Lecture 9: Document Clustering and Topic Modeling

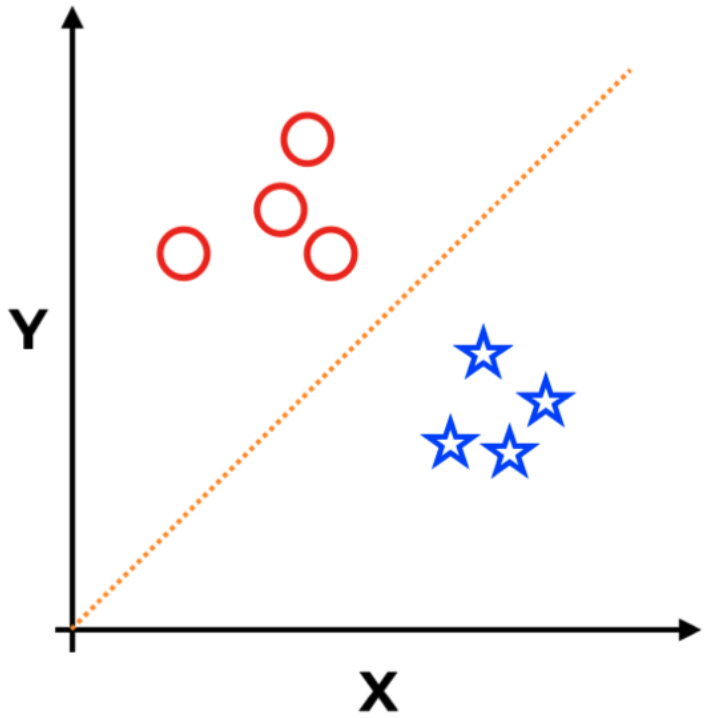
군집화 (Clustering)

군집화란?

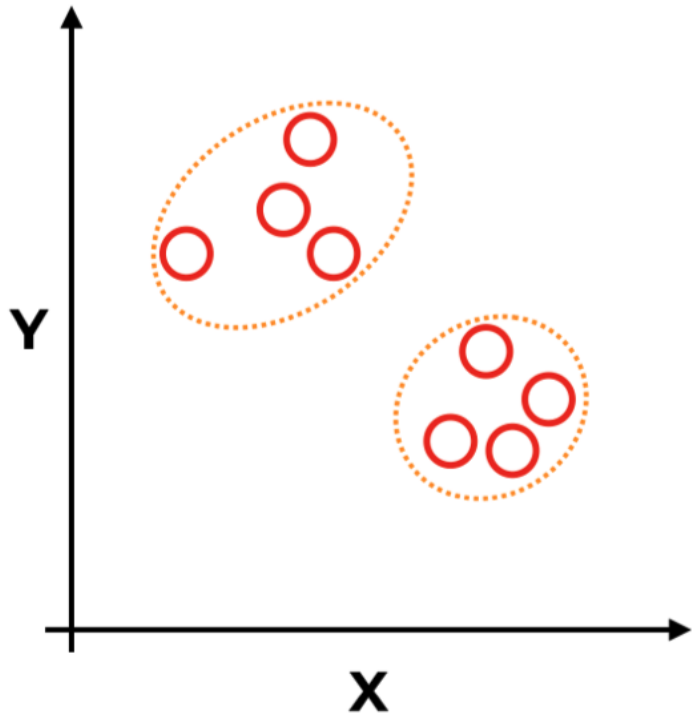
- 비지도학습(unsupervised learning)으로 데이터에서 자연스럽게 분류되는 그룹을 찾아내는 방법
- 지도학습(supervised learning) vs 비지도학습
  - 지도학습: y값(타깃값)이 사전에 정해진 경우 지도학습이며 (y=O or y=X), 타깃값을 알고 있는 데이터를이 용해 새로운 객체의 타깃값을 예측할 수 있는 패턴을 찾아내는 것
  - 비지도학습: y값이 정해지지 않고 자율적으로 학습하는 경우 비지도 학습이라 하며, 타깃 변수에 신경쓰지 않고 데이터 집합에 있는 어떤 규칙성을 찾으려는 것

군집화 (Clustering)

