HW 3

Kevin Lin

2/23/2026

1

Given $X_1, X_2, \ldots, X_n$ are n i.i.d. Bernoulli random variables following:

$$P(X = x; p) = p^x(1-p)^{1-x}, \quad x = 0, 1$$

where p is the probability of X being 1. We want to find the MLE of p using the given samples. The likelihood function is given by:

$$L(p) = \prod_{i=1}^{n} P(X_i = x_i; p) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i}$$

To find the MLE, we take the natural logarithm of the likelihood function:

$$\ell(p) = \ln L(p) = \sum_{i=1}^{n} \left(x_i \ln p + (1-x_i)\ln(1-p)\right)$$

Next, we differentiate $\ell(p)$ with respect to p and set it to zero to find the critical points:

$$\frac{d\ell(p)}{dp} = \sum_{i=1}^{n} \left(\frac{x_i}{p} - \frac{1-x_i}{1-p}\right) = 0$$

This simplifies to:

$$\sum_{i=1}^{n} \frac{x_i}{p} = \sum_{i=1}^{n} \frac{1-x_i}{1-p}$$

Multiplying both sides by $p(1-p)$ gives:

$$(1-p)\sum_{i=1}^{n} x_i = p\sum_{i=1}^{n}(1-x_i)$$

Let $S = \sum_{i=1}^{n} x_i$, then we can rewrite the equation as:

$$\begin{aligned}(1-p)S &= p(n-S)\\ S - pS &= pn - pS\\ S &= pn\\ p &= \frac{S}{n} \therefore\\ \hat{p} &= \frac{\sum_{i=1}^{n} x_i}{n}\end{aligned}$$

Thus, the MLE of p is the sample mean of the observed data.

2

We are given a binary classification problem with outputs $y \in \{0, 1\}$, and assume that $p(y|x)$ is Bernoulli $\hat{p}(x; \theta^*)$ for some parameter vector $\theta^* \in \mathbb{R}^d$, where for each θ, $\hat{p}(x; \theta)$ returns a number between $[0, 1]$.

(a) We know the cross-entropy loss for each parameter θ for a set of n examples with predictions $\hat{p}(x_i; \theta)$ and true labels y_i is:

$$CE(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \left(y_i \ln \hat{p}_i + (1-y_i)\ln(1-\hat{p}_i)\right)$$

We can derive this loss function from the principle of the MLE:

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} P(y_1, y_2, \ldots, y_n | x_1, x_2, \ldots, x_n; \theta)\\ &= \arg\max_{\theta} \prod_{i=1}^{n} P(y_i|x_i;\theta)\\ &= \arg\max_{\theta} \prod_{i=1}^{n} \hat{p}(x_i;\theta)^{y_i}(1-\hat{p}(x_i;\theta))^{1-y_i}\\ &= \arg\max_{\theta} \sum_{i=1}^{n} \left(y_i \ln \hat{p}(x_i;\theta) + (1-y_i)\ln(1-\hat{p}(x_i;\theta))\right)\\ &= \arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n} \left(y_i \ln \hat{p}(x_i;\theta) + (1-y_i)\ln(1-\hat{p}(x_i;\theta))\right)\\ &= \arg\min_{\theta} CE(\theta)\end{aligned}$$

Thus, minimizing the cross-entropy loss is equivalent to maximizing the likelihood of the observed data.

(b) Let's instantiate the prediction as the sigmoid function applied to a linear combination of input features $\hat{p}(x; \theta) = \sigma(\theta^T x)$, where $\sigma(z) = \frac{1}{1+e^{-z}}$.

Then, minimizing the cross-entropy loss is equivalent to minimizing the familiar logistic loss given by:

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-\tilde{y}_i \theta^T x_i}) \text{ where } \tilde{y}_i = \begin{cases} 1 & \text{if } y_i = 1 \\ -1 & \text{if } y_i = 0 \end{cases}$$

We can show this by substituting $\hat{p}(x_i; \theta) = \sigma(\theta^T x_i)$ into the cross-entropy loss:

$$\begin{aligned} CE(\theta) &= -\frac{1}{n}\sum_{i=1}^{n} \left(y_i \ln \sigma(\theta^T x_i) + (1 - y_i) \ln(1 - \sigma(\theta^T x_i))\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n} \left(y_i \ln \frac{1}{1 + e^{-\theta^T x_i}} + (1 - y_i) \ln\left(1 - \frac{1}{1 + e^{-\theta^T x_i}}\right)\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n} \left(y_i \ln \frac{1}{1 + e^{-\theta^T x_i}} + (1 - y_i) \ln \frac{1}{1 + e^{\theta^T x_i}}\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n} \left(-y_i \ln(1 + e^{-\theta^T x_i}) - (1 - y_i) \ln(1 + e^{\theta^T x_i})\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} \left(y_i \ln(1 + e^{-\theta^T x_i}) + (1 - y_i) \ln(1 + e^{\theta^T x_i})\right) \end{aligned}$$

