Technische Universität Berlin
Institut für Softwaretechnik
und theoretische Informatik.

Wintersemester 2019/2020
Prof. Dr. Klaus-Robert Müller

Machine Learning I - Homework I

Jacky 391049, Viktor 392636, Duc 395220, Laura 391342, Laura 392032

---

1. The Bayes decision rule of the two classes classification problem result in the Bayes error (e)

$$\mathbb{P}(\mathsf{e}) \coloneqq \int_{\mathbb{R}} \mathbb{P}(\mathsf{e}\,|x)p(x)\,\mathrm{d}x,$$

where $\mathbb{P}(\mathsf{e}\,|x) \coloneqq \min\big(\mathbb{P}(\omega_1|x), \mathbb{P}(\omega_2|x)\big)$ is the probability of error for a particular input x. Interestingly, while class posteriors $\mathbb{P}(\omega_1|x)$ and $\mathbb{P}(\omega_2|x)$ can be often expressed analytically and are integrable, the error function has discontinuities that prevent its analytical integration, and therefore, direct computation of the Bayes error.

   (a) Show that the full error can be upper-bounded as follows:

$$\mathbb{P}(\mathsf{e}) \leq \int_{\mathbb{R}} \frac{2}{\frac{1}{\mathbb{P}(\omega_1|x)} + \frac{1}{\mathbb{P}(\omega_2|x)}} p(x)\,\mathrm{d}x.$$

   Note that the integrand is now continuous and corresponds to the harmonic mean of the class posteriors weighted by $p(x)$.

   Proof. Since $p(x) > 0$ for all $x \in \mathbb{R}$ if suffices to show

$$\min(p_1, p_2) \leq \frac{2}{\frac{1}{p_1} + \frac{1}{p_2}},$$

   where $p_i \coloneqq P(\omega_i|x)$ for $i \in \{1, 2\}$, which is equivalent to

$$\frac{p_1 + p_2 - |p_1 - p_2|}{2} \leq \frac{2p_1p_2}{p_1 + p_2}.$$

   Method 1. Using that $p_1 + p_2 = 1$ (theorem of total probability) we can eliminate one variable, such that the inequality becomes

$$1 - |1 - 2p_2| \leq 4(1 - p_2)p_2$$

   This s true for all $p_2 \in [0, 1]$ since the RHS is a downward parabola with vertex (= max) at $\left(\frac{1}{2}, 1\right)$ and roots $(0, 0)$ and $(0, 1)$, whereas the LHS forms a triangle with endpoints $(0, 0)$, $\left(\frac{1}{2}, 1\right)$, $(0, 1)$. By symmetry reasons it therefore suffices to show the inequality for $p_2 \in \left(0, \frac{1}{2}\right)$, which is done by checking it for one point in the interval. We notice that $\frac{1}{2} = 1 - \left|1 - 2 \cdot \frac{1}{4}\right| \leq 4\left(1 - \frac{1}{4}\right)\frac{1}{4} = \frac{3}{4}$ holds.

Method 2. Cross-multiplication yields

$$\begin{aligned}
&(p_1+p_2)^2-(p_1+p_2)|p_1-p_2| \leq 4p_1p_2 \\
\iff &(p_1-p_2)^2-(p_1+p_2)|p_1-p_2| \leq 0 \\
\iff &(p_1-p_2)^2 \leq (p_1+p_2)|p_1-p_2| && (|p_1-p_2| \geq 0) \\
\iff &|p_1-p_2| \leq p_1+p_2 \\
\iff &p_1, p_2 \geq 0.
\end{aligned}$$

Since $p_{1,2} \in [0,1]$, the claims follows. □

(b) Show using the this result that for the univariate probability distributions

$$\bar{p}_{1,2} \coloneqq p(x|\omega_{1,2}) \coloneqq (\pi(1+(x \mp \mu)^2))^{-1}$$

the Bayes error can be upper-bounded by

$$\mathbb{P}(\mathsf{e}) \leq \frac{2p_1p_2}{\sqrt{1+4\mu^2p_1p_2}},$$

where $p_i \coloneqq \mathbb{P}(\omega_i)$ for $i \in \{1,2\}$. *Hint:* $\int_{\mathbb{R}}(ax^2+bx+c)^{-1}\,\mathrm{d}x = \frac{2\pi}{\sqrt{4ac-b^2}}$ *for* $b^2 < 4ac$.

Proof. By Bayes formula (B)

$$\begin{aligned}
\frac{p(x)}{\frac{1}{\mathbb{P}(\omega_1|x)}+\frac{1}{\mathbb{P}(\omega_2|x)}} &\overset{\text{(B)}}{=} \frac{\cancel{p(x)}}{\frac{\cancel{p(x)}}{p_1\bar{p}_1}+\frac{\cancel{p(x)}}{p_2\bar{p}_2}} = \frac{1}{\frac{1}{p_1\bar{p}_1}+\frac{1}{p_2\bar{p}_2}} \\
&= \left(\frac{\pi(1+(x-\mu)^2)}{p_1}+\frac{\pi(1+(x-\mu)^2)}{p_2}\right)^{-1} \\
&= \frac{p_1p_2}{\pi}\left(p_2(1+(x-\mu)^2)+p_1(1+(x-\mu)^2)\right)^{-1} \\
&= \frac{p_1p_2}{\pi}\left((p_1+p_2)x^2+2\mu(p_1-p_2)x+(p_1+p_2)(1+\mu^2)\right)^{-1} \\
&= \frac{p_1p_2}{\pi}\left(x^2+2\mu(p_1-p_2)x+1+\mu^2\right)^{-1} && (p_1+p_2=1)
\end{aligned}$$

holds. With the hint (h) we have

$$\begin{aligned}
\mathbb{P}(\mathsf{e}) &\overset{\text{(a)}}{\leq} \int_{\mathbb{R}} \frac{2}{\frac{1}{\mathbb{P}(\omega_1|x)}+\frac{1}{\mathbb{P}(\omega_1|x)}} \\
&= \frac{2p_1p_2}{\pi}\int_{\mathbb{R}}\left(x^2+2\mu(p_1-p_2)x+1+\mu^2\right)^{-1}\mathrm{d}x \\
&\overset{\text{(h)}}{=} \frac{2p_1p_2}{\cancel{\pi}}\frac{\cancel{2\pi}}{\sqrt{4(1+\mu^2)-(2\mu)^2(p_1-p_2)^2}} \\
&= \frac{4p_1p_2}{\sqrt{4(1+4\mu^2p_1p_2)}} = \frac{2p_1p_2}{\sqrt{1+4\mu^2p_1p_2}}. && (p_1+p_2=1 \implies 2p_1p_2 = 1-p_1^2-p_2^2)
\end{aligned}$$

We can use the hint since $b^2 < 4ac$ means $4\mu^2p_1p_2+1 \geq 0$, which is always true due to $\mu^2, p_{1,2} \geq 0$. □

(c) Explain how you would estimate the error if there was no upper bounds that are both tight and analytically integrable, distinguishing into two cases: the data being (1) low-dimensional and (2) high-dimensional.

2. One might speculate that, in some cases, the generated data $p(x|\omega_1)$ and $p(x|\omega_2)$ is of no use to improve the accuracy of a classifier, in which case one should only rely on prior class probabilities $\mathbb{P}(\omega_1)$ and $\mathbb{P}(\omega_2)$.

   For the first part of this exercise, we assume that the data for each class is generated by the univariate LAPLACIAN probability distributions:

$$p(x|\omega_{1,2}) \coloneqq \frac{1}{2\sigma} \exp\left(-\frac{|x \mp \mu|}{\sigma}\right), \quad \sigma, \mu > 0.$$

   (a) Determine for which values of $\mathbb{P}(\omega_1)$, $\mathbb{P}(\omega_2)$, μ, σ the optimal decision is to always predict the first class (i.e. under which conditions $\mathbb{P}(\mathsf{e}\,|x) = \mathbb{P}(\omega_2|x)$ for all $x \in \mathbb{R}$.

   Since $\mathbb{P}(\mathsf{e}\,|x) = \min\big(\mathbb{P}(\omega_1|x), \mathbb{P}(\omega_2|x)\big)$, we have to determine for which values of $\mathbb{P}(\omega_1)$, $\mathbb{P}(\omega_2)$, μ, σ we have

$$\mathbb{P}(\omega_2|x) < \mathbb{P}(\omega_1|x) \quad \forall x \in \mathbb{R}\,.$$

   Since $p(x) > 0$ for all $x \in \mathbb{R}$ this is equivalent to

$$p(x|\omega_2)\mathbb{P}(\omega_2) < p(x|\omega_1)\mathbb{P}(\omega_1) \quad \forall x \in \mathbb{R}$$

   by BAYES formula. By plugging in the definition of $p(x|\omega_i)$ and cross-multiplying we obtain for all $x \in \mathbb{R}$

$$\begin{aligned} &\exp\left(\frac{|x-\mu| - |x+\mu|}{\sigma}\right) < \frac{\mathbb{P}(\omega_1)}{\mathbb{P}(\omega_2)} \\ \iff &|x-\mu| - |x+\mu| < \sigma \log\left(\frac{\mathbb{P}(\omega_1)}{\mathbb{P}(\omega_2)}\right). \end{aligned} \tag{1}$$

   We can't have $\mathbb{P}(\omega_1) = \mathbb{P}(\omega_2)$ because then the RHS will be zero and the LHS is only smaller than zero for $x \geq 0$ since $\mu > 0$. Therefore we will only consider $\mathbb{P}(\omega_1) > \mathbb{P}(\omega_2)$, implying $\mathbb{P}(\omega_2) < \frac{1}{2}$ since $\mathbb{P}(\omega_1) + \mathbb{P}(\omega_2) = 1$.

   Rearranging (1) gives

$$\begin{aligned} &|x-\mu| - |x+\mu| < \sigma \log\left(\frac{1-\mathbb{P}(\omega_2)}{\mathbb{P}(\omega_2)}\right) \\ \iff &1 + \exp\left(\frac{|x-\mu| - |x+\mu|}{\sigma}\right) < \frac{1}{\mathbb{P}(\omega_2)} \\ \iff &\mathbb{P}(\omega_2) < \left(1 + \exp\left(\frac{|x-\mu| - |x+\mu|}{\sigma}\right)\right)^{-1} \xrightarrow{x \to -\infty} \left(1 + \exp\left(\frac{\mu}{\sigma}\right)\right)^{-1}. \end{aligned}$$

   As the sigmoid function $x \mapsto \frac{1}{1+e^{-x}}$ is monotonically increasing (chose $x = |t-\mu| - |t+\mu|$), and the inequality has to be true for all $x \in \mathbb{R}$, $\lim_{x \to -\infty}$ of the RHS must be an upper bound.

   (b) Repeat the exercise for the case where the data for each class is generated by the univariate GAUSSIAN probability distributions

$$p(x|\omega_{1,2}) \coloneqq \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x \mp \mu)^2}{2\sigma^2}\right), \quad \sigma > 0.$$

The same procedure as above yields for $\mu > 0$

$$\begin{aligned}
&\frac{(x-\mu)^2 - (x+\mu)^2}{2\sigma^2} < \log\left(\frac{\mathbb{P}(\omega_1)}{\mathbb{P}(\omega_2)}\right) \\
\iff &-\frac{2\mu x}{\sigma^2} < \log\left(\frac{\mathbb{P}(\omega_1)}{\mathbb{P}(\omega_2)}\right) \\
\iff &\log\left(\frac{\mathbb{P}(\omega_2)}{1-\mathbb{P}(\omega_2)}\right) < \frac{2\mu x}{\sigma^2} \\
\iff &\mathbb{P}(\omega_2) < \left(1 + e^{-\frac{2\mu x}{\sigma}}\right)^{-1} \xrightarrow{x \to -\infty} 0.
\end{aligned}$$

Since $\Phi : \mathbb{R} \to [0,1],\ x \mapsto (1+e^{-ax})^{-1}$ is monotonically increasing for all $a > 0$ we arrive at a contradiction: $\mathbb{P}(\omega_2) < 0$.

If $\mu = 0$ we have

$$-\frac{2 \cdot \mu \cdot 0}{\sigma^2} = 0 < \log\left((\frac{\mathbb{P}(\omega_1)}{\mathbb{P}(\omega_2)}\right),$$

implying $\mathbb{P}(\omega_1) > \mathbb{P}(\omega_2)$.

Programming Sheet 1: Bayes Decision Theory (40 P)

In this exercise sheet, we will apply Bayes decision theory in the context of small two-dimensional problems. For this, we will make use of 3D plotting. We introduce below the basics for constructing these plots in Python/Matplotlib.

The function `numpy.meshgrid`

To plot two-dimensional functions, we first need to discretize the two-dimensional input space. One basic function for this purpose is `numpy.meshgrid` . The following code creates a discrete grid of the rectangular surface $[0, 4] \times [0, 3]$. The function `numpy.meshgrid` takes the discretized intervals as input, and returns two arrays of size corresponding to the discretized surface (i.e. the grid) and containing the X and Y-coordinates respectively.

In [72]:
```
import numpy as np
X,Y = np.meshgrid([0,1,2,3,4],[0,1,2,3])
print(X)
print(Y)
```

```
[[0 1 2 3 4]
 [0 1 2 3 4]
 [0 1 2 3 4]
 [0 1 2 3 4]]
[[0 0 0 0 0]
 [1 1 1 1 1]
 [2 2 2 2 2]
 [3 3 3 3 3]]
```

Note that we can iterate over the elements of the grid by zipping the two arrays `X` and `Y` containing each coordinate. The function `numpy.flatten` converts the 2D arrays to one-dimensional arrays, that can then be iterated element-wise.

