

Chapter 12. Clustering

발표자: 김재민

최적화

- 변동성(variability)
 - c 는 단일 군집
 - 군집의 평균과 군집에 포함된 각 샘플 사이의 거리들을 모두 합한 것
- 비유사성(dissimilarity)
 - C 는 군집 집합
 - 모든 변이성을 더한 값

$$\text{variability}(c) = \sum_{e \in c} \text{distance}(\text{mean}(c), e)^2$$

$$\text{dissimilarity}(C) = \sum_{c \in C} \text{variability}(c)$$

최적화

- 군집의 크기로 나눈 값을 사용한다면?
 - 정규화
- 정규화 하였을 경우
 - 크기가 큰 분산이 가지는 패널티는 크기가 작은 군집에 비하면 작게 받음
- 정규화 하지 않은 경우
 - 크기가 크고 분산이 높은 군집에 더 높은 패널티를 줄 수 있는 것

목표 함수

- 최적화 하는 문제가 비유사성(dissimilarity)를 최소로 하는 C는 찾는 것이라고 할 수 있는가?
 - No 군집수가 샘플수와 같은 경우. 이럴 경우 변동성, 비유사성이 0.
- 어떻게 해야하는가?
 - 제약을 만들면 된다.
 - 각 군집들은 서로 최소한의 거리를 가져야 한다.
 - 군집의 개수를 제한

2가지 알고리즘

- 계층적 군집화 (Hierarchical clustering)
- k-평균 군집화 (K-means clustering)

병합 계층적 군집화

- N개의 요소를 N개의 집단으로 생성
- 가장 가까운 한쌍의 집단을 찾아서 합병. 집단의 수가 1만큼 감수
- 모든 요소를 가진 단 하나의 집단이 나올 때 까지 반복
- 거리는 어떻게 정의할 것인가?

연결 방식

- 단일 연결 : 한 집단과 다른 집단 사이의 임의의 요소간의 가장 짧은 거리
- 완전 연결 : 한 집단과 다른 집단 사이의 임의의 요소 간의 가장 긴 거리
- 평균 연결 : 한 집단과 다른 집단 사이의 모든 요소 간의 평균 거리

계층적 군집화의 예

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1949	3095	2979
NY		0	802	1771	2934	2815
CHI			0	966	2142	2013
DEN				0	1235	1307
SF					0	808
SEA						0

{BOS} {NY}

{BOS, NY}

{BOS, NY, CHI}

{BOS, NY, CHI}

{BOS, NY, CHI,
DEN}

{BOS, NY, CHI}

{CHI}

{CHI}

{DEN} {SF}

{DEN} {SF}

{DEN} {SF}

{DEN} {SF, SEA}

{SF, SEA} 단일 연결

or

{DEN, SF, SEA} 완전 연결

군집화 알고리즘

- 계층적 군집화의 단점
 - 너무 결정론적 (주어진 연결 방법에 대해서 항상 같은 결과만 나타남)
 - 탐욕 알고리즘
 - 각 지점에서 부분적으로 최적의 결정을 만드는 방식
 - 따라서 전체적으로 봤을 때는 최적이 아닐수 있음
 - 매우 느림
 - 단순 알고리즘 : N^3
 - 연결 방식에 따랄 N^2 도 존재
 - K-means 알고리즘
 - 탐욕 알고리즘 보다 빠름
 - 집단의 개수가 지정 되었을 때 가장 유용
 - 다만 K의 개수를 알지 못하면 문제가 됨.

K-means 알고리즘

randomly chose k examples as initial centroids while
true:

 create k clusters by assigning each example to
 closest centroid

 compute k new centroids by averaging examples in
 each cluster

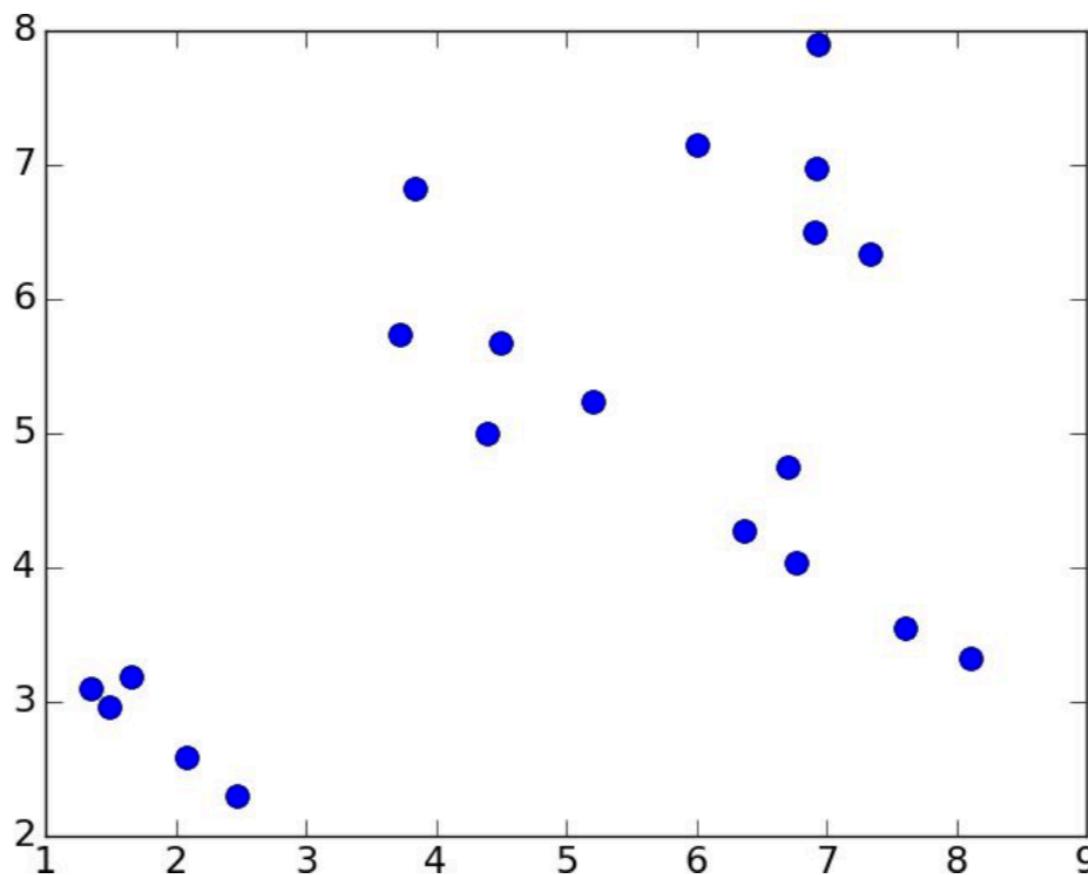
 if centroids don't change: break

- 초기 상태의 중심으로 설정할 K개의 샘플을 무작위로 선택
- K개의 군집 생성 되면, 각 샘플을 가장 가까운 중심에 할당
- K개의 새로운 중심을 계산
- 더이상 변화 없으면 스톱

