

연세대학교 통계 데이터 사이언스 학회 ESC 23-2 FINAL PROJECT

Spectral Clustering

[4조] 김근영 김시은 도현수 오동윤 허정웅



목차

1. Spectral Clustering & K-means Clustering
2. Code implementation: IRIS
3. Code implementation: Spiral data
4. Conclusion





Spectral Clustering & K-means Clustering

1. Spectral Clustering

Notations

• $W = (w_{ij}), i, j = 1, 2, \dots, n$: *Weighted adjacency matrix*

w_{ij} ?

1) *The ε – neighborhood graph*

2) *k – nearest neighbor graph*

3) *The fully connected graph e. g. Gaussian kernel*

• $d_i = \sum_{j=1}^n w_{ij}$: *degree of vertex v_i*

• $D = \text{diag}\{d_1, d_2, \dots, d_n\}$

• $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$

• $|A|$ = *the number of vertices in A*

• $\text{vol}(A) = \sum_{i \in A} d_i$



1. Spectral Clustering

Notations

- $cut(A_1, \dots, A_k) = \frac{1}{2} \sum_{i=1}^n W(A_i, \overline{A_i})$
- $RatioCut(A_1, \dots, A_k) = \frac{1}{2} \sum_{i=1}^n W(A_i, \overline{A_i}) / |A_i|$
- $NCut(A_1, \dots, A_k) = \frac{1}{2} \sum_{i=1}^n W(A_i, \overline{A_i}) / vol(A_i)$

intuition : minimizing graph cut



1. Spectral Clustering

Laplacian matrix

- $L = D - W$

- $f'Lf = \frac{1}{2} \sum_{i,j=1}^n w_{ij}(f_i - f_j)^2$, for all $f \in \mathbb{R}^n$

- L의 smallest eigenvalue는 0, 이때의 eigenvector은 k개의 indicator vector (다른 그룹 간 $w=0$ 가정 하)



1. Spectral Clustering

Unnormalized case - Approximating Ratocut

($k = 2$)

(1) $\min_{A \subset V} \text{Ratocut}(A_1, \dots, A_k)$ 문제는

(2) $\min_{A \subset V} f' L f$ subject to $f \perp 1, \|f\| = \sqrt{n}$, where f is defined as $f_i = \begin{cases} \sqrt{(|\bar{A}|/|A|)} & , \text{if } v_i \in A \\ \sqrt{(|A|/|\bar{A}|)} & , \text{if } v_i \in \bar{A} \end{cases}$ ($i = 1, \dots, n$) 로 바꿔 생각할 수 있음.

$$\begin{aligned}
 f' L f &= \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2 \\
 &\quad (\text{if } i \in A \text{ and } j \in \bar{A}, f_i = f_j, (f_i - f_j)^2 = 0) \\
 &= \frac{1}{2} \sum_{i \in A, j \in \bar{A}} w_{ij} \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + \frac{1}{2} \sum_{i \in \bar{A}, j \in A} w_{ij} \left(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 \\
 &= \frac{1}{2} \sum_{i \in A, j \in \bar{A}} w_{ij} \left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) + \frac{1}{2} \sum_{i \in \bar{A}, j \in A} w_{ij} \left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) \\
 &= \left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) + \frac{1}{2} \sum_{i \in \bar{A}, j \in A} w_{i,j} + \frac{1}{2} \sum_{i \in A, j \in \bar{A}} w_{i,j}
 \end{aligned}$$

$$\begin{aligned}
 \sum_{i=1}^n f_i &= \sum_{i \in A} \sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i \notin A} \sqrt{\frac{|A|}{|\bar{A}|}} = |A| \sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}| \sqrt{\frac{|A|}{|\bar{A}|}} = 0 \\
 \text{and } \|f\|^2 &= \sum_{i=1}^n f_i^2 = |A| \frac{|\bar{A}|}{|A|} + |\bar{A}| \frac{|A|}{|\bar{A}|} = |A| + |\bar{A}| = n \text{ (num of all vertices)}
 \end{aligned}$$



1. Spectral Clustering

Unnormalized case - Approximating Ratocut

($k = 2$)

(2) $\min_{A \subset V} f' L f$ subject to $f \perp 1, \|f\| = \sqrt{n}$, where f is defined as $f_i = \begin{cases} \sqrt{(|\bar{A}|/|A|)} & , \text{if } v_i \in A \\ \sqrt{(|A|/|\bar{A}|)} & , \text{if } v_i \in \bar{A} \end{cases}$ ($i = 1, \dots, n$) 문제는 집합의 분할 문제로, n 이 커짐에 따라 A 를 구성하는 경우의 수가 기하급수적으로 증가하므로 NP hard한 problem.

문제 조건을 *Relaxation*하여 위와 같은 *discrete*한 문제를 다음과 같은 *continuous*한 문제로 바꿔줌.

