Fecha entrega: Viernes 2 Mayo 2025

Taller 2 - corte 2

Perceptron

In [2]:
```
import random
import math
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
import numpy as np
```

Puntos a evaluar

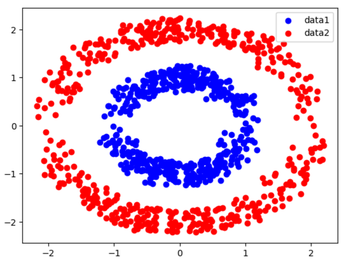
- Realizar la implementacion de un perceptron que logre clasificar el set de datos propuesto
- Generar una funcion de predicciones
- Generar un mapa de calor que identifique las zonas predichas por el modelo en conjunto con el set de datos

In [4]:
```
# generar data
data_1 = []
data_2 = []
for i in range(500):
    # data 1
    noise = 0.5
    radio = 1
    data1_random_x = (random.random()-0.5)*radio*2
    data1_y = math.sqrt( radio**2 - data1_random_x**2 )
    data_1.append( [data1_random_x+(random.random()-0.5)*noise , data1_y*random.sample([-1,1],1)[0]+(random.random()-0.5)*noise] )
    # data 2
    noise = 0.5
    radio = 2
    data2_random_x = (random.random()-0.5)*radio*2
    data2_y = math.sqrt( radio**2 - data2_random_x**2 )
    data_2.append( [data2_random_x+(random.random()-0.5)*noise , data2_y*random.sample([-1,1],1)[0]+(random.random()-0.5)*noise] )

# unificar data
X = data_1 + data_2
y = []
for _ in data_1:
    y.append(-1)
for _ in data_2:
    y.append(1)

plt.scatter( [d[0] for d in data_1], [d[1] for d in data_1], color="blue", label="data1" )
plt.scatter( [d[0] for d in data_2], [d[1] for d in data_2], color="red", label="data2" )
plt.legend()
```

Out[4]:
```
<matplotlib.legend.Legend at 0x245bfcb7710>
```



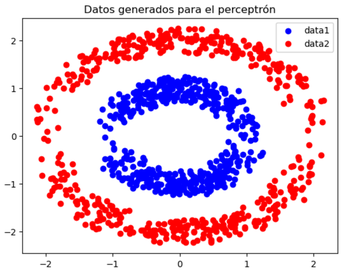
In [3]:
```
import random
import math
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm

data_1 = []
data_2 = []
for i in range(500):
    noise = 0.5
    radio = 1
    data1_random_x = (random.random() - 0.5) * radio * 2
    data1_y = math.sqrt(radio**2 - data1_random_x**2)
    data_1.append([
        data1_random_x + (random.random() - 0.5) * noise,
        data1_y * random.sample([-1, 1], 1)[0] + (random.random() - 0.5) * noise
    ])

    noise = 0.5
    radio = 2
    data2_random_x = (random.random() - 0.5) * radio * 2
    data2_y = math.sqrt(radio**2 - data2_random_x**2)
    data_2.append([
        data2_random_x + (random.random() - 0.5) * noise,
        data2_y * random.sample([-1, 1], 1)[0] + (random.random() - 0.5) * noise
    ])

X = np.array(data_1 + data_2)
y = np.array([-1] * len(data_1) + [1] * len(data_2))

plt.scatter([d[0] for d in data_1], [d[1] for d in data_1], color="blue", label="data1")
plt.scatter([d[0] for d in data_2], [d[1] for d in data_2], color="red", label="data2")
plt.legend()
plt.title("Datos generados para el perceptrón")
plt.show()
```



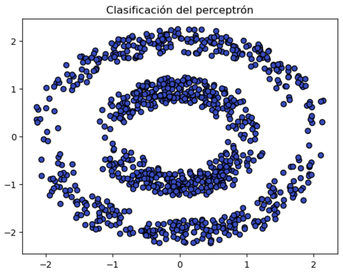
In [5]:
```
class Perceptron:
    def __init__(self, lr=0.01, epochs=1000):
        self.lr = lr
        self.epochs = epochs

    def fit(self, X, y):
        self.w = np.zeros(X.shape[1])
        self.b = 0
        for _ in range(self.epochs):
            for xi, yi in zip(X, y):
                update = self.lr * yi * (1 if (np.dot(xi, self.w) + self.b) * yi <= 0 else 0)
                self.w += update * xi
                self.b += update

    def predict(self, X):
        return np.sign(np.dot(X, self.w) + self.b)
```

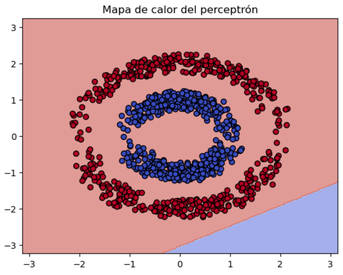
In [7]:
```
model = Perceptron(lr=0.01, epochs=1000)
model.fit(X, y)
preds = model.predict(X)

plt.scatter(X[:, 0], X[:, 1], c=preds, cmap='coolwarm', edgecolors='k')
plt.title("Clasificación del perceptrón")
plt.show()
```



In [8]:
```
xx, yy = np.meshgrid(np.linspace(X[:, 0].min()-1, X[:, 0].max()+1, 200),
                     np.linspace(X[:, 1].min()-1, X[:, 1].max()+1, 200))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap='coolwarm', alpha=0.5)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='coolwarm', edgecolors='k')
plt.title("Mapa de calor del perceptrón")
plt.show()
```



In [ ]: