

Project No.1

Learning Iris Data classification without tools

소프트웨어융합대학

2017 March 26

Student ID/Name : 20113337 최영근

Contents

1. 프로젝트 개요 및 환경
 - A. 개요
 - B. 컴파일 환경, 사용 언어, etc
2. 프로젝트 설명
 - A. Project 1-1-a
 - B. Project 1-1-b
 - C. Project 1-1-c
 - D. Project 1-2-a, 1-2-b, 1-2-e
 - E. Project 1-2-c
 - F. Project 1-2-d
 - G. Project 1-2-f

1. 프로젝트 개요 및 환경

A. 프로젝트 개요

주어진 Iris Data 를 Machine Learning tool 없이 분류할 수 있는 Classifier 를 만든다.

분류기를 만들기 위해 Machine Learning 의 기초인 Bayes Classifier 이론을 바탕으로 훈련 한다.

데이터(Training Data)로부터 객체를 분류하는 **지도 학습** (Supervised Learning) 방법을 사용한다. 훈련 데이터들의 값을 출력하고 주어진 입력 벡터가 어떤 종류의 값인지 분류(Classification) 하고 화면에 출력한다. 훈련 데이터로부터 주어진 데이터가 올바른지 확인 한다.

B. 컴파일 환경 및 사용언어, etc

OS : MAC OS

Compile : MAC OS Terminal

사용 언어 : Python ver2.7.4

사용 라이브러리 : numpy, matplotlib

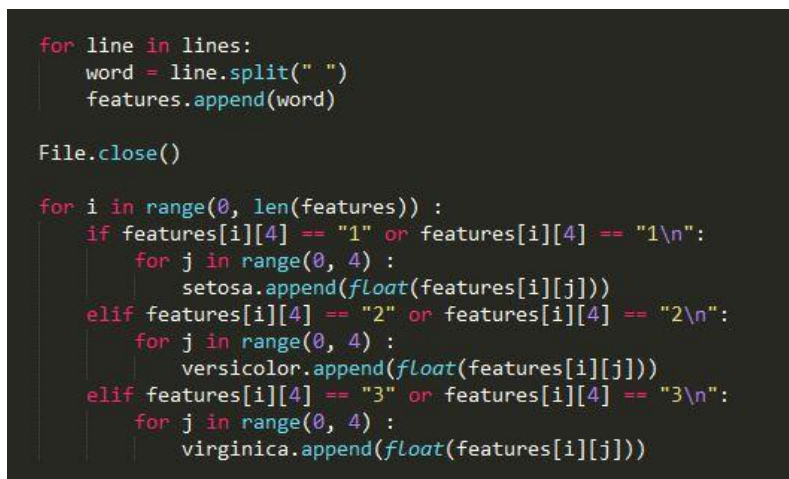
1. 프로젝트 개요 및 환경



The screenshot shows a text editor window titled 'Iris_test.dat.txt - 메모장'. The menu bar includes '파일(F)', '편집(E)', '서식(O)', '보기(V)', and '도움말(H)'. The text content is as follows:

```
5.0 3.6 1.4 0.2 1
4.9 3.1 1.5 0.1 1
5.8 4.0 1.2 0.2 1
5.1 3.8 1.5 0.3 1
4.8 3.4 1.9 0.2 1
4.7 3.2 1.6 0.2 1
4.9 3.1 1.5 0.2 1
5.1 3.4 1.5 0.2 1
5.1 3.8 1.9 0.4 1
5.0 3.3 1.4 0.2 1
```

➔ Data file format



```
for line in lines:
    word = line.split(" ")
    features.append(word)

File.close()

for i in range(0, len(features)) :
    if features[i][4] == "1" or features[i][4] == "1\n":
        for j in range(0, 4) :
            setosa.append(float(features[i][j]))
    elif features[i][4] == "2" or features[i][4] == "2\n":
        for j in range(0, 4) :
            versicolor.append(float(features[i][j]))
    elif features[i][4] == "3" or features[i][4] == "3\n":
        for j in range(0, 4) :
            virginica.append(float(features[i][j]))
```

➔ File read code

주어진 Input data 를 space 단위로 재 배열 -> \r, \n 을 제거하여 코드 상에서 space 단위로 쉽게 스플릿하여 저장한다.

