labsol2

January 27, 2022

1 Machine Learning - Laboratory 2

[1]:
```
import getopt
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from matplotlib import cm
from pdffuns import *
```

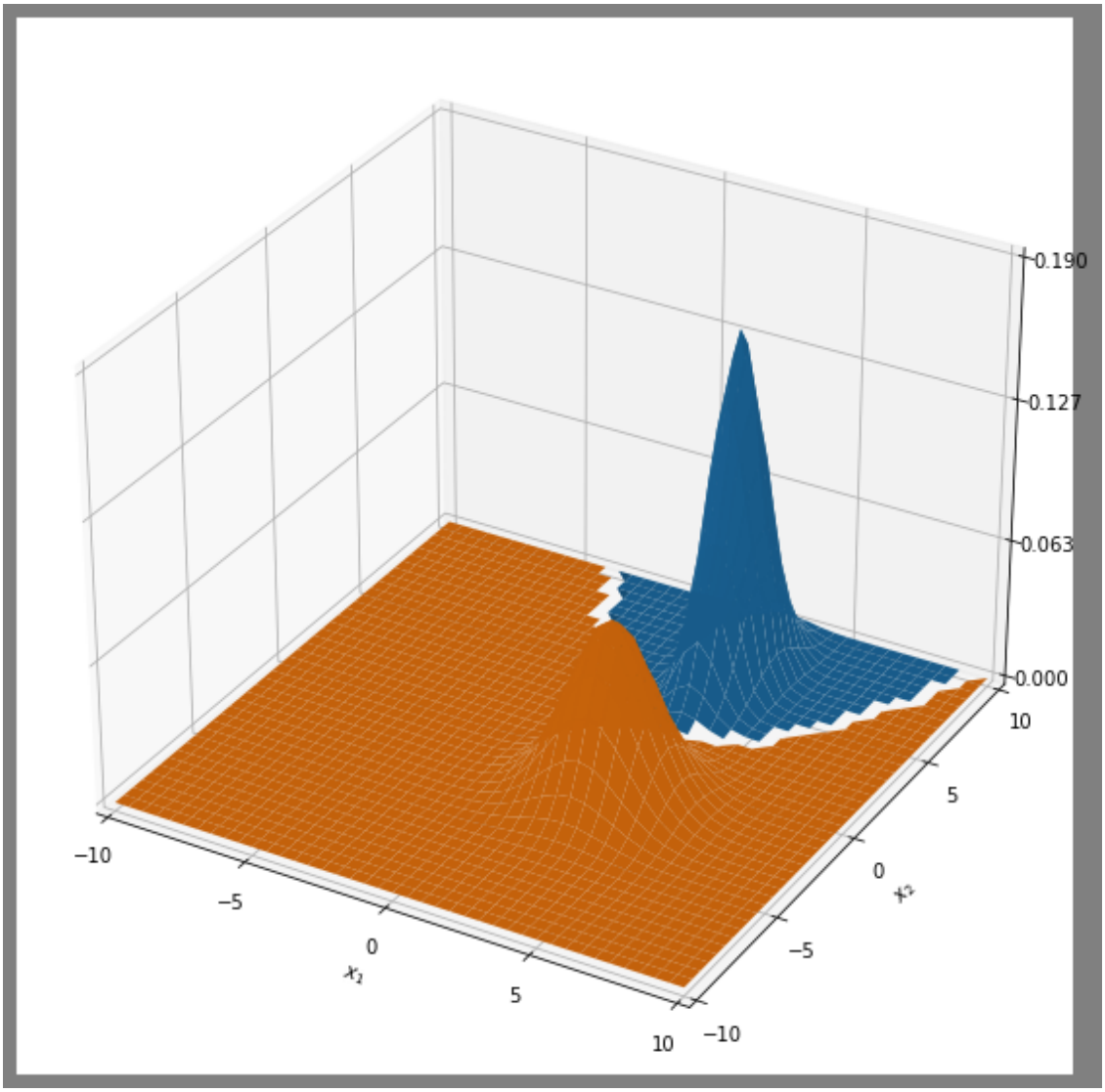
[2]:
```
def labsol2(my, Sgm, Pw, discr='pxw'):

    # Initialise values
    x1 = np.arange(-10,10.5,0.5).reshape(-1,1)
    x2 = np.arange(-10,10.5,0.5).reshape(-1,1)

    # Get coordinates grid
    X1, X2 = np.meshgrid(x1, x2)

    # Pack everything
    X = np.dstack((X1, X2))

    # Determine class specific probability density functions, pxw[i], i = 0,...
↪,M-1
    M = my.shape[0]
    # - initialise pxw as empty list
    pxw = np.empty(shape=(M, X.shape[0], X.shape[1]))
    # - initialise total density function, px as zero
    px = 0
    for i in range(M):
        pxw[i] = norm2D(my[i], Sgm[i], X)
        px = px + Pw[i] * pxw[i]

    # Determine discriminant functions, g[i], i = 0,...,M-1
    g = np.empty(shape=(M, X.shape[0], X.shape[1])) # - initialise g as empty␣
↪list
    # - iterate over classes, i = 0,...,M-1
```

```
    for i in range(M):
        #    - on condition of discr determine selected discriminant function
        if discr=='s_pxw':
            # - Scaled pdfs
            g[i] = Pw[i] * pxw[i]
        elif discr=='pp':
            # - Posterior probability
            g[i] = (Pw[i] * pxw[i]) / px
        elif discr=='pxw':
            # - pdfs (not really discriminant functions)
            g[i] = pxw[i]

    return x1, x2, g
```