We know that:

$$\tilde{y}_i = \begin{cases} 1 & \text{if } y_i = 1 \\ -1 & \text{if } y_i = 0 \end{cases}$$

Thus, we can define $\tilde{y}_i$ such that:

$$\begin{gathered} \tilde{y}_i = 2y_i - 1 \therefore \\ y_i = \frac{1 + \tilde{y}_i}{2}, 1 - y_i = \frac{1 - \tilde{y}_i}{2} \end{gathered}$$

Substituting this back into the expression for $CE(\theta)$ gives:

$$CE(\theta) = \frac{1}{n}\sum_{i=1}^{n} \left(\frac{1 + \tilde{y}_i}{2} \ln(1 + e^{-\theta^T x_i}) + \frac{1 - \tilde{y}_i}{2} \ln(1 + e^{\theta^T x_i})\right)$$

Thus, if we plug in $\tilde{y}_i = 1$, we get:

$$CE(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-\theta^T x_i})$$

And if we plug in $\tilde{y}_i = -1$, we get:

$$CE(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{\theta^T x_i})$$

which is equivalent to the logistic loss function:

$$L(\theta) = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-\theta^T x_i}) & \text{if } \tilde{y}_i = 1 \\ \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{\theta^T x_i}) & \text{if } \tilde{y}_i = -1 \end{cases}$$

3

See attached Jupyter notebook for code and plots.

(a) We know that data from class A are Gaussian $\mathcal{N}(\mu_1, I_d)$ and data from class B are also Gaussian $\mathcal{N}(\mu_2, I_d)$, where the prior probabilities of both classes are equal. We can derive the posterior probability $P(y|x)$ for each class using Bayes' theorem:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

$$P(x) = P(x|y=0)P(y=0) + P(x|y=1)P(y=1)$$

$$P(y=0|x) = \frac{P(x|y=0)P(y=0)}{P(x|y=0)P(y=0) + P(x|y=1)P(y=1)}$$

$$P(y=1|x) = \frac{P(x|y=1)P(y=1)}{P(x|y=0)P(y=0) + P(x|y=1)P(y=1)}$$

Since the prior probabilities are equal, we can simplify this to:

$$P(y=0|x) = \frac{P(x|y=0)}{P(x|y=0) + P(x|y=1)}$$

$$P(y=1|x) = \frac{P(x|y=1)}{P(x|y=0) + P(x|y=1)}$$

We can further simplify this by substituting the Gaussian probability density functions for $P(x|y=0)$ and $P(x|y=1)$:

$$P(x|y=0) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}\|x-\mu_1\|^2}$$

$$P(x|y=1) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}\|x-\mu_2\|^2}$$

Thus, the posterior probabilities can be expressed as:

$$P(y=0|x) = \frac{e^{-\frac{1}{2}\|x-\mu_1\|^2}}{e^{-\frac{1}{2}\|x-\mu_1\|^2} + e^{-\frac{1}{2}\|x-\mu_2\|^2}}$$
$$P(y=1|x) = \frac{e^{-\frac{1}{2}\|x-\mu_2\|^2}}{e^{-\frac{1}{2}\|x-\mu_1\|^2} + e^{-\frac{1}{2}\|x-\mu_2\|^2}}$$

The decision rule for the Bayes optimal classifier is to assign x to class A if $P(y=0|x) > P(y=1|x)$ and to class B otherwise. This can be simplified to:

$$P(y=0|x) > P(y=1|x)$$
$$\frac{e^{-\frac{1}{2}\|x-\mu_1\|^2}}{e^{-\frac{1}{2}\|x-\mu_1\|^2} + e^{-\frac{1}{2}\|x-\mu_2\|^2}} > \frac{e^{-\frac{1}{2}\|x-\mu_2\|^2}}{e^{-\frac{1}{2}\|x-\mu_1\|^2} + e^{-\frac{1}{2}\|x-\mu_2\|^2}}$$
$$e^{-\frac{1}{2}\|x-\mu_1\|^2} > e^{-\frac{1}{2}\|x-\mu_2\|^2}$$
$$-\frac{1}{2}\|x-\mu_1\|^2 > -\frac{1}{2}\|x-\mu_2\|^2$$
$$\|x-\mu_1\|^2 < \|x-\mu_2\|^2$$