(3) $\min_{f \in \mathbb{R}^n} f' L f$ subject to $f \perp 1, \|f\| = \sqrt{n}$



1. Spectral Clustering

Unnormalized case - Approximating Ratocut

($k = 2$)

Rayleigh-Ritz theorem에 의해 f 는 L 의 second smallest eigenvalue의 eigenvector이므로, 결론적으로 Ratocut의 minimizer은 L 의 second eigenvector로 approximate시킨 것이 됨.

위의 continuous한 해인 f 를 다시 re-transformation하여 discrete한 indicator vector로 나타내주어야 함. (Using K-means clustering)



1. Spectral Clustering

Unnormalized case - Approximating Ratocut

(arbitrary k)

1. *Minimizing Ratocut*(A_1, \dots, A_k)

2. $\min_{A_1, \dots, A_k} \text{Tr}(H' L H)$ subject to $H' H = I$ where $H = (h_{ij})$ is defined as $h_{ij} = \begin{cases} 1/\sqrt{|A_j|} & , \text{if } v_i \in A_j \\ 0 & , \text{otherwise} \end{cases}$ ($i = 1, \dots, n; j = 1, \dots, k$)

3. (Relaxation) $\min_{H \in \mathbb{R}^{n \times k}} \text{Tr}(H' L H)$ subject to $H' H = I$

4. By Rayleigh-Ritz theorem, Laplacian matrix L 의 k 번째 eigenvectors들로 구성된 H

5. K-means clustering to row of matrix를 통해 위에서 구한 해를 다시 discrete하게 변환



1. Spectral Clustering

Normalized case - Approximating Ncut (arbitrary k)

1. Minimizing $Ncut(A_1, \dots, A_k)$

2. $\min_{A_1, \dots, A_k} Tr(H' L H)$ subject to $H' D H = I$ where $H = (h_{ij})$ is defined as $h_{ij} = \begin{cases} 1/\sqrt{|vol(A_j)|} & , \text{if } v_i \in A_j \\ 0 & , \text{otherwise} \end{cases}$ ($i = 1, \dots, n; j = 1, \dots, k$)

3. (Relaxation) $\min_{H \in \mathbb{R}^{n \times k}} Tr(H' L H)$ subject to $H' D H = I \Leftrightarrow \min_{T \in \mathbb{R}^{n \times k}} Tr(T' D^{-1/2} L D^{-1/2} T)$ subject to $T' T = I$ ($T = D^{1/2} H$)

4. By standard Rayleigh-Ritz theorem, Laplacian matrix $L_{sym} = D^{-1/2} L D^{-1/2}$ 의 k 번째 eigenvectors들로 구성된 T

5. K-means clustering to row of matrix를 통해 위에서 구한 해를 다시 discrete하게 변환

(+) Normalized laplacian을 사용하는 이유?

normalized clustering은 between-clustering similarities를 minimize함과 동시에 within-clustering similarities를 maximize할 수 있음



2. K-means Clustering

Definition

각 집합별 중심점으로부터 집합 내 노드들간 거리의 제곱합을 최소화 하게 clustering 하는 방법

n 개의 d 차원 노드 $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ 이 주어 질 때,

$$\arg \min_{\mathcal{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mathbf{u}_i\|^2 \text{ where } \mathcal{S} = \{S_1, S_2, \dots, S_k\}, k \text{ is given constant}$$





Code implementation: IRIS

Code Implementation: IRIS

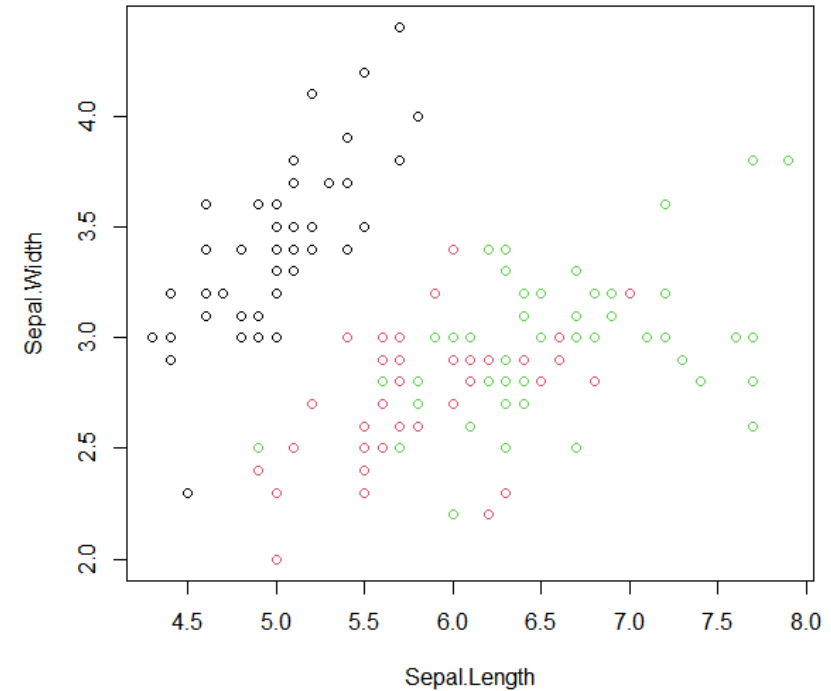
Data generation

```
my.data1 = iris[,1:4]  
plot(my.data1[,1:2], col = iris$species)
```