2 Sections a) and b)

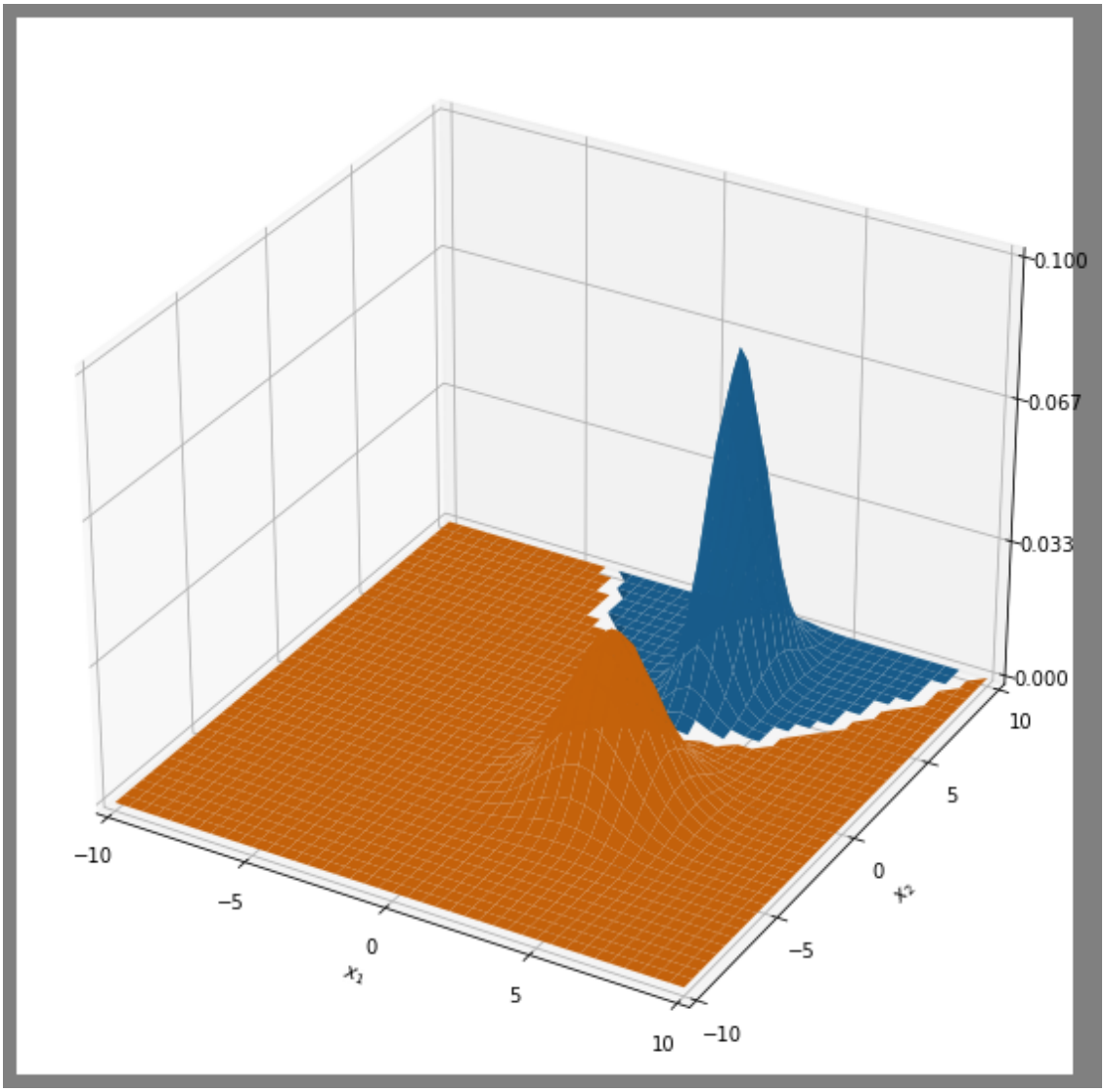
[3]:
```
# labsol2() uses the norm2D function to compute the discriminant function

# Define parameters
my = np.array([ [[3], [6]], [[3], [-2]] ])
Sgm = np.array([ [[0.5, 0], [0, 2]], [[2, 0], [0, 2]] ])
Pw = np.array([0.5, 0.5])

# Choose a discriminant function:
    # pxw --> Class-conditional PDF
    # pp --> Posterior probability
    # s_pxw --> Scaled PDF

x1, x2, g = labsol2(my, Sgm, Pw, 'pxw') # Generate discriminant functions
classplot(g, x1, x2, 1, gsv={'gsv': 1, 'figstr': 'pdf'})    # Plot both␣
↪discriminant functions
```

```
C:\Users\aeste\OneDrive - UNIVERSIDAD DE HUELVA\Universidad\4º Carrera
(Stavanger)\Spring Semester\ML\Laboratory\Lab2\pdffuns.py:60:
MatplotlibDeprecationWarning: Calling gca() with keyword arguments was
deprecated in Matplotlib 3.4. Starting two minor releases later, gca() will take
no keyword arguments. The gca() function should only be used to get the current
axes, or if no axes exist, create new axes with default keyword arguments. To
create a new axes with non-default arguments, use plt.axes() or plt.subplot().
  ax = fig.gca(projection='3d')
C:\Users\aeste\OneDrive - UNIVERSIDAD DE HUELVA\Universidad\4º Carrera
(Stavanger)\Spring Semester\ML\Laboratory\Lab2\pdffuns.py:72: UserWarning: Z
contains NaN values. This may result in rendering artifacts.
  obj = ax.plot_surface(X1, X2, G, facecolor=col[i])
```



3 Section c)

[4]:
```
x1, x2, g = labsol2(my, Sgm, Pw, 's_pxw') # We use s_pdf this time to get the␣
↪scaled pdf (Pw*pxw)
classplot(g, x1, x2, 1, gsv={'gsv': 1, 'figstr': 's_pdf'})    # Plot both␣
↪discriminant functions
```

