추알못의 슬기로운 추천생활 #2
prod. 추천 알고리즘 못 들어보신 분

글. 김희동 kt스카이라이프 AD/DX 팀

지난 추알못 시즌#1에서는 친구 녀석의 콘텐츠 추천 사례를 통해 콘텐츠 기반 필터링을 잠깐 언급하고 1세대 추천 방식과 2세대 중 연관관계분석(Association Rule Mining)의 개념과 다양한 사례를 통해 추천의 정의는 무엇이고 또 그 근거를 뒷받침해 줄 수 있는 여러 알고리즘에 대해서도 알아보았습니다. 2세대의 연관성 분석은 'What goes with what'을 규명하는 기법으로 상품과 상품 간의 상호 관계 또는 종속 관계를 찾아내는 분석법입니다. 이런 연관규칙에 대한 판단 기준은 전체 중 함께 발생하는 빈도수를 의미하는 지지도(Support)와 조건부 확률을 통해 항목 간 관련 정도를 측정하는 신뢰도(Confidence), 그리고 특정 상품이 함께 구매되는 확률 즉, 규칙과 상관없이 발생한 빈도수 가운데 함께 구매된 규칙이 얼마나 유효한지를 나타내주는 향상도(Lift) 값에 따라 상품 간의 연관성 의미를 분석해 낼 수 있다고 앞서 설명해 드린 바 있습니다. 지금부터는 본격적으로 콘텐츠 기반 필터링이 속해있는 2세대 추천시스템인 정보 필터링(Information Filtering)에 대해 설명하고자 합니다. 그보다 이에 앞서 이해를 돕기 위해 다시 한번 더 추천 시스템의 범주에 대해 상기해 보겠습니다.

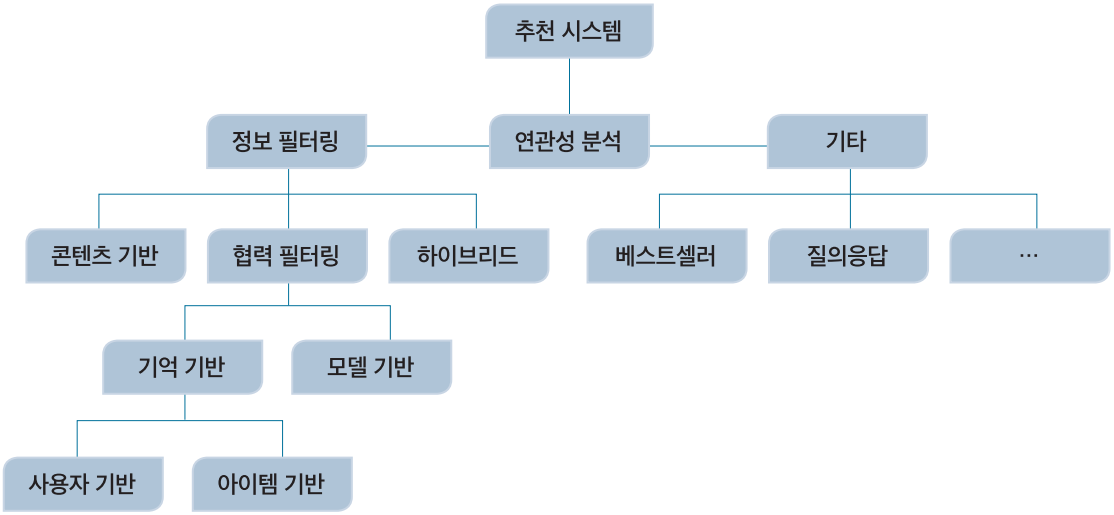


Figure 1. The Categorization of Recommender Systems, Son et al.,2015

정보 필터링은 User-Item Matrix의 많은 Item 중에서 특정 User가 관심이 갈만한 특정 Item이 추천에 활용될 수 있도록 다양한 정보를 기반으로 필터링해주는, 즉 걸러주는 것을 통칭합니다. 이때 아이템과 아이템 혹은 아이템과 사용자 선호도 간 유사성을 분석하여 이를 토대로 고객에게 아이템을 추천해 줄 것인지 아니면 특정 아이템에 대해 선호

도가 유사한 고객들은 다른 아이템에 대해서도 비슷한 선호도를 보일 것이라는 기본 가정을 바탕으로 사용자 혹은 아이템 간 유사도를 기반으로 선호도를 예측하여 추천하는 지에 따라 나뉘게 됩니다. 이렇듯 정보 필터링의 대표적인 두 가지 기법에는 [Figure 2]에서 보는 바와 같이 전자에 해당하는 콘텐츠 기반 필터링(Content-based filtering)과 후자의 협업 필터링(Collaborative filtering)이 있습니다.

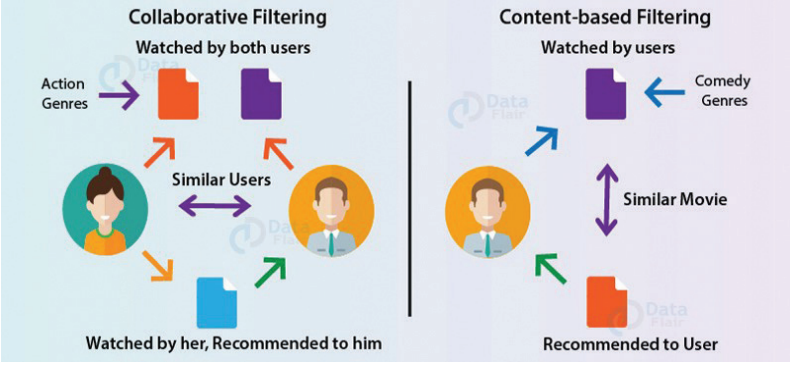


Figure 2. Content-based and Collaborative Filtering Comparison, Data Flair

먼저 콘텐츠 기반 필터링은 아래 [Figure 3]과 같이 콘텐츠 정보를 기반으로 다른 콘텐츠를 추천하는 방식으로 영화의 경우 장르, 스토리, 감독, 주연 배우 등을 데이터화하여 이와 유사한 다른 영화를 추천해 주는 방식입니다. 그렇기에 추천 대상만의 과거 구매 행태나 정보만을 독립적으로 프로파일링하면 다른 고객의 정보 유무와 상관없이 활용될 수 있는 장점이 있습니다. 그러나 구매 행태를 특정 지을 만한 데이터가 부족하고 선호도를 파악할 만한 아이템 평가점수가 없다면 콘텐츠 기반 접근방식 추천시스템을 구현하는 것 자체가 불가능해지는 치명적인 단점이 있습니다.

