이해동
정보과학과
202584-010115

머신러닝특론 과제2 리포트

서론

본 보고서는 베이즈 분류기를 직접 구현하고, 가우시안 확율밀도 함수를 적용한 경우의 베이즈 분류기의 성능을 테스트 해 봅니다. 또한 비 모수적인 방법인 K-NN 분류를 통하여 확율밀도 함수를 추정할 수 없는 경우에 대해서 분류를 할 수 있는 방법을 테스트 해 봅니다.

문제 1 베이지안 분류기 구현 및 테스트

문제 1.1 산점도 그리기

본 문제에서는 다음과 같은 평균과 공분산 행렬을 갖는 가우시안 분포를 따르는 두 개의 클래스 데이터를 각각 100개씩 생성하였습니다.

$$\mu 1=\begin{pmatrix}0\\0\end{pmatrix}, \Sigma 1=\begin{pmatrix}4 & 0\\0 & 4\end{pmatrix}, \mu 2=\begin{pmatrix}3\\5\end{pmatrix}, , \Sigma 2=\begin{pmatrix}3 & 0\\0 & 5\end{pmatrix}$$

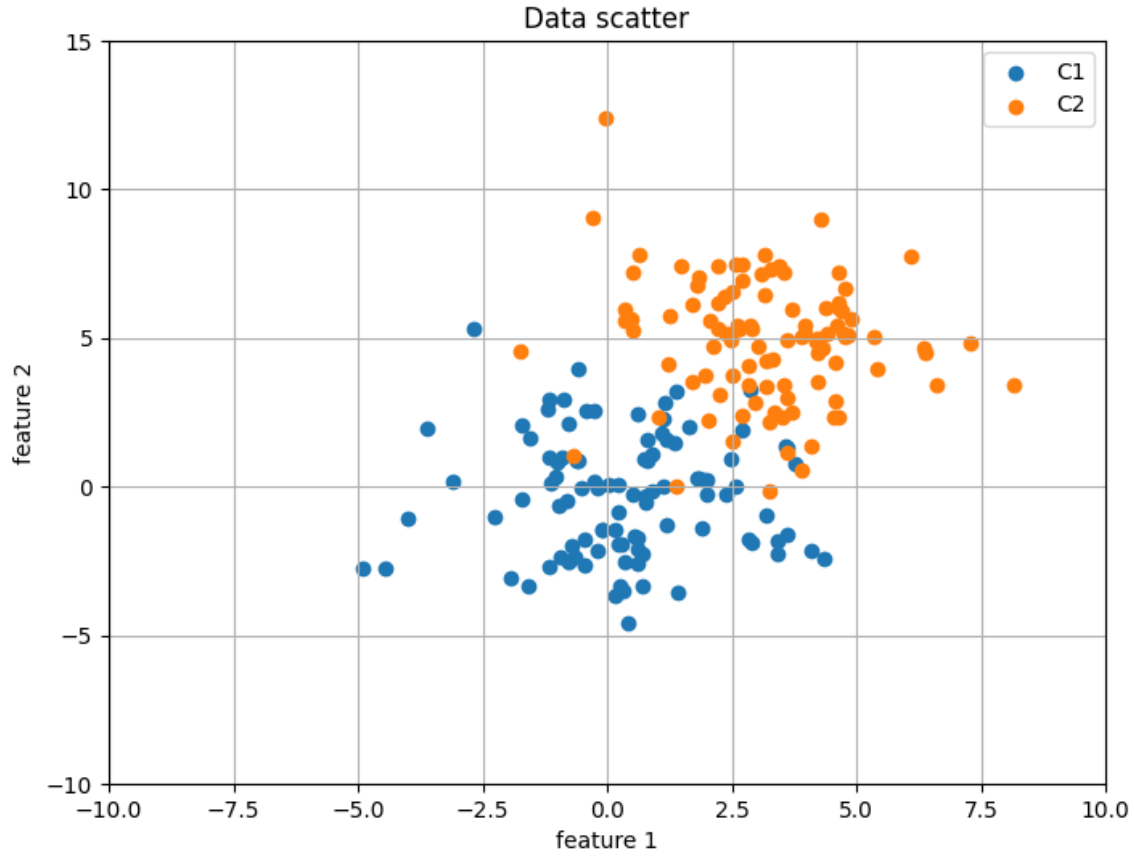
생성된 데이터의 산점도는 <그림 1>과 같습니다.

산점도는 python 의 numpy 라이브러의 np.random.multivariate_normal() 함수를 통하여 주어진 평균과 공분산을 가진 가우시안 분포를 따르는 데이터를 생성하였습니다.

