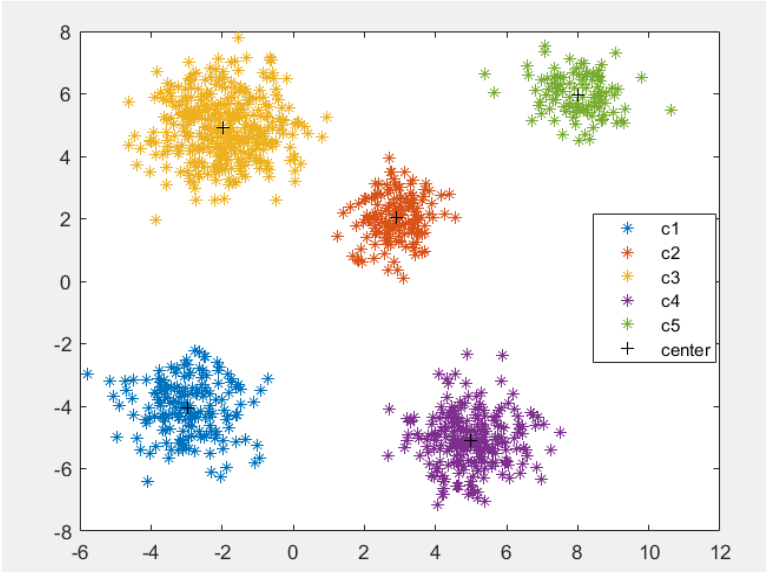
소프트웨어 특강

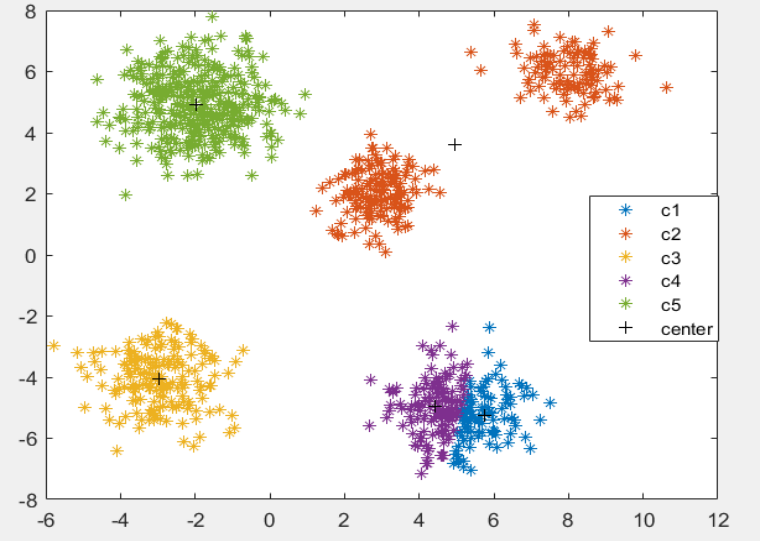
2012313779
소프트웨어학과
김진홍

1- 1

k = 5 로 설정하고 center들을 각각 (-6, 12) (-8, 8) 구간에서 random initialization을 통해
구현한 결과입니다.



분류가 잘 된 경우



분류가 잘 되지 않은 경우

가끔 선언한 center 여러 붙어
하나의 Cluster에서 생성되었을 경우, 하나의 Cluster를 두 개 혹은 세 개로 나누는 경우가 있습니다.
이러한 경우 반대쪽에서는 두 개의 Cluster가 하나의 Cluster로 인식이 되기 때문에 중심이 두 Cluster 사이에 위치하게 되여 objective function이 거의 수렴하지 않았습니다.

첨부한 사진과 같이 Cluster가 분류될 때에는 약 obj_function이 1575정도의 값으로 수렴이 되었고, 제대로 나뉘지 않은 경우는 약 3818 정도의 값을 가졌고 7515의 값을 가지기도 합니다.

1-2

분류가 잘 된 경우는 대부분이 5번의 iteration 이내로 수렴하는 결과를 확인하였습니다.

그러나 밑에 첨부한 두번째 사진은 Cluster가 제대로 분류되지 않아서 수렴을 하지 않은 경우입니다. Random initialization이기 때문에 원치 않는 경우가 나오기도 하는 것을 확인했습니다.