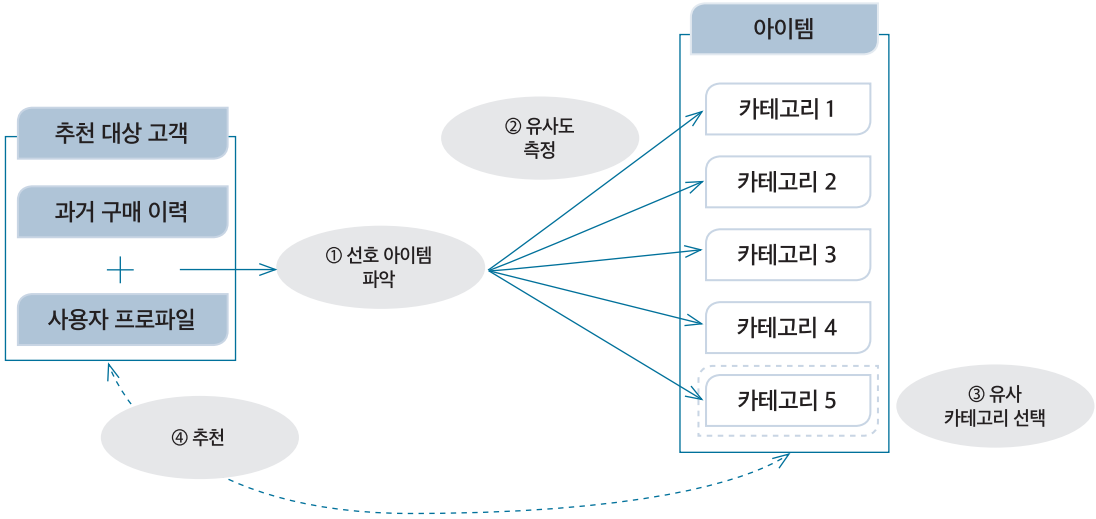


Figure 3. The Algorithm of Content-based Filtering , Son et al.,2015

협업 필터링에 비해 상대적으로 시스템이 간단하고 구현하기가 용이하다는 장점이 있으나 단순 콘텐츠 정보와 아이템 선호도만으로 다양한 형식의 항목을 걸러낼 수 없기에 프로파일을 구성하는데 한계가 있을 수밖에 없습니다. 따라서 핀셋으로 꼭 집어 분류를 할 수가 없고 그로 인해 노이즈(원치 않는 데이터가 포함되는 등)가 동반될 수 있는 단점이 있습니다. 만약 연출과 작가는 같지만 새로운 장르를 시도한 다른 드라마를 추천하였다면 제가 재밌게 봤을까란 질문을 지난 원고에서 드렸었는데 이제 그 답을 해야겠습니다. 아마도 처음에 보다가 중간에 안 볼 확률이 높습니다. 왜냐면 제가 선호도에 가중치를 더 주는 요소는 감독, 연출보다도 장르입니다. 콘텐츠 기반 필터링으로는 Feature 간 중요도를 파악할 수 없기에 3개 중에서 2개가 같지만 아쉽게도 실패입니다. 그럼에도 협업 필터링 방식에서 점수를 한 번

도 못 받은 아이템이 추천리스트에도 오르지 못하는 First rater를 해결해 줄 수 있어서 단독으로 활용되기보다는 하이브리드 등의 형태로 협업 필터링과 함께 추천시스템을 보완해 주는 역할을 하고 있습니다. 하이브리드 방식은 뒤에서 자세히 설명해 드리겠습니다. 이에 반해 협업 필터링은 추천을 해주고자 하는 수많은 사용자나 아이템에서 얻은 데이터를 분석하여 관심사 등을 자동으로 예측하고 이와 유사한 대상들을 그룹으로 분류하여 공통분모를 추출하는 방식입니다. 즉, 추천 대상이 되는 고객과 취향이 비슷한 사용자를 선정하고 그들이 선호하는 아이템을 추천 대상 고객에게 추천하므로 추천되는 아이템의 다양성을 보장할 수 있습니다. 대표적으로 사용자기반과 아이템 기반의 필터링이 있는데 상대적으로 데이터가 적고 변경이 자주 일어나는 경우 사용자기반 협업 필터링이 쓰이고 데이터가 방대하고 변경이 자주 일어나지 않는 경우에는 아이템 기반 필터링이 주로 사용됩니다.

Figure 4. The Algorithm of User-based Collaborative Filtering , Son et al.,2015

[Figure 4]는 사용자기반 협력 필터링의 기본적인 개념을 나타내고 있습니다. 추천 대상 고객이 선정되면, 구매 이력을 바탕으로 추천 대상 고객과 다른 사용자들 간의 유사도를 측정합니다. 즉, 구매한 아이템이 일치할수록 취향이 비슷한 것이므로 높은 유사도를 갖게 되며, 모든 사용자와 유사도를 측정했을 때 유사도가 가장 높은 사용자를 이웃으로 선택합니다. 예를 들어 추천 대상 고객과 취향이 가장 비슷한 사용자는, 사용자 a나 c보다 '아이템 1', '아이템 3', '아이템 5'를 구매한 '사용자 b' 입니다. 따라서 협력 필터링의 마지막 단계에서는, 아이템 추천 단계로써 유사도 측정을 통해 선택된 '사용자 b'는 구매하였으나 추천 대상 고객은 아직 구매하지 않은 '아이템 8'을 최종적으로 선택하여 추천 대상 고객에게 추천하게 됩니다.