In [73]:
```
print(list(zip(X.flatten(),Y.flatten())))
```

```
[(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (0, 1), (1, 1), (2, 1), (3,
1), (4, 1), (0, 2), (1, 2), (2, 2), (3, 2), (4, 2), (0, 3), (1, 3),
(2, 3), (3, 3), (4, 3)]
```

3D-Plotting

To enable 3D-plotting, we first need to load some modules in addition to `matplotlib` :

In [74]:
```
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

As an example, we would like to plot the L2-norm function $f(x, y) = \sqrt{x^2 + y^2}$ on the subspace $x, y \in [-4, 4]$. First, we create a meshgrid with appropriate size:

In [75]:
```
R = np.arange(-4,4+1e-9,0.1)
X,Y = np.meshgrid(R,R)
print(X.shape,Y.shape)
```

```
(81, 81) (81, 81)
```

Here, we have used a discretization with small increments of 0.1 in order to produce a plot with better resolution. The resulting meshgrid has size (81x81), that is, approximately 6400 points. The function f needs to be evaluated at each of these points. This is achieved by applying element-wise operations on the arrays of the meshgrid. The norm at each point of the grid is therefore computed as:

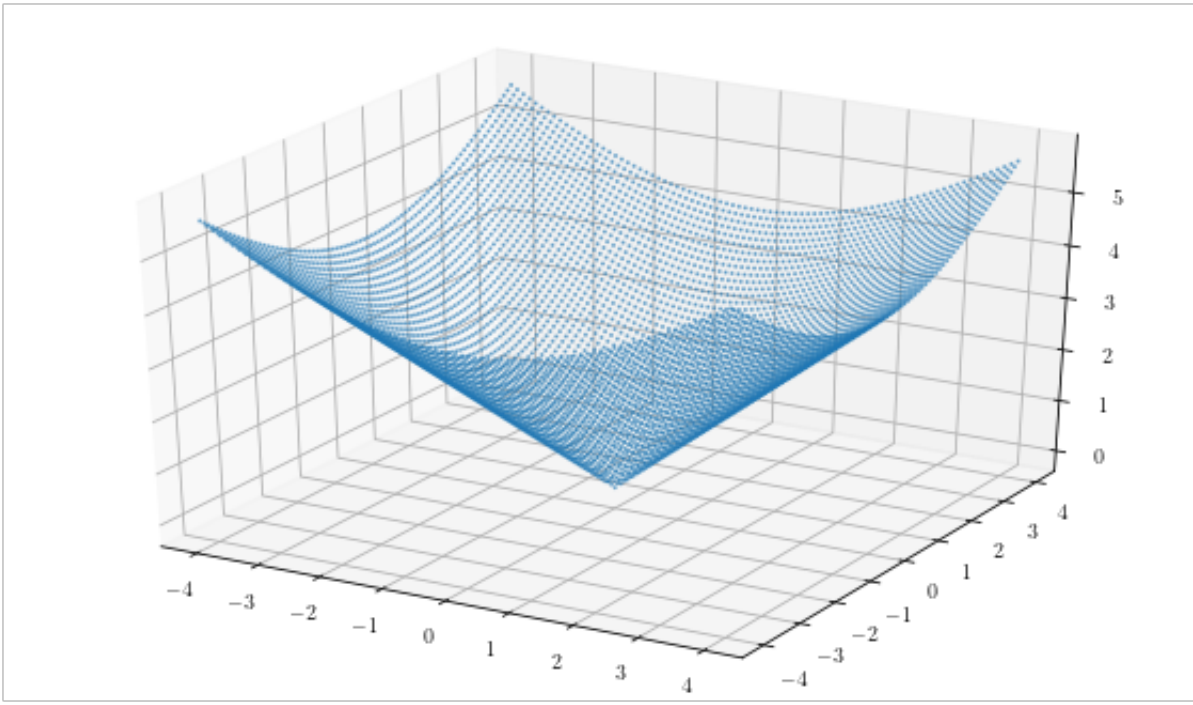
In [76]:
```
F = (X**2+Y**2)**.5
print(F.shape)
```

```
(81, 81)
```

The resulting function values are of same size as the meshgrid. Taking `X` , `Y` , `F` jointly results in a list of approximately 6400 triplets representing the x-, y-, and z-coordinates in the three-dimensional space where the function should be plotted. The 3d-plot can now be constructed easily by means of the function `scatter` of `matplotlib.pyplot` .

In [77]:
```
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,F,s=1,alpha=0.5)
```

Out[77]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x119c85210>
```



The parameter `s` and `alpha` control the size and the transparency of each data point. Other 3d plotting variants exist (e.g. surface plots), however, the scatter plot is the simplest approach at least conceptually. Having introduced how to easily plot 3D functions in Python, we can now analyze two-dimensional probability distributions with this same tool.

Exercise 1: Gaussian distributions (5+5+5 P)

Using the technique introduced above, we would like to plot a normal Gaussian probability distribution with mean vector $\mu = (0, 0)$, and covariance matrix $\Sigma = I$ also known as standard normal distribution. We consider the same discretization as above (i.e. a grid from -4 to 4 using step size 0.1). For two-dimensional input spaces, the standard normal distribution is given by:

$$p(x, y) = \frac{1}{2\pi} e^{-0.5(x^2+y^2)}.$$

This distribution sums to 1 when integrated over $\mathbb{R}^2$. However, it does not sum to 1 when summing over the discretized space (i.e. the grid). Instead, we can work with a discretized Gaussian-like distribution:

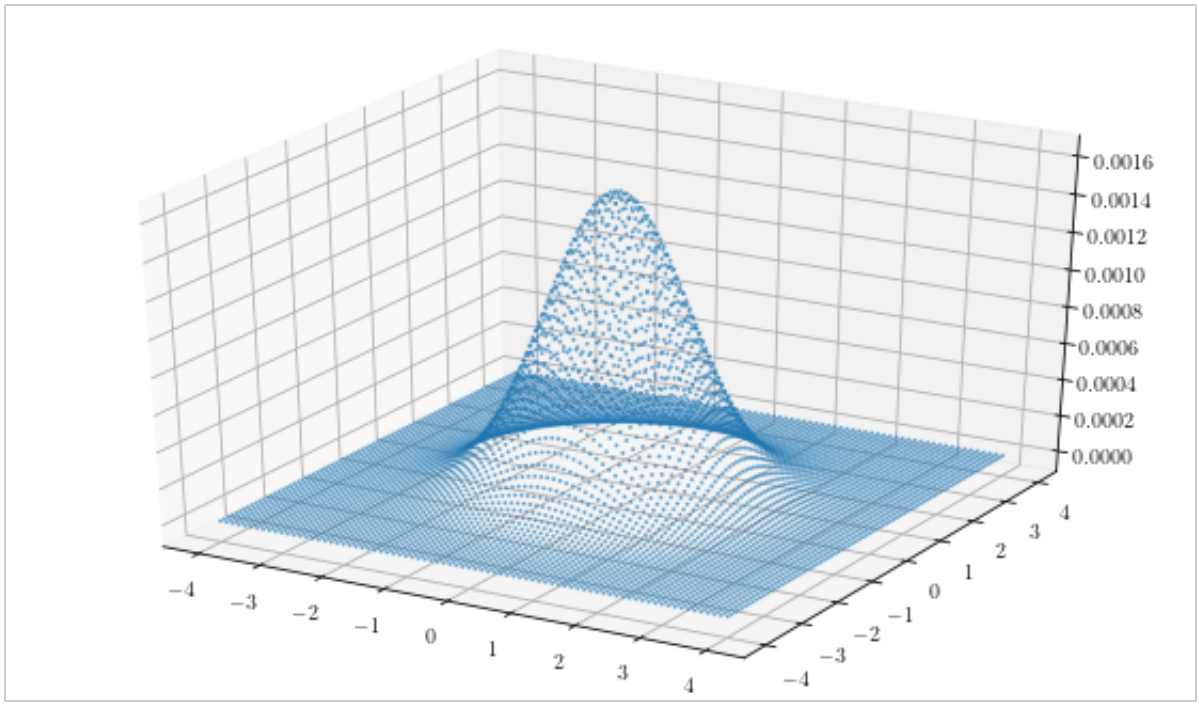
$$P(x, y) = \frac{1}{Z} e^{-0.5(x^2+y^2)} \qquad \text{with} \quad Z = \sum_{x,y} e^{-0.5(x^2+y^2)}$$