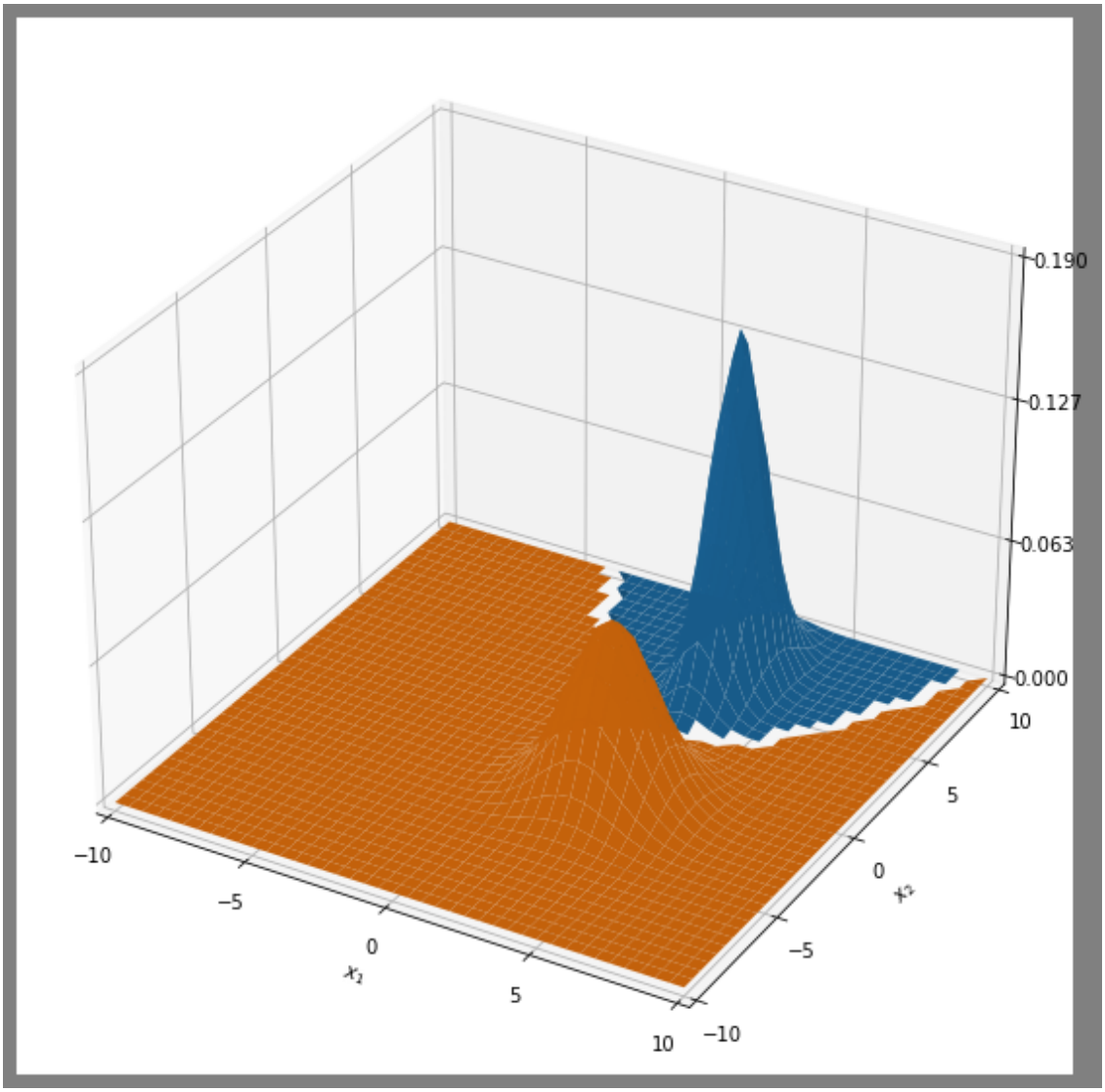


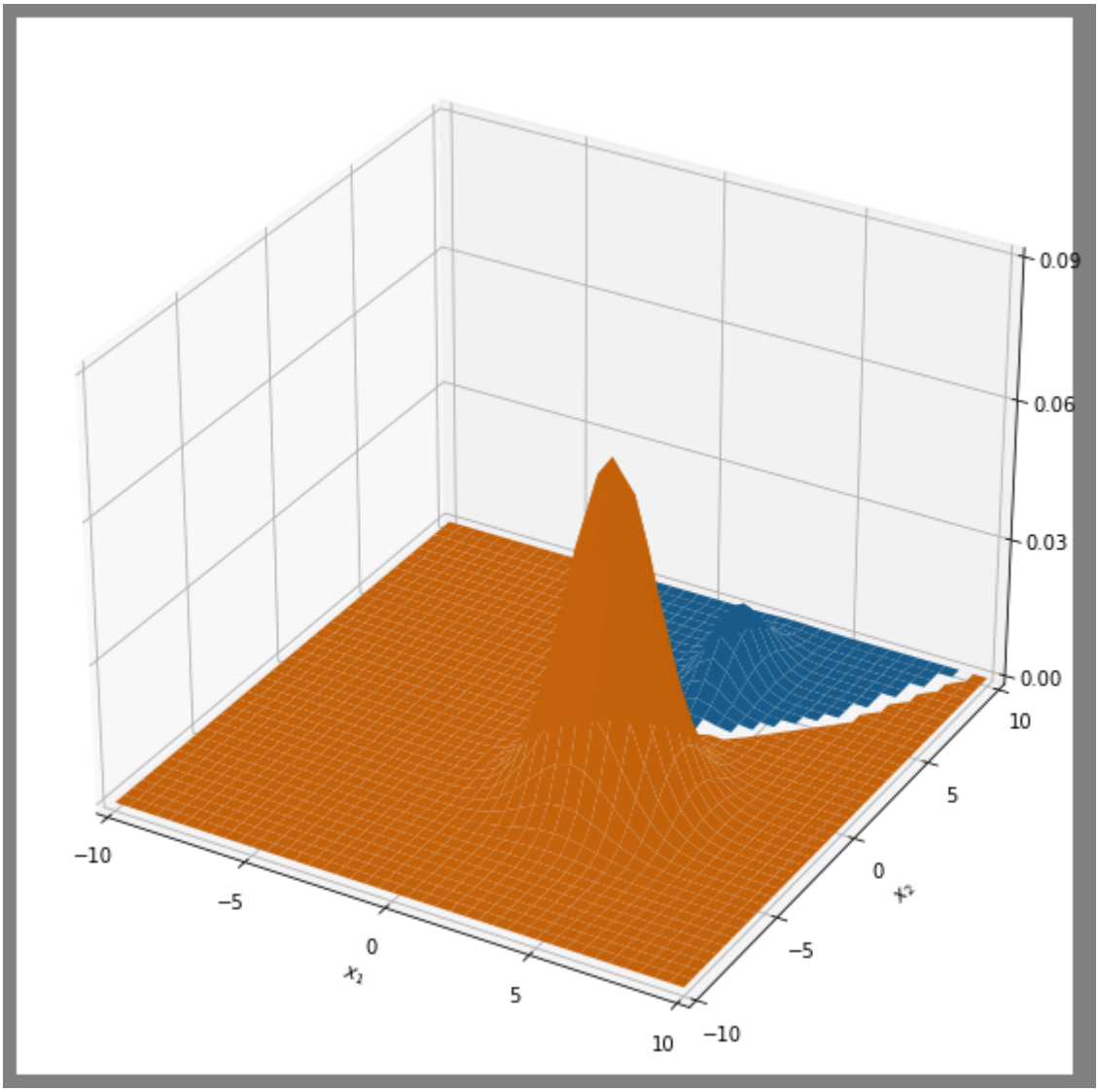
4 Section d)

The decision boundary is defined as $g_i(x) = g_j(x)$, in other words, those points were the value of g_i is the same as g_j.
We can see in the figures above the white line between both regions is the points were the functions have similar values.

5 Section e)

[5]:
```
# Define the new priori probabilities
Pw = np.array([0.1, 0.9])

x1, x2, g = labsol2(my, Sgm, Pw, 'pxw') # Generate discriminant functions
classplot(g, x1, x2, 1)    # Plot both discriminant functions

x1, x2, g = labsol2(my, Sgm, Pw, 's_pxw') # Generate discriminant functions
classplot(g, x1, x2, 1)    # Plot both discriminant functions
```