Thus, the decision boundary is the set of points x such that:

$$\|x-\mu_1\|^2 = \|x-\mu_2\|^2$$

This is the equation of a hyperplane that is equidistant from the two means μ_1 and μ_2. We can simplify this further by expanding the norms:

$$(x-\mu_1)^T(x-\mu_1) = (x-\mu_2)^T(x-\mu_2)$$
$$x^Tx - 2\mu_1^Tx + \mu_1^T\mu_1 = x^Tx - 2\mu_2^Tx + \mu_2^T\mu_2$$
$$-2\mu_1^Tx + \mu_1^T\mu_1 = -2\mu_2^Tx + \mu_2^T\mu_2$$
$$2(\mu_2-\mu_1)^Tx = \mu_2^T\mu_2 - \mu_1^T\mu_1$$
$$(\mu_2-\mu_1)^Tx = \frac{1}{2}(\mu_2^T\mu_2 - \mu_1^T\mu_1)$$

Given that $\mu_1 = (0,0)$ and $\mu_2 = (1,1)$, we can substitute these values into the equation for the decision boundary:

$$(\mu_2-\mu_1)^Tx = \frac{1}{2}(\mu_2^T\mu_2 - \mu_1^T\mu_1)$$
$$(1,1)^Tx = \frac{1}{2}(1^2+1^2-0^2-0^2)$$
$$x_1 + x_2 = 1$$

(b) If the prior probabilities are not equal, the Bayes optimal classifier decides based on which class has the highest posterior probability. Thus, the decision rule becomes:

$$P(y=0|x) > P(y=1|x)$$
$$P(x|y=0)P(y=0) > P(x|y=1)P(y=1)$$

as per our derivation of the posteriors in part (a), where we can cancel out the common denominator. We can then simplify this to:

$$P(x|y=0)p > P(x|y=1)(1-p)$$
$$\log P(x|y=0) + \log p > \log P(x|y=1) + \log(1-p)$$
$$-\frac{1}{2}\|x-\mu_1\|^2 + \log p > -\frac{1}{2}\|x-\mu_2\|^2 + \log(1-p)$$
$$\|x-\mu_1\|^2 - \|x-\mu_2\|^2 < 2(\log p - \log(1-p))$$
$$2(\mu_2-\mu_1)^T x + (\|\mu_1\|^2 - \|\mu_2\|^2) < 2(\log p - \log(1-p))$$
$$(\mu_2-\mu_1)^T x < \frac{1}{2}(\|\mu_2\|^2 - \|\mu_1\|^2) + \log\frac{p}{1-p}$$

For the specific case where $\mu_1 = (0,0)$ and $\mu_2 = (1,1)$, and $p = 0.8$, we get:

$$(1,1)^T x < \frac{1}{2}(1^2+1^2-0^2-0^2) + \log\frac{p}{1-p}$$
$$x_1 + x_2 < 1 + \log\frac{p}{1-p}$$

(c) If the covariance matrix for class A is $\sigma^2 I_d$ while class B's matrix is just I_d, the Bayes optimal classifier still chooses the larger posterior probability:

$$\log P(x|y=0) + \log p > \log P(x|y=1) + \log(1-p)$$
$$-\frac{d}{2}\log 2\pi - \frac{d}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\|x-\mu_1\|^2 + \log p > -\frac{d}{2}\log 2\pi - \frac{1}{2}\|x-\mu_2\|^2 + \log(1-p)$$
$$\log\frac{p}{1-p} - \frac{d}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\|x-\mu_1\|^2 + \frac{1}{2}\|x-\mu_2\|^2 > 0$$

For the specific case where $\mu_1 = (0,0)$ and $\mu_2 = (1,1)$, $p = 0.5$, and

$\sigma^2 = 0.25$, we get ($d = 2$):

$$-\log 0.25 + \frac{1}{2 \cdot 0.25}\|x\|^2 - \frac{1}{2}\|x - (1,1)\|^2 > 0$$
$$-\log 0.25 - 2\|x\|^2 + \frac{1}{2}\|x - (1,1)\|^2 > 0$$

(d) Now, if $\mu_1 = \mu_2 = (0,0)$, with $p = 0.5$, $\sigma^2 = 0.25$, the decision boundary is:

$$-\log 0.25 - 2\|x\|^2 + \frac{1}{2}\|x\|^2 > 0$$
$$-\log 0.25 - \frac{3}{2}\|x\|^2 > 0$$
$$\|x\|^2 < -\frac{2}{3}\log 0.25$$

(e) All the decision boundaries we derived in parts (a) through (d) can be realized by a Gaussian Naive Bayes classifier. This is because they have either equal covariance matrices, diagonal covariance matrices, and have no cross-feature correlations, which are all assumptions of the GNB model. For the linear cases of (a) and (b), the decision boundary is a hyperplane, which can be modeled by GNB with equal covariance matrices, even if the priors are unequal. For the quadratic cases of (c) and (d), the decision boundary is a perfect circle, which can be modeled by GNB as the covariance matrices are diagonal and scaled identitically, so there are no cross-feature correlations and the variance is the same across both features. Thus, GNB can model all of these decision boundaries.