Figure 5. The Algorithm of Item-based Collaborative Filtering , Son et al.,2015

[Figure 5]는 아이템 기반 협력 필터링의 기본 개념으로써 추천 대상 아이템을 기준으로 유사한 아이템을 선정한 뒤, 선정된 '아이템 2'는 구매하였으나 추천 대상 아이템은 구매하지 않은 '사용자 e'에게 추천 대상 아이템을 최종적으로 추천하게 됩니다. 간혹 Content-based Filtering과 Item-based Collaborative Filtering을 혼돈하시는 경우가 있습니다. 이는 두 필터링 방식 모두 아이템 간의 유사도를 따지기 때문일 걸로 추측됩니다. 그러나 Content-based Filtering은 해당 아이템의 특징을 바탕으로 하는데 반해 Item-based Collaborative Filtering은 사용자의 평가로 이뤄진다는 점에서 명확한 차이점이 있습니다. 따라서 콘텐츠 기반 필터링보다 결과가 직관적이며 항목의 구체적인 내용을 분석할 필요가 없게 됩니다. 또한 정확도(추천 성공률)가 향상되는 장점이 있을 수 있으나 이를 위해 많은 데이터가 필요로 하고 이것이 많아질수록 성공률은 올라갈 수 있으나 계산이 복잡해지고 충분한 학습데이터 부재 시 cold start 등의 단점이 있을 수 있습니다. 그리고 소위 잘나가는 제품이나 항목에만 추천이 집중되어 관심이 적은 다수는 정보조차 제공하지 못하게 되는 경우(LongTail)가 발생할 수도 있습니다.

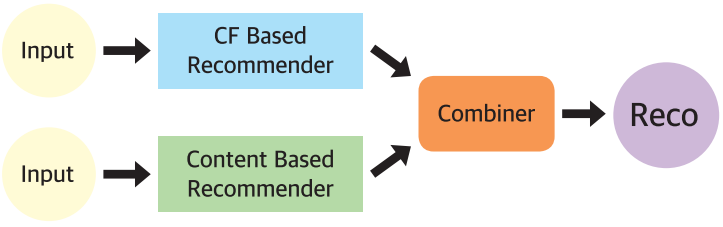


Figure 6. The Algorithm of Hybrid Filtering, Maruti

이렇듯 정보 필터링의 양대 산맥을 이루는 두 필터링 방식은 각각 장점이 있는 대신 단점 또한 분명하기에 이 둘의 장점을 살리고 단점을 줄여 시너지를 높이게끔 설계된 Hybrid Filtering 방식이 도입되었습니다. [Figure 6]은 Hybrid Filtering을 직관적으로 이해하기 쉽게 설명해 주고 있습니다. 하지만 그 내막을 자세히 들여다보면 매우 복잡합니다. 먼저 알고리즘 관점에서 본다면 다른 추천 기준을 지닌 여러 개의 알고리즘을 학습한 뒤, 각 알고리즘이 아이템 추천 점수의 가중평균 합으로 구할지 아니면 각각의 알고리즘에 사용되는 모든 변수를 하나의 알고리즘의 변수로 병합할지에 따라 달라집니다. 또한 한 알고리즘이 추천한 아이템을 다음 알고리즘의 후보로 이용할지 아니면 한 알고리즘의 추천 점수를 다른 알고리즘의 변수로 이용할지에 따라서도 천차만별의 Hybrid Filtering이 존재하게 됩니다. 다음 조합의 관점에서는 여러 추천 결과에 대해 적절한 가중치를 계산하기 위하여 초기 가중치를 모두 동일하게 설정한 뒤, 추천 결과에 대한 사용자 피드백 정보를 활용하여 가장 설명력이 좋은 가중치를 실시간으로 조절할지 아니면 데이터에 대해서 콘텐츠 기반 접근방식과 협업 필터링의 추천 점수를 40% : 60%로 선형 결합을 할지에 따라서도 달라집니다. 또 두 가지 방식을 병합하여 동시에 적용하는 것이 아니라, 구매 이력이 존재하는 사용자에 대해서는 협업 필터링을 통해 추천하고, 구매 이력이 없어서 협업 필터링 기법의 적용이 불가능한 사용자에 대해서는 사용자 프로필을 사용하여 유사도를 측정하는 기법도 있습니다. 이 방법은 새로운 고객에게 보다 효과적인 추천을 할 수 있다는 장점을 가지고 있습니다. 이렇듯 하이브리드는 협업 필터링과 콘텐츠 기반 필터링의 장점을 살리고 단점을 줄이기 위하여 다양한 방법들이 결합하게 됩니다. 따라서 추천시스템 설계 초기 단계부터 정확한 목표 설정이 필수이며 그에 따른 최적의 알고리즘과 데이터를 활용하여 어떻게 조합을 하는지 여부에 따라 성능 향상 또는 저하를 초래할 수도 있습니다.

요즘 활발히 연구되고 있는 분야는 협업 필터링입니다. 협업 필터링은 다시 메모리 기반과 모델기반으로 나뉘는데, 앞서 설명해 드린 사용자 기반과 아이템 기반 모두가 메모리 기반의 종류들이고 다음으로 등장할 모델 기반은 3세대 추천시스템이라 할 수 있습니다. 이 모델 기반은 메모리 기반에서 활용하는 군집화, 분류, 예측 등으로 유사도를 계산하는데 있어 기계학습 또는 데이터마이닝 기법을 활용하는 것을 뜻합니다. 나이브 베이지안 모델(Naïve Bayes), 선형 회귀 분석(Linear Regression), 마코프 결정 프로세스(Markov Process) 등 다양한 확률 및 통계 이론에 근거한 알고리즘이 사용되어 협력 필터링 적용 시에 부가적으로 발생하는 많은 문제점을 보완해 주는 역할

을 하고 있습니다. 모델 기반의 추천시스템에는 Autoencoder(AE), Multi-layer Perception(MLP), Convolution Neural Network(CNN), Recurrent Neural Network(RNN), Deep Semantic Similarity Model(DSSM), Restricted Boltzmann Machine(RBM), Neural Autoregressive Distribution Estimation(NADE), Generative Adversarial Network(GAN) 등 다양한 딥러닝을 기반으로 개발되고 있습니다. 이 가운데 오토인코더는 [Figure 7]에서 보는 바와 같이 신경망의 입출력 부분을 동일하게 설정, 효율적으로 은닉층을 학습하는 Artificial Neural Network(ANN)로서 후술할 행렬분해와 비슷하게 입력 데이터를 원래의 차원(Dimension)보다 적은 데이터로 줄인 후에, 변환한 데이터를 기반으로 원래의 데이터를 복원하는 Unsupervised Learning에 속합니다. 만약 은닉층을 z라 한다면 z는 입력 데이터(X1~X5)를 인코더 네트워크에 통과시켜 얻어진 압축된 값이 되고 이 압축된 z 벡터로부터 입력 데이터와 같은 크기의 출력값(X1~X5)을 생성하게 됩니다. 위 예제는 5차원 입력 데이터가 오토인코더를 통해 3차원으로 차원이 축소된 후 완벽히 재현(입력X1~5=출력X1~5)된 경우입니다. 실제는 에러가 존재하게 되는데 인코더를 통해 압축된 입력값과 디코더를 통해 복원된 출력값 사이의 차이가 Loss가 되고 AE의 알고리즘은 이 Loss가 0이 되도록 학습하게 되는 원리입니다.