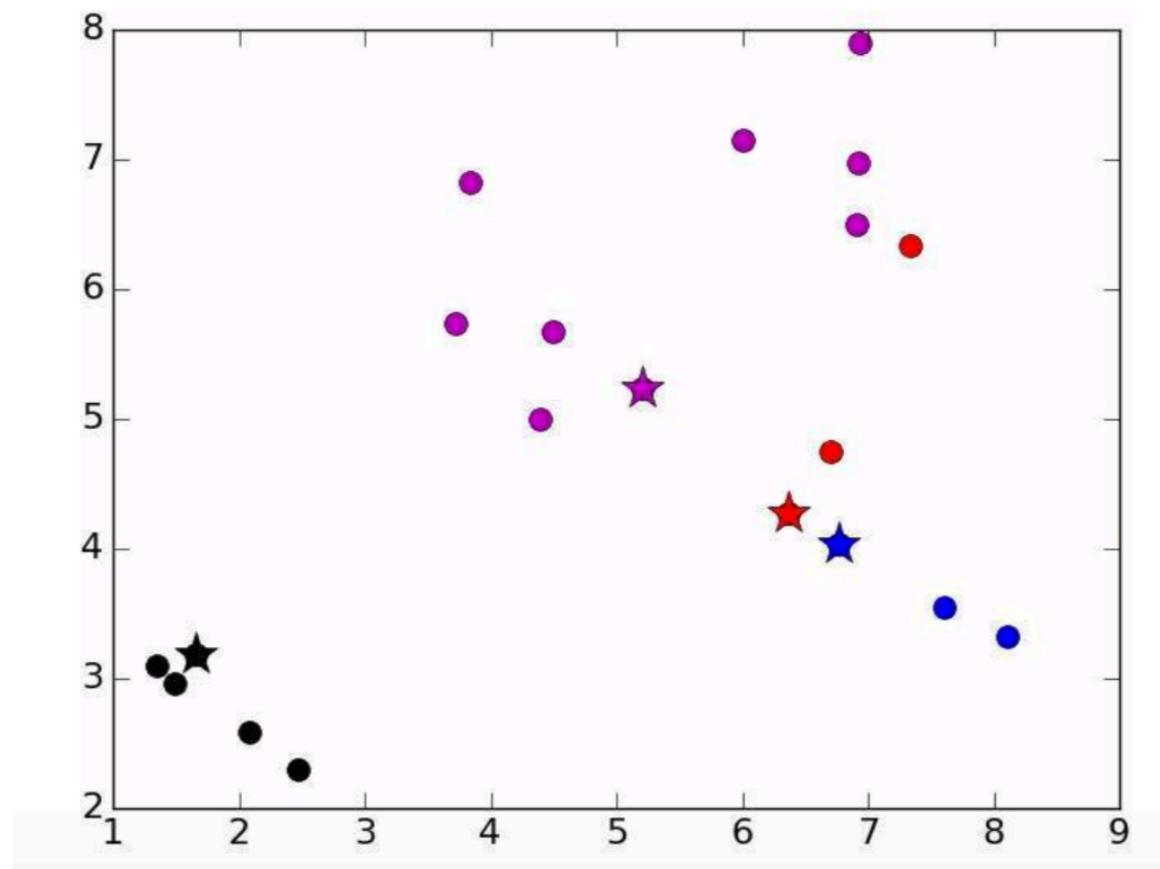
K-means 알고리즘

- 한 반복문의 복잡도는?
 - $K * n * d$
 - n은 점의 개수
 - d는 점들 사이의 거리 계산에 필요한 시간

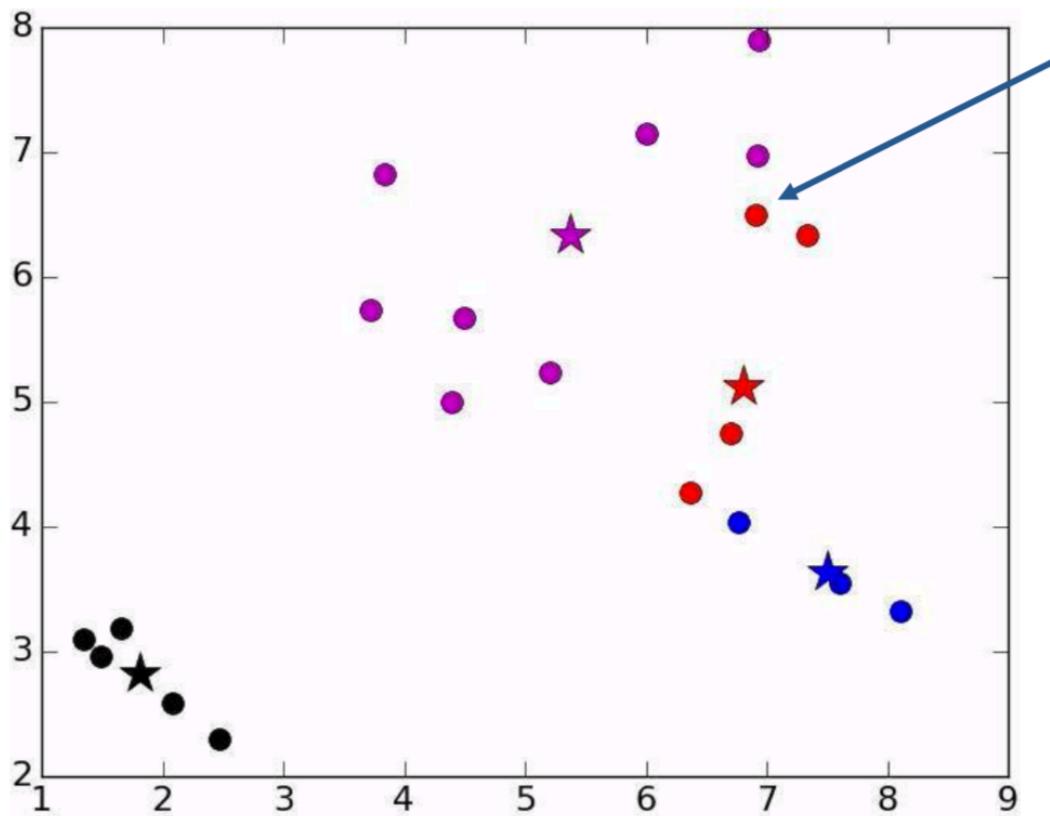
예시



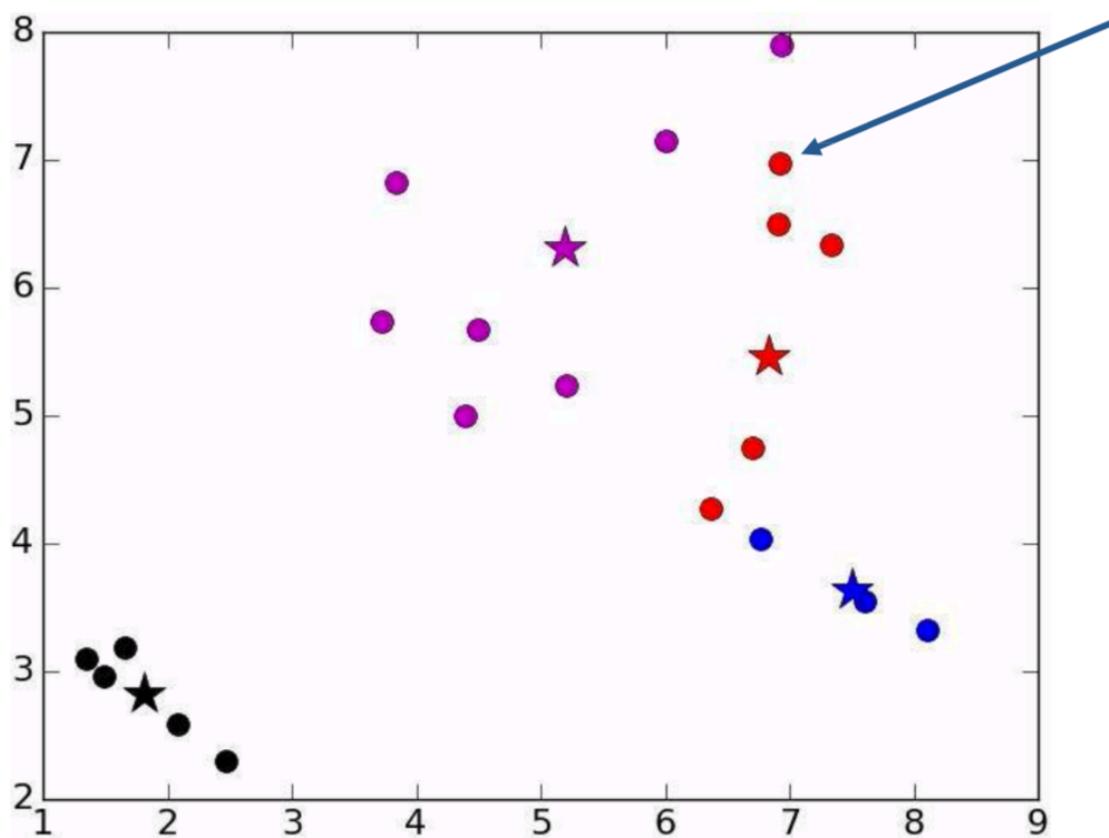
$k = 4$, 초기 상태



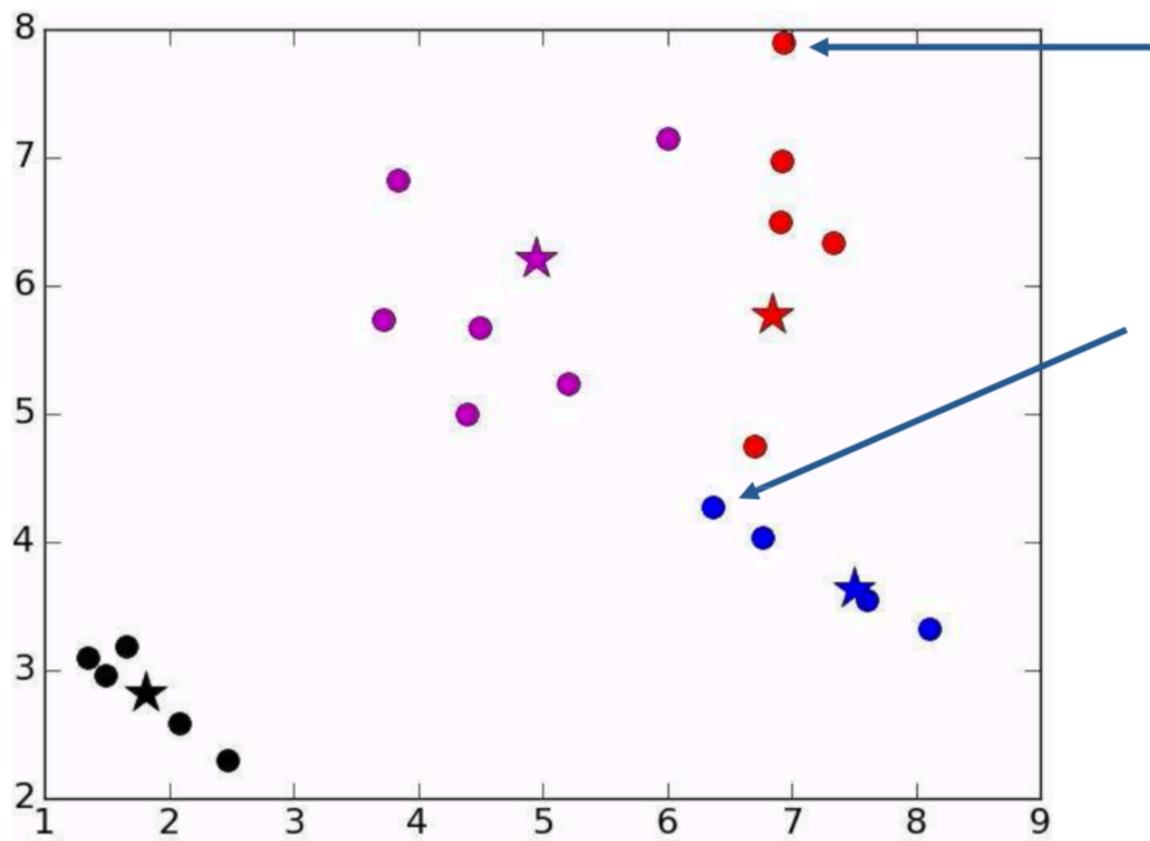
