Machine Learning 1 exercises 1

Jiayun's version

February 25, 2021

1 Estimating the Bayes Error

a) Show that the full error can be upper-bounded as corresponds to the harmonic mean of class posteriors weighted by $p(x)$.

$$p(error) = \int min[p(\omega_1|x), p(\omega_2|x)]p(x)dx$$

suppose $p(\omega_2|x) > p(\omega_1|x)$, then

$$p(\omega_2|x) > \frac{1}{2} > \min[p(\omega_1|x), p(\omega_2|x)]$$

$$2p(\omega_2|x) > \frac{1}{2} + p(\omega_2|x) > p(\omega_1|x) + p(\omega_2|x)$$

As the probability density is positive:

$$\frac{1}{2p(\omega_2|x)} < \frac{1}{p(\omega_1|x) + p(\omega_2|x)}$$

Multiply both sides $p(\omega_2|x)p(\omega_1|x)$,

$$p(\omega_1|x) < \frac{2p(\omega_2|x)p(\omega_1|x)}{p(\omega_1|x) + p(\omega_2|x)}$$

which indicates,

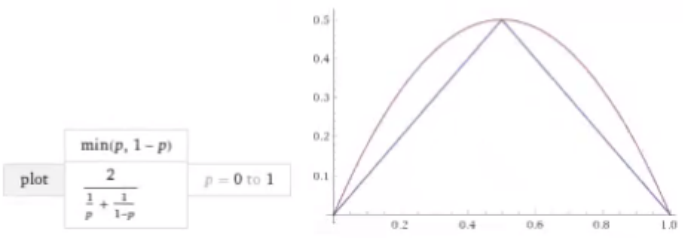
$$\min[p(\omega_1|x), p(\omega_2|x)] \leq \frac{2}{\frac{1}{p(\omega_1|x)} + \frac{1}{p(\omega_2|x)}}$$

Version 1: Using the *generalized mean inequality*.

$$\min[P(\omega_1|\boldsymbol{x}), P(\omega_2|\boldsymbol{x})] = M_{-\infty}(P(\omega_1|\boldsymbol{x}), P(\omega_2|\boldsymbol{x}))$$
$$\leq M_{-1}(P(\omega_1|\boldsymbol{x}), P(\omega_2|\boldsymbol{x})) = \frac{2}{\frac{1}{P(\omega_2|\boldsymbol{x})} + \frac{1}{P(\omega_2|\boldsymbol{x})}}$$

where M_d is the generalized mean of degree d. Multiplying, by the positive quantity $p(\boldsymbol{x})$ on both sides, and integrating, we get desired the inequality.

Version 2: Graphically (and observing $P(\omega_2|\boldsymbol{x}) = (1 - P(\omega_1|\boldsymbol{x}))$



b) Use the conclusion of a) combined with Bayesian formula gives,

$$p(error) \leq \int \frac{2p(x|\omega_1)p(\omega 1)p(x|\omega_2)p(\omega_2)}{p(x|\omega_1)p(\omega_1)+p(x|\omega_2)p(\omega_2)}dx$$

simplify the inequality and substitute the definition of $p(x|w_i)$ will give,

$$p(error) \leq 2p(\omega_1)p(\omega_2)\pi^{-1}\int \frac{1}{[p(\omega_1)+p(\omega_2)]x^2+2u[p(\omega_1)-p(\omega_2)]x+[p(\omega_1)+p(\omega_2)](u^2+1)}$$

use the hint identity in the problem text, we have

$$p(error) \leq 2p(\omega_1)p(\omega_2)\pi^{-1}\frac{2\pi}{\sqrt{4(u^2+1)-4u^2[p(\omega_1)-p(\omega_2)]^2}}$$

rearrange the right side,

$$p(error) \leq \frac{2p(\omega_1)p(\omega_2)}{\sqrt{u^2+1-u^2[p(\omega_1)-p(\omega_2)]^2}}$$

then use $[p(\omega_1)+p(\omega_2)]^2=[p(\omega_1)-p(\omega_2)]^2+4p(\omega_1)p(\omega_2)$ and under binary condition $p(\omega_1)+p(\omega_2)=1$ we will get the target inequality,

$$p(error) \leq \frac{2p(\omega_1)p(\omega_2)}{\sqrt{1+4u^2p(\omega_1)p(\omega_2)}}$$

c) No idea.

Standard Answer from Tutor.

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

(1) Low-dimensional: numerical integration of the original error function $P(\text{error}|\boldsymbol{x})$.