* IRIS: 세 가지 품종의 꽃들에 해당하는 각 샘플의

- 꽃받침 길이 (Sepal length)
- 꽃받침 넓이 (Sepal width)
- 꽃잎의 길이 (Petal length)
- 꽃잎의 너비 (Petal width)

수치를 담고 있는 dataset



Code Implementation: IRIS

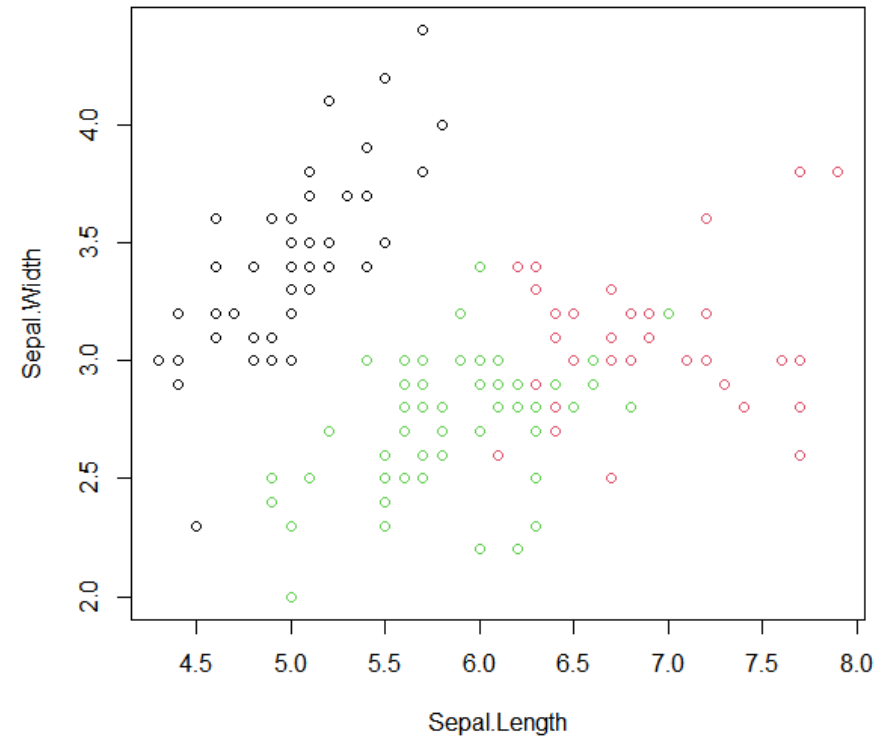
k-means clustering

```
km1<-kmeans(my.data1, centers = 3, iter.max = 10000)  
plot(my.data1[,1:2], col=km1$cluster)
```

kmeans() 함수를 활용하여 IRIS 데이터를 clustering

* Centers = n -> Cluster 개수 설정

iter.max = n -> 최대 반복 횟수 설정



Code Implementation: IRIS

Similarity Matrix

```
s <- function(x1, x2, alpha=1) {  
  exp(- alpha * norm(as.matrix(x1-x2), type="F"))  
}  
  
make.similarity <- function(my.data, similarity) {  
  N <- nrow(my.data)  
  S <- matrix(rep(NA,N^2), ncol=N)  
  for(i in 1:N) {  
    for(j in 1:N) {  
      S[i,j] <- similarity(my.data[i,], my.data[j,])  
    }  
  }  
  S  
}  
  
s1 <- make.similarity(my.data1, s)
```

* 함수 s : Gaussian kernel로 두 벡터 x1과 x2 사이의 유사도를 계산

* 함수 make.similarity의 인자로 IRIS 데이터와 함수 s를 사용하여 IRIS 데이터의 Similarity matrix 생성

* S1: IRIS 데이터셋의 유사도 행렬



Code Implementation: IRIS

Weighted Adjacency Matrix

```
make.affinity <- function(S, n.neighbor=5) {  
  N <- length(S[,1])  
  if (n.neighbor >= N) { # fully connected  
    W <- S  
  } else {  
    W <- matrix(rep(0,N^2), ncol=N)  
    for(i in 1:N) { # for each line  
      # only connect to those points with larger similarity  
      best.similarities <- sort(S[i,], decreasing=TRUE)[1:n.neighbor]  
      for (s in best.similarities) {  
        j <- which(S[i,] == s)  
        W[i,j] <- S[i,j]  
        W[j,i] <- S[i,j] # symmetric  
      }  
    }  
  }  
  return(W)  
}  
w1 <- make.affinity(s1, 10)
```