where the sum runs over the whole discretized space.

- **Compute the distribution $P(x, y)$, and plot it.**
- **Compute the conditional distribution $Q(x, y) = P((x, y)|\sqrt{x^2 + y^2} \geq 1)$, and plot it.**
- **Marginalize the conditioned distribution $Q(x, y)$ over y, and plot the resulting distribution $Q(x)$.**

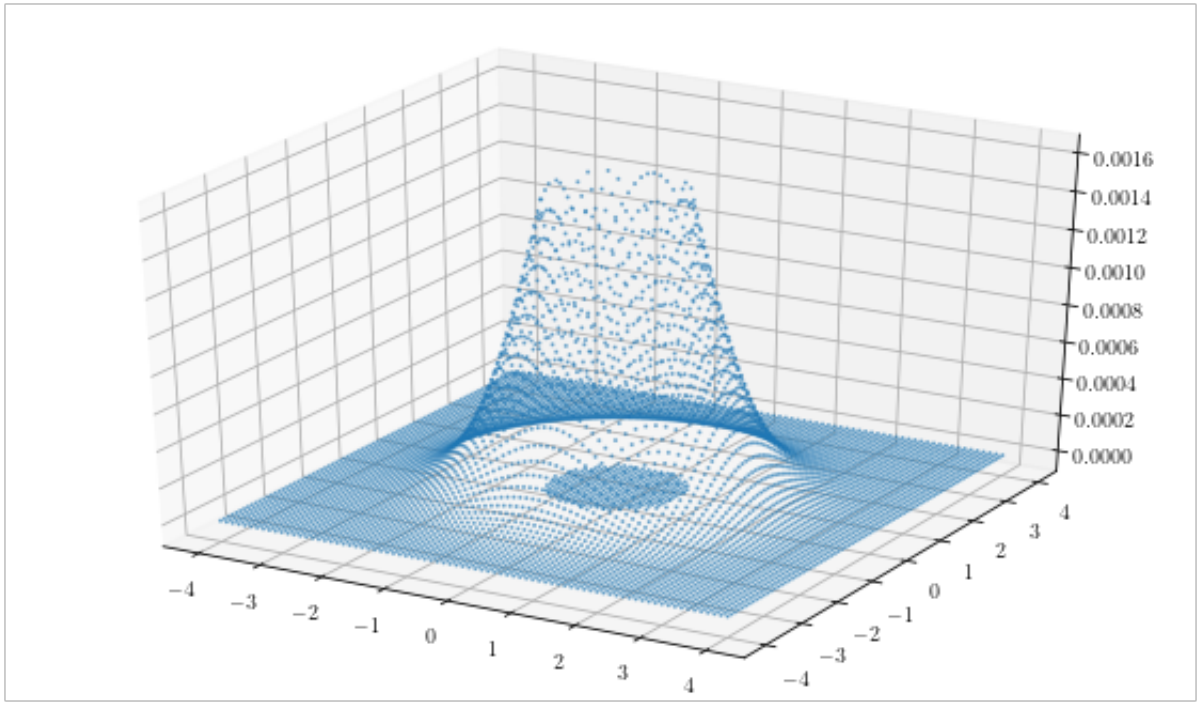
In [78]:
```
### REPLACE BY YOUR CODE
import numpy as np
import math
Z = np.sum(np.exp(-0.5*(X**2+Y**2)))
F = 1/Z * np.exp(-0.5*(X**2+Y**2))
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,F,s=1,alpha=0.5)
###
```

Out[78]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x11a058ed0>
```



In [79]:
```
### REPLACE BY YOUR CODE
W = np.sum(np.exp(-0.5*(X**2+Y**2))*(np.sqrt(X**2+ Y**2)>=1))
F = 1/W * np.exp(-0.5*(X**2+Y**2))*(np.sqrt(X**2+ Y**2)>=1)
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,F,s=1,alpha=0.5)
###
```

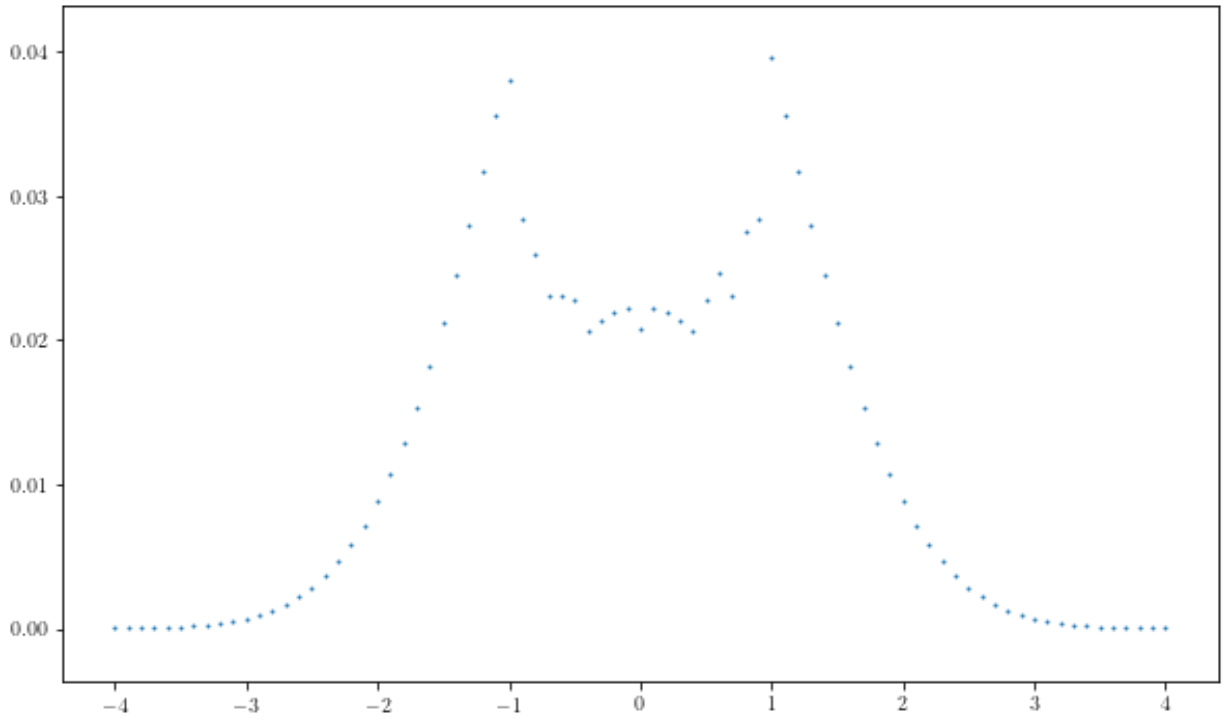
Out[79]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x11a330390>



In [80]:
```
### REPLACE BY YOUR CODE
W = np.sum([np.exp(-0.5*(X**2+Y**2))*(np.sqrt(X**2+ Y**2)>=1)])
F = np.sum(1/W * np.exp(-0.5*(X**2+Y**2))*(np.sqrt(X**2+ Y**2)>=1), ax
is = 0)
X1 = X[0]
fig = plt.figure(figsize=(10,6))
ax = plt.axes()
ax.scatter(X1,F,s=1,alpha=1)
###