(2) High-dimensional: (i) Build some discriminants $g_1(\boldsymbol{x}), g_2(\boldsymbol{x})$ supporting an optimal classifier, (ii) sample hierarchically: first the class from the priors $\omega_j \sim [P(\omega_1), P(\omega_2)]$ and then the data point from the class-conditioned distributions $P(\boldsymbol{x}\,|\,\omega_j)$, (iii) compute the average over some sufficiently large sample the disagreement between the true sampled class and the decision based on the discriminants.

2 Bayes Decision Boundaries

a) Determine for which values of $P(\omega_1), P(\omega_2), \mu, \sigma$ the optimal decision is to always predict the first class.

$P(error|x)=P(\omega_2|x) \quad \forall x \in \mathbb{R}$ means under the binary condition $p(error|x)=\min\{p(\omega_1|x), p(\omega_2|x)\}$, it will be $p(\omega_2|x)<p(\omega_1|x) \quad \forall x \in \mathbb{R}$. Use Bayesian formula gives:

$$\frac{p(x|\omega_2)p(\omega_2)}{p(x)} < \frac{p(x|\omega_1)p(\omega_1)}{p(x)}$$

eliminate $p(x)$ from both sides and rearrange the inequality,

$$\frac{p(\omega_2)}{p(\omega_1)} < \frac{p(x|\omega_1)}{p(x|\omega_2)}$$

Use the definition of the likelihood,

$$\frac{p(\omega_1)}{p(\omega_2)} > \frac{\exp(-\frac{|x+\mu|}{\sigma})}{\exp(-\frac{|x-\mu|}{\sigma})}$$

Use logarithm from both side,

$$\sigma \ln[\frac{p(\omega_1)}{p(\omega_2})] > |x-\mu|-|x+\mu|$$

The maximum of right hand side is 2μ, because $|a|-|b| \leq |a-b|$, so when the relationship between the four parameters satisfy $\sigma \ln[\frac{p(\omega_1)}{p(\omega_2}] > 2\mu$, the settings in the problem can be guaranteed over $\forall x, x \in \mathbb{R}$.

b) As the derivation from a) part, the relation should be $\frac{p(\omega_2)}{p(\omega_1)} < \frac{p(x|\omega_1)}{p(x|\omega_2)}$. Substitute the definition of the likelihood $p(x|\omega_i)$ and apply logarithm both sides, the inequality become:

$$2\sigma^2 \ln[\frac{p(\omega_1)}{p(\omega_2)}] > (x-\mu)^2 - (x+u)^2 = -4\mu x$$

Divide both sides with -4μ. If μ is positive, the inequality becomes:

$$x > -\frac{\sigma^2}{2\mu} \ln[\frac{p(\omega_1)}{p(\omega_2)}]$$

Where the $-\frac{\sigma^2}{2\mu}$ is a negative constant. If μ is negative,

$$x < -\frac{\sigma^2}{2\mu} \ln[\frac{p(\omega_1)}{p(\omega_2)}]$$

Where the $-\frac{\sigma^2}{2\mu}$ is a positive constant. So under both assumptions, the only requirement is, $|\frac{\sigma^2}{2\mu}| \ln[\frac{p(\omega_1)}{p(\omega_2)}] \to +\infty$, thus any presetting x will satisfy $P(error|x) = P(\omega_2|x) \quad \forall x \in \mathbb{R}$.

answer next page from math workmate, it's correct.

Exercise 2

a) Let $P(\omega_1), P(\omega_2), \mu, \sigma$ be such that the optimal decision is to always predict the first class. First of all we notice that $P(\omega_2) = 0$ is an obvious solution and that $P(\omega_1) \neq 0$ must hold. So going forward we assume $P(\omega_1) \neq 0$ and $P(\omega_2) \neq 0$. Further must hold

$$P(\omega_1|x) > P(\omega_2|x) \; \forall x \in \mathbb{R}$$

This gives us with the Bayesian Formula

$$\frac{p(x|\omega_1)P(\omega_1)}{p(x)} > \frac{p(x|\omega_2)P(\omega_2)}{p(x)} \; \forall x \in \mathbb{R}$$

So we get

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)} \; \forall x \in \mathbb{R}$$

Plugging in the Laplacian probability distributions yields

$$\frac{\exp(-\frac{|x-\mu|}{\sigma})}{\exp(-\frac{|x+\mu|}{\sigma})} > \frac{P(\omega_2)}{P(\omega_1)} \; \forall x \in \mathbb{R}$$

Using the natural logarithm on both sides finally gives us

$$\sigma \ln(\frac{P(\omega_2)}{P(\omega_1)}) < |x+\mu| - |x-\mu| \leq |x+\mu-(x-\mu)| = 2\mu \; \forall x \in \mathbb{R}$$

This means that under this condition $\sigma \ln(\frac{P(\omega_2)}{P(\omega_1)}) < 2\mu$ or if $P(\omega_2) = 0$ the optimal decision is to always predict the first class.