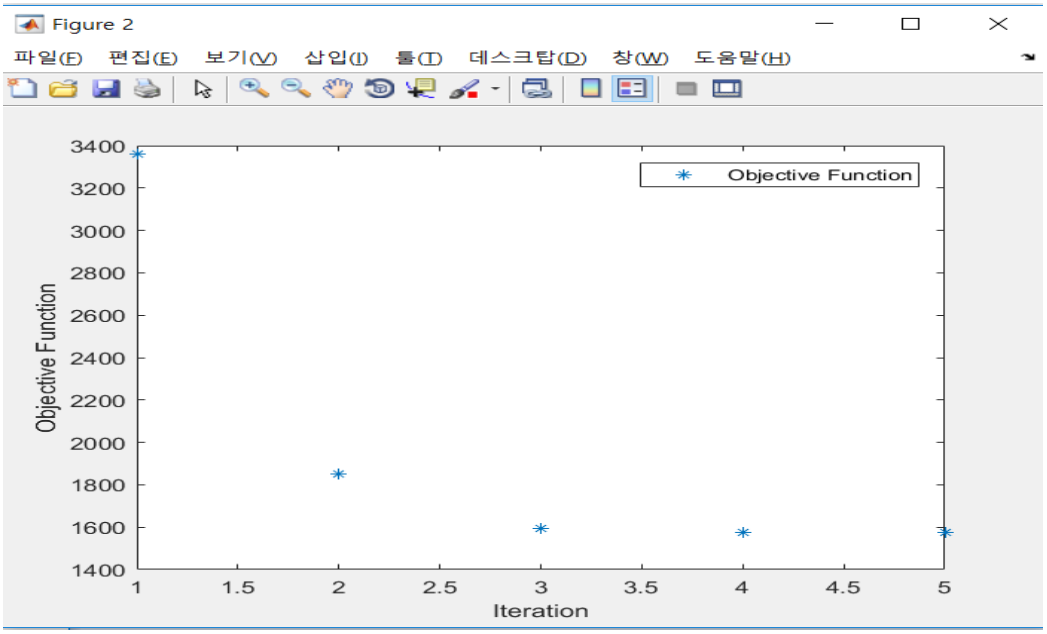
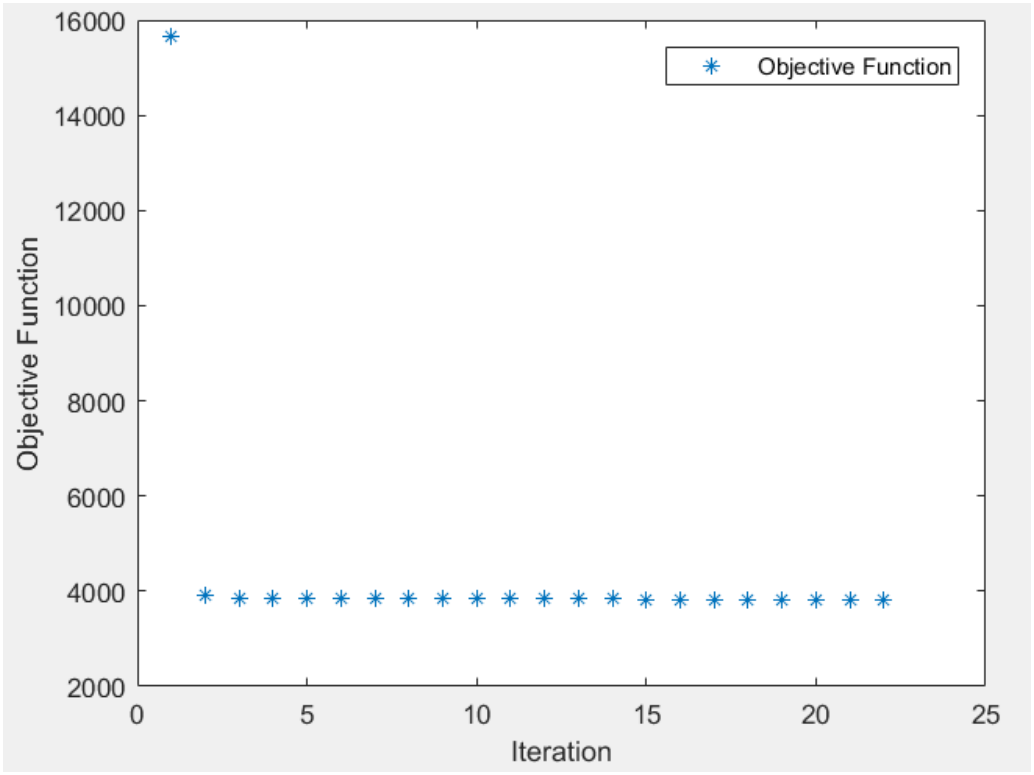
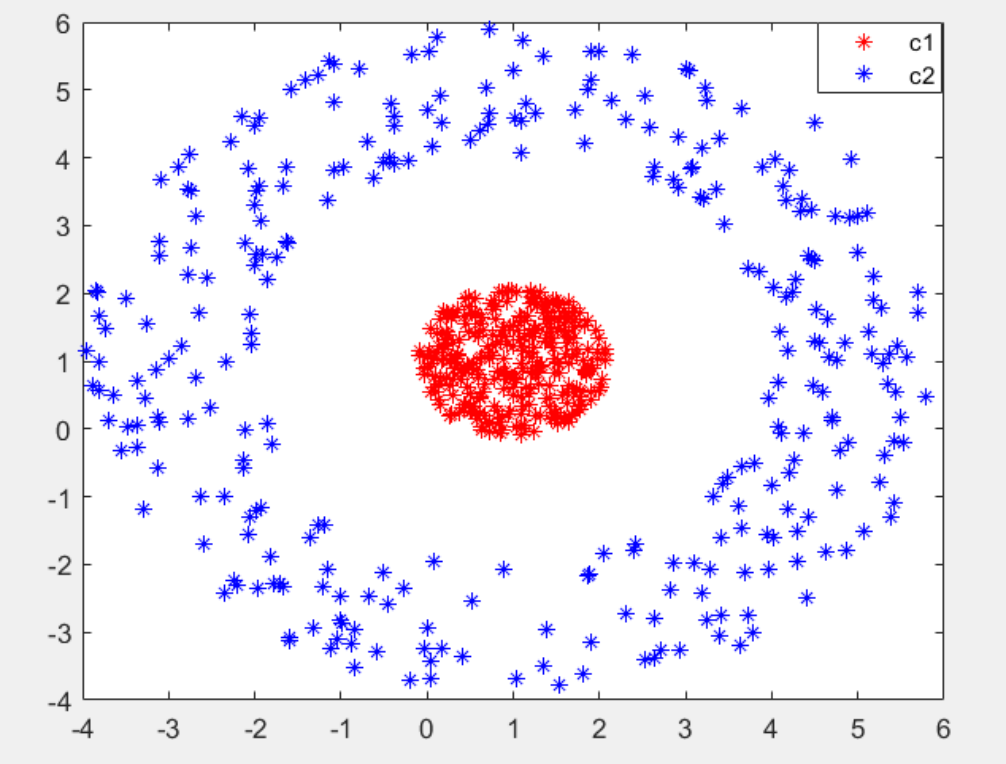