* make.affinity: 유사도 행렬 S와 이웃의 개수를 설정하여 affinity 행렬을 생성하는 함수

- If n.neighbor >= N (S의 크기)
→ 모든 데이터 포인트 간의 연결을 고려
→ S = affinity matrix

* W1: S1의 각 행에서 가장 큰 10개의 값만 남기고, 나머지 값은 0으로 만든 affinity 행렬

Code Implementation: IRIS

Degree Matrix

```
# Diagonal matrix
D1 <- diag(apply(w1, 1, sum)) # sum rows

eigen_result1<-eigen(D1, symmetric = TRUE)

D_minus_half1 <- eigen_result1$vectors %*% diag(1/sqrt(eigen_result1$values)) %*% t(eigen_result1$vectors)
```

*D1 : W1의 각 행의 합을 대각원소로 가지는 행렬

* eigen_result1: D1의 고유값과 고유벡터

* D_minus_half1: D1의 제곱근의 역행렬

-> Normalized Laplacian matrix를 계산하기 위해 쓰임



Code Implementation: IRIS

Laplacian Matrix

```
L1 <- D1 - W1  
L_sym1 <- (D_minus_half1 %**% L1 %**% D_minus_half1)
```

* L1: $D1 - W1$ 인 Laplacian matrix

* L_sym1: 앞서 구한 D_minus_half1 과 L1을 이용하여 생성한 Normalized Laplacian matrix



Code Implementation: IRIS

Eigen Gap

```
L_eigen1 <- eigen(L1, symmetric=TRUE)  
plot(rev(L_eigen1$values)[1:5])
```

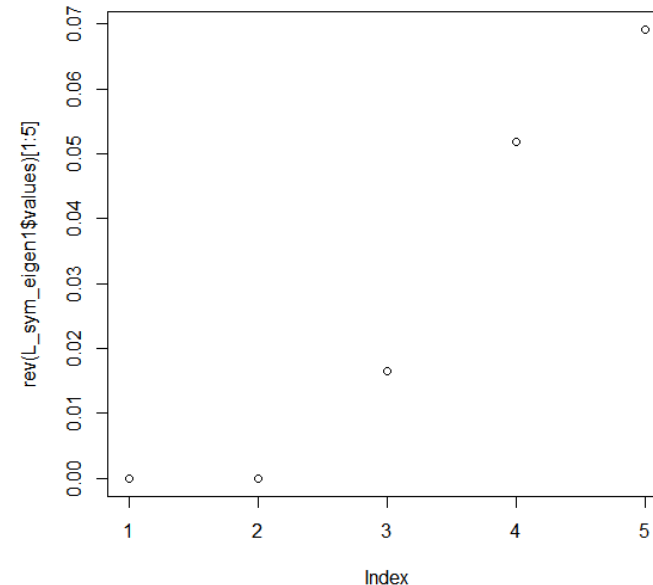
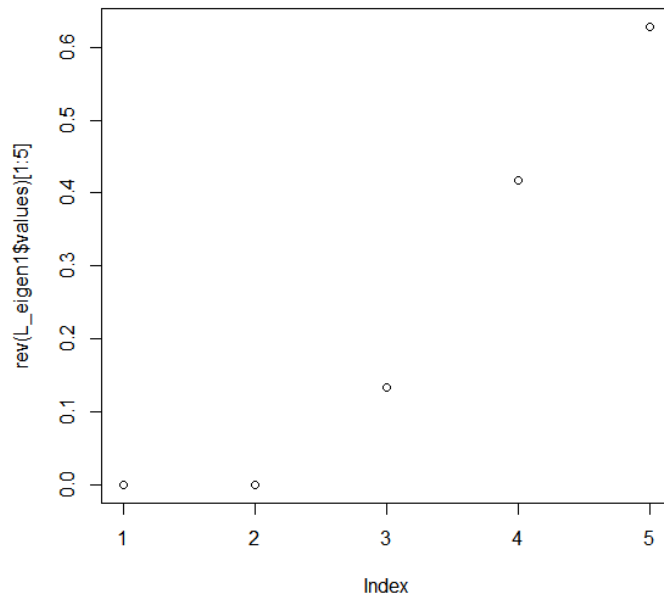
```
L_sym_eigen1 <- eigen(L_sym1, symmetric=TRUE)  
plot(rev(L_sym_eigen1$values)[1:5])
```

* L1과 L_sym1의 고유값, 고유벡터 계산

* 시각화 결과, 두 개의 경우 모두

4번째로 작은 고유값 지점에서 급격히 증가함을 확인

-> k=3 (Cluster 개수) 으로 설정하여 clustering 진행



Code Implementation: IRIS

Spectral Clustering

```
k1 = 3  
  
Z_unnormal1 <- L_eigen1$vectors[, (ncol(L_eigen1$vectors)-k1+1):ncol(L_eigen1$vectors)]  
Z_sym1 <- L_sym_eigen1$vectors[, (ncol(L_sym_eigen1$vectors)-k1+1):ncol(L_sym_eigen1$vectors)]
```