b) Let $P(\omega_1), P(\omega_2), \mu, \sigma$ be such that the optimal decision is to always predict the first class. First of all we notice that $P(\omega_2) = 0$ is an obvious solution and that $P(\omega_1) \neq 0$ must hold. We will see that there is no other solution for this problem apart from the obvious one. So let us assume $P(\omega_2) \neq 0$. We use the same argumentation as in **a)** until

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)} \; \forall x \in \mathbb{R}$$

We plug in the Gaussian probability distributions and get

$$\frac{\exp(-\frac{(x-\mu)^2}{2\sigma^2})}{\exp(-\frac{(x+\mu)^2}{2\sigma^2})} > \frac{P(\omega_2)}{P(\omega_1)} \; \forall x \in \mathbb{R}$$

Again using the natural logarithm on both sides gives

$$2\sigma^2 \ln(\frac{P(\omega_2)}{P(\omega_1)}) < (x+\mu)^2 - (x-\mu)^2 = 4\mu x \; \forall x \in \mathbb{R}$$

Note that if $\mu = 0$ both distributions are the same, so one cannot be larger than the other. So $\mu \neq 0$ which allows us to divide by μ and get

$$\frac{\sigma^2}{2\mu} \ln(\frac{P(\omega_2)}{P(\omega_1)}) < x \; \forall x \in \mathbb{R}$$

if $\mu > 0$. If $\mu < 0$ the inequality sign is just reversed. However in both cases this is a false statement since $\frac{\sigma^2}{2\mu} \ln(\frac{P(\omega_2)}{P(\omega_1)}) \in \mathbb{R}$ is equal to the lefthand side (There is no smallest/biggest real number!). So the assumption $P(\omega_2) \neq 0$ was wrong and thus $P(\omega_2) = 0$ is the only sufficient condition that the optimal decision is to always predict the first class.

Standard Answer from tutor.

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

$$p(x|\omega_1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big) \quad \text{and} \quad p(x|\omega_2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(-\frac{(x+\mu)^2}{2\sigma^2}\Big).$$

where $\sigma > 0$.

The Bayes optimal decision boundary is supported by the pair of discriminants

$$g_1(x) = -\frac{(x-\mu)^2}{2\sigma^2} + \log P(\omega_1)$$
$$g_2(x) = -\frac{(x+\mu)^2}{2\sigma^2} + \log P(\omega_2).$$

We look for parameters values for which $\forall_x : g_1(x) \geq g_2(x)$. The inequality becomes

$$-2x \cdot \mu \leq \sigma \cdot [\log P(\omega_1) - \log P(\omega_2)]$$

We observe from this last inequality that there is never a combination of parameters $\mu, \sigma, P(\omega_1), P(\omega_2)$ for which it is satisfied for all x. Hence, the set of parameters for which it is optimal to always predict ω_1 is $\varnothing$.

sheet01

November 2, 2020

1 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.

1.0.1 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.

[50]:
```
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.

[51]:
```
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)]
```

1.0.2 3D-Plotting

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

[52]:
```
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:

[53]:
```
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:

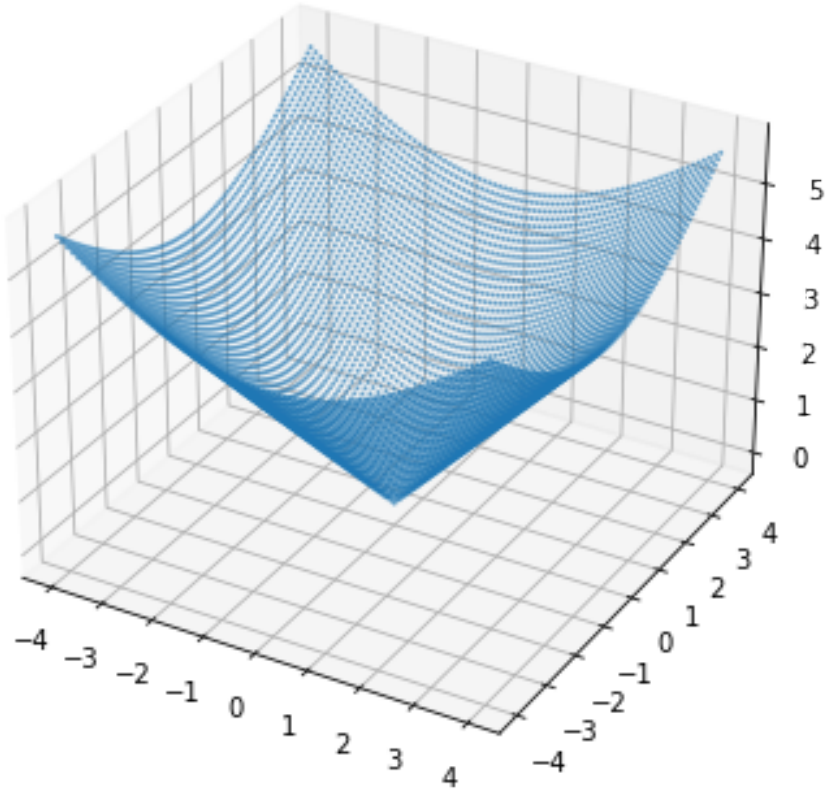
[54]:
```
F = (X**2+Y**2)**.5
print(F.shape)
```

```
(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`.