2. 프로젝트 설명

A. Project 1-1-a

```
for i in range(0, len(features)) :
    if features[i][4] == "1" or features[i][4] == "1\n":
        for j in range(0, 4) :
            setosa.append(float(features[i][j]))
    elif features[i][4] == "2" or features[i][4] == "2\n":
        for j in range(0, 4) :
            versicolor.append(float(features[i][j]))
    elif features[i][4] == "3" or features[i][4] == "3\n":
        for j in range(0, 4) :
            virginica.append(float(features[i][j]))
```

- 1) 각각의 클래스마다 리스트의 형태로 데이터를 저장 한다.
- 2) 리스트로 저장된 각 클래스의 데이터들에 대한 평균과 공분산 행렬을 계산

평균 : 1*4 Matrix

공분산 행렬 : 4*4 Matrix

```
----- project #1-1 -----
-----setosa Mean-----
[ 4.9975  3.4175  1.4425  0.2525]
-----versicolor Mean-----
[ 5.99  2.7775  4.31  1.3325]
-----virginica Mean-----
[ 6.61  2.97  5.5575  2.03 ]
-----setosa Covariance Matrix-----
[[ 0.13512179  0.10748077  0.02805769  0.01244231]
 [ 0.10748077  0.15686538  0.01564744  0.00854487]
 [ 0.02805769  0.01564744  0.02507051  0.00642949]
 [ 0.01244231  0.00854487  0.00642949  0.01230128]]
-----versicolor Covariance Matrix-----
[[ 0.28041026  0.10079487  0.20035897  0.06648718]
 [ 0.10079487  0.11666026  0.09561538  0.04767308]
 [ 0.20035897  0.09561538  0.23528205  0.0804359 ]
 [ 0.06648718  0.04767308  0.0804359  0.04327564]]
-----virginica Covariance Matrix-----
[[ 0.44194872  0.0854359  0.34453846  0.04687179]
 [ 0.0854359  0.09497436  0.06638462  0.03297436]
 [ 0.34453846  0.06638462  0.35173718  0.04951282]
 [ 0.04687179  0.03297436  0.04951282  0.05548718]]
```

➔ 각 클래스의 평균, 공분산 행렬 출력 결과

2. 프로젝트 설명

B. Project 1-1-b

Discriminant Functions 을 이용하여 Decision Boundary 을 계산 한다.

$$g_i(x) = x^t V_i x + v_i^t x + v_{i0}$$

$$\text{where } V_i = -\frac{1}{2} \Sigma_i^{-1}$$

$$v_i = \Sigma_i^{-1} \mu_i,$$

$$\text{and } v_{i0} = -\frac{1}{2} \mu_i^t \Sigma_i^{-1} \mu_i - \frac{1}{2} \ln |\Sigma_i| + \ln P(w_i)$$

이 공식을 이용하여 각 클래스의 $g(x)$ 값을 계산 한다.

```
-----
Name : calVi0()
Input = 2*1 mean matrix, 2*2 covarianceMatrix
Output = value of vi0
Description : Calculate the required value for the discriminantFunction
-----
def calVi0(meanMatrix, covarianceMatrix):

    vi0 = -0.5*np.dot(np.dot(np.transpose(meanMatrix), np.linalg.inv(covarianceMatrix)), meanMatrix)
    - 0.5*np.log2(numpy.linalg.det(covarianceMatrix))

    return vi0

-----
Name : discriminantFunctions()
Input = 2*n matrix, 2*1 mean matrix, 2*2 covarianceMatrix, value of vi0
Output = value(gi)
Description : Calculated using discriminantFunctions
-----
def discriminantFunctions(IrisTestData, meanMatrix, covarianceMatrix, vi0):

    cal1 = np.dot(np.dot(np.transpose(IrisTestData), -0.5*np.linalg.inv(covarianceMatrix)), IrisTestData)
    cal2 = np.dot(np.transpose(np.dot(np.linalg.inv(covarianceMatrix), meanMatrix)), IrisTestData)

    result = cal1 + cal2 + vi0
    return result
```

➔ Code of calculating discriminant function

각 클래스 마다 테스트 데이터에 대한 g 값의 list 를 구함 (size = 10)

2. 프로젝트 설명

C. Project 1-1-c

$$W12 \rightarrow g1(x) - g2(x) = 0$$

$$W23 \rightarrow g2(x) - g3(x) = 0$$

$$W31 \rightarrow g3(x) - g1(x) = 0$$

위의 공식을 이용하여 각 클래스의 Hyperplane 을 분리하여 입력된 테스트 데이터가 잘 분류 되었는지 계산 한다.