* Laplacian Matrix의 1, 2, 3번째 작은 고유값에 해당하는 고유벡터를 열로 가지는 행렬 Z 생성

*Z_unnormal1: L1의 경우

*Z_sym1: L_sym의 경우



Code Implementation: IRIS

Spectral Clustering

```
library(stats)
result_unnormal1 <- kmeans(Z_unnormal1, centers=k1, nstart=5)
result_sym1 <- kmeans(Z_sym1, centers=k1, nstart=5)

plot(my.data1[,1:2], col=iris$Species)
title(main = "true data")

plot(my.data1[,1:2], col=result_unnormal1$cluster)
title(main = "Clustering with Unnormalized laplacian")

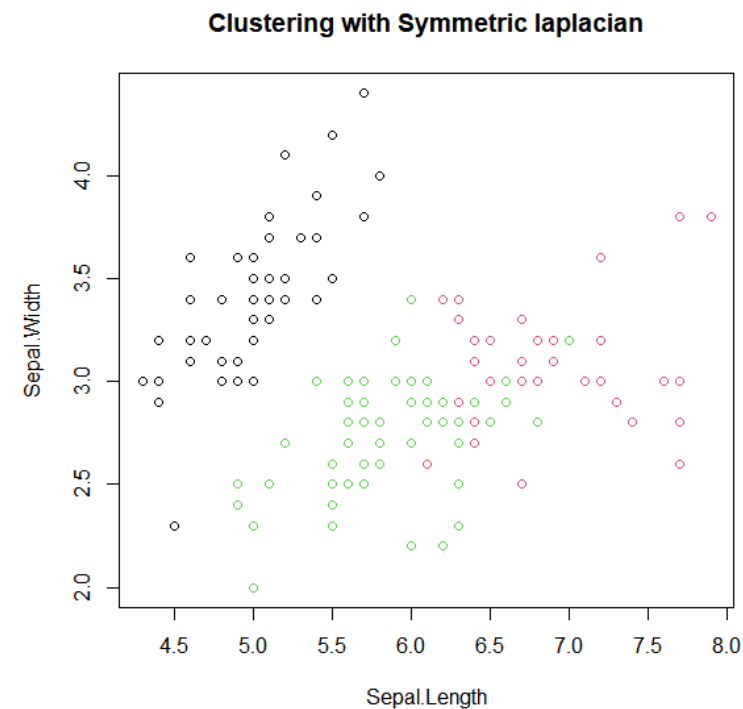
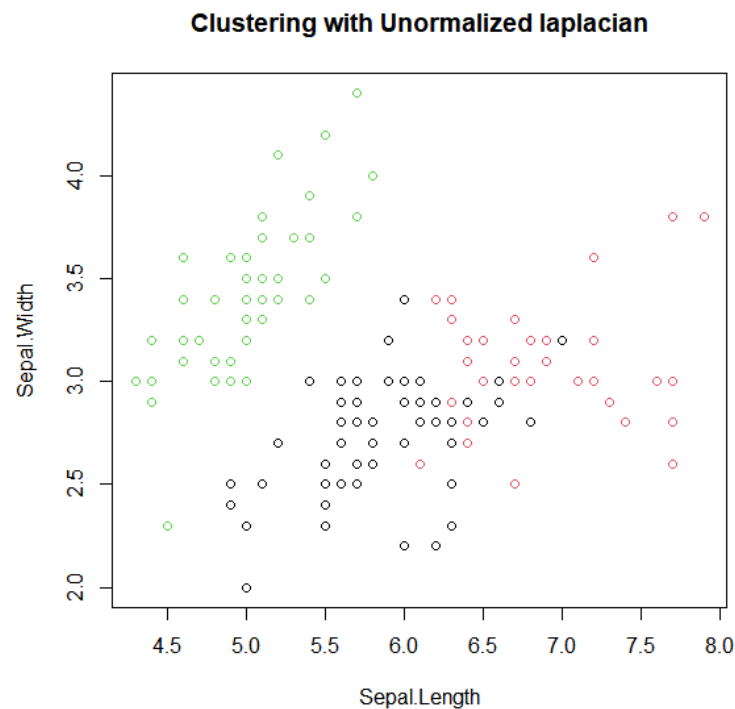
plot(my.data1[,1:2], col=result_sym1$cluster)
title(main = "Clustering with Symmetric laplacian")
```

- * Z 행렬들에 대해 k-means clustering 진행 및 시각화
- * "true data"는 원본 IRIS 데이터의 분포를 시각화한 것