4

(a) See attached Jupyter notebook for code and plots.

Dataset	Gaussian Naive Bayes	Linear Logistic Regression
1	Each feature has mixed overlap between 1 and 2. Because GNB assumes that each feature is independent, they are modeled separately and thus the boundary becomes a vertical line.	Dataset is linearly separable so logistic regression performs really well, giving us a clean fit.
2	The decision boundary becomes quadratic, so GNB can create a smooth circled boundary. However because the features are correlated in a slanted ellipse, naive bayes fails to recognize this and thus still somewhat poorly fits, but is still better than logistic regression.	Fails as data is not linearly separable but rather overlaps. Thus, it draws a line that minimizes classification error, but it cannot match the ellipse structure.
3	Each class is radially symmetric, and even though features are not independent (related to radius), GNB can still model it well in circles.	Fails as data is not linearly separable. Thus again, it draws a line that minimizes error but cannot match circular shape as it is constrained to 1 dimension

(b)

h3_code

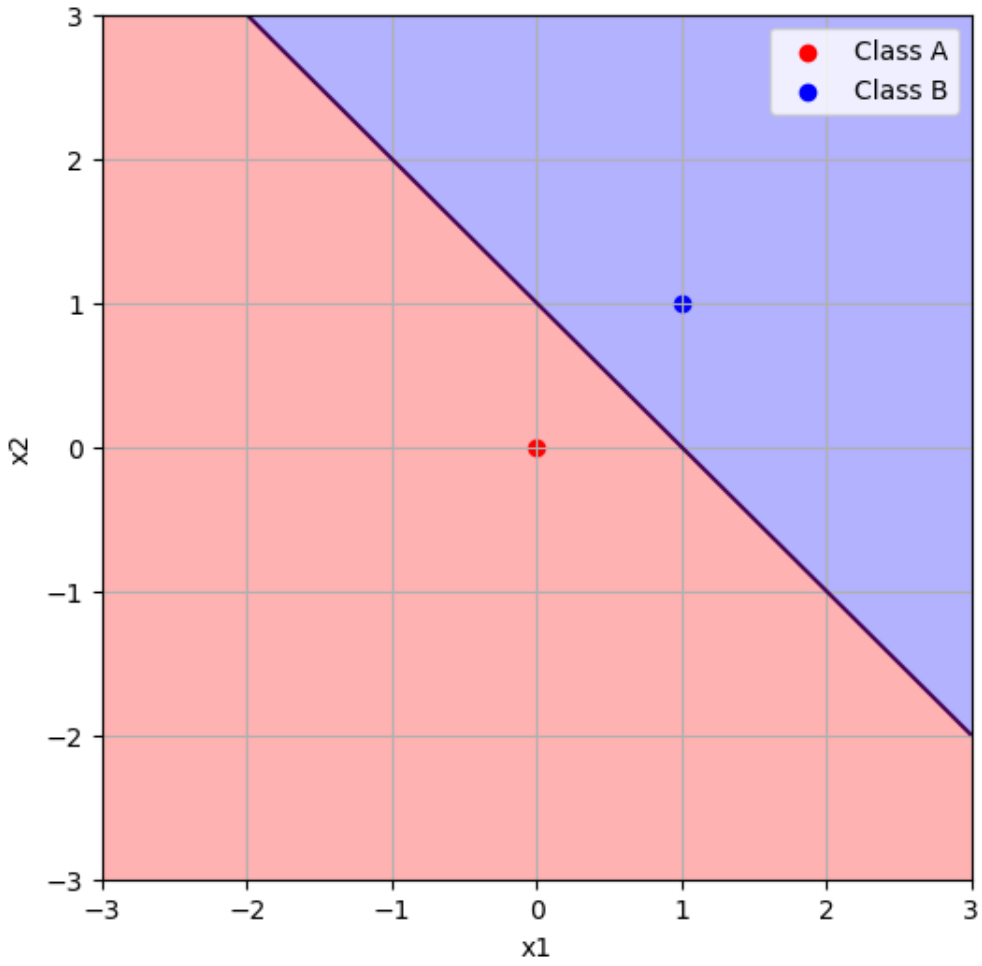
February 21, 2026

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

1 3

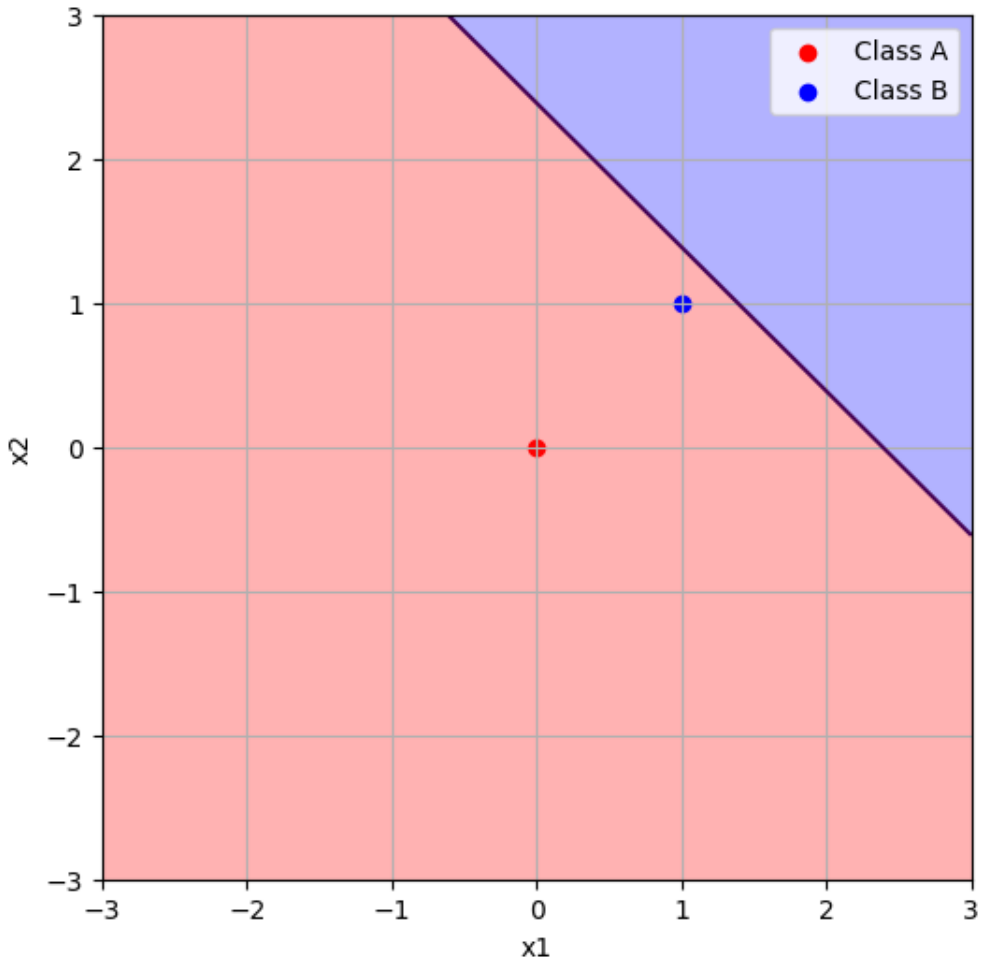
1.1 a

[9]:
```
mu1 = np.array([0, 0])
mu2 = np.array([1, 1])

x1 = np.linspace(-3, 3, 100)
x2 = np.linspace(-3, 3, 100)
X1, X2 = np.meshgrid(x1, x2)

Z = X1 + X2 - 1

plt.figure(figsize=(6, 6))
plt.contourf(X1, X2, Z, levels=[-10, 0, 10], colors=["red", "blue"], alpha=0.3)
plt.contour(X1, X2, Z, levels=[0])
plt.scatter(mu1[0], mu1[1], color='red', label='Class A')
plt.scatter(mu2[0], mu2[1], color='blue', label='Class B')
plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.grid(True)
plt.show()
```



