Lab3_LejunChen

In [1]:

```
%load_ext autoreload
%autoreload 2
```

In [2]:

```
import matplotlib
import matplotlib.pyplot as plt
import scipy.io
from mpl_toolkits.mplot3d import axes3d
from pdffuns import *
import sys
import getopt
import numpy as np
import scipy.io as io
import math
from pdffuns import norm2D,  classplot
import pickle

pfile='lab3.p'
with open(pfile, "rb") as fp:
     X=pickle.load(fp)
```

In [3]:

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "last" # all | last | last_expr | none
```

In [4]:

```
X[0]=np.matrix(X[0])
X[1]=np.matrix(X[1])
X
```

Out[4]:

```
[matrix([[2.8, 2. , 2.8, 3. ],
         [4.8, 5.8, 6.9, 5.8]]),
 matrix([[ 1.7,  2.9,  3.8],
         [-2.6,  0.2, -2.9]])]
```

In [5]:

```
X[1][:,1].reshape(2,1).shape
```

Out[5]:

```
(2, 1)
```

In [6]:

```
#caculate the prior probabilites for each class
np.shape(X[0])
np.shape(X[1])
X[0].shape[1]
pw0=X[0].shape[1]/(X[0].shape[1]+X[1].shape[1])
pw0=round(pw0,3)
pw1=X[1].shape[1]/(X[0].shape[1]+X[1].shape[1])
pw1=round(pw1,3)
Pw = np.array([pw0, pw1])
print("The prior probabilites for each class are: ",Pw)
```

```
The prior probabilites for each class are:  [0.571 0.429]
```

In [7]:

```
#check how many samples for each class in the datasets
N=[]
M=len(X)
for i in range(0, M):
    [l,p]=X[i].shape
    N.append(p)
print(l,N)
```

```
2 [4, 3]
```

a) mean for each class

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{N_i} \sum_{j=1}^{N} \mathbf{x}_j$$

In [8]:

```
mu=[None, None]
mu0=np.mean(X[0], axis=1)
mu1=X[1].mean(1)
mu[0]=mu0
mu[1]=mu1
print("The estimation of mean for class 1 is:\n",mu[0])
print("The estimation of mean for class 2 is:\n",mu[1])
```

```
The estimation of mean for class 1 is:
 [[2.65 ]
 [5.825]]
The estimation of mean for class 2 is:
 [[ 2.8       ]
 [-1.76666667]]
```

b) covariance matrix

$$\boldsymbol{\Sigma}_i = \frac{1}{N_i} \sum_{j=1}^{N} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T$$

In [9]:

```
S = [None, None]
S[0] = X[0]-mu[0]
S[1] = X[1]-mu[1]
X[0]
```

Out[9]:

```
matrix([[2.8, 2. , 2.8, 3. ],
        [4.8, 5.8, 6.9, 5.8]])
```

In [10]:

```
Sgm=[None, None]
Sgm[0]=1/4*(S[0][:,0]*S[0][:,0].T+S[0][:,1]*S[0][:,1].T+S[0][:,2]*S[0][:,2].T+S[0][:,3]*S[0
Sgm[0]
Sgm[1]=1/3*(S[1][:,0]*S[1][:,0].T+S[1][:,1]*S[1][:,1].T+S[1][:,2]*S[1][:,2].T)
Sgm[1]
print("The estimation of covariance matrix for class 1 is:\n",Sgm[0])
print("The estimation of covariance matrix for class 2 is:\n",Sgm[1])
```

```
The estimation of covariance matrix for class 1 is:
 [[0.1475   0.00375 ]
 [0.00375  0.551875]]
The estimation of covariance matrix for class 2 is:
 [[ 0.74       -0.00666667]
 [-0.00666667  1.94888889]]
```

In [11]:

```
def cov(Y_wi,my):
    [d,num]=Y_wi.shape
    sum=np.zeros((d, d))
    for k in range(0,num):
        temp=Y_wi[:,k].reshape(d,1)-my
        sum=sum+np.dot(temp,temp.transpose())
    cov=sum/num
    return cov
cov(X[0],mu[0])
```

Out[11]:

```
matrix([[0.1475  , 0.00375 ],
        [0.00375 , 0.551875]])
```

In [12]:

```
#cov1=np.cov(X[1])
#cov0=np.cov(X[0])
```

c) Plot the discriminant function for class 1.

In [13]:

```
#Define 25 points of computation along each axis
#x1 = np.arange(-5, 10, 0.6)
#x1 = x1.reshape(x1.size, 1)
#x2 = np.arange(-5, 10, 0.6)
#x2 = x2.reshape(x2.size, 1)
#x1.size
#x2.size
inc = 0.25
x1 = np.arange(-5, 10 + inc, inc)
x1 = x1.reshape(x1.size, 1)
x2 = np.arange(-5, 10 + inc, inc)
x2 = x2.reshape(x2.size, 1)
```

In [14]:

```
M = len(mu)
px = 0
pxw = []
g = []
```

In [15]:

```
for i in range(0, M):
    pxw.append([])
    [pxw[i], X1, X2] = norm2D(mu[i], Sgm[i], x1, x2)
    g.append(Pw[i] * pxw[i]) #scaled class probability function
    px=px+Pw[i] * pxw[i]
```

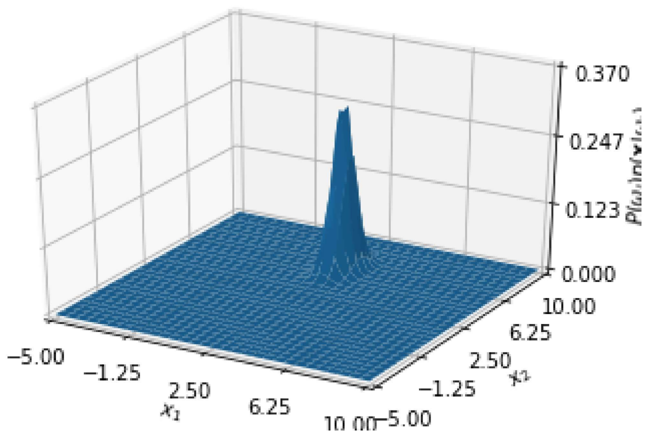
In [16]:

```
def posterior(g):
    posterior=[]
    M=len(g)
    px = sum(g)
    for i in range(0, M):
        posterior.append(np.divide(g[i], px))
    return posterior
```

In [17]:

```
print('Scaled')
gnan = 1
discr = 'Pwpxw'
q = []
q.append(Pw[0] * pxw[0])
classplot(q, x1, x2, gnan, discr, 1)
```

```
Scaled
```




```
<Figure size 432x288 with 0 Axes>
```

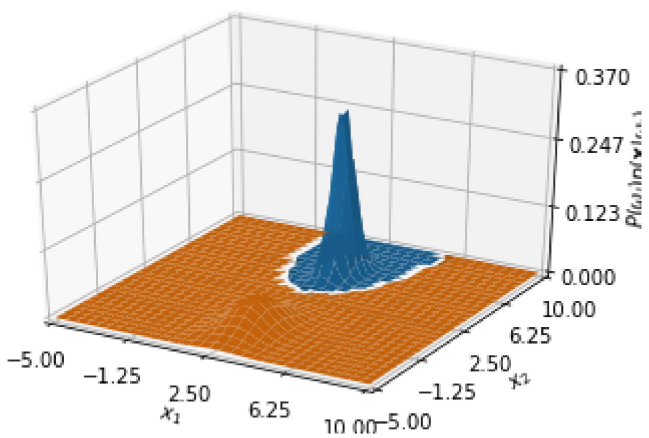
d) Plot two discriminant function in the same plot.

In [18]:

```
print('Scaled')
gnan = 1
discr = 'Pwpxw'
classplot(g, x1, x2, gnan, discr, 1)
```

```
Scaled

C:\Users\junec\OneDrive\Documents\cs\courses_term2\Machine Learning\assignme
nt_upload\exercise3\lab3_sol_lejun\pdffuns.py:64: UserWarning: Z contains Na
N values. This may result in rendering artifacts.
  obj = ax.plot_surface(X1, X2, G, facecolor=col[i])
```