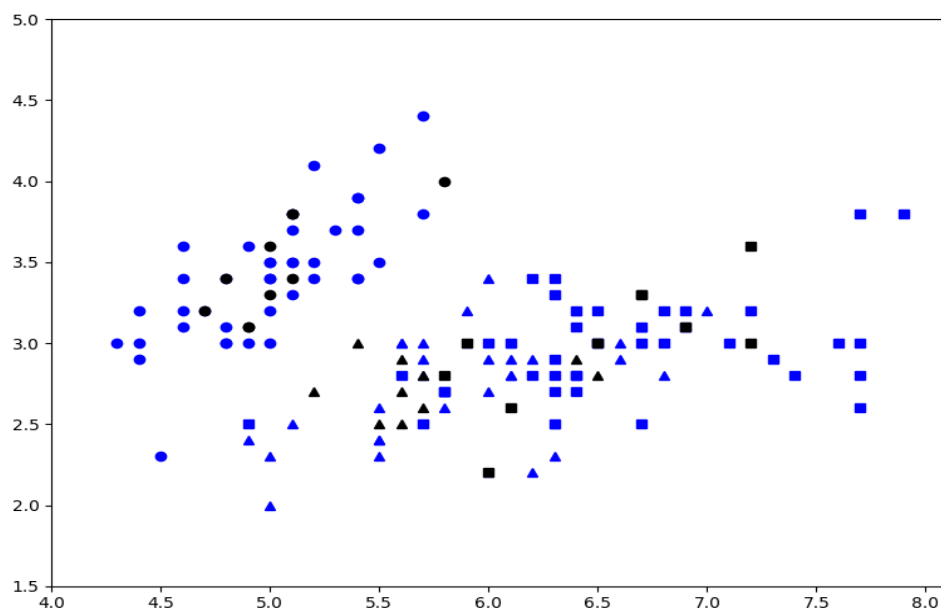
```
----confusion Matrix----
[[ 10.  0.  0.]
 [  0. 10.  0.]
 [  0.  0. 10.]]
```

➔ Discriminant function 을 사용하여 구한 confusion matrix 의 출력 결과

2. 프로젝트 설명

D. Project 1-2-a, 1-2-b, 1-2-e

Training data 의 2 가지 특징만 가지고 데이터를 분류하기 위해 각 클래스 별로 1-1 에서 저장했던 데이터, 평균, 공분산 행렬에서 2 가지 특징만 가져와서 새로운 변수에 저장하고 2 차원 데이터들을 출력 한다.



➔ Test data(black color), Training data(blue color)를 모두 출력한 결과

(원 : setosa data, 삼각형 : versicolor data, 사각형 : virginica data)

```
varianceMatrix[1] project #1-2 -----
---mean of setosa 2features---
[ 4.9975  3.4175]
varianceMatrix[1][1]
---mean of versicolor 2features---
[ 5.99  2.7775]
---mean of virginica 2features---
[ 6.61  2.97]
---covariance setosa of 2features---
[[ 0.13512179  0.10748077]
 [ 0.10748077  0.15686538]]
---covariance versicolor of 2features---
[[ 0.28041026  0.10079487]
 [ 0.10079487  0.11666026]]
---covariance virginica of 2features---
[[ 0.44194872  0.0854359 ]
 [ 0.0854359  0.09497436]]
```

