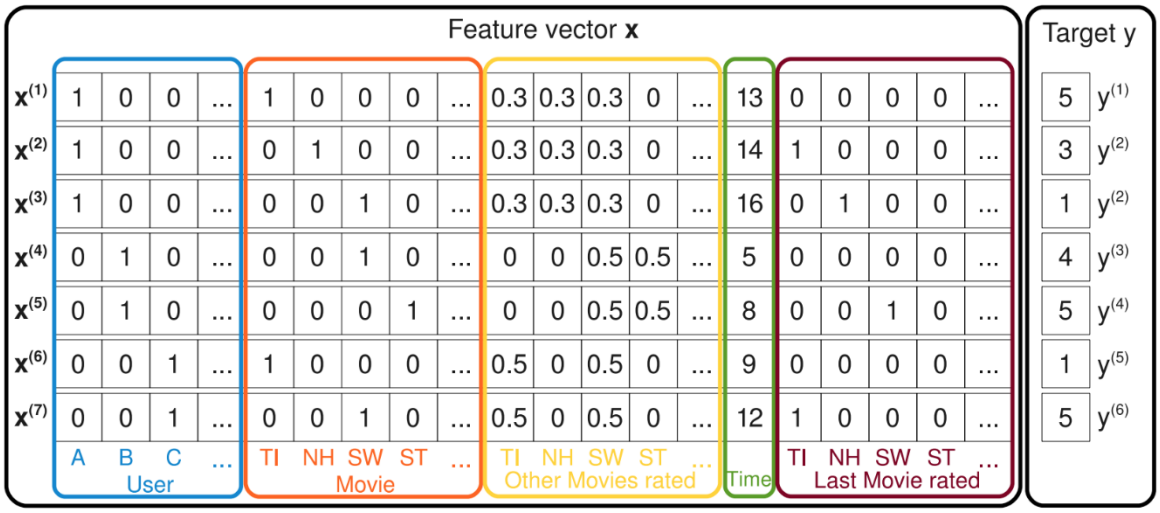
$$min_{q,p} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

λ는 cross-validation을 통해 결정할 수 있으며, 이와 같은 형태를 취했을 때의 계산 값이 minimum하게 나오는 q와 p를 찾는 것이 목표이다.

이 손실함수를 최소화하기 위한 방법으로써 지난 2주차 때 가볍게 살펴보았던 경사하강법(Stochastic Gradient Descent)를 활용할 수 있다. 또한, 손실함수가 convex하지 못함으로 인해 발생할 수 있는 문제를 보완한 Alternating Least Square(ALS) 방법을 통해 q_i와 p_u 값을 번갈아 가며 고정시켜 손실함수 값을 줄일 수도 있다.



마지막으로, Factorization Machine(FM)라는 이름의 알고리즘을 하나 더 가볍게 짚고 넘어가려 한다. MF, FM 이름은 비슷하지만 사실상 다른 모델이라고 이해하는 것이 편하다. FM은 다중회귀의 형태를 응용한 것으로, 이때 input으로 유저와 아이템을 취하고 output으로 평점을 산출하는 방식이다. MF 방법과 달리 모든 feature vector와 target rating을 y축으로서 구성하고, 각 데이터쌍(튜플)을 가로로 배치하는 것과 같은 형태이다. 아래 그림에서 볼 수 있듯 FM은 MF보다 더 많은 요소들을 고려하는데, 이처럼 풍부한 메타데이터를 사용하게 된다면 user와 item 사이의 interaction 패턴 파악을 더 쉽게 할 수 있다는 이점을 지닌다.



실습 타임

간단한 실습을 해보겠습니당! 학습 목표는 아래와 같습니당!

1. R (내장되어 있는 USArrests 정보를 사용합니당)

   -scaling을 통해 normalization을 달성하고, 결측치 처리를 해보겠습니다!

   -군집 개수를 결정하는 데 있어서 저는 kmeans()를 사용했는데요, 여러분은~~

   1) fviz_nbclust() 함수를 사용하여 군집 개수 결정, method 파라미터 갖고 놀기

   2) kmeans() 함수에서 사용할 수 있는 value 값으로는 어떤 것들이 있는지 찾아보기

   (예: kmeans()$tot.withinss)

   // 클러스터 내 분산은 최소화, 클러스터 간 분산은 최대화한다는 원리 유념하여.

이번에 부스팅 계열 모델 적합하는 내용이 패키지 과제로 출제 된다죠?

해보고 잘 안되면 질문 고고

클린업을 마무리하며~

3주간 부족한 팀장 믿고 따라와준 우리 데마팀 진모오빠 재성오빠 지현이 서영이

모두 너무 고맙고, 덕분에 저도 좋은 기운 받아서 열심히 공부할 수 있었습니당

3주간 많이 정신없고 힘들었을 텐데 잠시 중간고사 기간 동안 휴식을 취하고

(말에 어폐가 있는데… 무슨 말인지 대충 알겠죠? 찡긋) 주제분석 기간에 다시 만나요~~

코로나 시기 더 빨리 친해지지 못해 너무 아쉬운데 앞으로 자주 만나 더더더 친해지자구요~~