### REPLACE BY YOUR CODE
#X1 = X[0]
#fig = plt.figure(figsize=(10,6))
#ax = plt.axes()
#ax.scatter(X1,np.sum(F,axis=1),s=1,alpha=1)
###
```

Out[80]: <matplotlib.collections.PathCollection at 0x11a5a4450>



Exercise 2: Bayesian Classification (5+5+5 P)

Let the two coordinates x and y be now representated as a two-dimensional vector $\boldsymbol{x}$. We consider two classes ω_1 and ω_2 with data-generating Gaussian distributions $p(\boldsymbol{x}|\omega_1)$ and $p(\boldsymbol{x}|\omega_2)$ of mean vectors

$$\boldsymbol{\mu}_1 = (-0.5, -0.5) \quad \text{and} \quad \boldsymbol{\mu}_2 = (0.5, 0.5)$$

respectively, and same covariance matrix

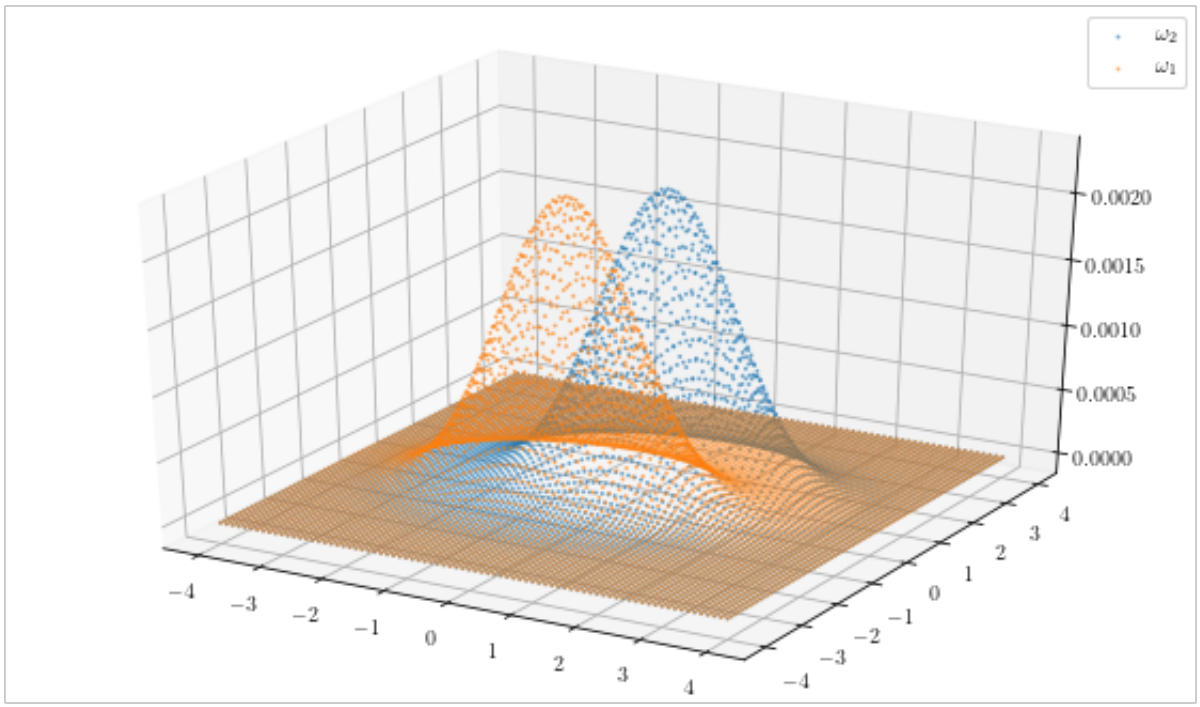
$$\Sigma = \begin{pmatrix} 1.0 & 0 \\ 0 & 0.5 \end{pmatrix}.$$

Classes occur with probability $P(\omega_1) = 0.9$ and $P(\omega_2) = 0.1$. Analysis tells us that in such scenario, the optimal decision boundary between the two classes should be linear. We would like to verify this computationally by applying Bayes decision theory on grid-like discretized distributions.

- **Using the same grid as in Exercise 1, discretize the two data-generating distributions $p(\boldsymbol{x}|\omega_1)$ and $p(\boldsymbol{x}|\omega_2)$ (i.e. create discrete distributions $P(\boldsymbol{x}|\omega_1)$ and $P(\boldsymbol{x}|\omega_2)$ on the grid), and plot them with different colors.**
- **From these distributions, compute the total probability distribution $P(\boldsymbol{x}) = \sum_{c\in\{1,2\}} P(\boldsymbol{x}|\omega_c) \cdot P(\omega_c)$, and plot it.**
- **Compute and plot the class posterior probabilities $P(\omega_1|\boldsymbol{x})$ and $P(\omega_2|\boldsymbol{x})$, and print the Bayes error rate for the discretized case.**

In [81]:
```
### REPLACE BY YOUR CODE
Z1 = np.sum(np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2)))
P1 = 1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2))
Z2 = np.sum(np.exp(-0.5*((X - 0.5)**2+ 2*(Y-0.5)**2)))
P2 = 1/Z2 * np.exp(-0.5*((X - 0.5)**2+ 2*(Y-0.5)**2))
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,P2,s=1,alpha=0.5,label="$\omega_2$")
ax.scatter(X,Y,P1,s=1,alpha=0.5,label="$\omega_1$")
ax.legend(loc=0)
###
```

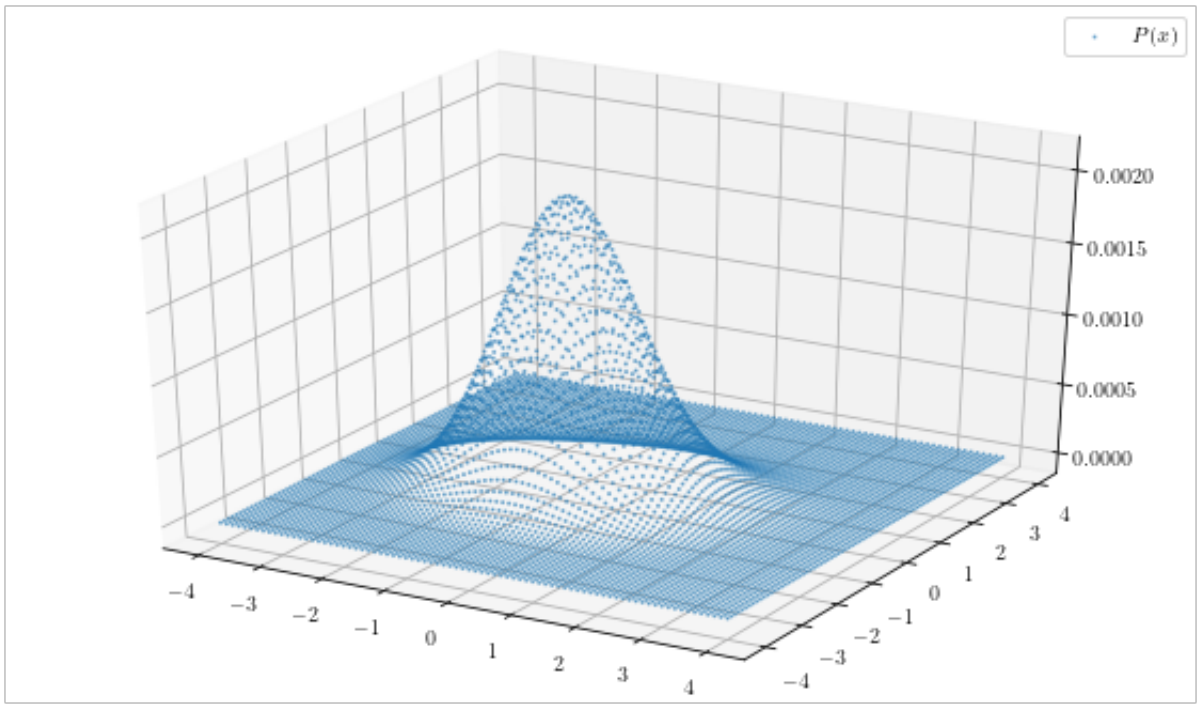
Out[81]: <matplotlib.legend.Legend at 0x11a5b8e10>



In [83]:
```
### REPLACE BY YOUR CODE
P =  1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2)) * 0.9 + 1/Z2 * n
p.exp(-0.5*((X - 0.5)**2+ 2*(Y-0.5)**2)) * 0.1
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,P,s=1,alpha=0.5,label="$P(x)$")
ax.legend(loc=0)
###