[55]:
```
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.scatter(X,Y,F,s=1,alpha=0.5)
```

[55]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fac6b11fca0>
```



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.

1.1 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:

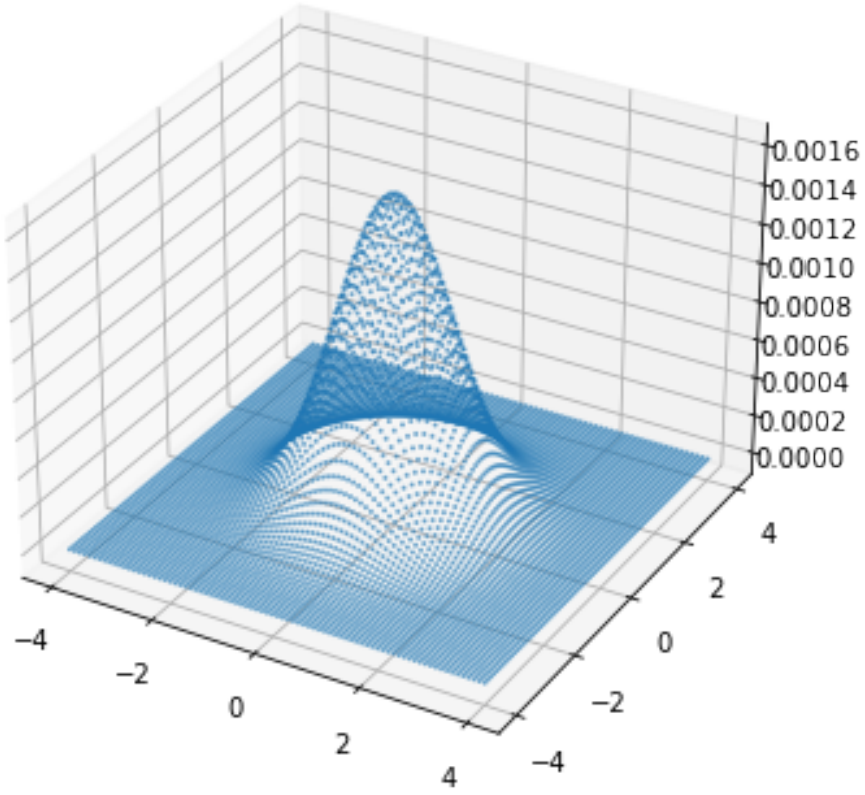
$$P(x, y) = \frac{1}{Z} e^{-0.5(x^2+y^2)} \qquad \text{with} \quad Z = \sum_{x,y} e^{-0.5(x^2+y^2)}$$

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 $Q(x)$.**

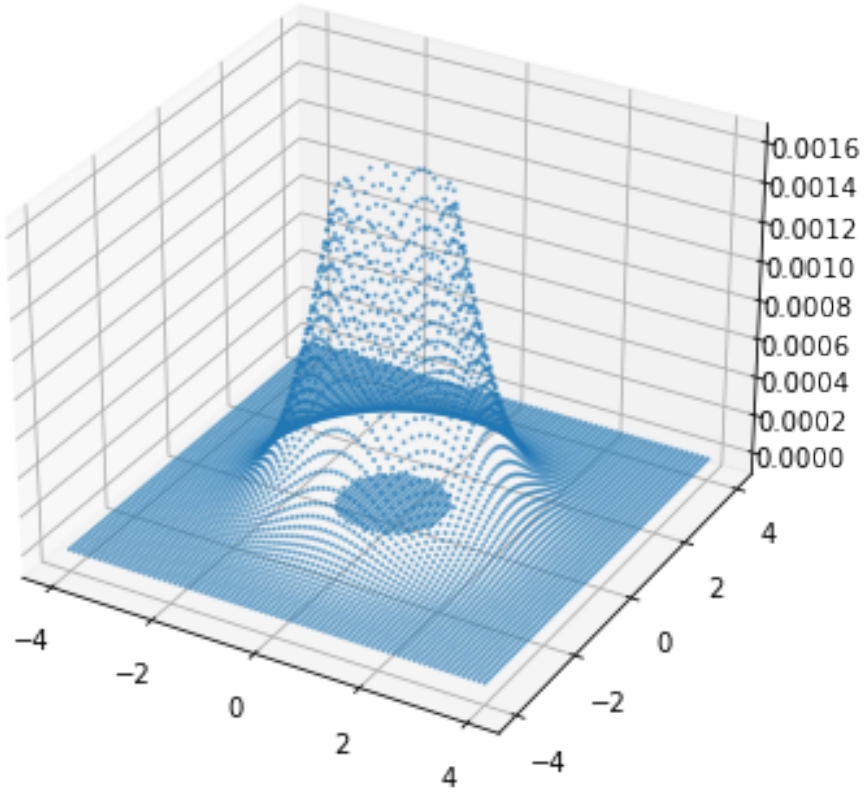
[56]:
```
### s1a
def Prob(u,CoV):
    E = np.exp(-0.5 * ((X-u[0])**2/CoV[0,0] + (Y-u[1])**2/CoV[1,1]))
    Z = np.sum(E)
    return E / Z