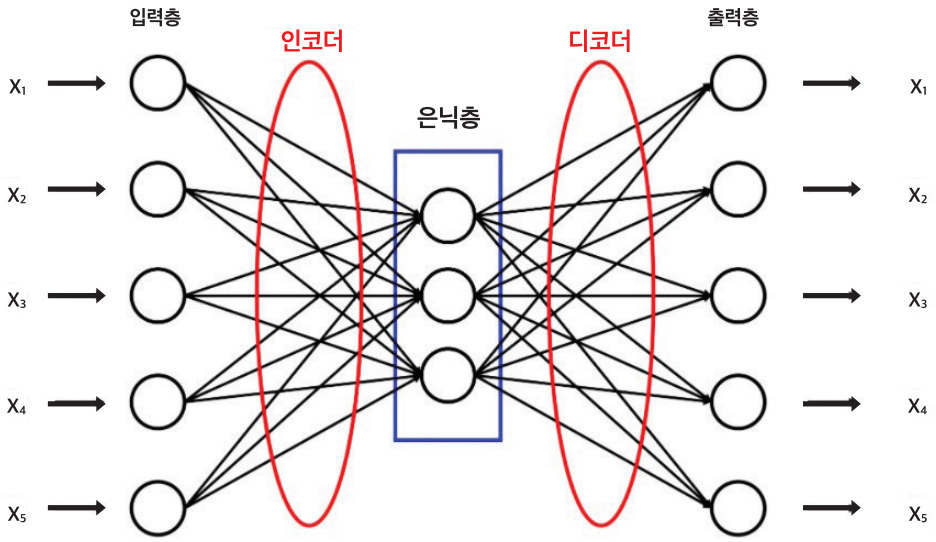


Figure 7. The Algorithm of Autoencoder, Github

이 AE는 비교적 가장 최근에 등장한 비지도 학습 알고리즘 중 하나이며 관련 논문 또한 양이 상당하니 관심 있는 분들은 꼭 한번 참조하시면 좋을 듯합니다. AE에서 Denoising AE, Contractive AE, 또 Variational AE VAE에서 Conditional VAE 등으로 계속 발전해 나가고 있습니다.

이제 추천시스템의 모든 종류에 대해 알아봤으니 필터링을 위해 필요한 유사도(Similarity) 알고리즘과 협업 필터링의 숙원과도 같은 데이터 희소성(Sparsity)의 문제를 해결하기 위한 머신러닝 기법에 대하여 살펴보겠습니다. 유사도 알고리즘에도 자카드(Jaccard), 맨하탄(Manhattan), 마할노비스(Mahalanobis) 등 무수히 많지만 가장 기본이라 할 수 있는 거리기반의 유클리디안(Euclidean Distance)과 각도 기반의 코사인(Cosine Similarity) 유사도에 대하여 알아보겠습니다. 또 데이터 희소성(Sparsity)과 관련해서는 차원축소는 무엇이고 왜 필요한지 그리고 핵심이라 할 수 있는 행렬분해(Matrix Factorization)가 기다리고 있습니다.

유사도 알고리즘은 두 객체가 얼마나 닮았는지를 가늠해주는 척도입니다. 두 객체가 떨어진 거리로 판단할지, 또는 두 벡터 사이의 각도를 측정할지는 상황에 맞게 정해줘야 합니다. [Figure 8]에서 보듯이 유클리디안 거리는 2차원 평면에

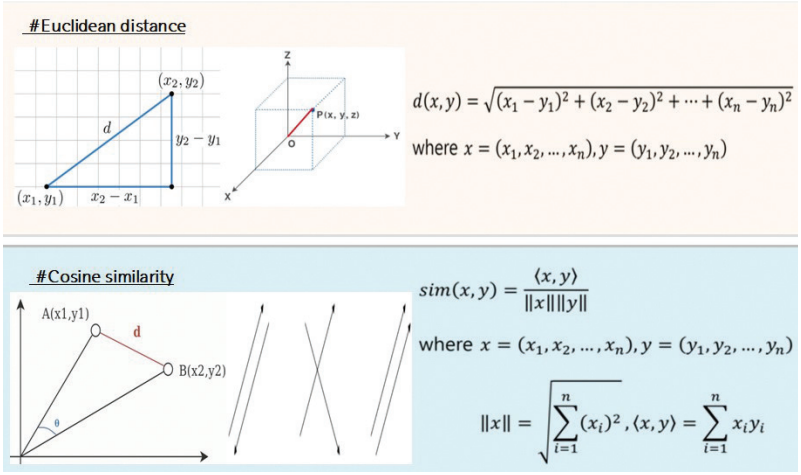


Figure 8. Euclidean Distance and Cosine Similarity Comparison, WIKI

서 두 지점 사이의 거리입니다. 이는 피타고라스 정리에서 대각선의 길이와도 같습니다. 이를 공간좌표(x, y, z)로 확장해보면 OP의 길이에 해당하고 일반화한다면 수식과 같습니다. 코사인 유사도는 두 벡터 A와 B 사이의 앵글 θ에 따라서 -1~1까지의 값을 갖게 됩니다. 유사도 sim(x, y)는 θ를 구하는 공식이고 (x, y)의 내적(Inner Product)이 됩니다. 글로는 쉽게 이해가 가지 않을 수 있어 이번에도 예제를 들어보겠습니다.

Empirical Example(a.k.a. Mentor Hunting)

Which class should Q take next semester? Then, who is much more suitable for track choice and detail recommendation?