Source code : Bayes1_1.py

결과

- 클래스 1 : (0,0)을 중심으로 퍼져있는 형태
- 클래스 2 : (3,5)를 중심으로 퍼져있는 형태



<그림 1>

문제 1.2 베이즈 분류기를 통해 데이터 분류

앞서 생성한 산점도 데이터를 기반으로 베이즈 분류기를 통해 클래스를 판별할 판별 함수 2개를 작성하였습니다. 새로운 데이터 x1, x2 를 전달하여 공통공분산 판별함수와 일반공분산 판별함수의 결과를 계산합니다.

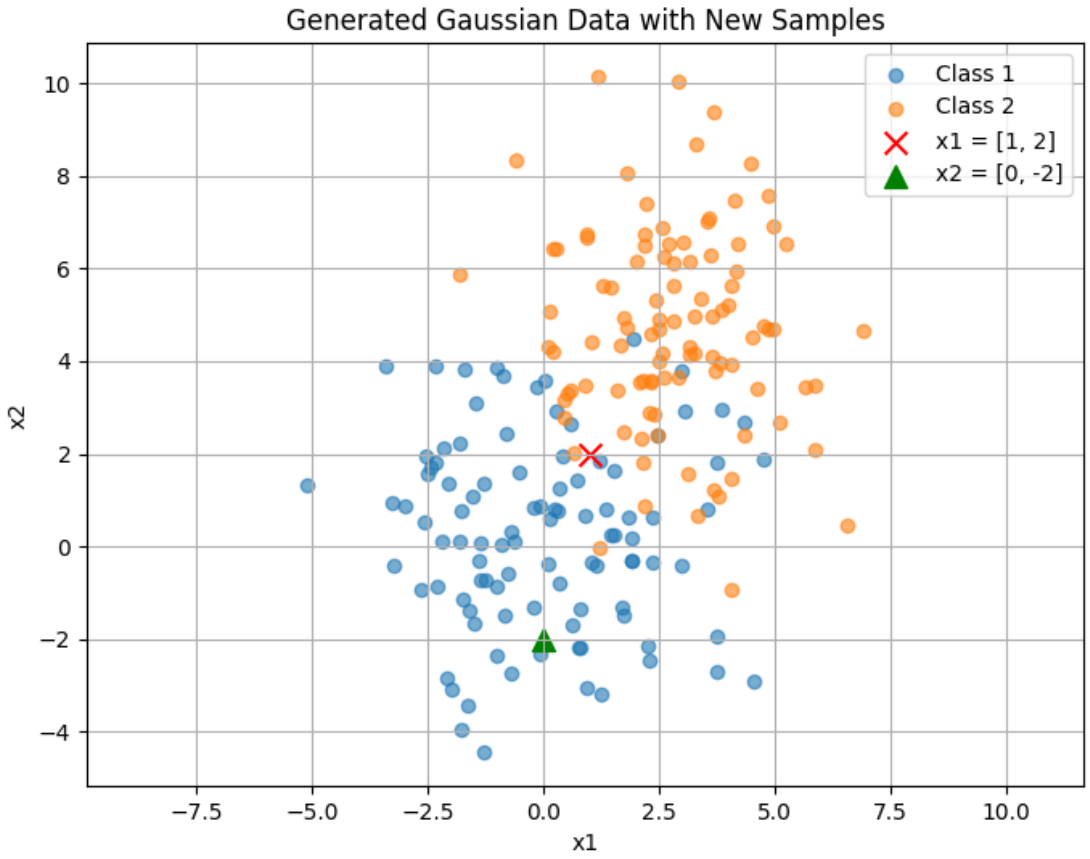
- classify_common_cov() : 클래스 공통 공분산 행렬 계산
- classify_general_cov() : 일반적인 공분산 행렬 계산

새로운 데이터 x1, x2 를 베이즈 분류기로 분류한 결과는 모든 상황에서 클래스 1으로 분류되었습니다.

Source code : Bayes1_2.py

결과

```
hoolworks/MachineLearning/HW2/Bayes1_2.py
x1 = [1 2] => 공통공분산: 클래스 1, 일반공분산: 클래스 1
x2 = [ 0 -2] => 공통공분산: 클래스 1, 일반공분산: 클래스 1
```



<그림 2>

문제 2

(1) iris 데이터를 K-NN 분류기로 분류

첨부된 iris 데이터는 [150, 4] 의 shape 을 가진 데이터와 [150, 1] 의 shape 을 가진 class 분류값으로 되어 있는 데이터 입니다.

K-NN 분류기 구현은 '참고교재의 프로그램 5-1' 을 참고하여 제작 되었습니다.