➔ Mean matrix & covariance Matrix of 2features training data

2. 프로젝트 설명

E. Project 1-2-c

마할라노비스 거리 (Mahalanobis distance) : 평균과의 거리가 표준편차의 몇 배인지 나타내는 값

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$

➔ Mahalanobis distance formula

```
"""
-----
Name : calMahalanobisDistance()
Input = 2*n matrix, 2*1 mean matrix, 2*2 covariance matrix
Output = expression of Z
Description : calculate and plot the each mahalanobis distance in graph
-----
"""
def calMahalanobisDistance(Iris, meanVector, covarianceMatrix):

    irisMBPlot = np.zeros([2, 40])

    count = 0
    for i in range(0, 160):
        if i%4 == 0:
            irisMBPlot[0][count] = Iris[i]
            irisMBPlot[1][count] = Iris[i+1]
            count += 1

    xx, yy = np.meshgrid(np.arange(4, 8.1, 0.05), np.arange(1, 6, 0.05))

    matrix = np.linalg.inv(covarianceMatrix)

    temp1 = (meanVector[0]-xx)*(matrix[0][0])+(meanVector[1]-yy)*(matrix[1][0])
    temp2 = (meanVector[0]-xx)*(matrix[0][1])+(meanVector[1]-yy)*(matrix[1][1])

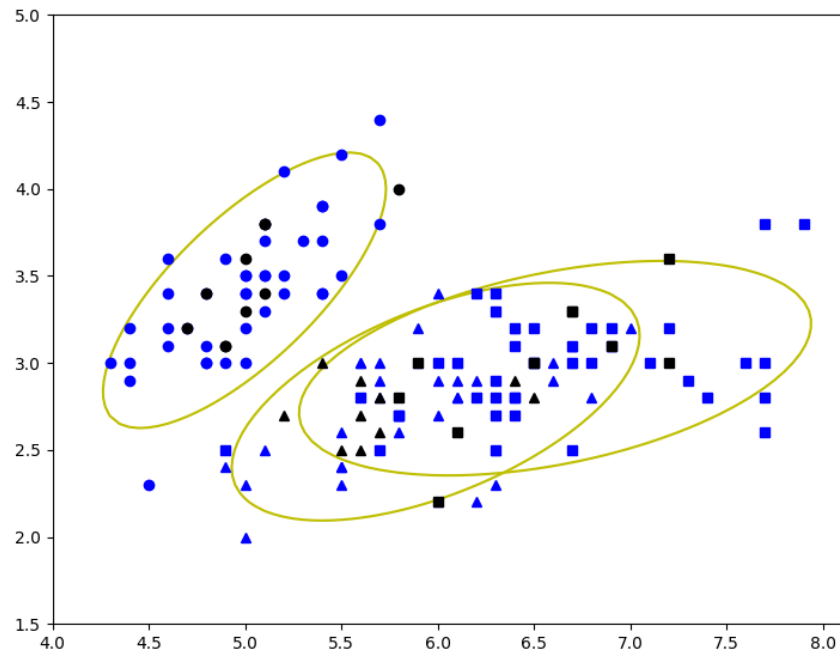
    Z = np.sqrt(temp1*(meanVector[0]-xx) + temp2*(meanVector[1]-yy)) - 2
    plt.contour(xx, yy, Z, [0], colors = 'y')

    return Z
```

➔ Code of calculating & plotting Mahalanobis distance = 2

주어진 트레이닝 데이터가 어느 일정범위안에 존재하기 때문에 x와 y의 범위를 지정하고 그 사이 데이터들을 0.05의 단위로 쪼개어 마할라노비스 거리 2를 갖는 데이터(Z)들을 모두 출력한다.

2. 프로젝트 설명



➔ 세 클래스의 마할라노비스 거리가 2 인 데이터들을 출력한 결과

(원 : setosa data, 삼각형 : versicolor data, 사각형 : virginica data)

2. 프로젝트 설명

F. Project 1-2-d

1-1-b 의 방법과 동일하게 계산

이번엔 마할라노비스 거리가 2 인 것들의 집합인 $g(x)$ 리스트를 가지고 Hyperplane 을 분리 한다.

```
"""
Name : calGlistofMD()
Input = 2*n matrix, 2*1 mean matrix, 2*2 covariance matrix
Output = g value list of each class
Description : Generate list to determine test data
"""
def calGlistofMD(Iris, meanVector, covarianceMatrix):

    glist = np.zeros((1,10))
    matrix = np.linalg.inv(covarianceMatrix)

    for i in range(0, len(Iris)):
        temp1 = (meanVector[0]-Iris[i][0])*(matrix[0][0])+(meanVector[1]-Iris[i][1])*(matrix[1][0])
        temp2 = (meanVector[0]-Iris[i][0])*(matrix[0][1])+(meanVector[1]-Iris[i][1])*(matrix[1][1])

        Z = np.sqrt(temp1*(meanVector[0]-Iris[i][0]) + temp2*(meanVector[1]-Iris[i][1])) -2
        glist[0][i] = Z

    return glist
```