The MOT Faculty → Graduate Student ↓				
D1(9th AI Track)	3	2	0	2
D2(9th IE Track)	1	2	3	0
D3(9th ETM Track)	2	2	2	2
Q(10.5th ? Track)	1	5	0	0

Figure 9. The Empirical Example of Similarity used in Decision Making, Author

그림의 실증적 사례 1)은 제가 직접 겪었던 일로 대학원에 진학하여 선택과목이나 트랙을 정해야 할 때 유용하게 쓰일 수 있을 거 같아 소개해 드립니다. 일명 멘토 찾기인데 신입생이 들어와 조언을 해줄 수 있는 선배로 누가 가장 적합한지에 대해 유사도 알고리즘을 활용해 보았습니다. 사진은 교수진이고 D1~D3는 9기 선배, 그리고 Q는 10.5기(가을 입학) 신입생입니다. 참고로 D1이 접니다. AI 트랙을 선택하여 '인공지능 통계수학', '알고리즘적 사고와 사회현상의 이해', '의사결정 지원을 위한 인공지능의 활용' 수업을 듣기 위해 해당 교수님 과목을 수강하였습니다. 점수의 의미는 전체 수강한 학점 중 사진 속 교수님의 과목을 얼마나 들었는지를 나타내주는 가중치가 적용된 수치입니다. D2와 D3는 각각 다른 트랙을 선택하여 이수한 과목이 다르고 Q는 2학기 안으로 정해야 하는 트랙 선택도 못 하였고 따라서 Syllabus나 Curriculum과 같은 단순 교과목 정보만으로 수강신청을 한 상태입니다. Q는 트랙에 관한 조언도 얻고 싶고 선수과목 등 관련 세부정보도 궁금합니다. 과연 누가 Q의 멘토로서 가장 적합하다고 결론을 낼 수 있을까요?

Euclidean distance between D1 & Q: $\sqrt{(3-1)^2+(2-5)^2+(0-0)^2+(2-0)^2}=\sqrt{17}\fallingdotseq 4.123$

Euclidean distance between D2 & Q: $\sqrt{(1-1)^2+(2-5)^2+(3-0)^2+(0-0)^2}=\sqrt{18}\fallingdotseq 4.243$

Euclidean distance between D3 & Q: $\sqrt{(2-1)^2+(2-5)^2+(2-0)^2+(2-0)^2}=\sqrt{18}\fallingdotseq 4.243$

Cosine similarity between D1 & Q: $\cos\theta(Q,D_1)=\frac{(1\times3)+(5\times2)+(0\times0)+(0\times2)}{\sqrt{1^2+5^2}\sqrt{3^2+2^2+0^2+2^2}}=\frac{13}{\sqrt{26}\sqrt{17}}\fallingdotseq 0.618$

Cosine similarity between D2 & Q: $\cos\theta(Q,D_2)=\frac{(1\times1)+(5\times2)+(0\times3)+(0\times0)}{\sqrt{1^2+5^2}\sqrt{1^2+2^2+3^2+0^2}}=\frac{11}{\sqrt{26}\sqrt{14}}\fallingdotseq 0.577$

Cosine similarity between D3 & Q: $\cos\theta(Q,D_3)=\frac{(1\times2)+(5\times2)+(0\times2)+(0\times0)}{\sqrt{1^2+5^2}\sqrt{2^2+2^2+2^2+2^2}}=\frac{12}{\sqrt{26}\sqrt{16}}\fallingdotseq 0.588$

Solve. 1 Euclidean Distance and Cosine Similarity

[Figure 8] 공식에 대입하여 수기로 풀어보면 [Solve 1]과 같은 유사도 값이 나옵니다. 유클리드 거리에 따라서 D1이 Q와 가장 가깝다는 것을 알 수 있고 D2와 D3는 값이 같게 나왔습니다. 끝까지 우열을 가려야 직성이 풀리니 이번엔 코사인 유사도를 계산해 봤습니다. 이번에도 D1이 가장 가깝게 나왔고 D3이 D2보다 약간 더 유사하다는 결과가 나왔습니다. 사실 학생-교수 매트릭스가 4차원이었기에 직접 풀 수 있었지, 전 과목으로 확장한다면 엄두가 안 날 겁니다. 그래서 계산값이 맞는지 Python을 확인해 봐야겠습니다. [Solve 1]과 값이 같습니다.

```
In [1]:
import numpy as np
def dist(x,y): #Euclid distance definition
    return np.sqrt(np.sum((x-y)**2))

D1 = np.array((3,2,0,2))
D2 = np.array((1,2,3,0))
D3 = np.array((2,2,2,2))
Q = np.array((1,5,0,0))

print(dist(D1,Q)) #Euclid distance between D1 & Q
print(dist(D2,Q)) #Euclid distance between D2 & Q
print(dist(D3,Q)) #Euclid distance between D3 & Q

4.123105625617661
4.242640687119285
4.242640687119285
```

```
In [2]:
from numpy import dot
from numpy.linalg import norm
import numpy as np
def cos_sim(A, B): #Cosine similarity definition
      return dot(A, B)/(norm(A)*norm(B))
D1=np.array([3,2,0,2])
D2=np.array([1,2,3,0])
D3=np.array([2,2,2,2])
Q=np.array([1,5,0,0])
print(cos_sim(D1, Q)) #Cosine similarity between D1 & Q
print(cos_sim(D2, Q)) #Cosine similarity between D2 & Q
print(cos_sim(D3, Q)) #Cosine similarity between D3 & Q

0.6183469424008424
0.5765566601970551
0.5883484054145521
```

Code 1. The calculation of Similarity

이제 끝이 보이기 시작합니다. 조금만 더 힘을 내 보겠습니다. 젤 중요한 협업 필터링의 꽃과도 같은 행렬분해입니다. 행렬분해라... 지금껏 수학에서 봤던 분해는 중학교 때 배운 인수분해가 전부였는데 이게 누구십니까? "니가 왜 거기서 나와?" 아니 추천을 하는데 행렬은 왜 나오고 분해는 또 여기서 무슨 뜻인지 헷갈려 하는 분들이 많을 거라 추측됩니다. 저 역시 첨엔 그랬으니까요. 행렬분해가 Matrix Factorization인데 여기서 Factorization은 어디서 많이 보시지 않았나요? 맞습니다. Factor가 인수란 뜻이니 직역하자면 인수화하는 것? 정도가 되겠네요. 고차다항식의 합의 형태를 인수들의 곱의 형태로 분해하는 것을 말하는데 바로 인수분해입니다. 하는 이유는 분명합니다. 고차다항식을 알아보기 쉽도록 해서 해를 구하기 위해서죠. 행렬분해도 마찬가지입니다. 행렬분해는 행렬을 특정한 구조를 가진 다른 행렬의 곱으로 나타내는 것을 의미합니다. 이를 통해 선형 방정식의 해를 구하거나, 행렬 계산을 효율적으로 하기 위해 사용됩니다. 하지만 추천시스템에서는 사뭇 다른 목적이 있습니다. 그게 뭘까요?