"""
PX = P1 * 0.9 + P2 * 0.1
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,PX,s=1,alpha=0.5,label="$P(x)$")
ax.legend(loc=0)
"""
```

Out[83]:
```
'\nPX = P1 * 0.9 + P2 * 0.1\nfig = plt.figure(figsize=(10,6))\nax =
plt.axes(projection=\'3d\')\nax.scatter(X,Y,PX,s=1,alpha=0.5,label="
$P(x)$")\nax.legend(loc=0)\n'
```



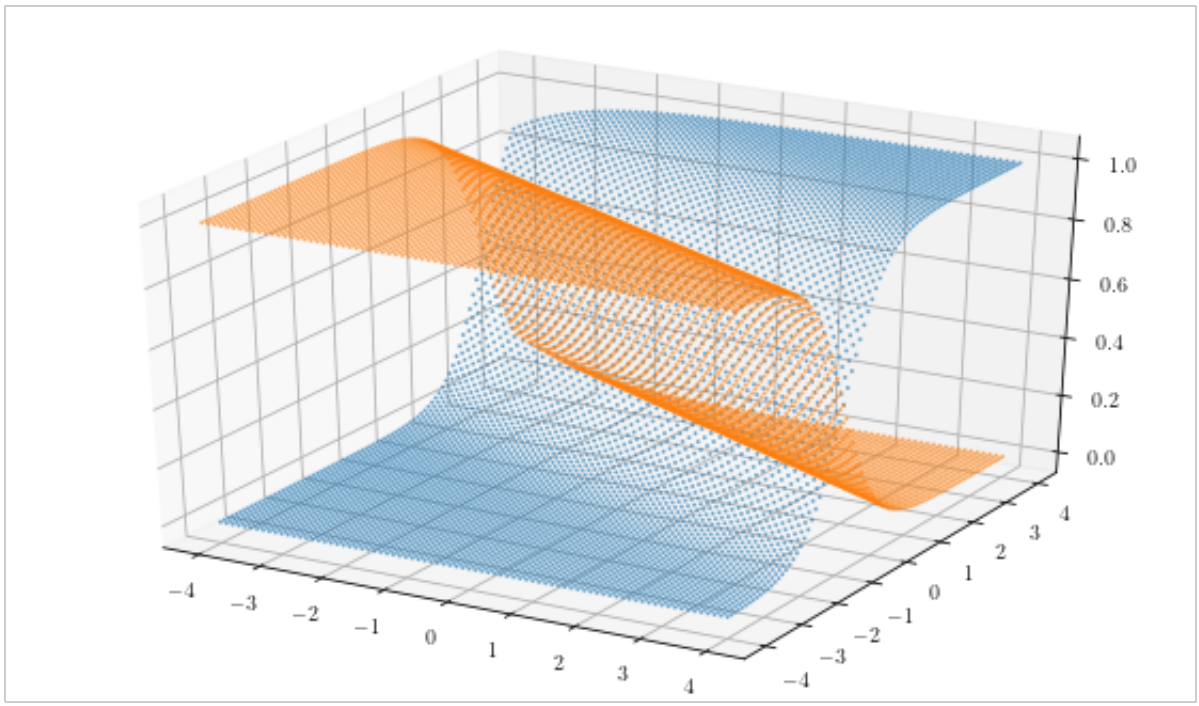
In [86]:
```
### REPLACE BY YOUR CODE
PW1= (1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2)) * 0.9)/P
PW2= (1/Z2 * np.exp(-0.5*((X - 0.5)**2+ 2*(Y-0.5)**2))* 0.1)/P
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,PW2,s=1,alpha=0.5)
ax.scatter(X,Y,PW1,s=1,alpha=0.5)
E = np.sum(np.minimum(PW1,PW2)*P)
print("Bayes error rate = {}".format(E))
###

"""
PW1 = P1 * 0.9 * (1 / PX)
PW2 = P2 * 0.1 * (1 / PX)
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,PW2,s=1,alpha=0.5,label="$P(w_2|x)$")
ax.scatter(X,Y,PW1,s=1,alpha=0.5,label="$P(w_1|x)$")
ax.legend(loc=0)

R = np.sum(np.fmin(PW1, PW2) * PX)
print("Bayes Error Rate = {}".format(R))
"""
```