지도학습 *(Supervised Learning)*



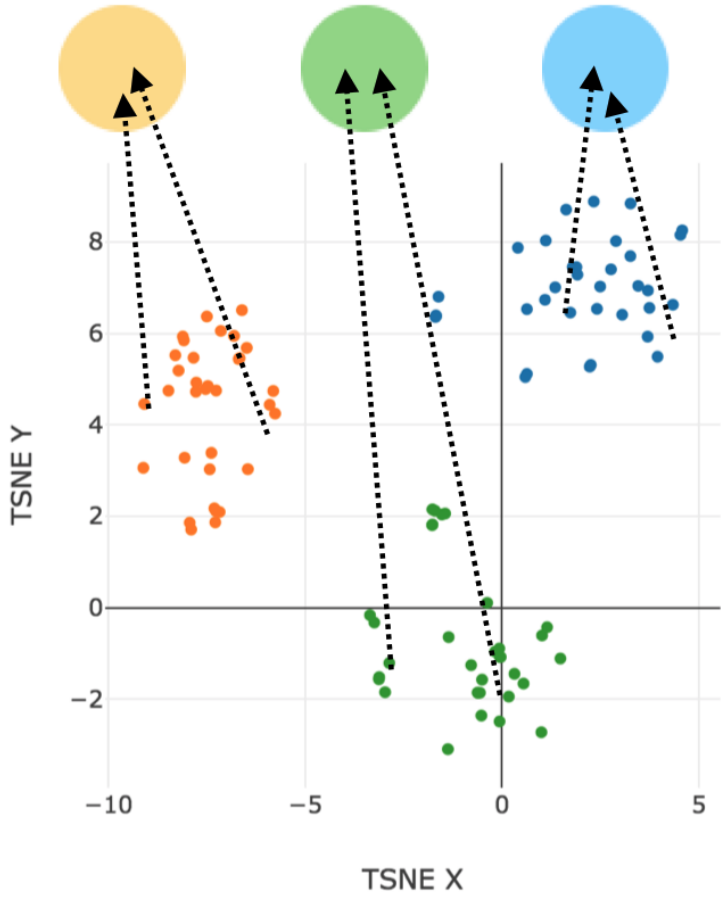
비지도학습 *(Unsupervised Learning)*

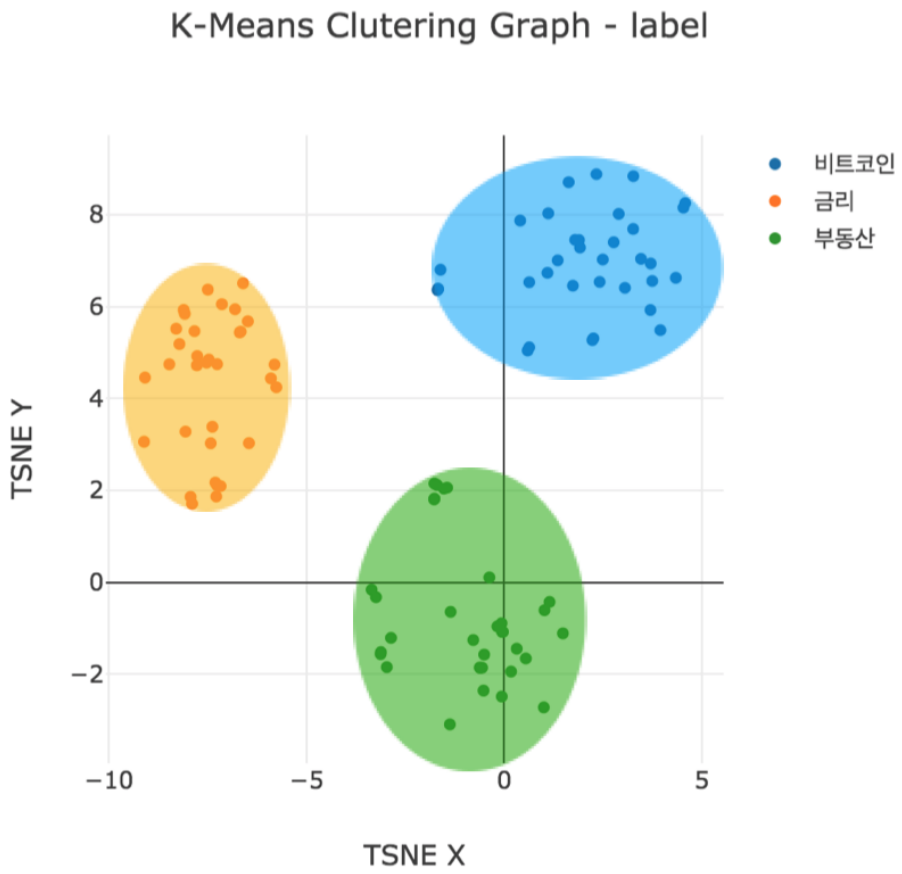


문서 군집화

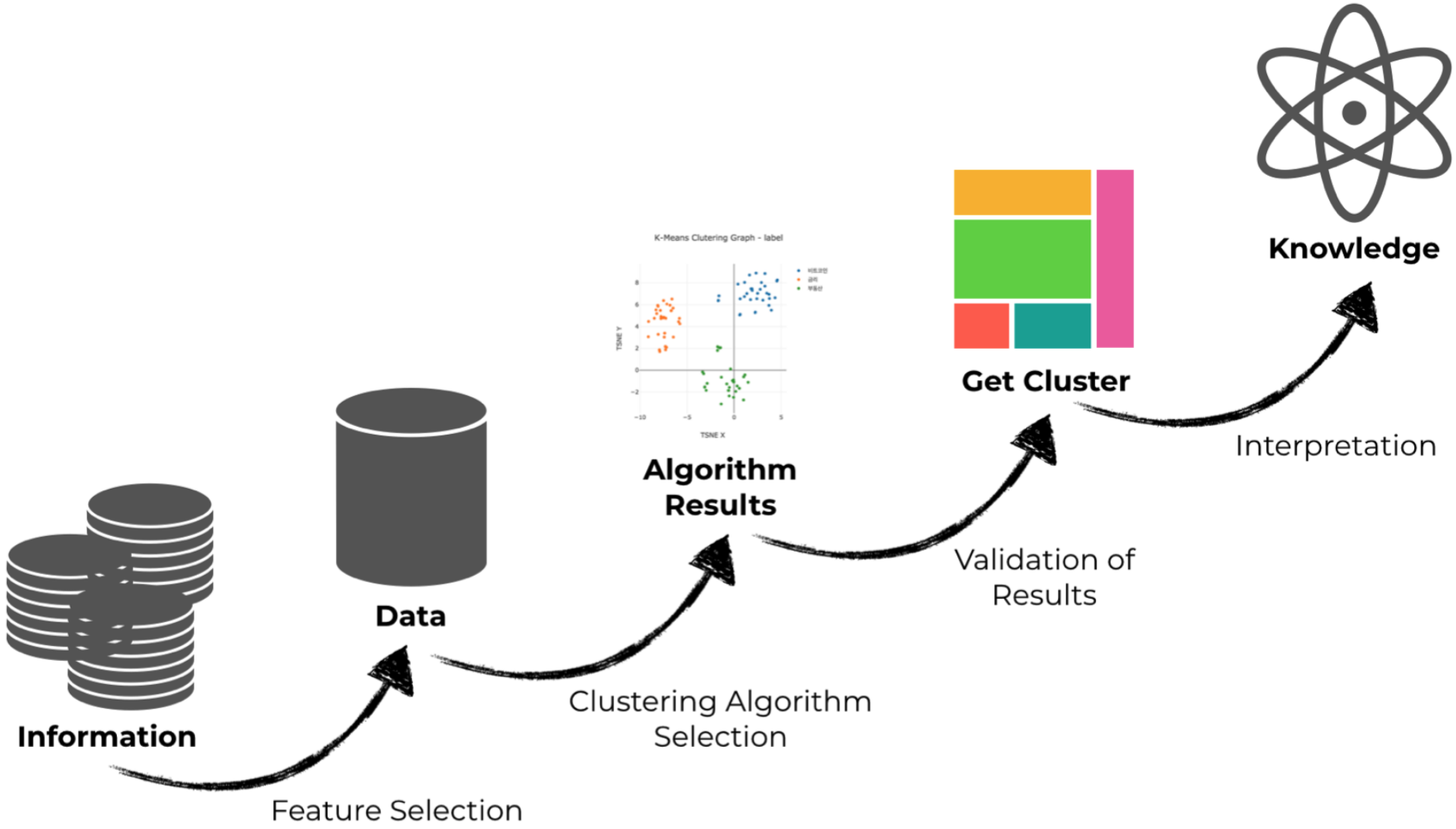
- 문서들 사이에서 발견되는 자연스러운 그룹(군집)을 발견하고 주제를 제시하는 방법
- 분류 vs 군집화
  - 분류 (Classification) : 문서의 집합을 이미 정해진 유형의 개수와 속성에 따라서 분류하는 방법
  - 군집화 : 사전에 유형의 개수와 속성이 알려지지 않은 상태로 그룹을 발견하는 방법

문서 군집화


8
6
4
2
0
−2
−10
−5
0
5
TSNE Y
TSNE X


K-Means Clutering Graph - label
비트코인
금리
부동산
8
6
4
2
0
−2
−10
−5
0
5
TSNE Y
TSNE X

문서를 이해하기 위한 과정



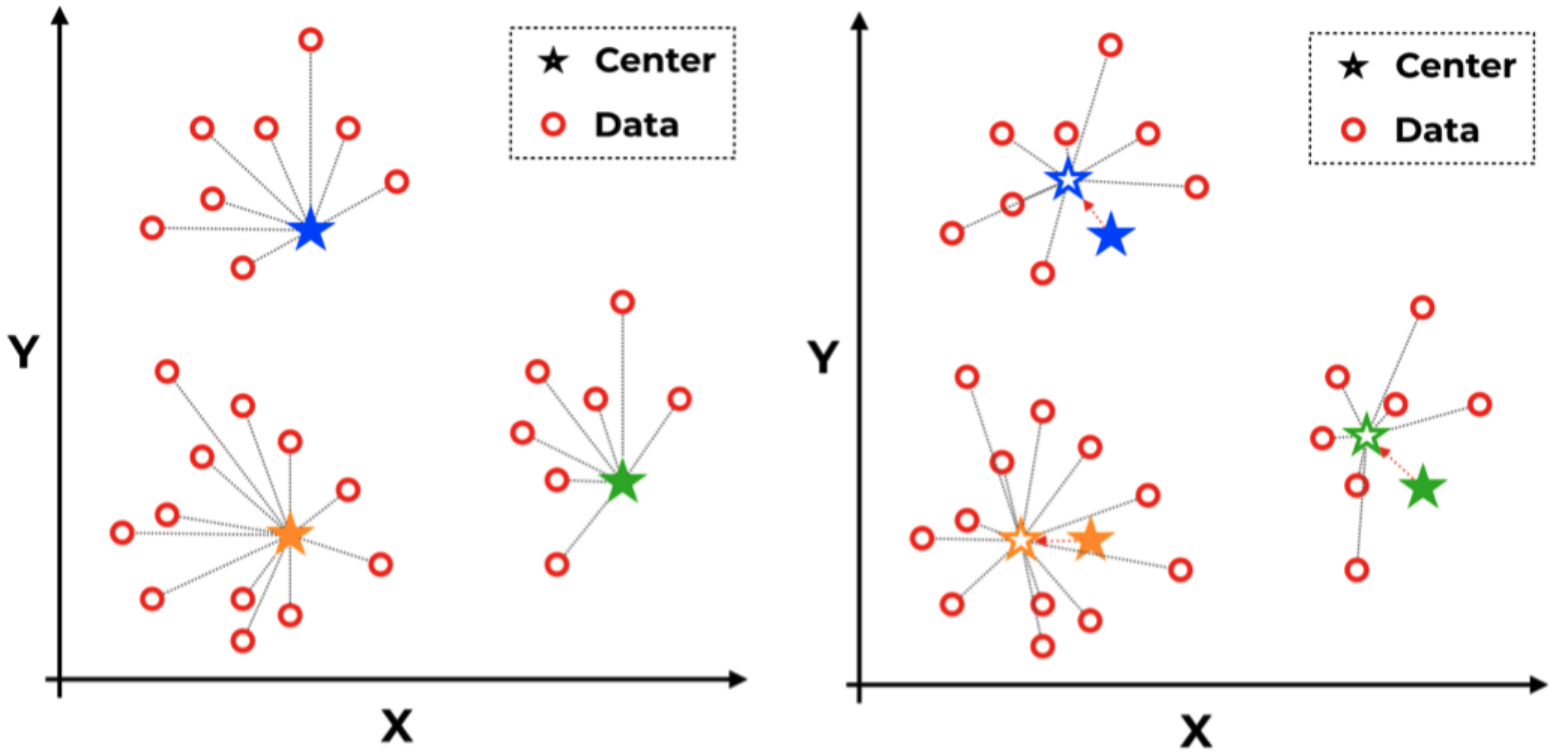
Clustering

- 군집화 방법들은 각 객체들의 유사도(거리) 정보를 이용 유사한 객체를 하나의 군집으로 분류
- 좋은 군집에 대한 기준은 다양하지만 공통적으로 “군집 내 객체들은 비슷하며, 군집 간 객체들은 이질적“임을 추구
- 객체의 표현과 거리를 이용하는 방식에 따라 다양한 방법 존재
    - centroids models
    - connectivity models
    - density models
    - graph based models
- 근본적으로 데이터의 representation 과 dissimilarity measure 가 잘 정의되어야 함

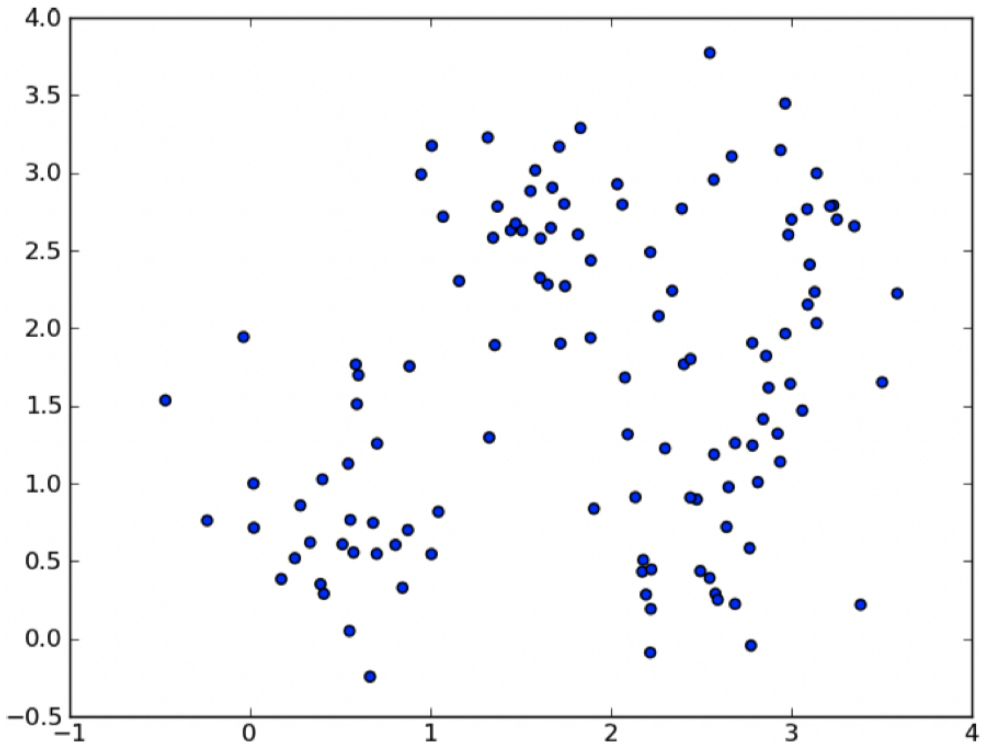
k-평균 군집 (K-means Clustering)