행렬분해에는 대표적인 예로 주성분분석(Principal Component Analysis, PCA)과 특이값 분해(Singular Value Decomposition, SVD) 그리고 비음수행렬분해(Non-negative Factorization, NMF)가 있습니다. 지면 관계상 PCA까지 말씀드리고 SVD와 NMF는 각각의 차이점을 비교하며 다음 호 맨 마지막인 시즌#3에서 마저 설명드리겠습니다. 차원축소의 핵심은 고유값 분해(Eigenvalue Decomposition)라 할 수 있습니다. PCA가 이에 해당하고 후술할 SVD의 특이값 분해 또한 특이벡터의 고유값을 분해하기에 그렇습니다. PCA를 한 문장으로 요약하자면 분석하고자 하는 데이터의 분산을 최대한 보존하면서 서로 직교하는 새 축을 찾아, 고차원 공간의 표본들을 연관성이 없는 저차원의 공간으로 선형변환하는 차원축소 방법론입니다. 아직 이해가 안 가신다고요? 괜찮습니다. 다음 [Figure 10]은 Matthias Scholz의 생물정보학(Bioinformatics) 관련 박사학위 논문에서 발췌한 것으로 이는 3차원 이상의 고차원 데이터를 차원축소를 통해 시각화하였을 때 얼마나 효과적인지를 나타내고 있습니다. 왼쪽 그래프는 3차원 공간의 유전자 발현 정보(Gene Expression Data)를 보여주고 있고, 오른쪽 그래프는 이를 원래 변수(Original Variables), 여기서는 Gene 1부터 3까지의 변수를 선형조합(Linear Combination)으로 이뤄진 주성분 1(Principal Component 1)과 주성분 2(Principal Component 2)의 Subspace로 차원 축소된 결과입니다.

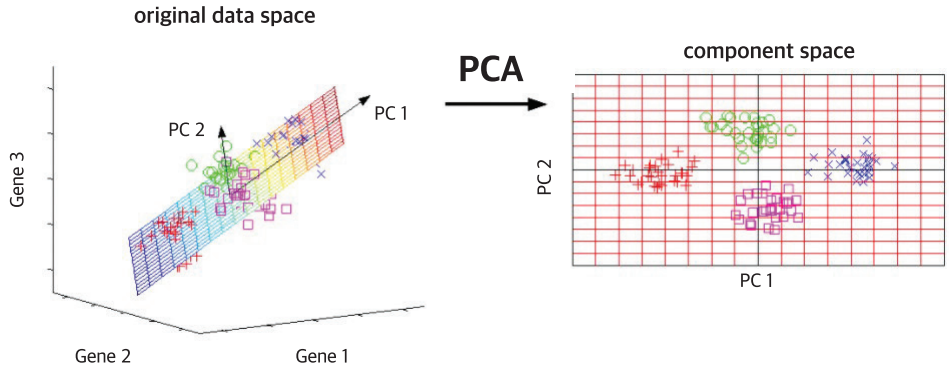


Figure 10. The Principle of PCA Process, Matthias Scholz

다시 말해 PCA를 사용하여 Data Set의 가장 높은 분산을 최적으로 설명하는 PC 1과 PC 2의 선형조합으로 2차원 Subspace로 변환해 줄 수 있다면 이 공간을 회전하여 오른쪽 그래프와 같이 표시됩니다. 그러면 이와 같은 차원축소와 데이터 시각화를 통해 4개의 샘플로 명확히 식별이 가능하게 되어 해당 실험 조건에 관한 정성적(Qualitative) 결론을 쉽게 도출할 수 있게 되는 것입니다. 감이 조금 오셨나요? 아직도 이해가 안 될 수 있습니다. 그럼 [Figure 10]을 바탕으로 앞서 언급한 PCA의 정의를 단락별 의미로 쪼개어 찬찬히 들여다보겠습니다. ① 분산을 최대한 보존하면서 → 데이터들을 새로운 축으로 프로젝션해야 하는데 분산이 가장 최대가 되게끔 유지하면서 ② 서로 직교하는 새 축을 찾아 → 주성분 간의 서로 겹쳐서 발생하는 불필요함 즉, 다중공선성 문제를 고려하여 주성분(Principal Component) 끼리는 서로 Uncorrelated 되게 ③ 고차원 공간의 표본들을 연관성이 없는 저차원의 공간으로 → p개 변수를 k개 변수로, where p > k ④ 선형변환하는 차원축소 방법론 → 모든 행렬은 선형변환(Linear Transform) 가능, where 직교행렬(Orthogonal)의 선형변환 = 회전변환(Rotate) and 대각행렬(Diagonal, Σ)의 선형변환 = 스케일변환(Stretch)하는 차원축소 방법론입니다. 그럼 이해를 돕기 위해 수식으로 살펴봐야겠습니다.