1.2 b

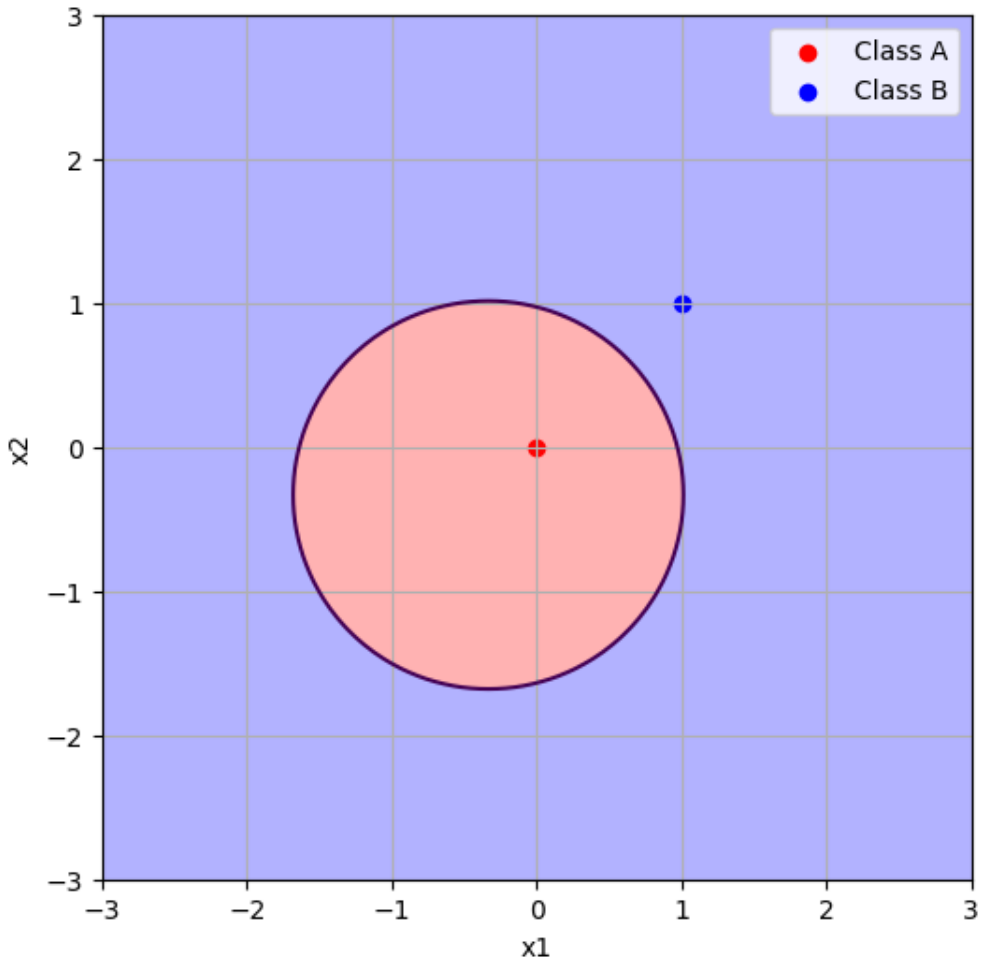
[10]:
```
p = 0.8
Z = (X1 + X2) - (1 + np.log(p / (1 - p)))

plt.figure(figsize=(6, 6))
plt.contourf(X1, X2, Z, levels=[-10, 0, 10], colors=["red", "blue"], alpha=0.3)
plt.contour(X1, X2, Z, levels=[0])
plt.scatter(mu1[0], mu1[1], color='red', label='Class A')
plt.scatter(mu2[0], mu2[1], color='blue', label='Class B')
plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.grid(True)
plt.show()
```



1.3 c

[11]:
```
p = 0.5
sigma2 = 0.25
d = 2

A_dist2 = (X1 - mu1[0])**2 + (X2 - mu1[1])**2
B_dist2 = (X1 - mu2[0])**2 + (X2 - mu2[1])**2

Z = -(d/2)*np.log(sigma2) - (1/(2*sigma2))*A_dist2 + 0.5*B_dist2

plt.figure(figsize=(6, 6))
plt.contourf(X1, X2, Z, levels=[-1e9, 0, 1e9], colors=["blue", "red"], alpha=0.
↪3)
plt.contour(X1, X2, Z, levels=[0])
plt.scatter(mu1[0], mu1[1], color='red', label='Class A')
```