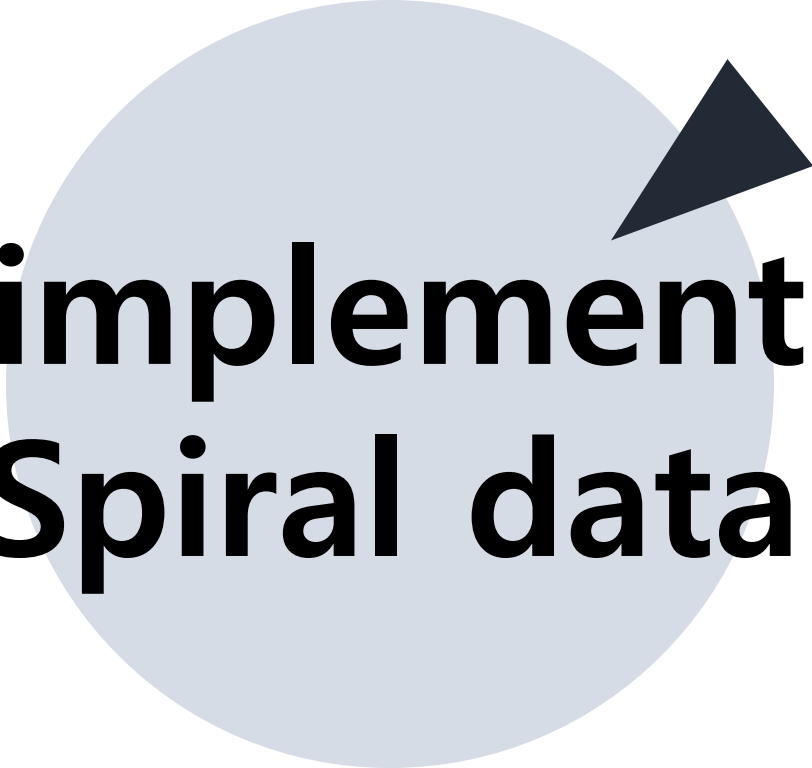


Code Implementation: IRIS

Spectral Clustering



* k-means clustering 결과와 비교하였을 때, 유의미한 차이점을 발견할 수 없음.



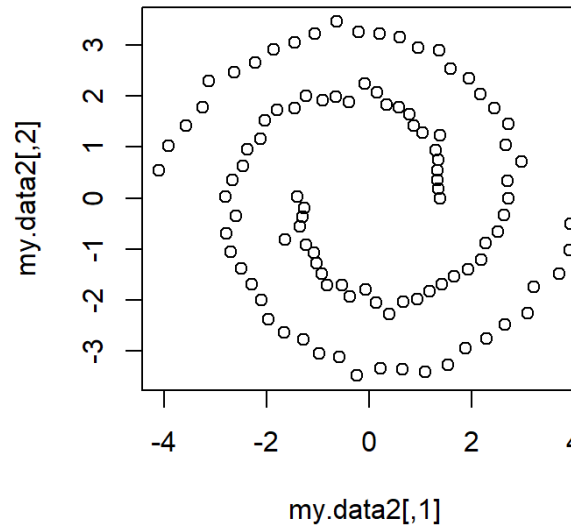
**Code implementation:
Spiral data**

Code Implementation: Spiral data

Data generation

```
library(mlbench)
obj <- mlbench.spirals(100,1,0.025)
my.data2 = 4 * obj$x
plot(my.data2)
```

mlbench 라이브러리를 활용하여 나선형 데이터 생성

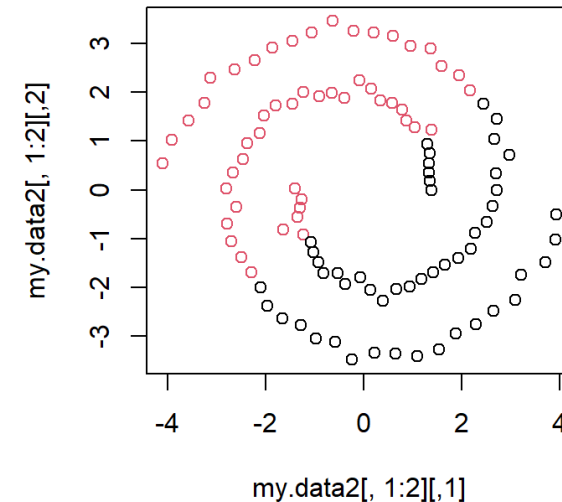


Code Implementation: Spiral data

k-means clustering

```
km2<-kmeans(my.data2, centers = 2, iter.max = 10000)  
plot(my.data2[,1:2], col=km2$cluster)
```

kmeans() 함수를 활용하여 나선형 데이터를 clustering
이때, centers = 2로 설정해줌으로써 2개의 그룹으로 clustering 진행



Code Implementation: Spiral data

Similarity Matrix

```
s <- function(x1, x2, alpha=1) {  
  exp(- alpha * norm(as.matrix(x1-x2), type="F"))  
}  
  
make.similarity <- function(my.data, similarity) {  
  N <- nrow(my.data)  
  S <- matrix(rep(NA,N^2), ncol=N)  
  for(i in 1:N) {  
    for(j in 1:N) {  
      S[i,j] <- similarity(my.data[i,], my.data[j,])  
    }  
  }  
  S  
}  
  
s2 <- make.similarity(my.data2, s)
```