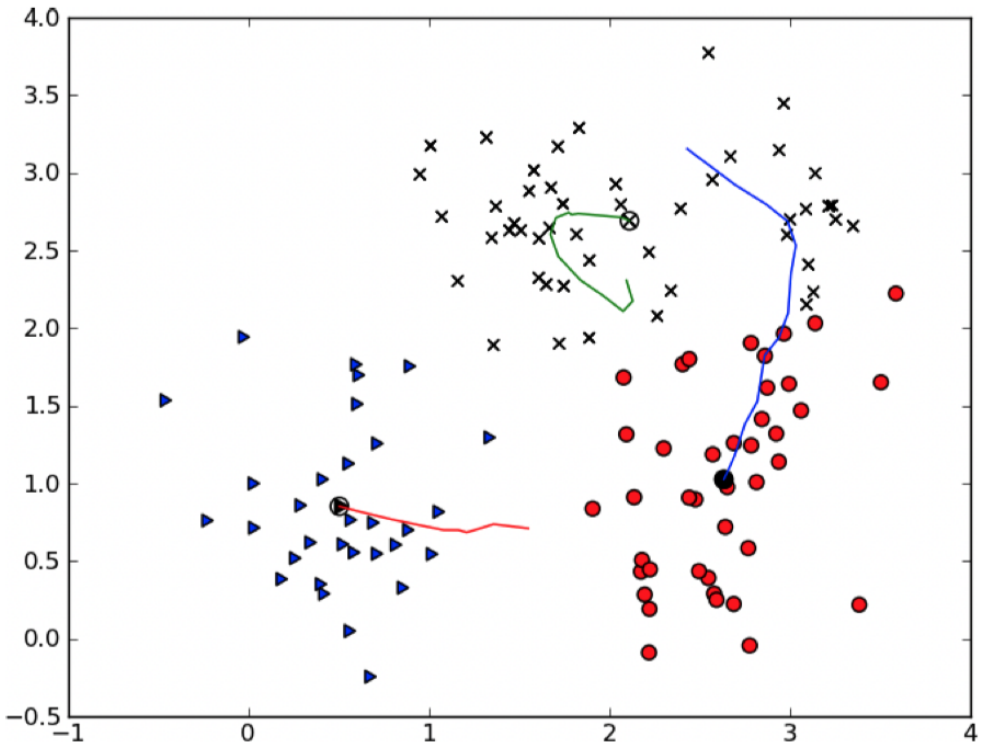
- 중점 주변 군집화 중 가장 널리 사용되며, 찾아내려는 군집 k개를 파악 (k=군집수)
- 평균은 각 군집의 중점이며, 군집에 들어있는 객체들의 각 차원 별 산술평균값으로 표현
- k-평균알고리즘의3단계
  - Step 1 : 임의로 k개의 데이터 포인트를 시드로 선택
  - Step 2 : 각 레코드를 가장 가까운 시드에 배정
  - Step 3 : 군집의 중심점 찾기 (다시 계산)
- 군집 중점이 이동하면 각점마다 어느 군집에 속하는지 다시 계산해야 하고, 각 점이 속한 군집을 다시 계산한 후에는 군집의 중점을 다시 계산
- 더 이상 군집에 변화가 생기지 않을 때까지 Step 2와 Step 3 과정을 반복

k-평균 군집 (K-means Clustering)



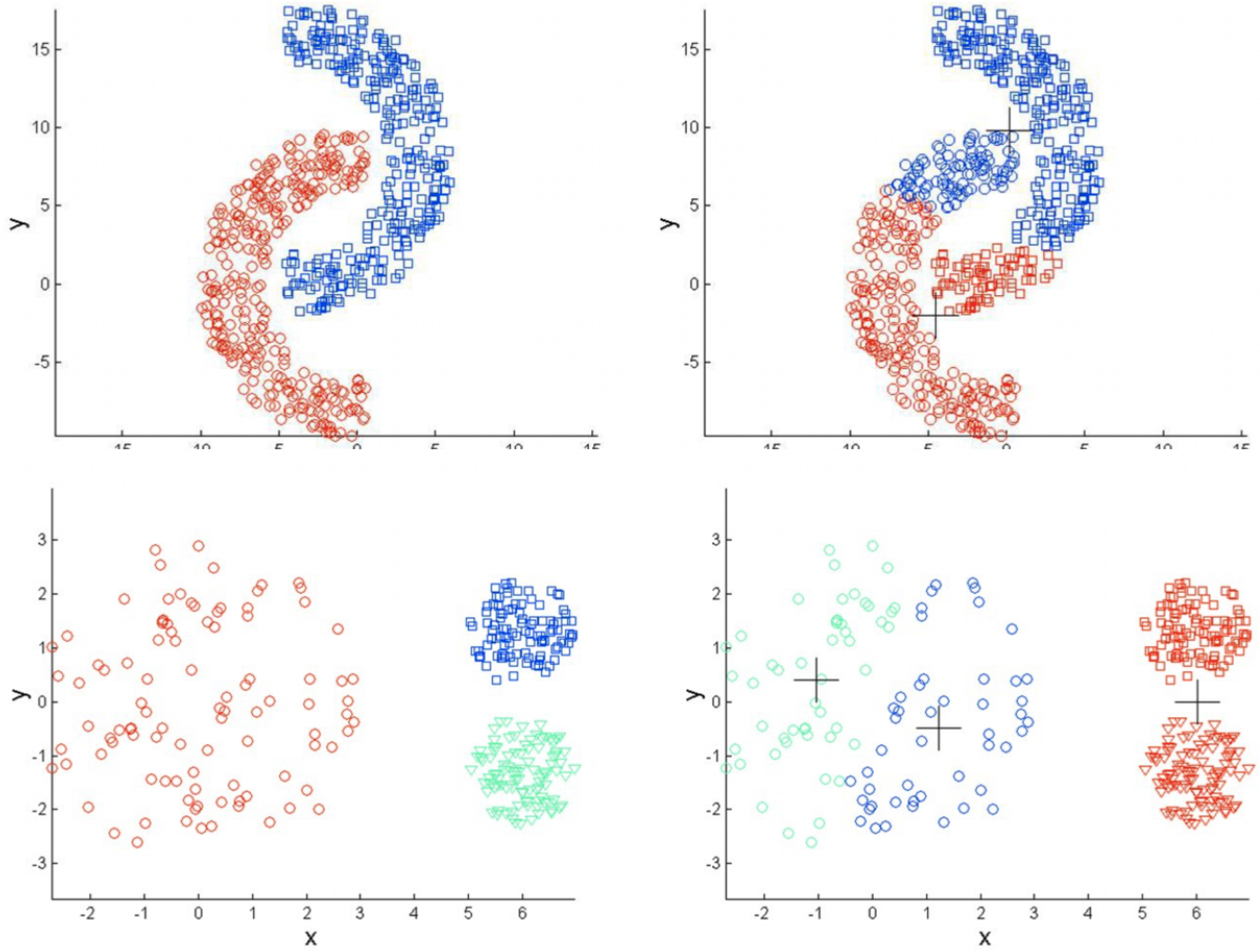
k-평균 군집 (K-means Clustering)





k-평균 군집의 한계점

- 구형이 아닌 데이터 분포, 밀도가 다른 군집



k-평균 군집의 한계점

- k-means은 고차원 데이터에는 의미가 없다.
  - 고차원 데이터에서는 가까운 점의 거리는 0 이지만,
  - 조금만 멀어져도 대부분의 점들 간 거리의 값이 비슷
  - k-means은 저차원 데이터에 적합
- 특히 sparse data + Cosine 에서는 대부분의 거리가 1
  - 비싼 random sampling