- C : covariance matrix of x
- $C = P\Sigma P^T$ (P: orthogonal, Σ: diagonal)

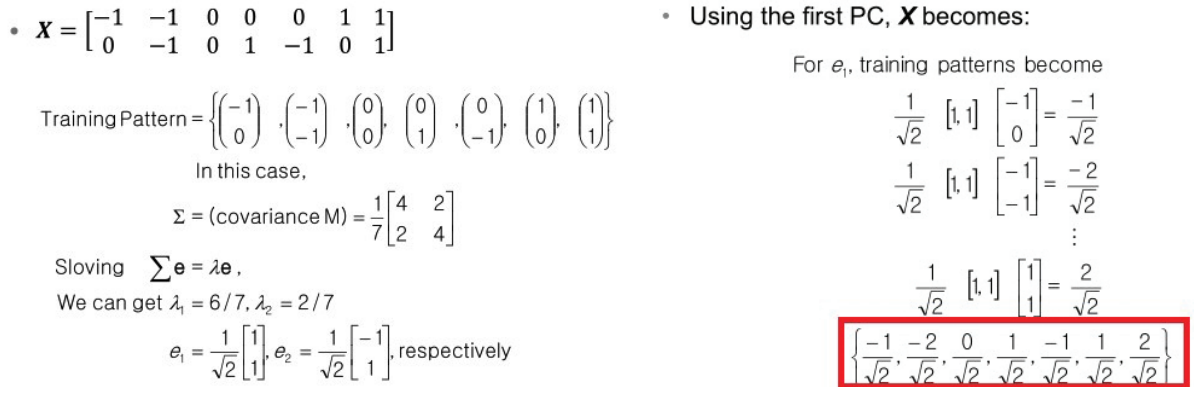
$$C = \begin{pmatrix} e_1 & \cdots & e_n \end{pmatrix} \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix} \begin{pmatrix} e_1^T \\ \vdots \\ e_n^T \end{pmatrix}$$

- P : n×n orthogonal matrix
- Σ : n×n diagonal matrix
- $Ce_i = \lambda_i e_i$
  - e_i : eigenvector of C, direction of variance
  - λ_i : eigenvalue, e_i 방향으로의 분산
  - $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$

Figure11. The Formula of PCA, WIKI

수식 [Figure 11]에서 C는 x란 행렬의 공분산(Covariance) 행렬입니다. 이때 정방행렬(Square matrix) C를 선형변환으로 봤을 때, 선형변환 C에 의한 변환 결과가 자기 자신의 상수 배가 되는 0이 아닌 벡터를 고유벡터(Eigenvector)라고 하고, 이 상수배 값을 고유값(Eigenvalue)이라고 합니다. 고유값, 고유벡터는 정방행렬에 대해서만 정의됩니다. 다시 말해, 정방행렬 C에 대해서 $Ce_i = \lambda e_i$를 만족하는 0이 아닌 열벡터 e_i를 고유벡터, 상수 λ를 고유값이라고 합니다. 여기서 λe_i에 단위행렬(Identity Matrix, 곱해도 자기 자신이 나오는 행렬)을 곱해주고 좌변으로 넘겨 식을 정리하면 $(C-\lambda I)e_i=0$이 됩니다. e_i 즉, 고유벡터는 0벡터가 될 수 없다고 정의하였기 때문에 $(C-\lambda I)=0$이 되어야 하고 2×2 행렬의 경우 구성 성분 ad-dc=0을 만족하는 행렬식(Determinant)으로부터 고유값 λ와 고유벡터 e_i를 구하게 되고 이 둘의 곱의 형태로 공분산 행렬 C를 분해하게 되는 원리입니다. 역시나 글로는 이해가 충분치 않은 것 같아 예제를 들어보겠습니다.

예 2) X에는 2차원의 Original Data Space 두 평면 Xi=[-1, -1, 0, 0, 0, 1, 1] Xj=[0, -1, 0, 1, -1, 0, 1] 축에 총 7개의 Data Set이 있을 때, 분산을 최대한 유지하면서 새로운 1차원 Principal Component 축의 Subspace로 투영한다면 즉, 어떻게 차원을 축소할 수 있을까요?



Solve. 2 The Eigenvalue Decomposition of PCA

수기로 풀어야 한다면 일단 공분산 행렬 C를 구해야 합니다. Xi-Xj 공분산은 X1의 각 성분(Xi1~Xi7)에서 X1 전체 평균을 뺀 값과 Xj의 각 성분(Xj1~Xj7)에서 Xj 전체 평균을 뺀 값을 서로 곱해주고 총 성분이 7개씩 있으므로 7로 나눈 값이 됩니다. 따라서 계산 편의상 Xi와 Xj의 평균이 0인 예제를 들었습니다. 그럼 2×7 행렬과 이 행렬의 전치행렬(Transpose) 7×2 형태의 행렬을 곱하고 7로 나눠주면 공분산 행렬은 2×2 형태가 되고 그 값은 [Solve. 2]에서 Σ가 됩니다. 이제 이 공분산 행렬을 행렬분해를 통해서 고유값을 구하고 그로부터 고유벡터를 찾으면 됩니다. 두 평면이니까 2차원이 되겠고 따라서 2차 방정식의 고유값은 2개가 나오게 됩니다. (C-λI)=0이 되어야 하니 ad-dc=0을 만족하는 행렬식(Determinant)으로부터 각각 λ_1(=6/7)과 λ_2(=2/7)의 고유값이 나오고 각각의 고유벡터는 λ_1일 때 e_1과 λ_2일 때 e_2가 됩니다. 여기서 고유값은 데이터의 변동성의 정도를 가장 잘 나타내주는 순서대로 나열되며 차원 축소는 p개 변수를 k개 변수로 줄여야 하니까 전체 고유값 p개 중에서 가장 높은 k개를 선택해 줌으로써 차원이 줄어들게 됩니다. 이 예제에서는 2차원이므로 가장 높은 고유값 6/7을 취해서 원래의 X를 새로운 1차원의 서브스페이스로 선형변환이 됩니다. PCA는 데이터 선택(Selection)이 아닌 추출(Extraction)이기 때문에 원래로 복원할 수 없고 λ_1(=6/7)과 λ_2(=2/7)의 고유값의 Sum은 8/7(=6/7+2/7)이 되고 이중에서 λ_1을 취했기 때문에 총 분산의 설명역(Fraction of Variance Explained)은 75%{=(6/7)/(6/7+2/7)}가 됩니다. 즉, 2차원의 변수를 1차원으로 줄이면서 원래의 특성을 3/4만큼 나타내 줄 수 있다는 뜻이 됩니다. 사실 2×2의 정방행렬이라 2차 방정식을 직접 풀 수 있었지 3×3부터는 행렬식이 복잡해지고 인수분해의 해도 정수가 아닌 실수(소수점)면 근의 공식으로도 풀기 싫어집니다. 그럼 Python으로 확인해보겠습니다.