함수 s 는 Gaussian kernel 로 두 벡터 x1과 x2 간의 유사도를 계산

함수 make.similarity 의 인자로 나선형 데이터와 함수 s를 사용하여
나선형 데이터의 Similarity matrix 생성



Code Implementation: Spiral data

Weighted Adjacency Matrix

```
make.affinity <- function(S, n.neighbor=5) {  
  N <- length(S[,1])  
  if (n.neighbor >= N) { # fully connected  
    W <- S  
  } else {  
    W <- matrix(rep(0,N^2), ncol=N)  
    for(i in 1:N) { # for each line  
      # only connect to those points with larger similarity  
      best.similarities <- sort(S[i,], decreasing=TRUE)[1:n.neighbor]  
      for (s in best.similarities) {  
        j <- which(S[i,] == s)  
        W[i,j] <- S[i,j]  
        W[j,i] <- S[i,j] # to make an undirected graph  
      }  
    }  
  }  
  return(W)  
}  
W2 <- make.affinity(S2, 3)
```

make.affinity 함수를 사용하여 W2 행렬 생성

W2는 앞서 만든 Similarity matrix의 각 행에 대해서

가장 큰 3개의 값만 남기고 나머지 값은 0으로 만든 행렬



Code Implementation: Spiral data

Laplacian Matrix

```
# Diagonal matrix
D2 <- diag(apply(W2, 1, sum)) # sum rows
eigen_result2<-eigen(D2, symmetric = TRUE)
D_minus_half2 <- eigen_result2$vectors %*% diag(1/sqrt(eigen_result2$values)) %*% t(eigen_result2$vectors)
L2 <- D2 - W2
L_sym2 <- (D_minus_half2 %*% L2 %*% D_minus_half2)
```

앞서 만든 W2 행렬에서 각 행의 합을 대각 원소로 가지는 D2 행렬 생성

D2 행렬과 W2 행렬을 빼서 Unnormalized Laplacian Matrix L2 생성

L2 행렬의 앞 뒤에 D_minus_half2 행렬을 곱해서 Normalized Laplacian Matrix L_sym2 생성



Code Implementation: Spiral data

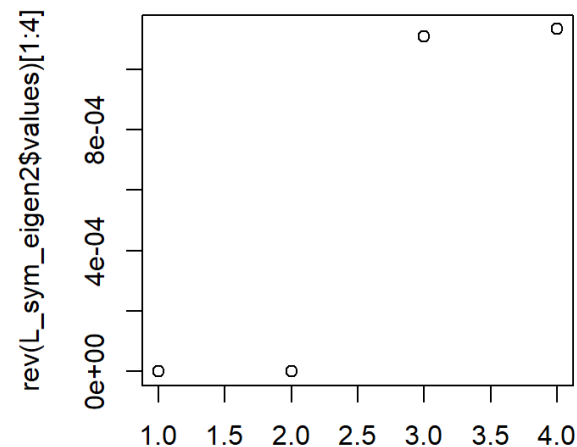
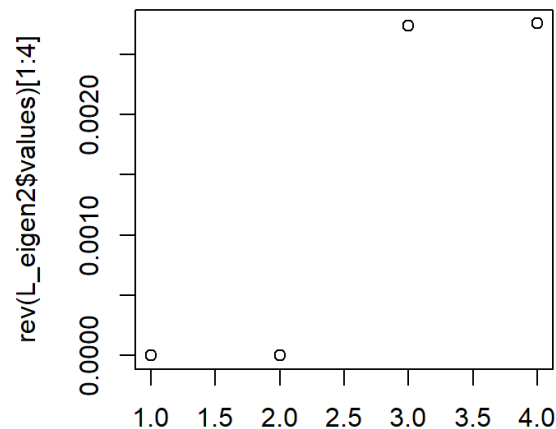
Eigen Gap

```
L_eigen2 <- eigen(L2, symmetric=TRUE)  
plot(rev(L_eigen2$values)[1:4])
```

```
L_sym_eigen2 <- eigen(L_sym2, symmetric=TRUE)  
plot(rev(L_sym_eigen2$values)[1:4])
```

Unnormalized Laplacian Matrix 와 Normalized Laplacian Matrix 모두
세 번째로 작은 고유값에서 급격히 증가함을 확인 가능

이를 통해, $k = 2$ 로 설정



Code Implementation: Spiral data

Spectral Clustering

```
k2 = 2
```

```
Z_unnormal2 <- L_eigen2$vectors[, (ncol(L_eigen2$vectors)-k2+1):ncol(L_eigen2$vectors)]
```

```
Z_sym2 <- L_sym_eigen2$vectors[, (ncol(L_sym_eigen2$vectors)-k2+1):ncol(L_sym_eigen2$vectors)]
```