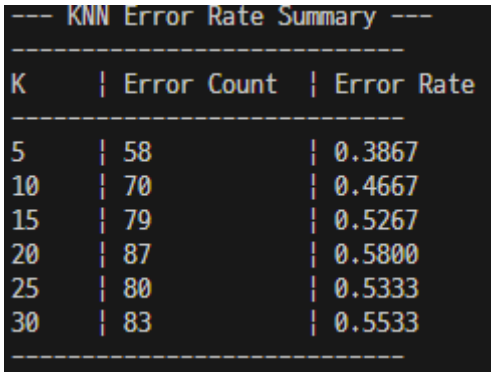
150개의 각각의 데이터가 다른 데이터간의 거리(norm) 를 계산하여 가장 상위 K 개만큼의 클래스 투표를 통해 가장 많은 클래스로 판명된 것으로 클래스를 할당합니다.

classify_KNN() 함수는 위의 내용을 구현한 코드 입니다.

K 값을 변경해가며 시험해 본 결과 K 가 5일 때 가장 좋은 괄과를 보였습니다.

Source code : KNN2_1.py

결과



```
--- KNN Error Rate Summary ---
---------------------------
K     | Error Count  | Error Rate
---------------------------
5     | 58           | 0.3867
10    | 70           | 0.4667
15    | 79           | 0.5267
20    | 87           | 0.5800
25    | 80           | 0.5333
30    | 83           | 0.5533
---------------------------
```

<그림 3>

(2) iris 데이터를 120개의 학습데이터, 30개의 테스트 데이터로 나누어 테스트 데이터에 대한 오분류율 표로 정리

기존의 데이터를 40 + 10 으로 분리하여 테스트 데이터와 학습데이터를 분리 하였습니다.

40	10
40	10
40	10

=>

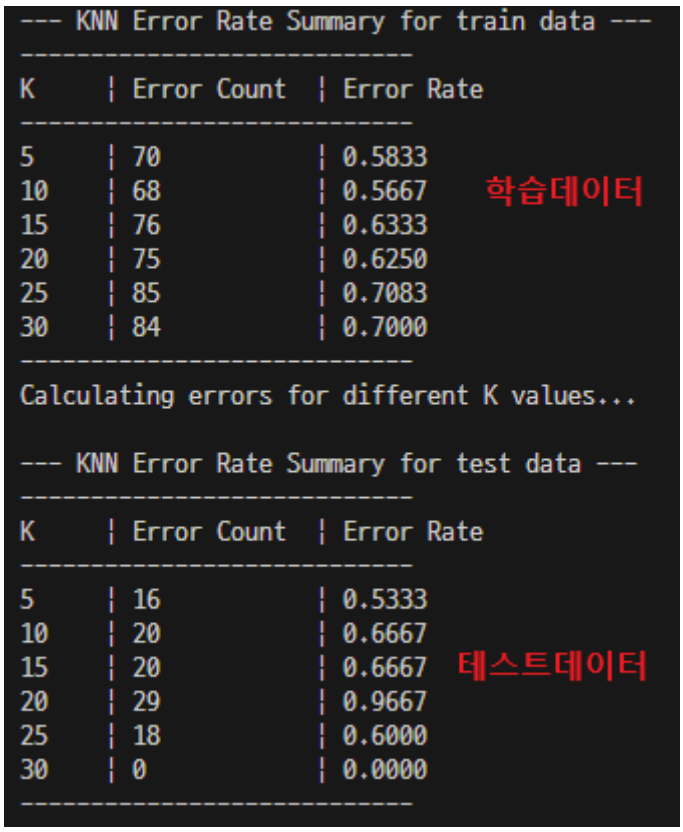
40 + 40 + 40	10 + 10 + 10

코드는 앞서 문제 2-(1) 에서 작성한 코드를 거의 그대로 사용하였습니다.

결과값을 비교해 보면 코드에 문제가 있는지, 테스트 데이터의 결과가 전반적으로 오류율이 늘어난 결과를 보입니다.

Source code : KNN2_2.py

결과


--- KNN Error Rate Summary for train data ---
K | Error Count | Error Rate
5 | 70 | 0.5833
10 | 68 | 0.5667
15 | 76 | 0.6333
20 | 75 | 0.6250
25 | 85 | 0.7083
30 | 84 | 0.7000
Calculating errors for different K values...
--- KNN Error Rate Summary for test data ---
K | Error Count | Error Rate
5 | 16 | 0.5333
10 | 20 | 0.6667
15 | 20 | 0.6667
20 | 29 | 0.9667
25 | 18 | 0.6000
30 | 0 | 0.0000
학습데이터
테스트데이터

<그림 4>