```
Bayes error rate = 0.08042117524744927
```

Out[86]:
```
'\nPW1 = P1 * 0.9 * (1 / PX)\nPW2 = P2 * 0.1 * (1 / PX)\nfig = plt.f
igure(figsize=(10,6))\nax = plt.axes(projection=\'3d\')\nax.scatter(
X,Y,PW2,s=1,alpha=0.5,label="$P(w_2|x)$")\nax.scatter(X,Y,PW1,s=1,al
pha=0.5,label="$P(w_1|x)$")\nax.legend(loc=0)\n\nR = np.sum(np.fmin(
PW1, PW2) * PX)\nprint("Bayes Error Rate = {}".format(R))\n'
```



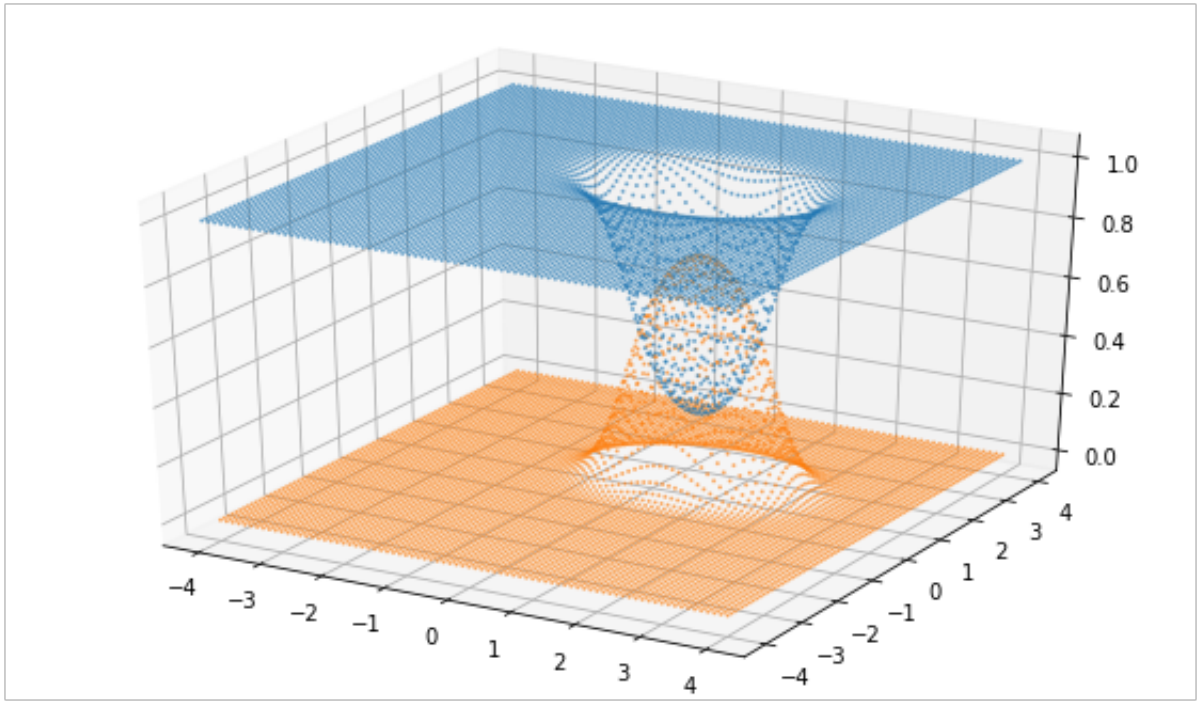
Exercise 3: Reducing the Variance (5+5 P)

Suppose that the data generating distribution for the second class changes to produce samples much closer to the mean. This variance reduction for the second class is implemented by keeping the first covariance the same (i.e. $\Sigma_1 = \Sigma$) and dividing the second covariance matrix by 4 (i.e. $\Sigma_2 = \Sigma/4$). For this new set of parameters, we can perform the same analysis as in Exercise 2.

- **Plot the new class posterior probabilities $P(\omega_1|\boldsymbol{x})$ and $P(\omega_2|\boldsymbol{x})$ associated to the new covariance matrices, and print the new Bayes error rate.**

In [15]:
```
### REPLACE BY YOUR CODE
Z1 = np.sum(np.exp(-0.5*((X +0.5)**2+ 2*(Y+0.5)**2)))
P1 = 1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2))
Z2 = np.sum(np.exp(-0.5*(4*(X -0.5)**2+ 8*(Y-0.5)**2)))
P2 = 1/Z2 * np.exp(-0.5*(4*(X - 0.5)**2+ 8*(Y-0.5)**2))
PN=  1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2)) * 0.9 + 1/Z2 * n
p.exp(-0.5*(4*(X - 0.5)**2+ 8*(Y-0.5)**2)) * 0.1
PW1= (1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2)) * 0.9)/PN
PW2= (1/Z2 * np.exp(-0.5*(4*(X - 0.5)**2+ 8*(Y-0.5)**2))* 0.1)/PN
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,PW1,s=1,alpha=0.5)
ax.scatter(X,Y,PW2,s=1,alpha=0.5)
E = np.sum(np.minimum(PW1,PW2)*PN)
print(E)
###
```