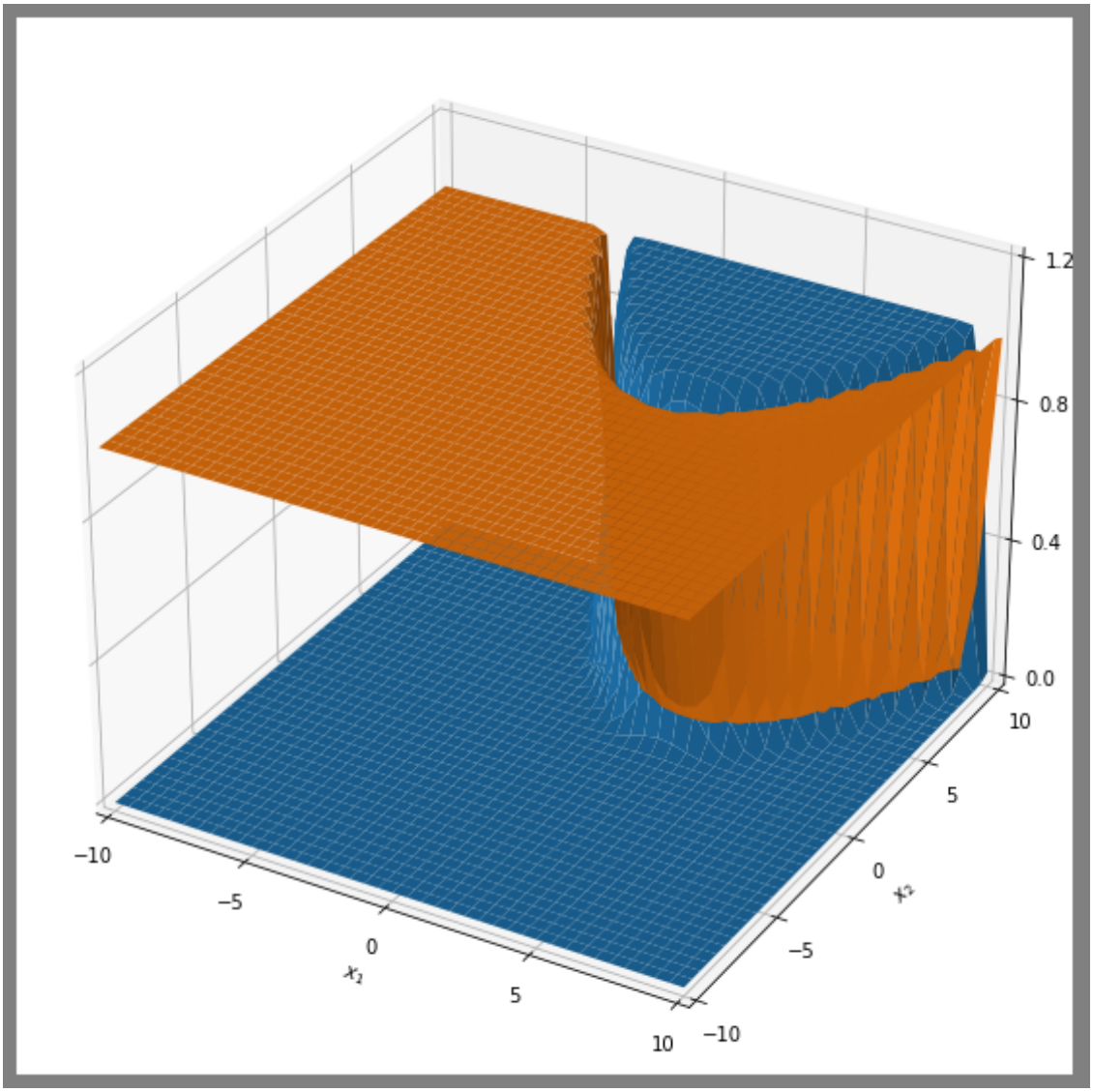




We can notice that the **PDF** remains the same while the scaled **PDF** seems very different, this is due to the direct dependence of the scaled **PDF** on the prior probabilities: $g(x) = P(w) * p(x|w)$.

6 Section f)

[6]:
```
# Restore priori probabilities
Pw = np.array([0.5, 0.5])

x1, x2, g = labsol2(my, Sgm, Pw, 'pp') # Generate discriminant functions
classplot(g, x1, x2, 0)    # Plot both discriminant functions
```



6.0.1 Students information

Matthew Bwye Lera
264491@uis.no

Álvaro Esteban Muñoz
a.estebanmunoz@stud.uis.no

Gero Kolhof
264546@uis.no