```
<Figure size 432x288 with 0 Axes>
```

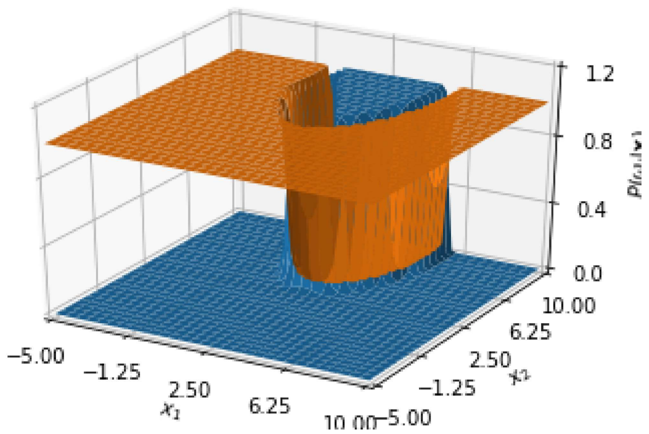
The corresponding posterior probability function plotting

$$p(x \mid \omega_1) = (p(\omega_1)p(\omega_1 \mid x))/(p(\omega_1)p(\omega_1 \mid x) + p(\omega_2)p(\omega_2 \mid x))$$

In [19]:

```
print('Posterior')
discr='Pwx'
gnan=0
classplot(posterior(g), x1, x2, gnan, discr, 1)
```

```
Posterior
```




```
<Figure size 432x288 with 0 Axes>
```

e) The decision border is shown in the above plot.

f) Repeat subtaske c-d for the Parzen classifier with window size h1 = 0.5.

In [20]:

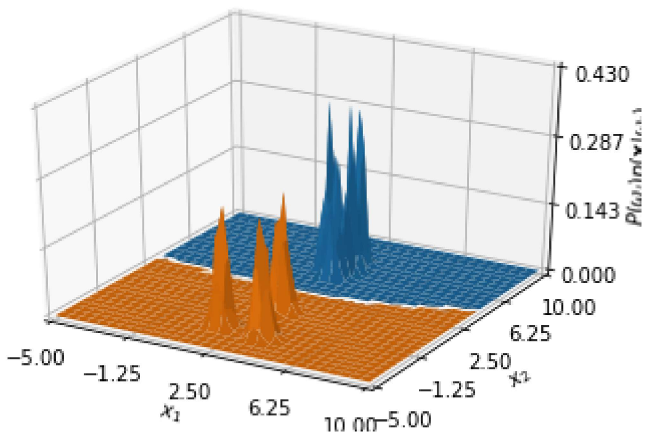
```
def ParW(h1):
    g = []
    pxw=[]
    for k in range(0,M):
        hn=h1/np.sqrt(N[k])
        hnI=hn**2*np.identity(l)
        pxw.append([])
        pxw[k]=0
        for i in range(0,N[k]):
            xk=X[k][:,i].reshape(l,1)
            [pn,X1,X2]=norm2D(xk,hnI,x1,x2)
            print("norm2D value",pn[0])
            print("norm2D value len",len(pn[0]))
            pxw[k]=pxw[k]+pn
        pxw[k]=pxw[k]/N[k]
        g.append(Pw[k] * pxw[k])
    return g
```

In [21]:

```
h1=0.5
print('Scaled')
gnan = 1
discr = 'Pwpxw'
g=ParW(h1)
classplot(g, x1, x2, gnan, discr, 1)
```