반복 1



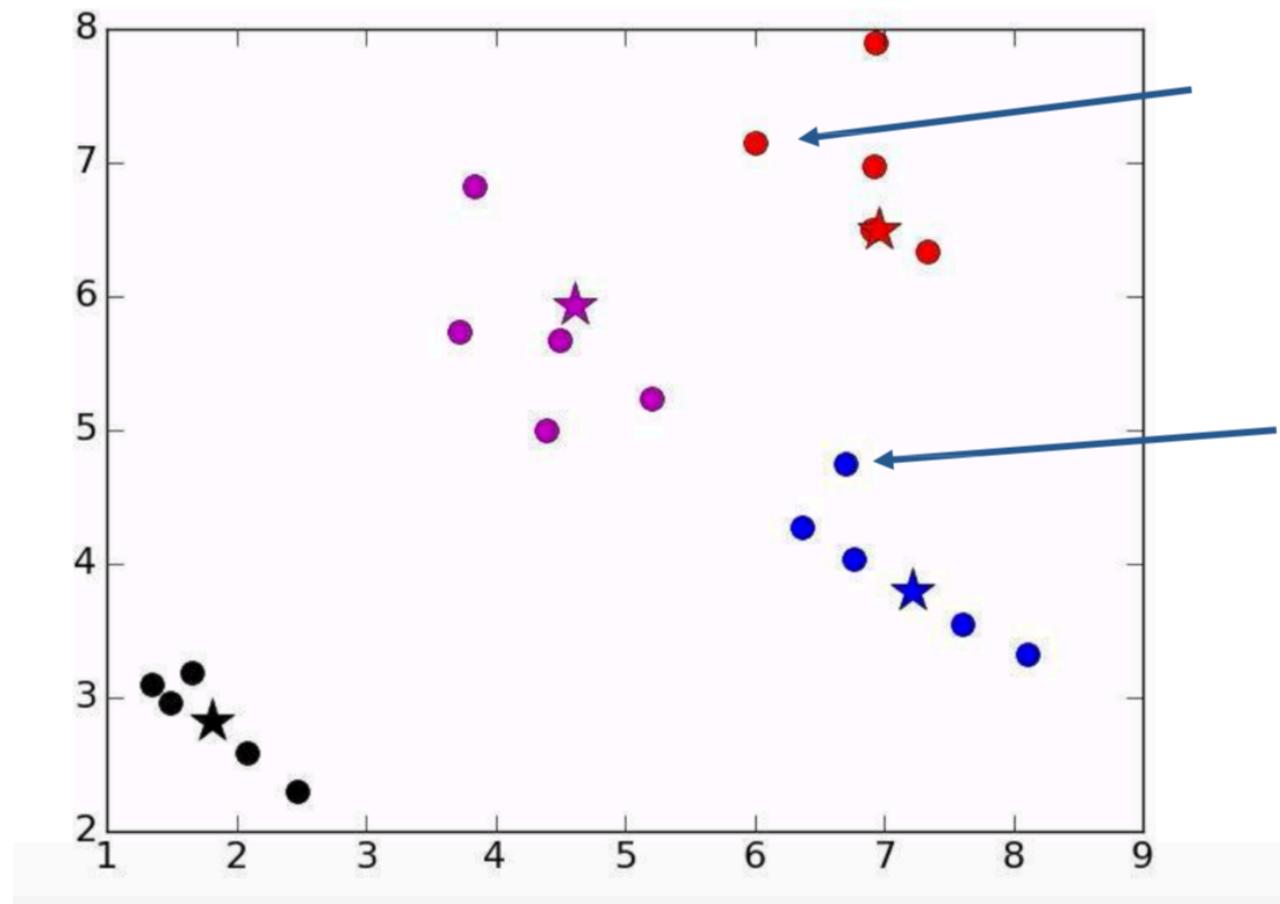
반복 2



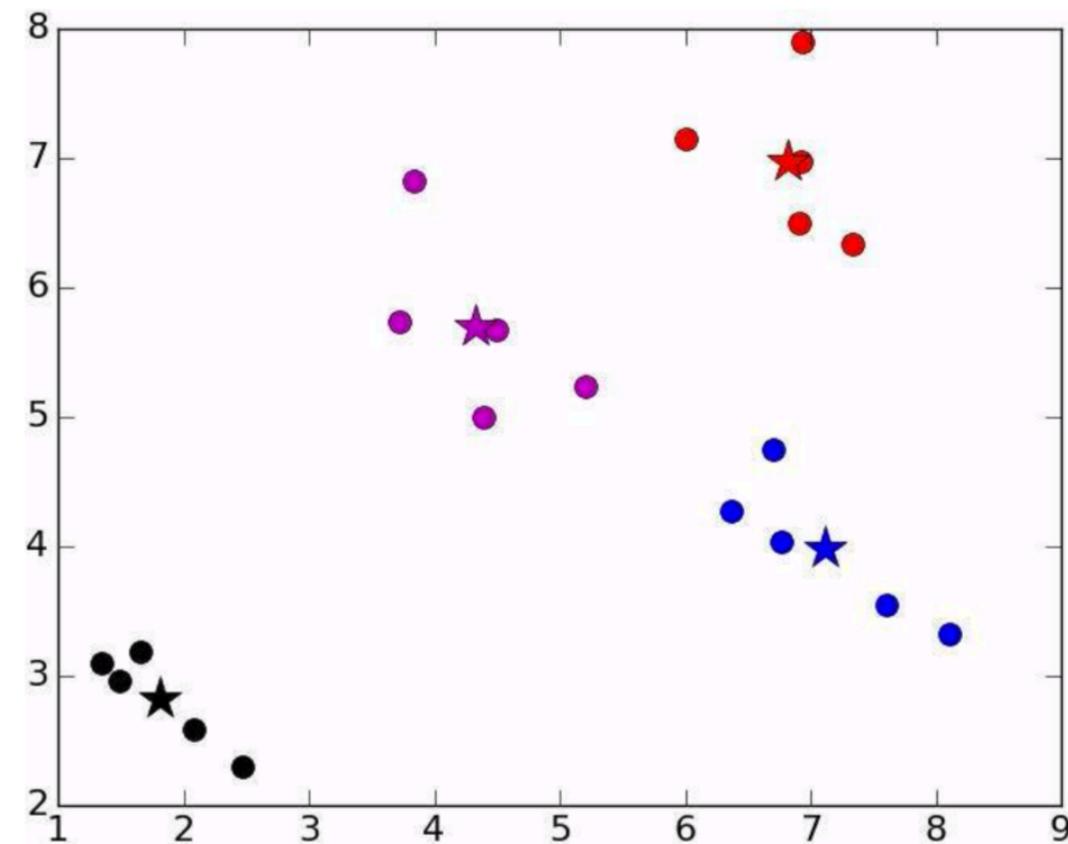
반복 3



반복 4



반복 5