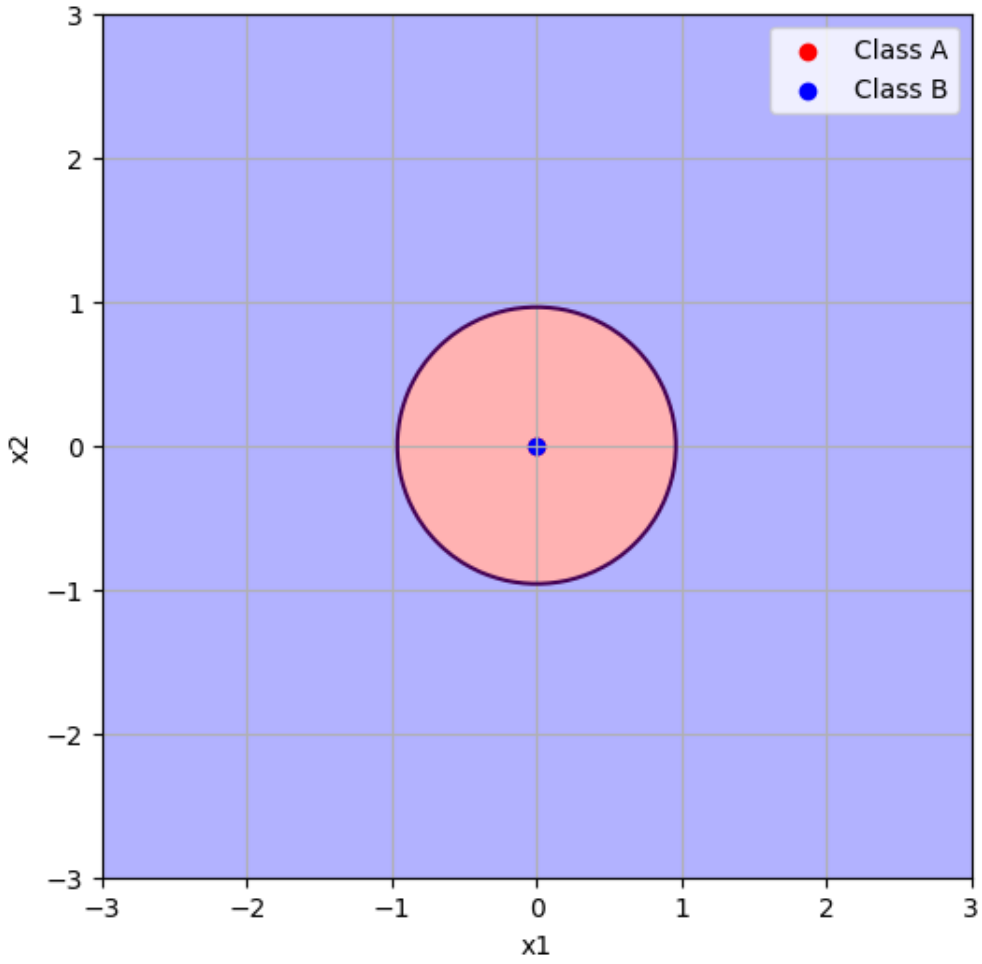
```
plt.scatter(mu2[0], mu2[1], color='blue', label='Class B')
plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.grid(True)
plt.show()
```



1.4 d

[13]:
```
dist2 = X1**2 + X2**2

Z = -(d/2)*np.log(sigma2) - (1/(2*sigma2))*dist2 + 0.5*dist2

plt.figure(figsize=(6, 6))
plt.contourf(X1, X2, Z, levels=[-1e9, 0, 1e9], colors=["blue", "red"], alpha=0.
↪3)
```

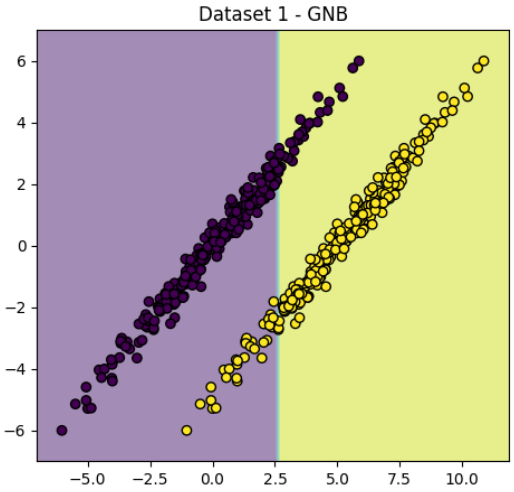
```
plt.contour(X1, X2, Z, levels=[0])
plt.scatter(mu1[0], mu1[1], color='red', label='Class A')
plt.scatter(mu1[0], mu1[1], color='blue', label='Class B')
plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.grid(True)
plt.show()
```