```
Scaled
norm2D value [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
norm2D value len 61
norm2D value [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
norm2D value len 61
norm2D value [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
norm2D value len 61
norm2D value [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
norm2D value len 61
norm2D value [1.99025318e-132 7.33444038e-124 1.27674680e-115 1.04983660e-
107
 4.07772356e-100 7.48157426e-093 6.48406469e-086 2.65448836e-079
 5.13326224e-073 4.68905380e-067 2.02328061e-061 4.12388133e-056
 3.97040985e-051 1.80569193e-046 3.87910125e-042 3.93638629e-038
 1.88687637e-034 4.27236469e-031 4.56953837e-028 2.30863608e-025
 5.50957078e-023 6.21096660e-021 3.30734698e-019 8.31915974e-018
 9.88458195e-017 5.54774241e-016 1.47079928e-015 1.84191534e-015
 1.08959490e-015 3.04466559e-016 4.01877112e-017 2.50568276e-018
 7.37968860e-020 1.02666587e-021 6.74682359e-024 2.09434740e-026
 3.07098169e-029 2.12708513e-032 6.95939769e-036 1.07556726e-039
 7.85204239e-044 2.70773921e-048 4.41072693e-053 3.39385002e-058
 1.23354291e-063 2.11784910e-069 1.71757182e-075 6.57981881e-082
 1.19067190e-088 1.01776945e-095 4.10947039e-103 7.83793107e-111
 7.06148698e-119 3.00517678e-127 6.04119736e-136 5.73660784e-145
 2.57315791e-154 5.45201265e-164 5.45665391e-174 2.57973504e-184
 5.76106710e-195]
norm2D value len 61
norm2D value [1.56730291e-233 2.11383814e-223 1.34669628e-213 4.05272119e-
204
 5.76106710e-195 3.86846149e-186 1.22702316e-177 1.83842697e-169
 1.30112537e-161 4.34981756e-154 6.86913545e-147 5.12403699e-140
 1.80551695e-133 3.00517678e-127 2.36274915e-121 8.77494385e-116
 1.53939551e-110 1.27566084e-105 4.99343463e-101 9.23299613e-097
 8.06427031e-093 3.32710601e-089 6.48406469e-086 5.96907696e-083
 2.59565015e-080 5.33168241e-078 5.17322689e-076 2.37103475e-074
 5.13326224e-073 5.24962308e-072 2.53595727e-071 5.78675211e-071
 6.23744838e-071 3.17583689e-071 7.63815651e-072 8.67756858e-073
 4.65679044e-074 1.18046869e-075 1.41351816e-077 7.99516531e-080
 2.13615441e-082 2.69598192e-085 1.60723895e-088 4.52608867e-092
```

```
 6.02066977e-096 3.78308148e-100 1.12286028e-104 1.57429054e-109
 1.04261313e-114 3.26167459e-120 4.81989094e-126 3.36444110e-132
 1.10934796e-138 1.72783308e-145 1.27120303e-152 4.41781161e-160
 7.25234748e-168 5.62379077e-176 2.05996027e-184 3.56424600e-193
 2.91310164e-202]
norm2D value len 61
norm2D value [9.97657140e-214 2.00214352e-202 1.89796516e-191 8.49885472e-
181
 1.79767721e-170 1.79614816e-160 8.47718660e-151 1.88990714e-141
 1.99025318e-132 9.90045895e-124 2.32638434e-115 2.58218137e-107
 1.35385190e-099 3.35300896e-092 3.92263055e-085 2.16770030e-078
 5.65848551e-072 6.97718626e-066 4.06386773e-060 1.11809306e-054
 1.45309991e-049 8.92056039e-045 2.58683030e-040 3.54342202e-036
 2.29275135e-032 7.00761406e-029 1.01172511e-025 6.89976065e-023
 2.22271949e-020 3.38231795e-018 2.43121485e-016 8.25489187e-015
 1.32397132e-013 1.00305536e-012 3.58963562e-012 6.06813130e-012
 4.84550126e-012 1.82768602e-012 3.25644397e-013 2.74072062e-014
 1.08959490e-015 2.04618312e-017 1.81511051e-019 7.60572786e-022
 1.50541983e-024 1.40751691e-027 6.21625395e-031 1.29682990e-034
 1.27795752e-038 5.94879443e-043 1.30803885e-047 1.35859954e-052
 6.66563336e-058 1.54479381e-063 1.69113686e-069 8.74512717e-076
 2.13615441e-082 2.46478075e-089 1.34339301e-096 3.45865317e-104
 4.20620156e-112]
norm2D value len 61
```

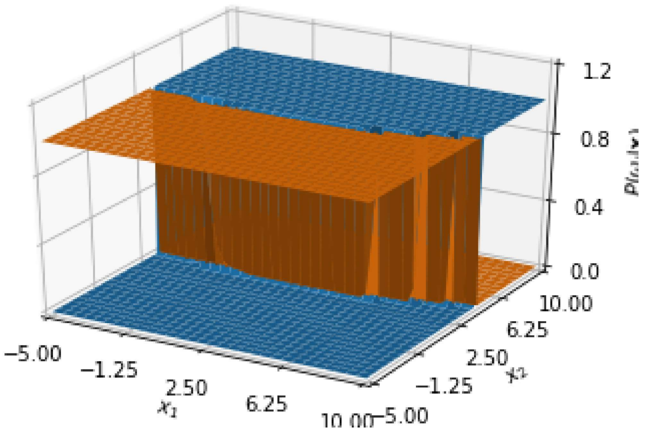