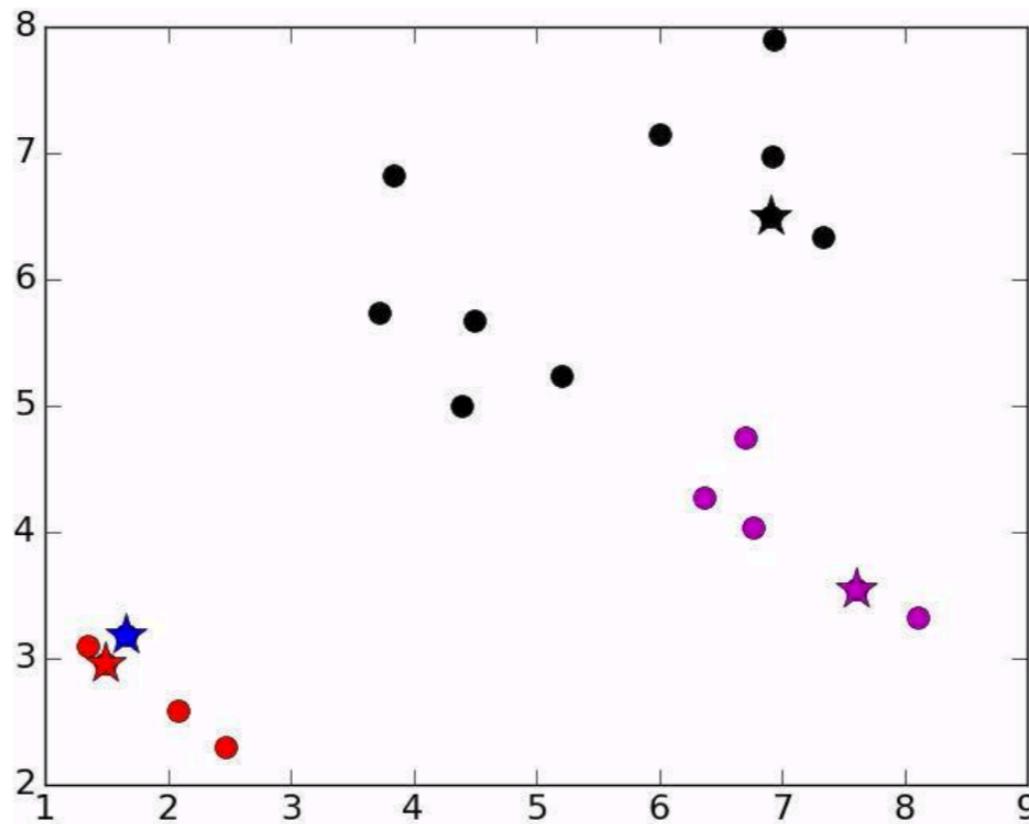
K-means의 문제점

- 결과가 초기값에 의존
- 잘못된 k를 선택하여 원치 않은 결과 초래
 - 반복 횟수가 더 늘어날 수도 있음
 - 최종 결과가 이상하게 나올 수 있음