u1 = [0, 0]
CoV1 = np.array([[1,0],[0,1]])
P1 = Prob(u1,CoV1)

# plot it
fig1 = plt.figure(figsize=(10,6))
ax1 = plt.axes(projection='3d')
ax1.set_xticks([-4,-2,0,2,4])
ax1.set_yticks([-4,-2,0,2,4])
ax1.scatter(X,Y,P1,s=1,alpha=0.5)
###
```

[56]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fac6b0b4dc0>
```



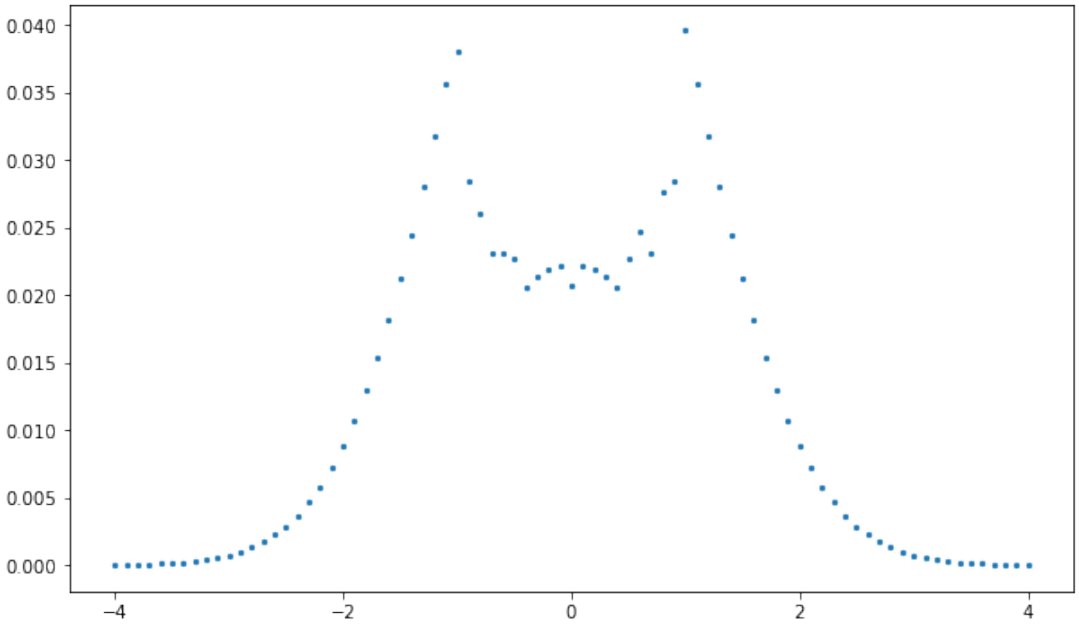
[57]:
```
### s1b
C = (X**2 + Y**2)**.5
u_reduce = [0, 0]
CoV_reduce = np.array([[1,0],[0,1]])
E = np.exp(-0.5 * ((X-u_reduce[0])**2/CoV_reduce[0,0] + (Y-u_reduce[1])**2/
 ↪CoV_reduce[1,1]))
E[C<1] = 0
Z = np.sum(E)
Q = E/Z

# plot it
fig2 = plt.figure(figsize=(10,6))
ax2 = plt.axes(projection='3d')
ax2.set_xticks([-4,-2,0,2,4])
ax2.set_yticks([-4,-2,0,2,4])
ax2.scatter(X,Y,Q,s=1,alpha=0.5)
###
```

[57]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fac6b013850>



[58]:
```
### s1c
Qx = Q.sum(axis = 1)

# plot it
fig3 = plt.figure(figsize=(10,6))
ax3 = plt.axes()
ax3.set_xticks([-4,-2,0,2,4])
ax3.scatter(R,Qx,s=5,alpha=1)
###
```

[58]: <matplotlib.collections.PathCollection at 0x7fac6aff27f0>



1.2 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

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

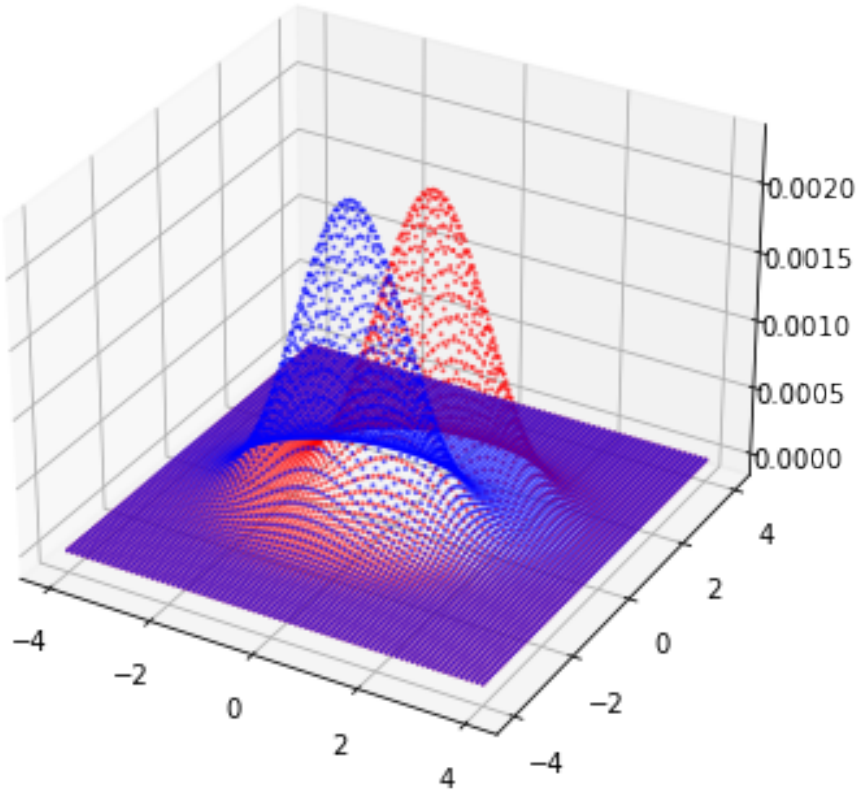
Classes occur with probability $P(\omega_1) = 0.9$ and $P(\omega_2) = 0.1$. Analysis tells us that in such scenario, the optimal decision boundary between the two classes should be linear. We would like to verify this computationally by applying Bayes decision theory on grid-like discretized distributions.

- ** 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.**

[59]:
```
### s2a
u1 = np.array([-0.5, -0.5])
u2 = np.array([0.5, 0.5])
```

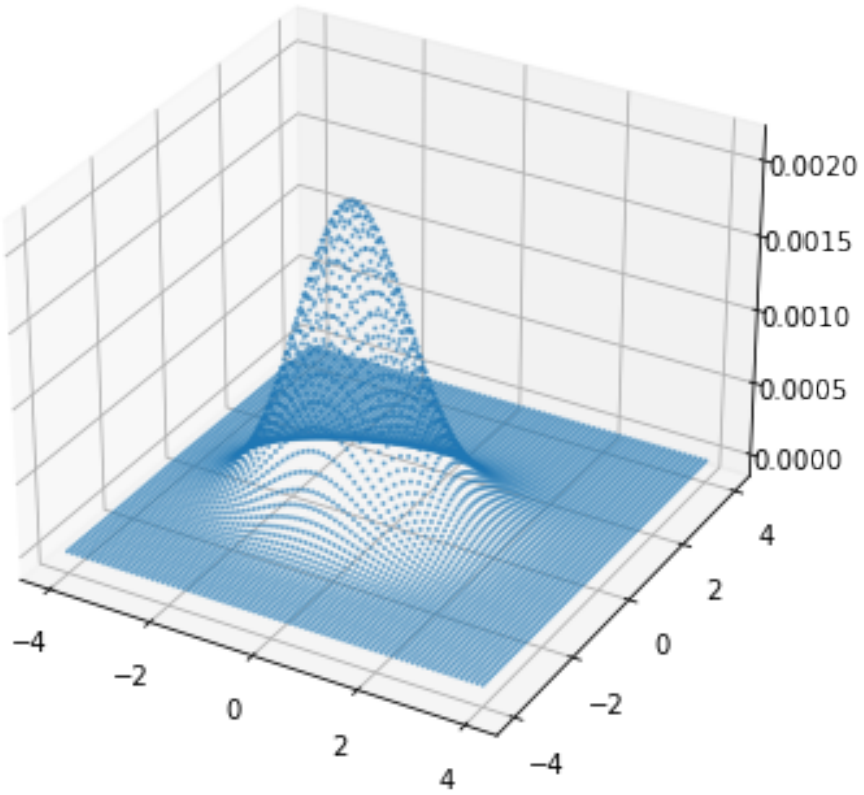
```
CoV2_1 = np.array([[1.0, 0],[0, 0.5]])
CoV2_2 = CoV2_1

P2_1 = Prob(u1, CoV2_1)
P2_2 = Prob(u2, CoV2_2)

# plot it
fig4 = plt.figure(figsize=(10,6))
ax4 = plt.axes(projection='3d')
ax4.set_xticks([-4,-2,0,2,4])
ax4.set_yticks([-4,-2,0,2,4])
ax4.scatter(X,Y,P2_1,s=1,alpha=0.5,color='b')
ax4.scatter(X,Y,P2_2,s=1,alpha=0.5,color='r')
###
```