➔ 각 클래스마다 마할라노비스 거리가 2 인 데이터들을 리스트에 넣고 반환하는 함수

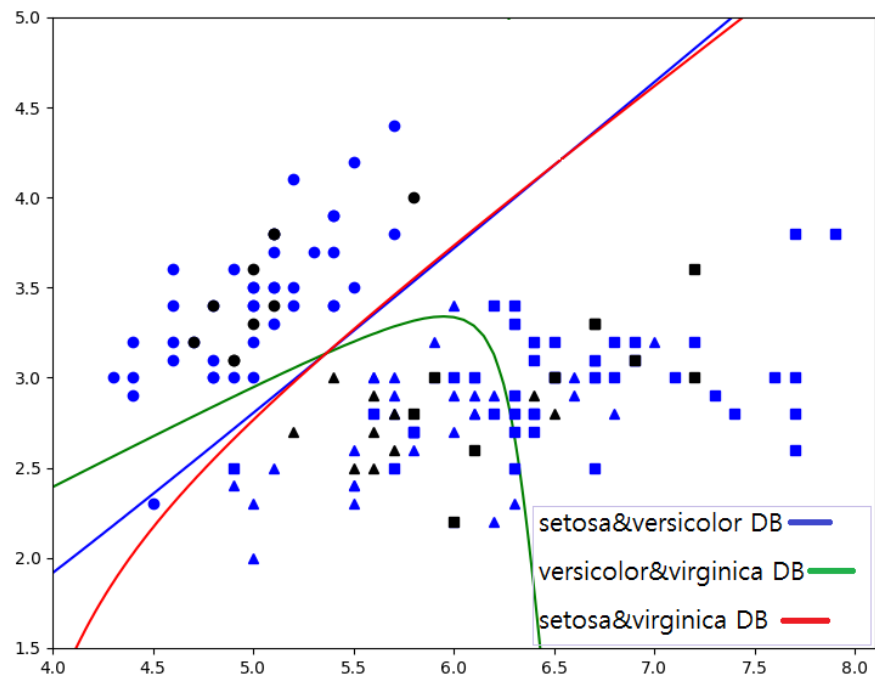
```
"""
Name : plotThedecisionBoundaries()
Input = three of Iris g value
Description : plot the each decision boundary in graph
"""
def plotThedecisionBoundaries(g1, g2, g3):

    xx, yy = np.meshgrid(np.arange(4, 8.1, 0.05), np.arange(1, 6, 0.05))

    plt.contour(xx, yy, g1-g2, [0], colors = 'b')
    plt.contour(xx, yy, g2-g3, [0], colors = 'g')
    plt.contour(xx, yy, g3-g1, [0], colors = 'r')
```