Figure 2
파일(F) 편집(E) 보기(V) 삽입(I) 툴(T) 데스크탑(D) 창(W) 도움말(H)
Objective Function
Objective Function
Iteration
3400
3200
3000
2800
2600
2400
2200
2000
1800
1600
1400
1
1.5
2
2.5
3
3.5
4
4.5
5


Objective Function
Objective Function
Iteration
16000
14000
12000
10000
8000
6000
4000
2000
0
5
10
15
20
25

2. k = 2로 설정하고 Kernel option에 true를 준 결과입니다.



Y의 윗부분 300개는 1번 데이터로 아랫부분 300개는 2번 데이터로 수렴하는 것을 확인하였습니다.

Gaussian RBF kernel을 적용하는데 있어서 c = [0.1, 0.1] 을 사용하였습니다.
[0.5, 0.5], [1, 1]과도 같은 값을 넣어보았을 때 실행시간은 체감상 조금 더 빨랐지만 objective function이 더 큰 값을 가졌고, [0.01, 0.01]과 같은 값을 넣어보니 시간도 오래 걸리고 데이터를 linear하게 구분하는 것을 확인하였기에
[0.1, 0.1]이 가장 적합하다 생각하였습니다. Y 데이터의 표준편차를 대입하여도 마찬가지로 linear하게 구분할 뿐이었습니다.

이렇게 적용한 Kernel matrix를 사용한 결과로
kernel objective function은 267 정도의 값을 얻었습니다.