[59]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fac6af5c8b0>



[60]:
```
### s2b
P2_x = P2_1*0.9 + P2_2*0.1

# plot it
fig5 = plt.figure(figsize=(10,6))
ax5 = plt.axes(projection='3d')
ax5.set_xticks([-4,-2,0,2,4])
ax5.set_yticks([-4,-2,0,2,4])
ax5.scatter(X,Y,P2_x,s=1,alpha=0.5)
###
```

[60]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fac6aebfa90>

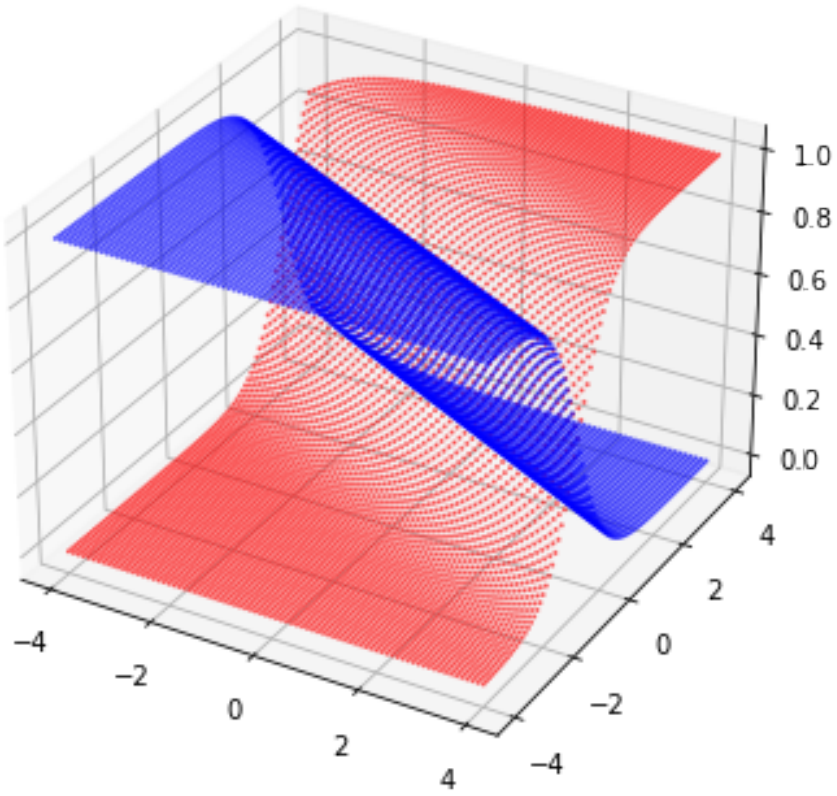


[61]:
```
### s2c
P2_w1_x = P2_1*0.9/P2_x
P2_w2_x = P2_2*0.1/P2_x

BER = np.round(np.sum(np.minimum(P2_w1_x*P2_x, P2_w2_x*P2_x)),3)
print("Bayesian error rate: {}".format(BER))
```

```
# plot it
fig6 = plt.figure(figsize=(10,6))
ax6 = plt.axes(projection='3d')
ax6.set_xticks([-4,-2,0,2,4])
ax6.set_yticks([-4,-2,0,2,4])
ax6.scatter(X,Y,P2_w1_x,s=1,alpha=0.5,color='b')
ax6.scatter(X,Y,P2_w2_x,s=1,alpha=0.5,color='r')
###
```

```
Bayesian error rate: 0.08
```

```
[61]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fac6ae9e9d0>
```



1.3 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. $\Sigma_2 = \Sigma/4$).