[Solve 2]의 붉은색으로 칠한 결과값과 [Code 2]의 첫 번째 고유값을 토대로 주성분 분석을 통해 차원축소를 한 결과와 동일함을 알 수 있습니다. 근데 여기서 의문점 하나가 더 있을 수 있습니다. 위 예2)는 쉬운 이해를 돕기 위해 2차원을 1차원으로 줄여 보았는데 필드에서는 이보다 훨씬 많은 변수를 사용하고 그에 따라 차원 수가 엄청 많아질 텐데 어디까지 얼마만큼 차원을 줄여도 원래의 데이터의 특성을 잘 보여줄지 여부입니다. 아무리 분석을 하고자 하는 도메인의 지식이 충분하다 할지라도 임의적으로 변수의 수(차원)를 줄이다가는 자칫 원래의 데이터가 내포하고 있는 의미를 왜곡할 수 있기에 Python의 사이킷런(Sklearn)을 Import 하여 쉽게 해결할 수 있습니다. Python은 다양한 Library

```
import numpy as np
import numpy, linalg as linalg
X_transpose = np.array([[-1,-1,0,0,0,1,1], [0,-1,0,1,-1,0,1]])
X = np.transpose(X_transpose)
X_mean = np.array([np.mean(X[:,0]), np.mean(X[:,1])])
X_norm = X-X_mean
X_norm_transpose = np.transpose(X_norm)
C= np.matmul(X_norm_transpose, X_norm)
C=1/7*C
D, V = np.linalg.eig(C)
X_new = np.matmul(V[:,0], X_norm_transpose)
print('eigen value:₩n',D)
print('eigen vector:₩n',V)
print('Using the first PC:₩n',X_new)
print('PCA Variance:₩n',X_new.var())
print('PCA Variance explained:₩n',(6/7)/(6/7+2/7))
```

```
eigen value:
 [0.85714286 0.28571429]
eigen vector:
 [[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
Using the first PC:
[-0.70710678 -1.41421356  0.          0.70710678 -0.70710678  0.70710678
 1.41421356]
PCA Variance:
 0.857142857142857
PCA Variance explained:
 0.75
```

Code 2. The calculation of Eigenvalue Decomposition

를 제공하고 있다하였는데 Scikit-learn은 머신러닝에 특화된 모듈입니다. 이 Sklearn으로 우선 데이터의 분산을 0-1까지 스케일을 조정하고 다시 가로축은 주성분 수, 세로축은 정규화된 0-1까지의 누적 분산비율을 표시해주는 그래프를 그릴 수 있습니다. 예를 들어 PCA로 설명되는 분산이 95% 이상이 되게 한다면 Python 코드로는 pca = PCA(n_components = 0.95)라 입력하면 되는데 95%가 정형화된 값은 아닙니다. 그래서 명확한 문제 정의와 그에 따른 모델 설계가 Domain Knowledge에 의해 영향을 받게 됩니다.

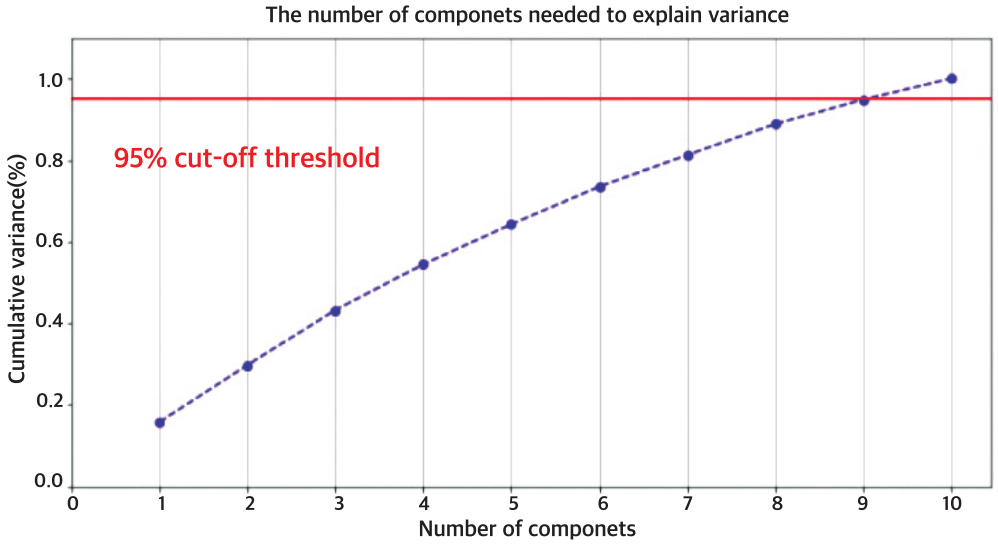


Figure 12. The Reasonable Selection of PC Number, Bartosz Mikulski

위 [Figure 12] 특정 주성분 수에 따른 누적 분산을 보여주고 있습니다. 예시와 같이 95%의 원본 데이터를 설명해 줄 수 있는 차원축소를 원한다면 이 경우에는 9개의 주성분을 선택해야 합니다. 만약 80%만 설명해줘도 상관없다 한다면 7개만 선택해도 되고 이는 2개의 변수를 줄여줘 2차원 더 줄어드는 효과가 있게 됩니다.

이것으로써 시즌#2에서는 2세대 추천시스템인 정보 필터링(Information Filtering)의 종류와 이와 함께 사용되는 여러 유사도 알고리즘에 대하여 알아보았습니다. 그중에서 협업 필터링에서 추천시스템 알고리즘의 핵심이라 할 수 있는 행렬분해(Matrix Factorization)를 설명하면서 비교적 단순한 주성분 분석 PCA에 대해 먼저 살펴보았습니다.
다음 시즌#3에서는 마지막 남은 특이값 분해 SVD와 비음수행렬분해 NMF 알고리즘의 원리는 무엇이고 또 추천시스템에 어떻게 활용될 수 있는지에 대해 지금까지와 마찬가지로 여러 사례를 통해 Python으로 구현해 보겠습니다. 그럼 '추알못의 슬기로운 추천생활 #3'에서 다시 만나요.