k-평균 군집의 한계점

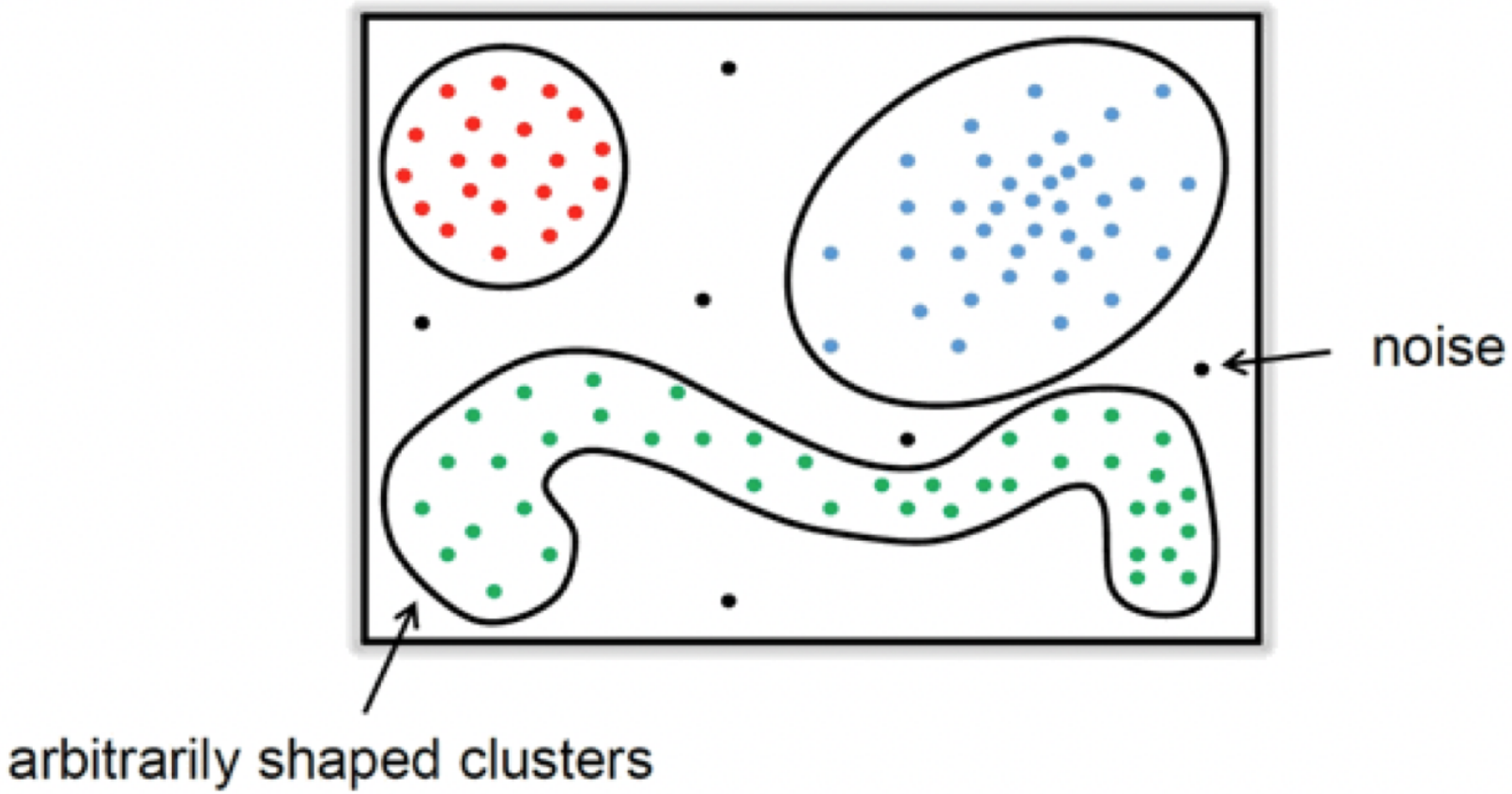
> 하루의 뉴스 데이터 (30,091 개)의 pairwise distance 예시

- 고차원의 데이터에서는 비슷하다는 것 외의 거리는 의미가 없다.

Cosine distance	Percentage
0.00 ~ 0.10	0.31%
0.10 ~ 0.20	0.31%
0.20 ~ 0.30	0.32%
0.30 ~ 0.40	0.19%
0.40 ~ 0.50	0.30%
0.50 ~ 0.60	0.32%
0.60 ~ 0.70	0.54%
0.70 ~ 0.80	2.11%
0.80 ~ 0.90	**10.22%**
0.90 ~ 1.00	**85.39%**

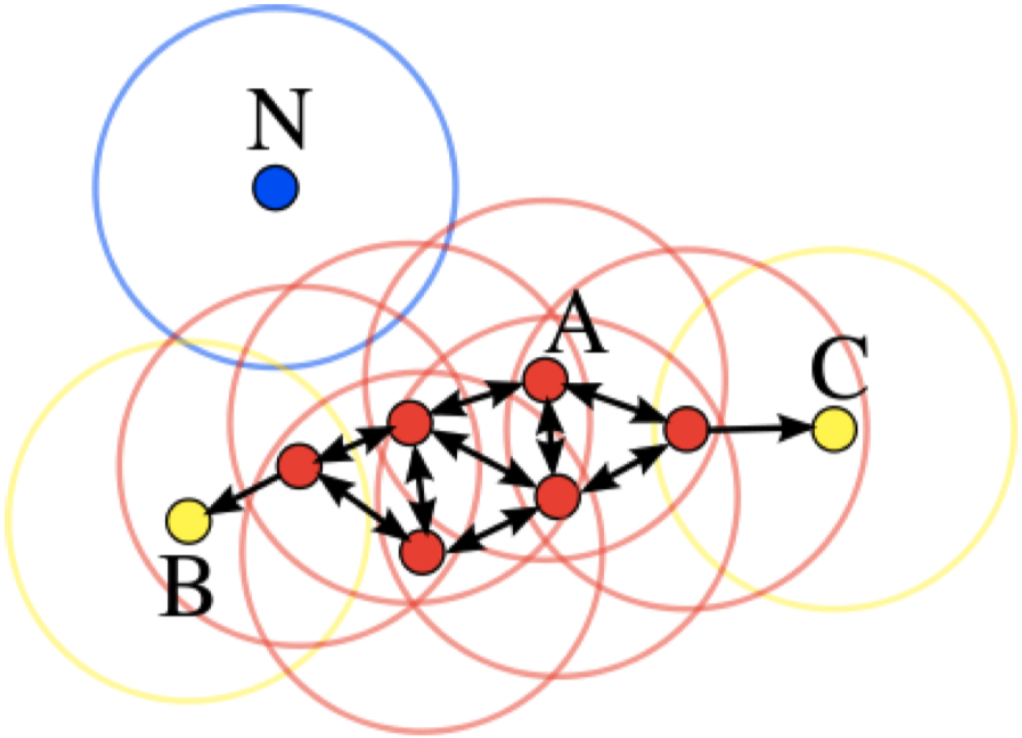
Density-Based Clustering (밀도 기반 클러스터링)

- 데이터의 분포와 밀도 (densitty)를 고려하여 클러스터를 구성하는 방법
- 구형이 아닌 임의의 모양으로 생긴 클러스터도 잘 찾을 수 있음
- 클러스터링 과정에서 노이즈를 제거하는 것이 가능



Density-Based Spatial Clustering of Applications with Noise

- 핵심 데이터 : 임의의 점 p에서 부터 거리 e 이하의 점이 m(=3)개 이상
- 이웃 데이터 : 임이의 점 p와의 거리가 e 이하인 데이터
- 외각 데이터 : 핵심 데이터는 아니지만, 임이의 핵심 데이터의 이웃 데이터인 점
- 노이즈 데이터 : 핵심 데이터도 아니고 외각데이터도 아닌 점



군집의 평가 (Validation) - Silhouette Score

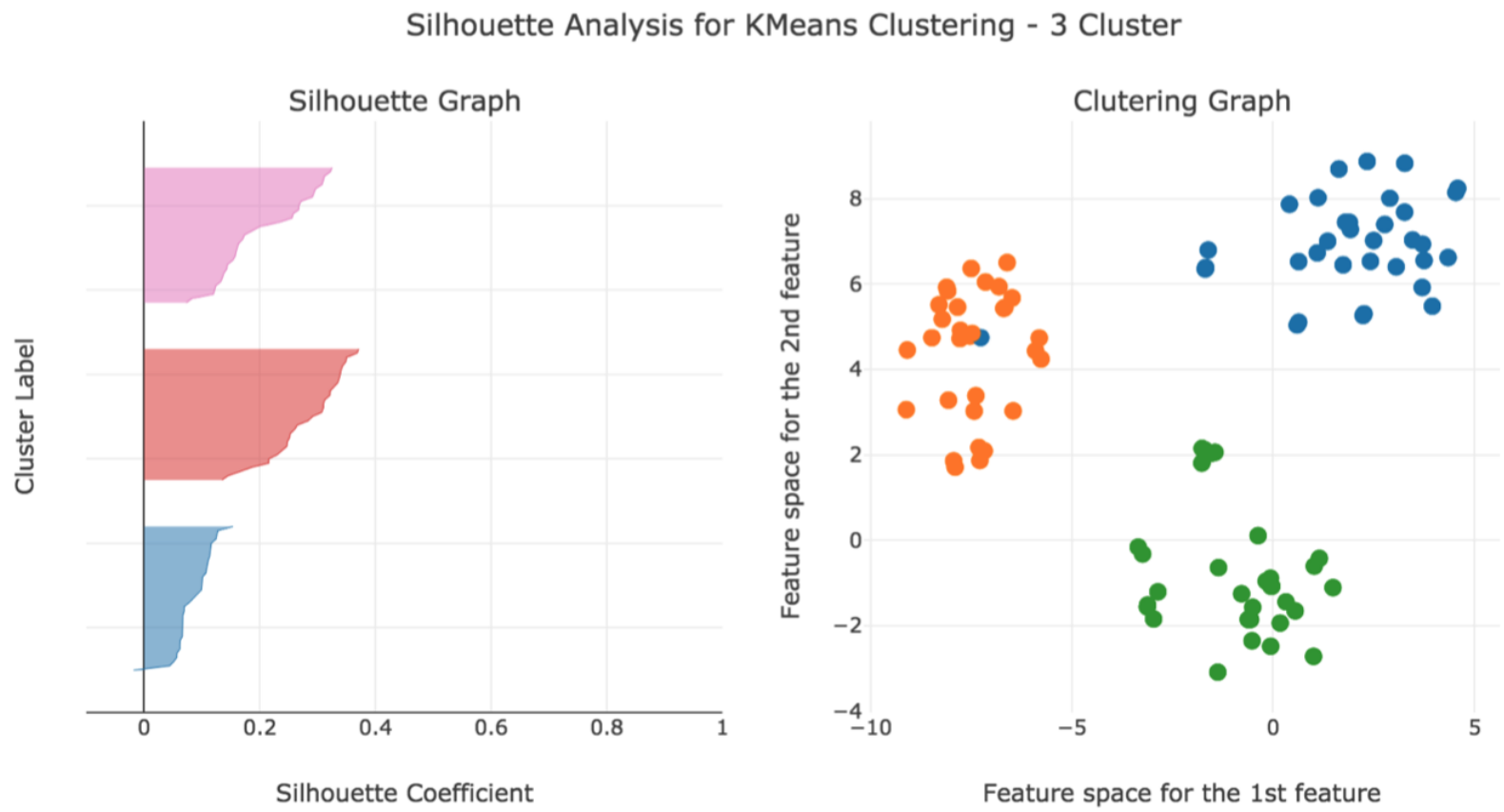
- 클러스터 내의 일관성과 유효성을 검증하는 척도 중 하나로, 각 객체가 얼마나 잘 분류되었는가를 판단 하고 시각화하기 위해 활용됨
- 다른 클러스터와 비교하여 객체가 자체 클러스터와 얼마나 비슷한지 (cohesion) 또는 분리되어 있는 지 (separation)에 대한 척도
- Silhouette 은 군집화 품질의 척도로, 적절한 k 를 정하는데 이용