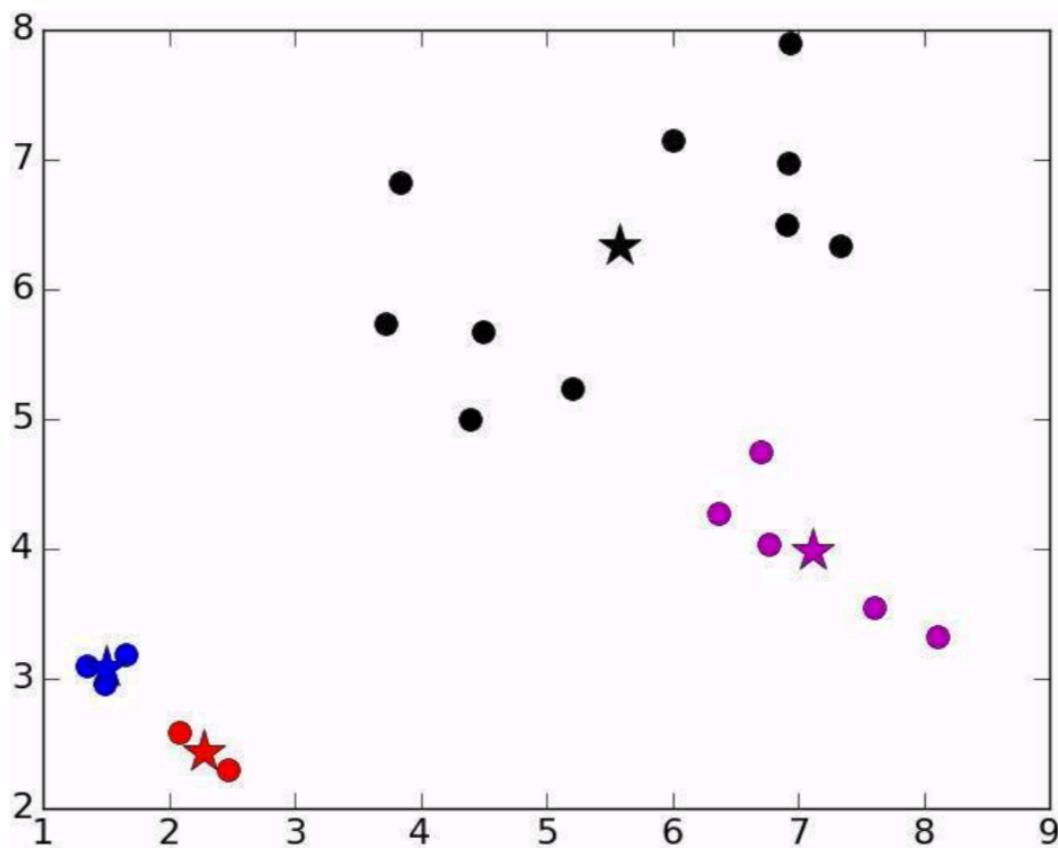
K를 선택하는 방법

- 어플리케이션 도메인에 대한 선 경험적 지식
 - 박테리아에는 다섯 종류가 있다. $k = 5$ (실제로 큰 분류로는 5종류)
- 적절한 K 찾기
 - 다른 k 값을 시도한 뒤 결과를 평가
 - 데이터의 부분 집합에 계층적 군집화를 적용

적절하지 못한 초기값



수렴



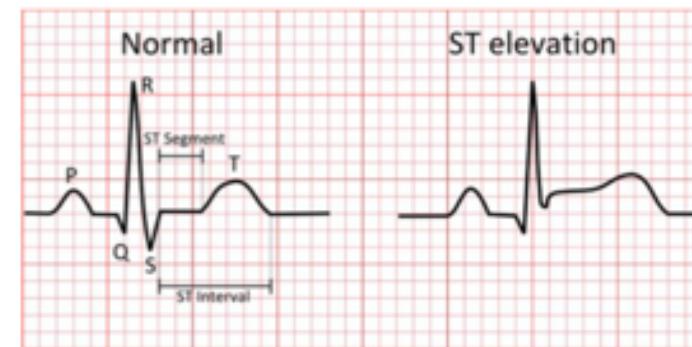
초기값에 대한 의존성 완화

- 첫번째 방법은 코너 같은 곳으로 강제로 넓게 펼치는 것
- 두번째 방법은 최고의 결과를 선택

```
best = kMeans(points)
for t in range(numTrials): C = kMeans(points)
    if dissimilarity(C) < dissimilarity(best):
        best = C
return best
```

예시(심경근색)

- 많은 환자들의 4 가지 특성
 - 분당 심박수
 - 심장마비 경력
 - 나이
 - ST 상승 (이진)
- 특성에 따른 결과 (죽음)
 - 결정론적이 아닌 확률론적
 - E.g., 심장마비 경력이 있는 나이 든 사람이 더 큰 위험
- 결과를 바탕으로 집단의 순수성 검사