	[,1]	[,2]		[,1]	[,2]
[1,]	-0.009416527	-0.141107509	[1,]	0.12252956	0.02934129
[2,]	0.141107509	-0.009416527	[2,]	-0.03360311	0.14032697
[3,]	-0.009416527	-0.141107509	[3,]	0.13207745	0.03162765
[4,]	0.141107509	-0.009416527	[4,]	-0.03458169	0.14441353
[5,]	0.141107509	-0.009416527	[5,]	-0.03885677	0.16226635
[6,]	0.141107509	-0.009416527	[6,]	-0.03453488	0.14421806
[7,]	-0.009416527	-0.141107509	[7,]	0.17238056	0.04127875
[8,]	-0.009416527	-0.141107509	[8,]	0.13395644	0.03207760
[9,]	-0.009416527	-0.141107509	[9,]	0.13728129	0.03287378
[10,]	0.141107509	-0.009416527	[10,]	-0.03441773	0.14372884
[11,]	0.141107509	-0.009416527	[11,]	-0.03393120	0.14169709
[12,]	0.141107509	-0.009416527	[12,]	-0.03320219	0.13865275
[13,]	-0.009416527	-0.141107509	[13,]	0.13778607	0.03299465
[90,]	0.141107509	-0.009416527	[90,]	-0.03178803	0.13274719
[91,]	-0.009416527	-0.141107509	[91,]	0.13213081	0.03164043
[92,]	-0.009416527	-0.141107509	[92,]	0.13660503	0.03271184
[93,]	0.141107509	-0.009416527	[93,]	-0.03148435	0.13147903
[94,]	0.141107509	-0.009416527	[94,]	-0.03119661	0.13027742
[95,]	0.141107509	-0.009416527	[95,]	-0.03378265	0.14107673
[96,]	-0.009416527	-0.141107509	[96,]	0.13333664	0.03192918
[97,]	0.141107509	-0.009416527	[97,]	-0.03143929	0.13129085
[98,]	-0.009416527	-0.141107509	[98,]	0.13453714	0.03221666
[99,]	-0.009416527	-0.141107509	[99,]	0.15951846	0.03819876
[100,]	0.141107509	-0.009416527	[100,]	-0.02968076	0.12394719

앞서 k = 2로 설정했으므로 Laplacian Matrix의 첫 번째로 작은 고유값에

대응하는 고유벡터와 두 번째로 작은 고유값에 대응하는 고유벡터를 열로 가지는 행렬 생성

Z_unnormal2 는 Unnormalized Laplacian Matrix 의 경우

Z_sym2 는 Normalized Laplacian Matrix 의 경우



Code Implementation: Spiral data

Spectral Clustering

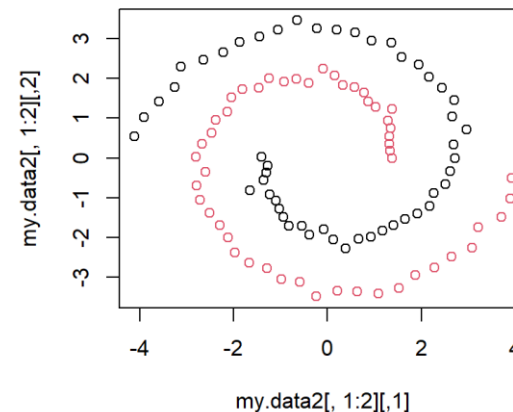
```
library(stats)
result_unnormal1 <- kmeans(Z_unnormal1, centers=k1, nstart=5)
result_sym1 <- kmeans(Z_sym1, centers=k1, nstart=5)

plot(my.data1[,1:2], col=iris$species)
title(main = "true data")

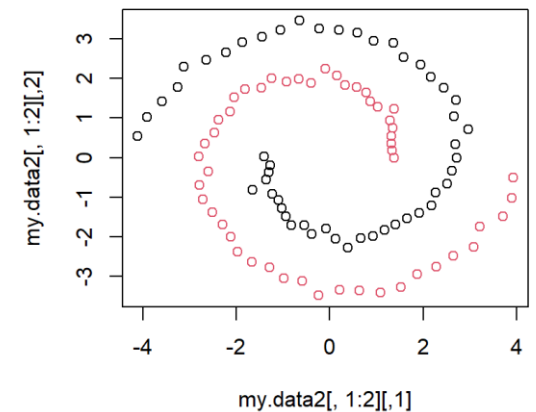
plot(my.data1[,1:2], col=result_unnormal1$cluster)
title(main = "Clustering with Unnormalized laplacian")

plot(my.data1[,1:2], col=result_sym1$cluster)
title(main = "Clustering with Symmetric laplacian")
```

Clustering with Unnormalized laplacian



Clustering with Symmetric laplacian



앞서 만든 Z_unnormal2 와 Z_sym2 행렬의 각 행에 대해 k-means clustering 을 진행
나선형 데이터에 대해서 Spectral Clustering 이 잘 적용되었음을 확인 가능





Conclusion

comparison

Spectral clustering

vs

k-means clustering

- 고차원의 복잡한 형태의 데이터셋에서 효과적
- 계산 복잡성이 높아 너무 큰 데이터 세트에는 적합하지 않을 수 있다.

- 간단하고 직관적이다 (매개변수 선택 간단)
- 원형 데이터에서 잘 작동

Why?

Relaxation?

Of course, the relaxation we discussed above is not unique. For example, a completely different relaxation which leads to a semi-definite program is derived in Bie and Cristianini (2006), and there might be many other useful relaxations. The reason why the spectral relaxation is so appealing is not that it leads to particularly good solutions. Its popularity is mainly due to the fact that it results in a standard linear algebra problem which is simple to solve.

NP hard problem → Linear Algebra problem!



Contribution

김시은, 김근영, 허정웅 : A tutorial on Spectral Clustering 논문 리딩, 요약

도현수, 오동윤 : Code Implementation