$$s(x) = \frac{b - a}{\max(a, b)}, \qquad s(X) = mean\big(s(x)\big)$$

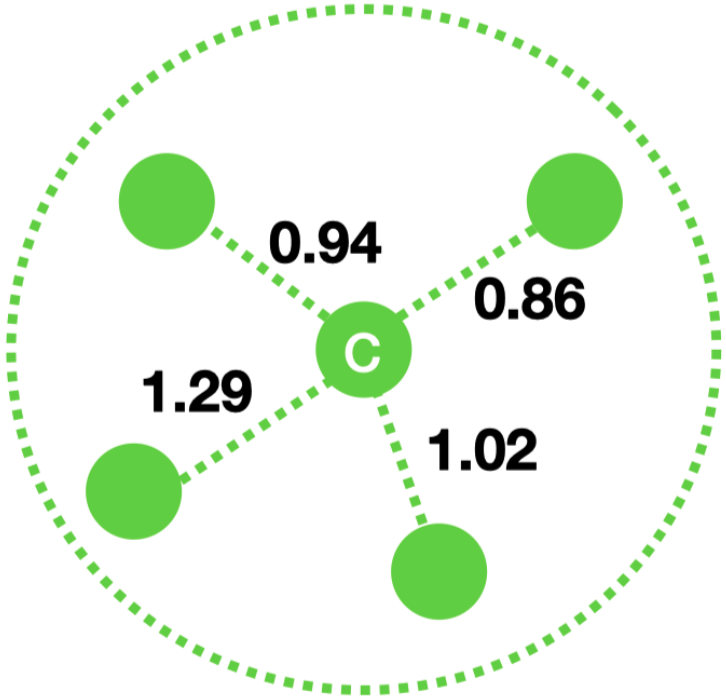
a: mean distance between a sample and all other points in the same class

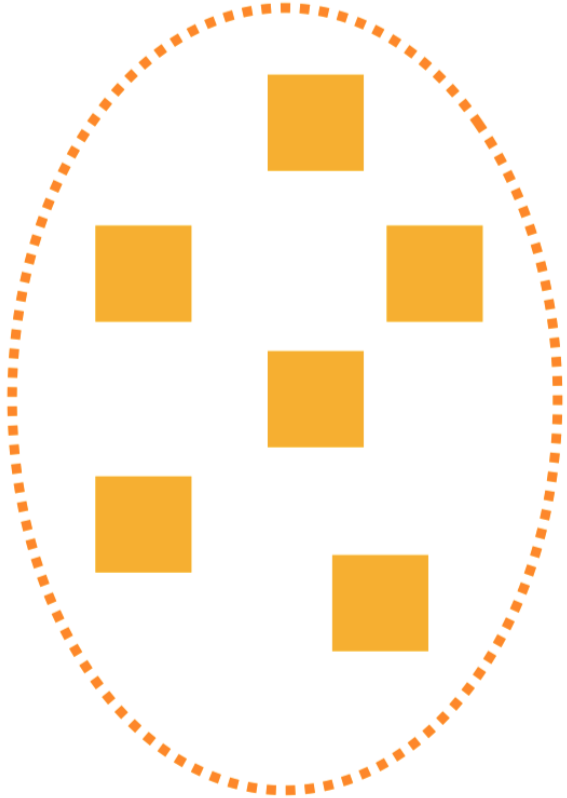
b: mean distance between a sample and all other points in the next nearest class

Silhouette Score

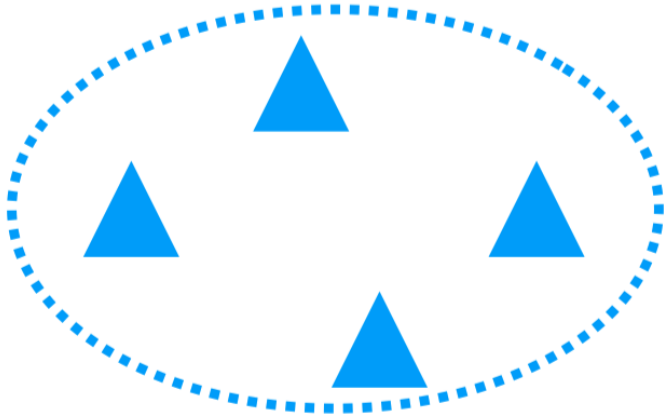


Silhouette Score

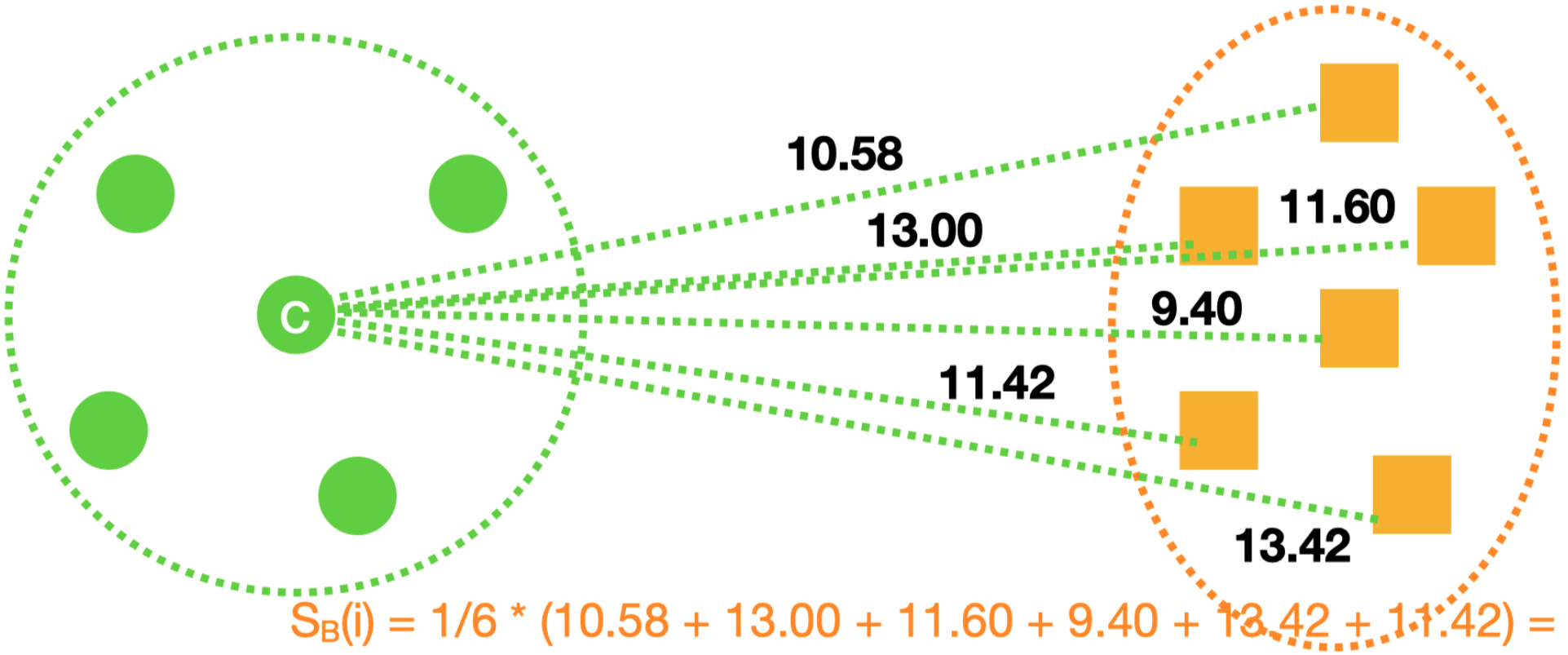




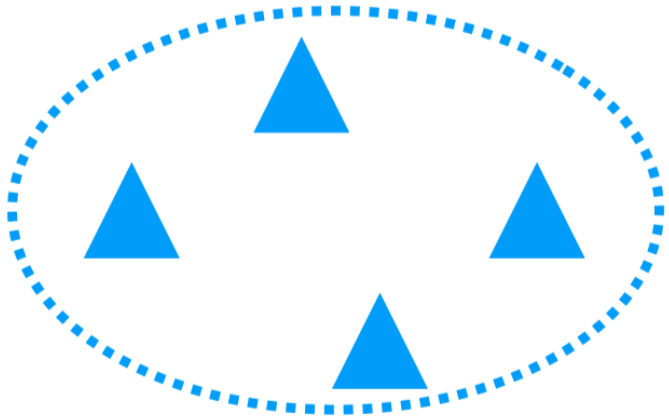
$S_A(i) = 1/4 * (0.94 + 0.86 + 1.02 + 1.29) = 1.03$