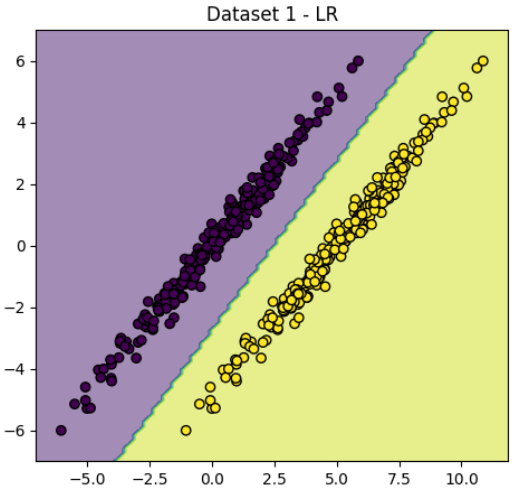


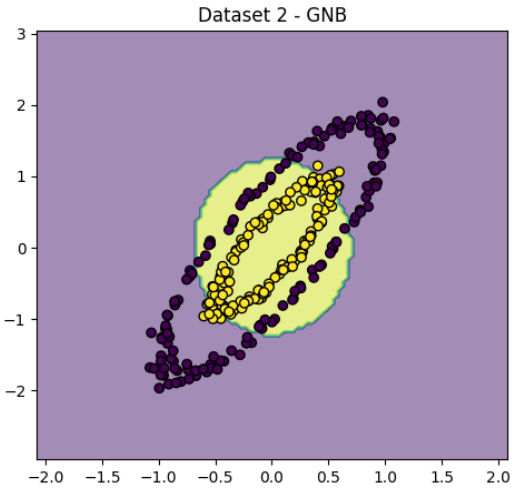
2 4

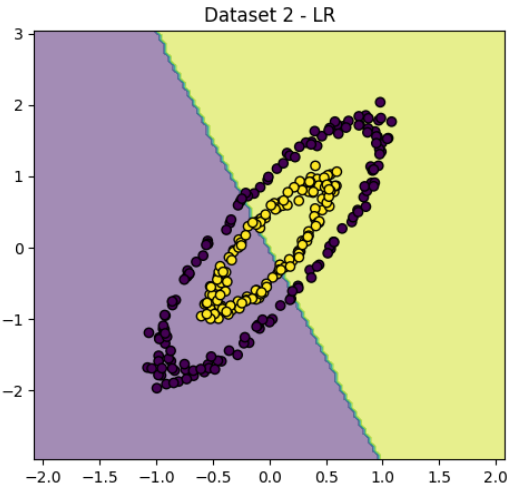
[14]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression

data1 = np.load('data-1.npy')
```

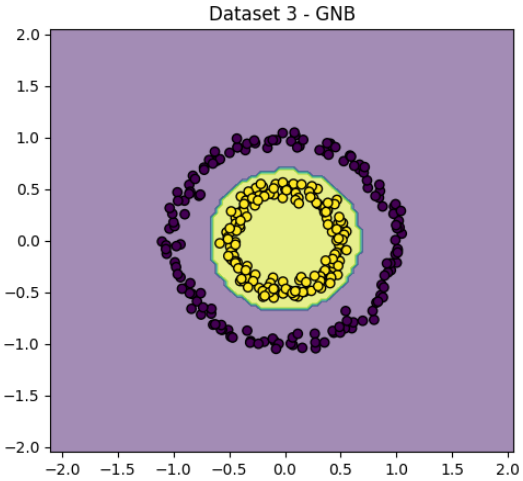
```
data2 = np.load('data-2.npy')
data3 = np.load('data-3.npy')

datasets = [data1, data2, data3]
for i, data in enumerate(datasets):
    X = data[:, :2]
    y = data[:, 2]

    # Fit Gaussian Naive Bayes
    gnb = GaussianNB()
    gnb.fit(X, y)

    # Fit Logistic Regression
    lr = LogisticRegression()
    lr.fit(X, y)

    # Plotting
    plt.figure(figsize=(12, 5))

    # Plot decision boundaries
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min,␣
 ↪y_max, 100))

    Z_gnb = gnb.predict(np.c_[xx.ravel(), yy.ravel()])
    Z_gnb = Z_gnb.reshape(xx.shape)

    Z_lr = lr.predict(np.c_[xx.ravel(), yy.ravel()])
    Z_lr = Z_lr.reshape(xx.shape)

    plt.subplot(1, 2, 1)
    plt.contourf(xx, yy, Z_gnb, alpha=0.5, cmap='viridis')
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', marker='o')
    plt.title(f'Dataset {i+1} - GNB')

    plt.subplot(1, 2, 2)
    plt.contourf(xx, yy, Z_lr, alpha=0.5, cmap='viridis')
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', marker='o')
    plt.title(f'Dataset {i+1} - LR')
    plt.show()
```

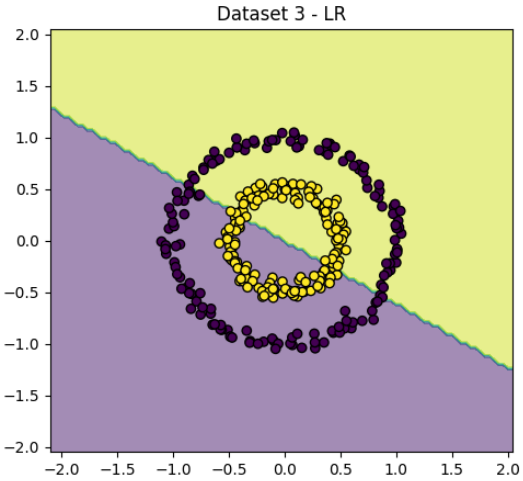

Dataset 1 - GNB
6
4
2
0
−2
−4
−6
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Dataset 1 - LR
6
4
2
0
−2
−4
−6
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Dataset 2 - GNB
3
2
1
0
−1
−2
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Dataset 2 - LR
3
2
1
0
−1
−2
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Dataset 3 - GNB
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0


Dataset 3 - LR
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0