데이터 샘플

	<u>HR</u>	<u>Att</u>	<u>STE</u>	<u>Age</u>	<u>Outcome</u>
P000:[89.	1.	0.	66.]	:1
P001:[59.	0.	0.	72.]	:0
P002:[73.	0.	0.	73.]	:0
P003:[56.	1.	0.	65.]	:0
P004:[75.	1.	1.	68.]	:1
P005:[68.	1.	0.	56.]	:0
P006:[73.	1.	0.	75.]	:1
P007:[72.	0.	0.	65.]	:0
P008:[73.	1.	0.	64.]	:1
P009:[73.	0.	0.	58.]	:0
P010:[100.	0.	0.	75.]	:0
P011:[79.	0.	0.	31.]	:0
P012:[81.	0.	0.	58.]	:0
P013:[89.	1.	0.	50.]	:1
P014:[81.	0.	0.	70.]	:0

클래스

클러스터 클래스

```
class Cluster(object):

    def __init__(self, examples):
        """Assumes examples a non-empty list of Examples"""
        ...

    def update(self, examples):
        """Assume examples is a non-empty list of Examples
           Replace examples; return amount centroid has
           changed"""
        ...

    def computeCentroid(self):
        vals = pylab.array([0.0]*self.examples[0].\
                           dimensionality())
        for e in self.examples: #compute mean
            vals += e.getFeatures()
        centroid = Example('centroid', vals/len(self.examples))
        return centroid
```

클러스터 클래스

```
def variability(self):
    totDist = 0
    for e in self.examples:
        totDist += (e.distance(self.centroid))**2
    return totDist

def members(self):
    for e in self.examples:
        yield e
```

군집화 평가

```
def dissimilarity(clusters):
    """Assumes clusters a list of clusters
       Returns a measure of the total dissimilarity of the
       clusters in the list"""
    totDist = 0
    for c in clusters:
        totDist += c.variability()
    return totDist
```

환자

```
import cluster, pylab, numpy

class Patient(cluster.Example):
    pass

def scaleAttrs(vals):
    vals = pylab.array(vals)
    mean = sum(vals)/len(vals)
    sd = numpy.std(vals)
    vals = vals - mean
    return vals/sd

def getData(toScale = False):
    #read in data
    ...
    if toScale:
        hrList = scaleAttrs(hrList)
    ...
    #Build points
    ...
    return points
```

Z-Scaling

Mean = ?

Std = ?

K-means

```
def kmeans(examples, k, verbose = False):
    #Get k randomly chosen initial centroids,
    #create cluster for each
    ...
    #Iterate until centroids do not change
    ...
    #Associate each example with closest centroid
    ...
    for c in newClusters: #Avoid having empty clusters
        if len(c) == 0:
            raise ValueError('Empty Cluster')

    #Update each cluster; check if a centroid has changed
    ...

def trykmeans(examples, numClusters, numTrials, verbose=False):
    """Calls kmeans numTrials times and returns the result with
       the lowest dissimilarity"""
    ...
```

결과 검사

```
def printClustering(clustering):
    """Assumes: clustering is a sequence of clusters
       Prints information about each cluster
       Returns list of fraction of pos cases in each cluster"""
    ...

def testClustering(patients, numClusters, seed = 0,
                   numTrials = 5):
    random.seed(seed)
    bestClustering = trykmeans(patients, numClusters,
                               numTrials)
    posFracs = printClustering(bestClustering)
    return posFracs

patients = getData()
for k in (2,):
    print('\n      Test k-means (k = ' + str(k) + ')')
    posFracs = testClustering(patients, k)
```

실험 결과

k-평균 실험 ($k = 2$)

118의 크기 집단의 긍정 비율 = 0.3305

132의 크기 집단의 긍정 비율 = 0.3333

만족하나요?

patients = getData(True) k-평균 실험 ($k = 2$)

244의 크기 집단의 긍정 비율 = 0.2902

26의 크기 집단의 긍정 비율 = 0.6923

민감도?

얼마만큼의 긍정이 있을까?

```
numPos = 0
for p in patients:
    if p.getLabel() == 1:
        numPos += 1
print('Total number of positive patients =', numPos)
```

긍정 반응 환자의 총 수 = 83

k-평균 실험 (k = 2)

224의 크기 집단의 긍정 비율 = 0.2902

26의 크기 집단의 긍정 비율 = 0.6923

가설

- 서로 다른 긍정 환자 부분 집단은 다른 성격을 지님
- 다른 k값으로 시도

다양한 K시도

k-평균 실험 (k = 2)

224의 크기 집단의 긍정 비율 = 0.2902

26의 크기 집단의 긍정 비율 = 0.6923

k-평균 실험 (k = 4)

26의 크기 집단의 긍정 비율 = 0.6923

86의 크기 집단의 긍정 비율 = 0.0814

76의 크기 집단의 긍정 비율 = 0.7105

62의 크기 집단의 긍정 비율 = 0.0645

k-평균 실험 (k = 6)

49의 크기 집단의 긍정 비율 = 0.0204

26의 크기 집단의 긍정 비율 = 0.6923

45의 크기 집단의 긍정 비율 = 0.0889

54의 크기 집단의 긍정 비율 = 0.0926 Cluster of

36의 크기 집단의 긍정 비율 = 0.7778 Cluster of

k 를 선택