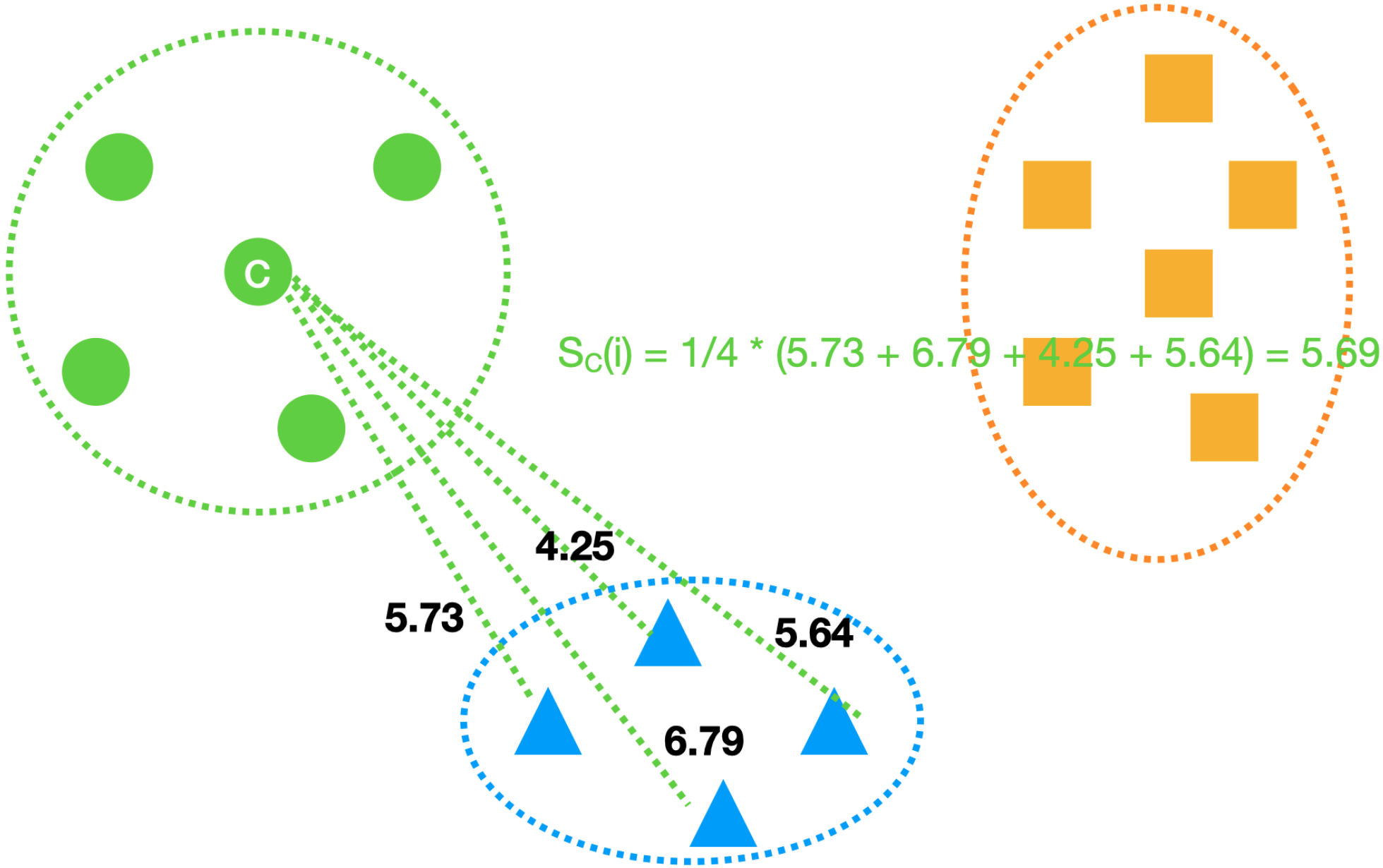
Silhouette Score



$S_B(i) = 1/6 * (10.58 + 13.00 + 11.60 + 9.40 + 13.42 + 11.42) = 11.57$

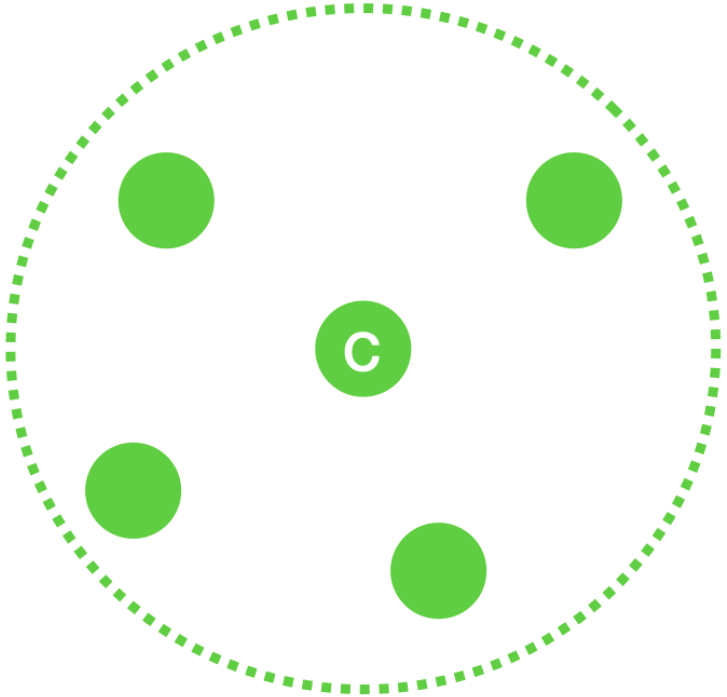


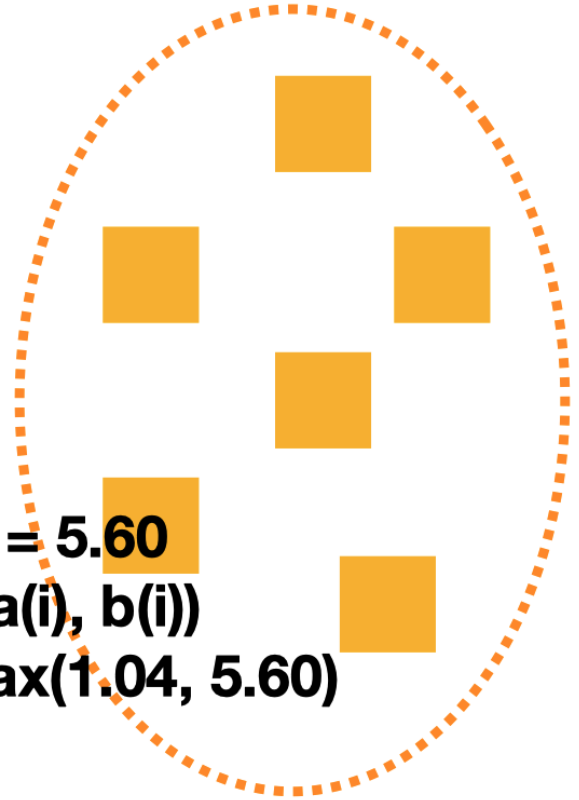
Silhouette Score

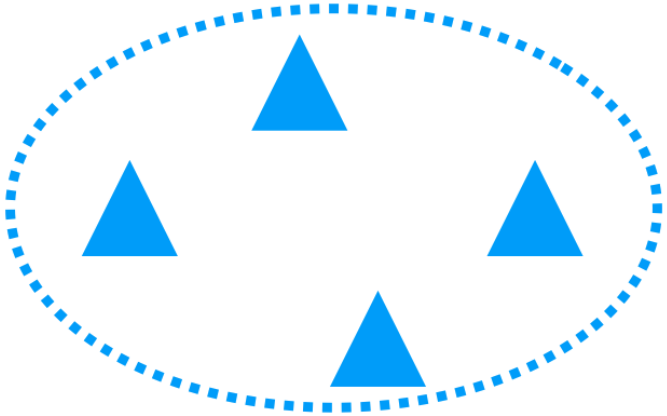


$S_C(i) = 1/4 * (5.73 + 6.79 + 4.25 + 5.64) = 5.69$

Silhouette Score







b(i) = min(SB, SC)
= min(11.57, 5.60) = 5.60
S(i) = (b(i) - SA)/ max(a(i), b(i))
= (5.60 - 1.03) / max(1.04, 5.60)
= 4.57/5.60 = 0.82

Silhouette Score

Silhouette은 사후 평가 방법

- 적절한 k 를 찾았더라도, 그 군집이 최선이라고는 말하지 못한다.
- 모든 k 에 대하여 테스트 할 비용 큼

Silhouette은 model fitness measure

- 높은 값이 “우리가 예상하는“ 좋은 군집화 결과를 의미하지는 않는다.
- LDA 의 perplexity 역시 수리적으로는 설명력이 있지만, 현실적이지 않은 measure 라는 주장도 존재

토픽모델링 (Topic Modeling)

구조화되지 않은 방대한 문헌집단에서 주제를 찾아내기 위한 방법

- 뉴스, 블로그, 웹페이지, 기사 등의 형태로 온라인 상에서 방대한 양의 문서가 생성되고 저장되면서 사람들이 찾고자 하는 것을 발견하는 것이 어려워짐
- 방대한 양의 문서를 관리하고, 검색하고, 이해를 돕기 위한 새로운 도구로 텍스트 마이닝 기법이 주목받기 시작했으며, 그 중 문서 요약과 검색을 위해 토픽모델링 기법이 제안됨
- 맥락과 관련된 단서들을 이용하여 유사한 의미를 가진 단어들을 클러스터링하는 방식으로 주제를 추론하며, 같은 맥락에서 나타날 가능성이 있는 단어들을 그룹화함

잠재 의미 분석 (Latent Semantic Analysis, LSA)

DTM과 TF-IDF의 단점

- BoW에 기반한 DTM이나 TF-IDF는 기본적으로 단어의 빈도 수를 이용한 수치화 방법이기 때문에 단어의 의미를 고려하지 못한다 는 단점이 있다.
  (이를 토픽 모델링 관점에서는 단어의 토픽을 고려하지 못한다 고도 한다.)

단점 보완 방법

- DTM의 잠재된(Latent) 의미를 이끌어내는 방법으로 잠재 의미 분석(Latent Semantic Analysis, LSA)이라는 방법이 있다.
- 잠재 의미 분석(Latent Semantic Indexing, LSI)이라고도 불린다.

특이값 분해(Singular Value Decomposition, SVD)

- SVD란 A가 $m \times n$ 행렬일 때, 다음과 같이 3개의 행렬의 곱으로 분해(decomposition)하는 것을 말한다.

$$a = U \sum V^T$$

- 여기서 각 3개의 행렬은 다음과 같은 조건을 만족한다.
  - U
    - $m \times m$ 직교 행렬
    - $A\,A^T = U \left(\sum \sum^T \right) U^T$
  - V
    - $n \times n$ 직교 행렬
    - $A^T\,A = V \left(\sum^T \sum \right) V^T$
  - $\sum$
    - $m \times n$ 직사각 대각행렬

직교행렬(orthogonal matrix)

- 자신과 자신의 전치 행렬(transposed matrix)의 곱 또는 이를 반대로 곱한 결과가 `단위행렬(identity matrix)`이 되는 행렬

대각행렬(diagonal matrix)

- 주대각선을 제외한 곳의 원소가 모두 0인 행렬

특이값(singular value)

- 이 때 SVD로 나온 대각 행렬의 대각 원소의 값을 행렬 A의 `특이값(singular value)`라고 한다.

전치 행렬 (Transposed Matrix)

- 원래의 행렬에서 행과 열을 바꾼 행렬
- 즉, 주 대각선을 축으로 반사 대칭을 하여 얻는 행렬
- 기호 : 기존 행렬 표현의 우측 위에 T를 붙인다. (ex) 기존 행렬 : M $\rightarrow$ 전치 행렬 : M^T)

$$M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \qquad M^T = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

단위 행렬 (Identity Matrix)

- 주대각선의 원소가 모두 1이며, 나머지 원소는 모두 0인 정사각 행렬
- 기호 : I

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

역행렬 (Inverse Matrix)

- 행렬 A와 어떤 행렬을 곱했을 때, 결과로서 단위 행렬이 나오면 이 때의 어떤 행렬을 행렬 A의 역행렬이라고 한다.
- 기호 : A^{-1}

$$A \times A^{-1} = I$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} \\ ? \\ \\ \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

직교 행렬 (Orthogonal Matrix)

- 실수 $n \times n$ 행렬 A에 대해서 다음 2가지 조건을 동시에 만족하면 행렬 A를 직교 행렬이라고 한다.
  - $A \times A^T = I$
  - $A^T \times A = I$
- 역행렬 정의에 따라 직교 행렬은 $A^{-1} = A^T$를 만족한다.

대각 행렬 (Diagonal Matrix)

- 주대각선을 제외한 곳의 원소가 모두 0인 행렬
- 주대각선의 원소를 a로 표기

정사각 대각 행렬

- 대각 행렬 $\sum$가 3×3 행렬인 경우

$$\sum = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix}$$

직사각 대각 행렬 ($m > n$)

- 행의 크기가 열의 크기보다 큰 경우
- 즉, $m \times n$ 행렬일 때, $m > n$인 경우

$$\sum = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \\ 0 & 0 & 0 \end{bmatrix}$$

직사각 대각 행렬 ($m < n$)

- 행의 크기가 열의 크기보다 작은 경우
- 즉, $m \times n$ 행렬일 때, $m < n$인 경우

$$\sum = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & a & 0 \end{bmatrix}$$

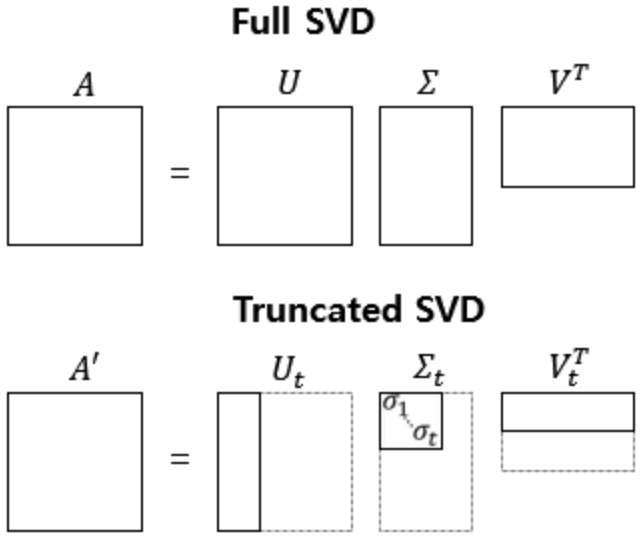
SVD를 통해 나온 대각 행렬 $\sum$의 추가적인 성질

- 대각 행렬 $\sum$의 주대각 원소를 행렬 A의 특이값(singular value) 라고 한다.
- 이 특이값을 $\sigma_1, \sigma_2, \cdots, \sigma_r$라고 표현한다고 했을 때 특이값 $\sigma_1, \sigma_2, \cdots, \sigma_r$은 내림차순으로 정렬 되어 있다는 특징을 가진다.
- 아래의 그림은 특이값 12.4, 9.5, 1.3이 내림차순으로 정렬되어져 있는 모습니다.

$$\sum = \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}$$

절단된 SVD (Truncated SVD)

- 위에서 설명한 SVD를 풀 SVD(full SVD)라고 한다.
- 하지만 LSA의 경우 풀 SVD에서 나온 3개의 행렬에서 일부 벡터들을 삭제시킨 절단된 SVD(truncated SVD)를 사용하게 된다.



t의 의미

- 절단된 SVD는 대각 행렬 $\sum$의 대각 원소의 값 중에서 상위값 t개만 남게 된다.
- 절단된 SVD를 수행하면 값의 손실이 일어나므로 기존의 행렬 A를 복구할 수 없다.
- 또한, U행렬과 V행렬의 t열까지만 남는다.
- 여기서 t는 우리가 찾고자 하는 토픽의 수를 반영한 하이퍼파라미터값 이다.
    - 하이퍼파라미터 : 사용자가 직접 값을 선택하며 성능에 영향을 주는 매개변수
- t를 선택하는 것은 쉽지 않은 일이다.
    - t를 크게 잡는 경우 : 기존의 행렬 A로부터 다양한 의미를 가져갈 수 있다.
    - t를 작게 잡는 경우 : 노이즈를 제거할 수 있다.

데이터의 차원 줄이기의 효과

- 이렇게 일부 벡터들을 삭제하는 것을 데이터의 차원을 줄인다고 말하기도 한다.
- 데이터의 차원을 줄이게 되면 당연히 풀 SVD를 했을 때보다 직관적으로 계산 비용이 낮아지는 효과를 얻을 수 있다.
- 계산 비용이 낮아지는 것 외에도 상대적으로 중요하지 않은 정보를 삭제하는 효과를 갖는다.
    - 영상 처리 분야에서는 노이즈를 제거한다는 의미를 가짐
    - 자연어 처리 분야에서는 설명력이 낮은 정보를 삭제하고 설명력이 높은 정보를 남긴다는 의미를 가짐

LSA의 아이디어

- 기존의 DTM이나 DTM에 단어의 중요도에 따라 가중치를 주었던 TF-IDF 행렬은 단어의 의미를 전혀 고려하지 못한다는 단점을 갖고 있다.
- LSA는 기본적으로 DTM이나 TF-IDF 행렬에 절단된 SVD(truncated SVD)를 사용하여 차원을 축소시키고, 단어들의 잠재적인 의미를 끌어낸다는 아이디어를 갖고 있다.

LSA의 장단점 (Pros and Cons of LSA)

장점

- LSA는 쉽고 빠르게 구현이 가능하다.
- 단어의 잠재적인 의미를 이끌어낼 수 있어 문서의 유사도 계산 등에서 좋은 성능을 보여준다

단점

- SVD의 특성상 이미 계산된 LSA에 새로운 데이터를 추가하여 계산하려고하면 보통 처음부터 다시 계산해야 한다.
- 즉, 새로운 정보에 대해 업데이트가 어렵다.
- 이는 최근 LSA 대신 Word2Vec 등 단어의 의미를 벡터화할 수 있는 또 다른 방법론인 인공 신경망 기반의 방법론이 각광받는 이유이기도 하다.

잠재 디리클레 할당 (Latent Dirichlet Allocation, LDA)

- 문서들은 토픽들의 혼합으로 구성되어져 있으며, 토픽들은 확률 분포에 기반하여 단어들을 생성한다고 가정한다.
- 데이터가 주어지면, LDA는 문서가 생성되던 과정을 역추적한다.

잠재 디리클레 할당 (Latent Dirichlet Allocation, LDA)