➔ 위에서 구한 각 클래스의 g 값을 가지고 decision boundary 를 계산하고 화면에 출력 하는 함수

2. 프로젝트 설명



→ Decision boundary 을 출력한 결과

2. 프로젝트 설명

G. Project 1-2-f

위에서 구한 Decision boundary 를 가지고 test data 를 분류하고

confusion matrix 를 계산한다. 예측과 다른 데이터들은 다른 색으로 화면에 출력한다.

```
-----
Name : plotClassifyTestData()
Input = 2*n matrix of each class, g value list of each class, state num
Output = 3*3 confusion matrix
Description : Use the boundary to display data in the correct area
              and calculate confusion matrix
-----
def plotClassifyTestData(Iris1, Iris2, Iris3, gList1, gList2, gList3, num):

    cf1, cf2, cf3 = 0, 0, 0
    matrix = np.zeros(3)
    for i in range(0, 10):
        if num == 0:
            if gList1[0][i] - gList1[1][i] > 0:
                cf2+=1
                plt.plot(Iris1[i][0], Iris1[i][1], 'ro')
            elif gList1[0][i] - gList1[2][i] > 0:
                cf3+=1
                plt.plot(Iris1[i][0], Iris1[i][1], 'ro')
            else:
                cf1+=1
        elif num == 1:
            if gList2[1][i] - gList2[0][i] > 0:
                cf1+=1
                plt.plot(Iris2[i][0], Iris2[i][1], 'r^')
            elif gList2[1][i] - gList2[2][i] > 0:
                cf3+=1
                plt.plot(Iris2[i][0], Iris2[i][1], 'r^')
            else:
                cf2+=1
        elif num == 2:
            if gList3[2][i] - gList3[0][i] > 0:
                cf1+=1
                plt.plot(Iris3[i][0], Iris3[i][1], 'rs')
            elif gList3[2][i] - gList3[1][i] > 0:
                cf2+=1
                plt.plot(Iris3[i][0], Iris3[i][1], 'rs')
            else:
                cf3+=1

    matrix[0], matrix[1], matrix[2] = cf1, cf2, cf3
    return matrix
```

➔ Code of classify & plot Testdata and calculating confusion matrix

2. 프로젝트 설명

1-1-b 에서 구했던 것처럼 Discriminant Functions 을 이용하여 Decision Boundary 을 계산한다.

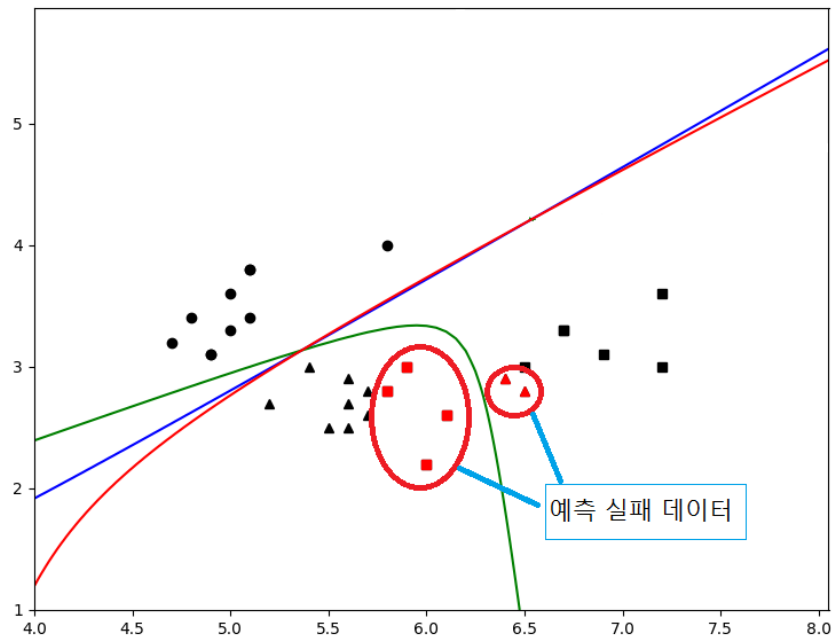
인자로 각각의 테스트 데이터, $g(x)$ 리스트 값, 테스트 데이터의 예측 값이 들어간다.

테스트 데이터의 예측 값에 따라 각각의 if 문 안으로 들어가게 되고 해당 테스트 데이터가 decision boundary 안에 있는지 판별하여 다른 클래스에 들어가 있다면 다른 색(빨간색)으로 출력한다.

마지막으로 클래스 마다 confusion matrix 를 계산하여 반환한다.

```
----Confusion Matrix 2f of test data
[[ 10.  0.  0.]
 [  0.  8.  2.]
 [  0.  4.  6.]]
```

➔ 최종적인 confusion matrix 출력 결과



➔ 분류한 테스트 데이터들의 최종적인 출력 결과

Setosa data -> 모두 정상

Versicolor data -> 2 개 오류 (virginica 데이터에 2 개 포함됨)

Virginica data -> 4 개 오류 (versicolor 데이터에 4 개 포함됨)

의 결과를 도출하였습니다.