```
<Figure size 432x288 with 0 Axes>
```

In [22]:

```
print('Posterior')
discr='Pwx'
gnan=0
classplot(posterior(g), x1, x2, gnan, discr, 1)
```

Posterior



<Figure size 432x288 with 0 Axes>

g) Repeat subtaske c-d for the Parzen classifier with window size h1 = 5

In [23]:

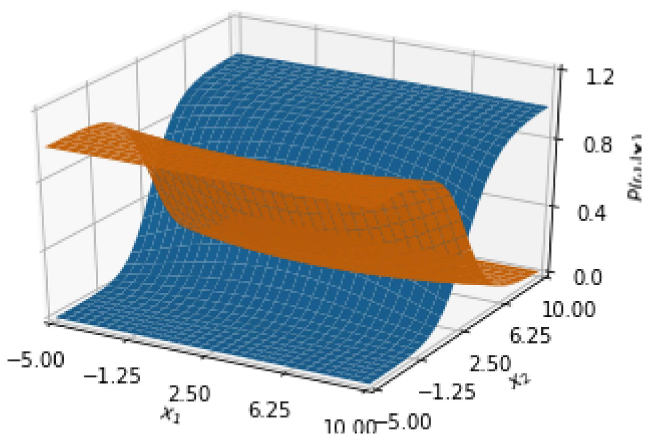
```
h1=5
print('Scaled')
gnan = 1
discr = 'Pwpxw'
g=ParW(h1)
classplot(g, x1, x2, gnan, discr, 1)
```

```
 7.96177550e-06 7.25471809e-06 6.54467687e-06 5.84538247e-06
 5.16885931e-06 4.52515589e-06 3.92219728e-06 3.36575406e-06
 2.85951499e-06 2.40524562e-06 2.00301196e-06 1.65144726e-06
 1.34804046e-06 1.08942717e-06 8.71666858e-07 6.90494016e-07
 5.41534787e-07 4.20484369e-07 3.23243952e-07 2.46018640e-07
 1.85379914e-07]
norm2D value len 61
norm2D value [4.47730908e-08 5.89451474e-08 7.68309310e-08 9.91473678e-08
 1.26672790e-07 1.60229523e-07 2.00659072e-07 2.48789537e-07
 3.05395393e-07 3.71150376e-07 4.46574977e-07 5.31980751e-07
 6.27414437e-07 7.32605463e-07 8.46920904e-07 9.69332128e-07
 1.09839720e-06 1.23226264e-06 1.36868719e-06 1.50508899e-06
 1.63861607e-06 1.76623829e-06 1.88485714e-06 1.99142812e-06
 2.08308929e-06 2.15728832e-06 2.21190034e-06 2.24532893e-06
 2.25658369e-06 2.24532893e-06 2.21190034e-06 2.15728832e-06
 2.08308929e-06 1.99142812e-06 1.88485714e-06 1.76623829e-06
 1.63861607e-06 1.50508899e-06 1.36868719e-06 1.23226264e-06
 1.09839720e-06 9.69332128e-07 8.46920904e-07 7.32605463e-07
 6.27414437e-07 5.31980751e-07 4.46574977e-07 3.71150376e-07
 3.05395393e-07 2.48789537e-07 2.00659072e-07 1.60229523e-07
```

In [24]:

```
print('Posterior')
discr='Pwx'
gnan=0
classplot(posterior(g), x1, x2, gnan, discr, 1)
```

```
Posterior
```




```
<Figure size 432x288 with 0 Axes>
```

h) Repeat the previous subtask the kN-nearest neighbour classifier with kn = 1.

In [25]:

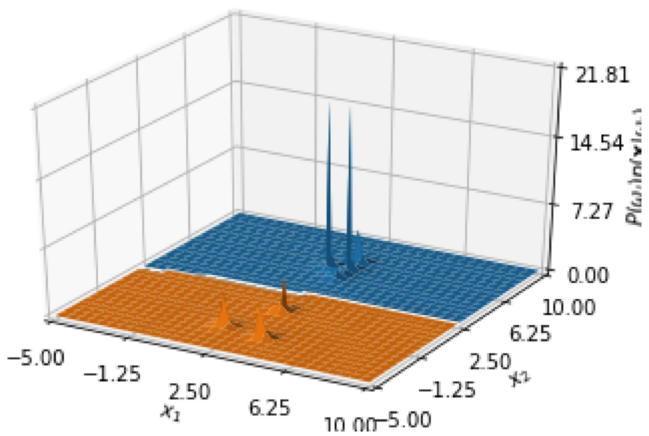
```
def NearNeib(kn,x1,x2):
    a_list=[]
    g=[]
    X1,X2 = np.meshgrid(x1,x2)
    [m,n]=np.shape(X1)
    f=0
    for k in range(0,M):
        p = np.zeros(np.shape(X1))
        for i in np.arange(0, m):
                for j in np.arange(0, n):
                    a=np.array([[X1[i,j]],[X2[i,j]]])
                    a_list.append(a)
                    dist=[]
                    for q in range(0,N[k]):
                        b=X[k][:,q]
                        d = np.linalg.norm(a-b)
                        dist.append(d)

                    dist_sort=sorted(dist)
                    p[i,j]=Pw[k]*kn/N[k]/(math.pi*dist_sort[kn-1]**2)
        g.append(p)
    return g
```

In [26]:

```
print('Scaled')
gnan = 1
discr = 'Pwpxw'
g=NearNeib(1,x1,x2)
classplot(g, x1, x2, gnan, discr, 1)
```

Scaled



<Figure size 432x288 with 0 Axes>

In [27]:

```
g
```

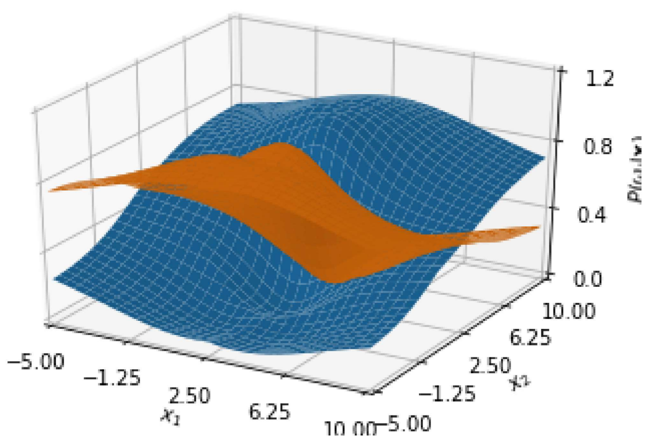
Out[27]:

```
[array([[0.00028964, 0.0002969 , 0.00030428, ..., 0.00032242, 0.0003148 ,
         0.00030727],
        [0.00029886, 0.00030659, 0.00031447, ..., 0.00033388, 0.00032571,
         0.00031766],
        [0.00030841, 0.00031666, 0.00032507, ..., 0.00034586, 0.0003371 ,
         0.00032848],
        ...,
        [0.00072482, 0.00076687, 0.00081228, ..., 0.00087974, 0.00082522,
         0.00077541],
        [0.00070336, 0.00074289, 0.00078542, ..., 0.00085713, 0.00080529,
         0.00075779],
        [0.00068185, 0.00071894, 0.0007587 , ..., 0.00083374, 0.00078461,
         0.00073944]]),
 array([[0.00089868, 0.00096106, 0.00102983, ..., 0.00123356, 0.00114332,
         0.00106227],
        [0.00091933, 0.00098471, 0.00105703, ..., 0.00126748, 0.0011724 ,
         0.00108733],
        [0.00093852, 0.00100676, 0.00108248, ..., 0.00129867, 0.00119903,
         0.0011102 ],
        ...,
        [0.0003057 , 0.00031389, 0.00032225, ..., 0.00035001, 0.00034118,
         0.00033249],
        [0.00029632, 0.00030401, 0.00031185, ..., 0.00033777, 0.00032954,
         0.00032143],
        [0.00028727, 0.0002945 , 0.00030185, ..., 0.00032606, 0.00031839,
         0.00031081]])]
```

In [28]:

```
print('Posterior')
discr='Pwx'
gnan=0
classplot(posterior(g), x1, x2, gnan, discr, 1)
```

Posterior



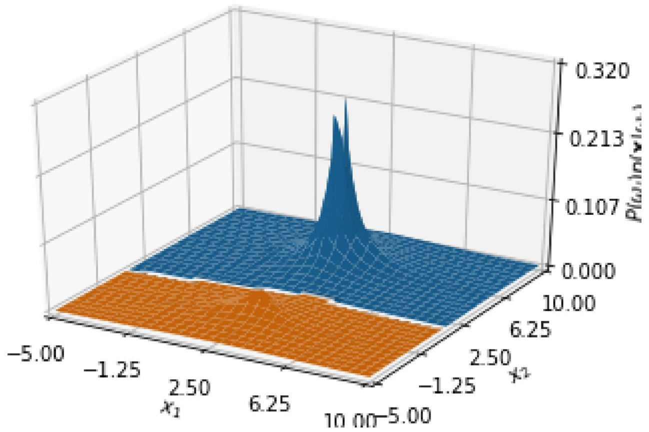

```
<Figure size 432x288 with 0 Axes>
```

i) Repeat the previous subtask the kN-nearest neighbour classifier with kn = 3.

In [29]:

```
print('Scaled')
gnan = 1
discr = 'Pwpxw'
g=NearNeib(3,x1,x2)
classplot(g, x1, x2, gnan, discr, 1)
```

```
Scaled
```

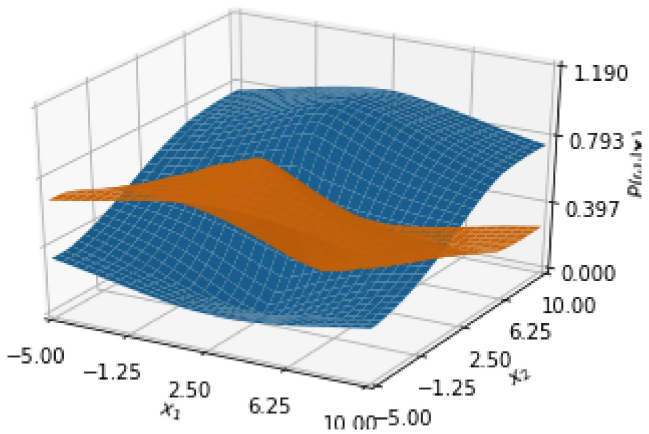



```
<Figure size 432x288 with 0 Axes>
```

In [30]:

```
print('Posterior')
discr='Pwx'
gnan=0
classplot(posterior(g), x1, x2, gnan, discr, 1)
```

```
Posterior
```




```
<Figure size 432x288 with 0 Axes>
```

j) Repeat the previous subtask the kN-nearest neighbour classifier with kn = 5. (Why will this not work?)

In [ ]:

```

```

Search for the fifth nearest neighbour, but there are just 4 neighbours for class 1, and 3 neibghbours for class 2.

k) Add functionality so that the figure display the a posteriori probability for the two classes: *See the corresponding parts above.*