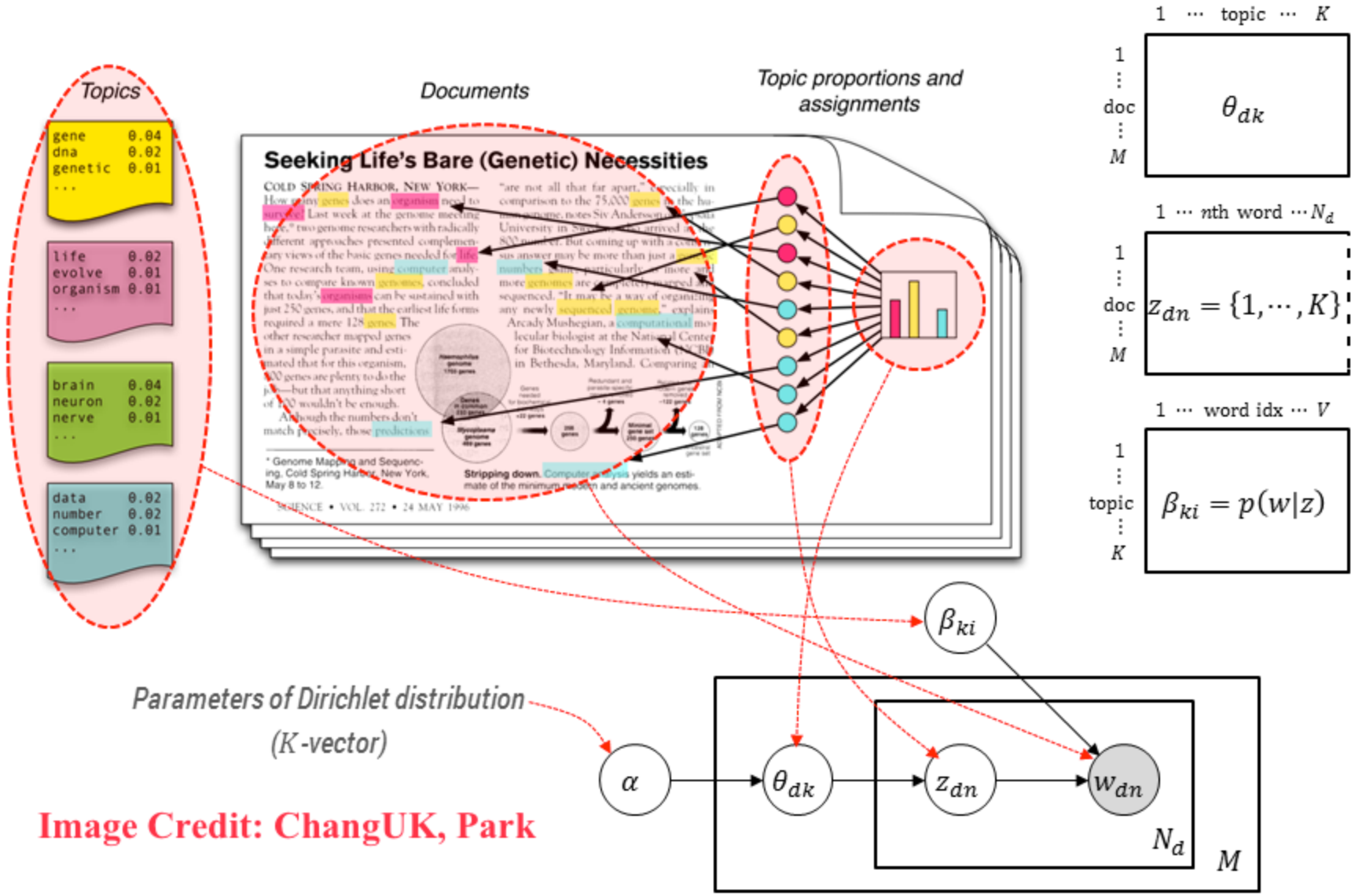


Image Credit: ChangUK, Park

LDA 블랙박스

- 아래와 같은 3개의 문서가 있다고 하자.

> 문서1 : 저는 사과랑 바나나를 먹어요
> 문서2 : 우리는 귀여운 강아지가 좋아요
> 문서3 : 저의 깜찍하고 귀여운 강아지가 바나나를 먹어요 2) 토픽의 개수 지정

- LDA를 수행할 때 문서 집합에서 토픽이 몇 개가 존재할 지 가정하는 것은 사용자가 해야 할 일.
- 여기서는 LDA에 2개의 토픽을 찾으라고 요청한다.
    - 토픽의 개수를 의미하는 변수를 k라고 했을 때, k를 2로 한다.
- k의 값을 잘못 선택하면 원치않는 이상한 결과가 나올 수 있다.
- 하이퍼파라미터의 선택은 여러 실험을 통해 얻은 값일 수도 있고, 우선 시도해보는 값일 수도 있다.

LDA 수행 결과

- 여기서는 LDA 입력 전에 주어와 불필요한 조사 등을 제거하는 전처리 과정은 거쳤다고 가정
- LDA는 각 문서의 토픽 분포 와 각 토픽 내의 단어 분포를 추정한다.

각 문서의 토픽 분포

```
- 문서1
  - 토픽 A 100%
- 문서2
  - 토픽 B 100%
- 문서3
  - 토픽 B 60%
  - 토픽 A 40%
```

각 토픽의 단어 분포

```
- 토픽 A
  - 사과 20%
  - 바나나 40%
  - 먹어요 40%
  - 귀여운 0%
  - 강아지 0%
  - 깜찍하고 0%
  - 좋아요 0%
- 토픽 B
  - 사과 0%
  - 바나나 0%
  - 먹어요 0%
  - 귀여운 33%
  - 강아지 33%
  - 깜찍하고 16%
  - 좋아요 16%
```

LDA 결과 해석

- LDA는 토픽의 제목을 정해주지 않지만, 이 시점에서 알고리즘의 사용자는 위 결과로부터 두 토픽이 각각 과일에 대한 토픽과 강아지에 대한 토픽이라고 판단해볼 수 있다.

LDA의 가정

- LDA는 문서의 집합으로부터 어떤 토픽이 존재하는 지를 알아내기 위한 알고리즘이다.
- LDA는 빈도수 기반의 표현 방법인 BoW의 행렬 DTM 또는 TF-IDF 행렬을 입력으로 한다.
  $\rightarrow$ LDA는 단어의 순서는 신경쓰지 않겠다는 것을 알 수 있다.

LDA가 가정하는 문서 작성 시나리오

- LDA는 문서들로부터 토픽을 뽑아내기 위해서 이러한 가정을 염두해두고 있다.
    - 모든 문서 하나, 하나가 작성될 때 그 문서의 작성자는 다음과 같은 생각을 했을 것이다.

      > "나는 이 문서를 작성하기 위해서 이런 주제들을 넣을거고, 이런 주제들을 위해서는 이런 단어들을 넣을 거야"

- 각각의 문서는 다음과 같은 과정을 거쳐서 작성되었다고 가정한다.

  i. 문서에 사용할 단어의 개수 N을 정한다.

    - ex) 5개의 단어를 사용하기로 한다.

  ii. 문서에 사용할 토픽의 혼합을 확률 분포에 기반하여 결정한다.

    - ex) 위 예제와 같이 토픽이 2개라고 했을 때 강아지 토픽을 60%, 과일 토픽을 40%와 같이 선택할 수 있다.

  iii. 문서에 사용할 각 단어를 아래와 같이 정한다.

- 토픽 분포에서 토픽 T를 확률적으로 고른다.
    - ex) 60% 확률로 강아지 토픽을 선택하고, 40%의 확률로 과일 토픽을 선택할 수 있다.
- 선택한 토픽 T에서 단어의 출현 확률 분포에 기반에 문서에 사용할 단어를 고른다.
    - ex) 강아지 토픽을 선택 $\rightarrow$ 33% 확률로 강아지란 단어를 선택할 수 있다.
- 이제 iii) 을 반복하면서 문서를 완성한다.
- 이러한 과정을 통해 문서가 작성되었다는 가정 하에 LDA는 토픽을 뽑아내기 위하여 위 과정을 역으로 추적하는 역공학(reverse engneering) 을 수행한다.

LDA의 수행 과정 정리

- 사용자는 알고리즘에게 토픽의 개수 k를 알려준다
  - LDA에게 토픽의 개수를 알려주는 역할은 사용자의 역할이다.
  - LDA는 토픽의 개수 k를 입력받으면, k개의 토픽이 M개의 전체 문서에 걸쳐 분포되어 있다고 가정한다.
- 모든 단어를 k개 중 하나의 토픽에 할당한다.
  - LDA는 모든 문서의 모든 단어에 대해 k개 중 하나의 토픽을 랜덤으로 할당한다.
  - 이 작업이 끝나면 각 문서는 토픽을 가지며, 토픽은 단어 분포를 가지는 상태이다.
  - 랜덤으로 할당했기 때문에 이 결과는 전부 틀린 상태이다.
  - 만약 한 단어가 한 문서에서 2회 이상 등장하였다면, 각 단어는 서로 다른 토픽에 할당되었을 수도 있다.

- 모든 문서의 모든 단어에 대해서 아래의 사항을 반복 진행한다. (iterative)
- 어떤 문서의 각 단어 w는 자신은 잘못된 토픽에 할당되어져 있지만, 다른 단어들은 전부 올바른 토픽에 할당되어져 있는 상태라고 가정한다.
    - 이에 따라 단어 w는 아래의 두 가지 기준에 따라서 토픽이 재할당된다.
    - $p(topic\,t \mid document\,d)$
        - 문서 d의 단어들 중 토픽 t에 해당하는 단어들의 비율
    - $p(word\,w \mid topic\,t)$
        - 단어 w를 갖고 있는 모든 문서들 중 토픽 t가 할당된 비율
- 이를 반복하면, 모든 할당이 완료된 수렴 상태가 된다.

doc1

word	apple	banana	apple	dog	dog
topic	B	B	???	A	A

doc2

word	cute	book	king	apple	apple
topic	B	B	B	B	B

- 위의 그림은 두 개의 문서 doc1과 doc2를 보여준다.
- 여기서는 doc1의 세 번째 단어 apple의 토픽을 결정하고자 한다.

doc1

word	apple	banana	apple	dog	dog
topic	B	B	???	A	A

doc2

word	cute	book	king	apple	apple
topic	B	B	B	B	B

- 우선 첫 번째로 사용하는 기준은 문서 doc1의 단어들이 어떤 토픽에 해당하는 지를 확인한다.
- doc1의 모든 단어들은 토픽 A와 토픽 B에 50 대 50의 비율로 할당되어져 있다.
- 이 기준에 따르면 단어 apple은 토픽 A 또는 토픽 B 둘 중 어디에도 속할 가능성이 있다.

doc1

word	apple	banana	apple	dog	dog
topic	B	B	???	A	A

doc2

word	cute	book	king	apple	apple
topic	B	B	B	B	B

- 두 번째 기준은 단어 apple이 전체 문서에서 어떤 토픽에 할당되어져 있는 지를 본다.
- 이 기준에 따르면 단어 apple은 토픽 B에 할당된 가능성이 높다.
- 이러한 두 가지 기준을 참고하여 LDA는 doc1의 apple을 어떤 토픽에 할당할 지 결정한다.

잠재 디리클레 할당과 잠재 의미 분석의 차이

잠재 의미 분석 (Latent Semantic Analysis, LSA)

- DTM을 차원 축소하여 축소 차원에서 근접 든어들을 토픽으로 묶는다.

잠재 디리클레 할당 (Latent Dirichlet Allocation, LDA)

- 단어가 특정 토픽에 존재할 확률과 문서에 특정 토픽이 존재할 확률을 결합확률로 추정하여 토픽을 추출한다.