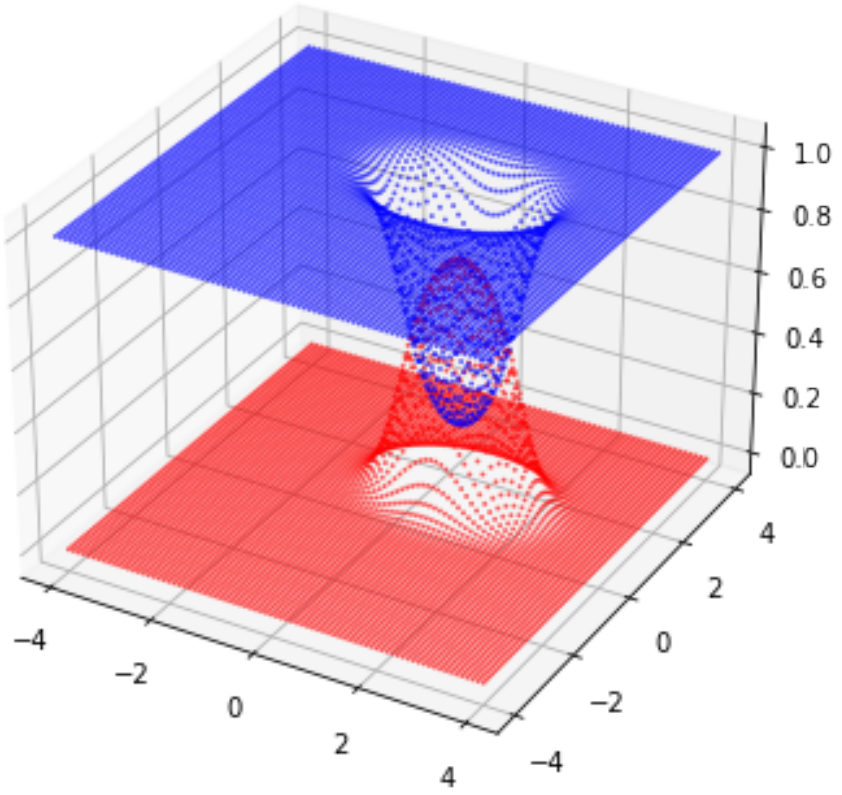
For this new set of parameters, we can perform the same analysis as in Exercise 2.

- **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.**

[62]:
```
### s3a
CoV3_1 = CoV2_1
CoV3_2 = CoV2_2/4

P3_1 = Prob(u1, CoV3_1)
P3_2 = Prob(u2, CoV3_2)

P3_x = P3_1*0.9 + P3_2*0.1

P3_w1_x = P3_1*0.9/P3_x
P3_w2_x = P3_2*0.1/P3_x

BER = np.round(np.sum(np.minimum(P3_w1_x*P3_x, P3_w2_x*P3_x)),3)
print("Bayesian error rate: {}".format(BER))

# plot it
fig6 = plt.figure(figsize=(10,6))
ax6 = plt.axes(projection='3d')
ax6.set_xticks([-4,-2,0,2,4])
ax6.set_yticks([-4,-2,0,2,4])
ax6.scatter(X,Y,P3_w1_x,s=1,alpha=0.5,color='b')
ax6.scatter(X,Y,P3_w2_x,s=1,alpha=0.5,color='r')
###
```

```
Bayesian error rate: 0.073
```

[62]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fac6ae077c0>
```



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.**

[EXPLANATION HERE]

- In the previous exercise, the variance of the class 2 is equal to class 1, which means both distributions will have the same expansion. In the region, beyond the some fixed sigma value, although the prior of class 2 is smaller than class 1, the relative propotion likelihood will decay to 0. So there exits one region in the graph, where the $P(\omega_2|x)$ close to 1. The posterior probability of class 2 could be written as $P(\omega_2|x) = \frac{P(\omega_2)}{P(\omega_2)+p(\omega_2)\frac{P(x|\omega_1)}{P(x|\omega_2)}}$.
- In the last exercise, the variance of class 2 is 4 time less than class 1. The posterior probability of class 2 can be also written out, they act in similar way depending on the likelood proportion. So the decaying rate of probability density of class 2 will larger the class 1, thus there exits no situation in the rand, the posterior of class 2 converges to 1 but to 0. Only in the peak region of class 2, where the propotion of two likelihood apparently not 0, the posterior of class 2 will

surpass class 1's, and will approach 1, if the variance of class 2 continue to shrink.

No idea about the tutor answer here.

In [31]:
```
import solutions
solutions.s3b()
```

```
- In the first case, full certainty for the second class is reached in regions of very low data density
- When the variance is reduced for the second class, the presence of the second class in these low-density becomes ev
en smaller
```