```
0.07290780555695717
```



Intuition tells us that by variance reduction and resulting concentration of generated data for class 2 in a smaller region of the input space, it should be easier to predict class 2 with certainty at this location. Paradoxally, in this new "dense" setting, we observe that class 2 does not reach full certainty anywhere in the input space, whereas it did in the previous exercise.

- **Explain this paradox.**

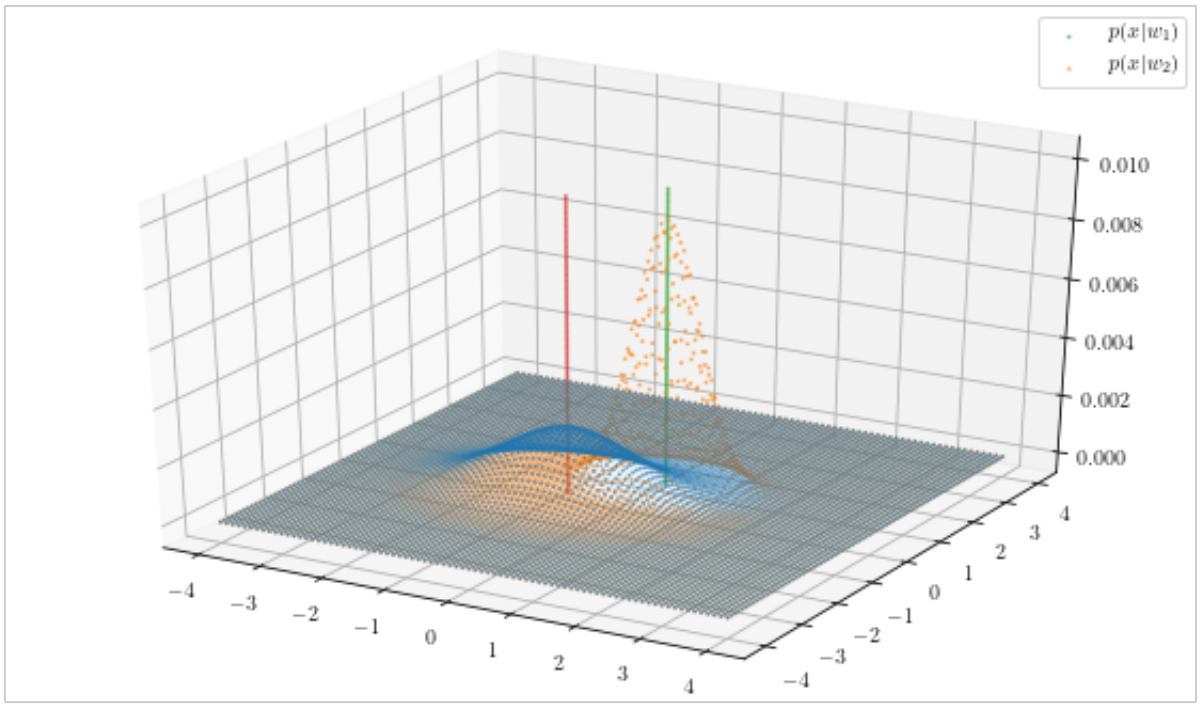
This is not a paradox because of the following reason. The decision boundary depends on how close $\mathbf{x} = (x, y)$ is to the mean μ with respect to the variance Σ. If one decreases the variance for class two, points that are near to μ may not be regarded as close to μ anymore! In other words, the point is actually far away from μ compared to the variance Σ. Thus, considering that class one also has a much higher probability, it makes much sense that there is no instance of $\mathbf{x}$ where the posterior probability is computed to one.

Playground

This is not part of any homework exercise

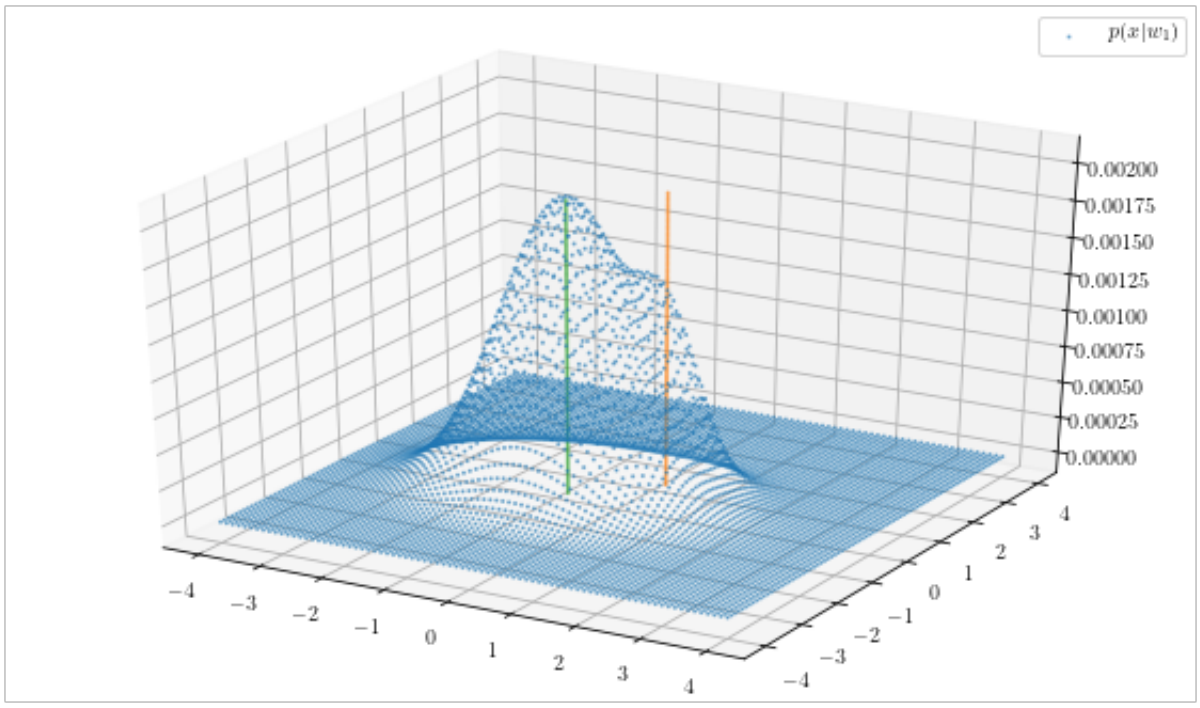
In [69]:
```
### REPLACE BY YOUR CODE
Z1 = np.sum(np.exp(-0.5*((X +0.5)**2+ 2*(Y+0.5)**2)))
P1 = 1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2))
Z2 = np.sum(np.exp(-0.5*(4*(X -0.5)**2+ 8*(Y-0.5)**2)))
P2 = 1/Z2 * np.exp(-0.5*(4*(X - 0.5)**2+ 8*(Y-0.5)**2))
PN=  1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2)) * 0.9 + 1/Z2 * n
p.exp(-0.5*(4*(X - 0.5)**2+ 8*(Y-0.5)**2)) * 0.1
PW1= (1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2)) * 0.9)/PN
PW2= (1/Z2 * np.exp(-0.5*(4*(X - 0.5)**2+ 8*(Y-0.5)**2))* 0.1)/PN
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,P1,s=1,alpha=0.5,label="$p(x|w_1)$")
ax.scatter(X,Y,P2,s=1,alpha=0.5,label="$p(x|w_2)$")
ax.scatter([0.5] * 100,[0.5] * 100, np.linspace(0,0.01,100),s=1,alpha=
0.5)
ax.scatter([-0.5] * 100,[-0.5] * 100, np.linspace(0,0.01,100),s=1,alph
a=0.5)
ax.legend(loc=0)
###
```

Out[69]: <matplotlib.legend.Legend at 0x1192f1090>



In [71]:
```
### REPLACE BY YOUR CODE
Z1 = np.sum(np.exp(-0.5*((X +0.5)**2+ 2*(Y+0.5)**2)))
P1 = 1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2))
Z2 = np.sum(np.exp(-0.5*(4*(X -0.5)**2+ 8*(Y-0.5)**2)))
P2 = 1/Z2 * np.exp(-0.5*(4*(X - 0.5)**2+ 8*(Y-0.5)**2))
PN=  0.9 * P1 + P2* 0.1
PW1= (1/Z1 * np.exp(-0.5*((X + 0.5)**2+ 2*(Y+0.5)**2)) * 0.9)/PN
PW2= (1/Z2 * np.exp(-0.5*(4*(X - 0.5)**2+ 8*(Y-0.5)**2))* 0.1)/PN
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,PN,s=1,alpha=0.5,label="$p(x|w_1)$")
ax.scatter([0.5] * 100,[0.5] * 100, np.linspace(0,0.002,100),s=1,alpha
=0.5)
ax.scatter([-0.5] * 100,[-0.5] * 100, np.linspace(0,0.002,100),s=1,alp
ha=0.5)
ax.legend(loc=0)
###
```

Out[71]: <matplotlib.legend.Legend at 0x1196e3f50>



In [ ]: