Course: MYZ307 **Homework 2**
Student: Ata Ataş **Term:** Fall 2025
Student ID: 040230771

I already uploaded the report within the previous time constraints. But I wanted to upload a revised version since the deadline was postponed.

(a) Data Generation and CSV Export

Two 2D Gaussian classes (each with 500 samples) were generated and saved to `distribution.csv`. Parameters used:

$$\mu_1 = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} -1.0 \\ 0.8 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1.2 & -0.4 \\ -0.4 & 0.8 \end{bmatrix}.$$

Essentials:

```
n_samples = 500
mean1 = [1.5, -0.5]
mean2 = [-1.0,  0.8]
cov   = [[1.2, -0.4], [-0.4, 0.8]]

X1 = np.random.multivariate_normal(mean1, cov, n_samples)
X2 = np.random.multivariate_normal(mean2, cov, n_samples)
y1 = np.ones(n_samples); y2 = -np.ones(n_samples)

X = np.vstack((X1, X2))
y = np.hstack((y1, y2))

df = pd.DataFrame({"x1":X[:,0], "x2":X[:,1], "label":y})
df.to_csv("distribution.csv", index=False)
```

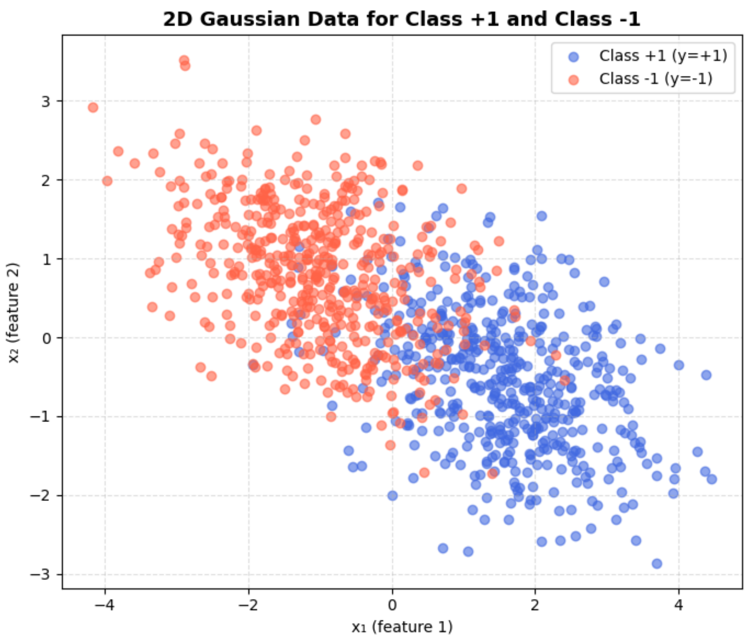

Figure 1: Scatter plot of the generated data

(b) Perceptron Learning Algorithm

Update rule (only if $y_i(w^\top x_i + b) \leq 0$):
The method given in the course files was updated by shuffling the data in each epoch. The reason for this was although the loss value was reasonable in the loss function, the classification boundary line was graphed in a way that it classified the entire data as either one of the classes. I depicted the reason for this was the loss function was optimized according to a local minima in the gradient descent process. And since the data was small enough to do these kind of optimizations in the stochastic gradient descent process, this method was used to prevent it.
10, 20, 50 and 100 epochs were tested in this code to analyze the effects of different epoch numbers on the optimization. The learning rate was chosen 0.5 after some testing to prevent small epoch numbers from performing too poor for comparison.

```
def perceptron_train(X, y, lr=0.05, epochs=20, seed=42):
    rng = np.random.default_rng(seed)
    n, d = X.shape
    w = np.random.rand(d); b = np.random.rand()
    for epoch in range(epochs):
        idx = rng.permutation(n)  # shuffle each epoch
        for i in idx:
            margin = y[i] * (np.dot(w, X[i]) + b)
            if margin <= 0:
                w += lr * y[i] * X[i]
                b += lr * y[i]
    return w, b

    # Start training
    w_0, b_0, fl_0 = perceptron_train(X, y, lr=0.05, epochs=10)
    w_1, b_1, fl_1 = perceptron_train(X, y, lr=0.05, epochs=20)
    w_2, b_2, fl_2 = perceptron_train(X, y, lr=0.05, epochs=50)
    w_3, b_3, fl_3 = perceptron_train(X, y, lr=0.05, epochs=100)
```

(c) Loss vs. Epoch (10, 20, 50, 100)

Average misclassification proxy per epoch recorded for 4 runs (10/20/50/100 epochs).
Figure 2. Loss progression subplots for 4 epoch counts.

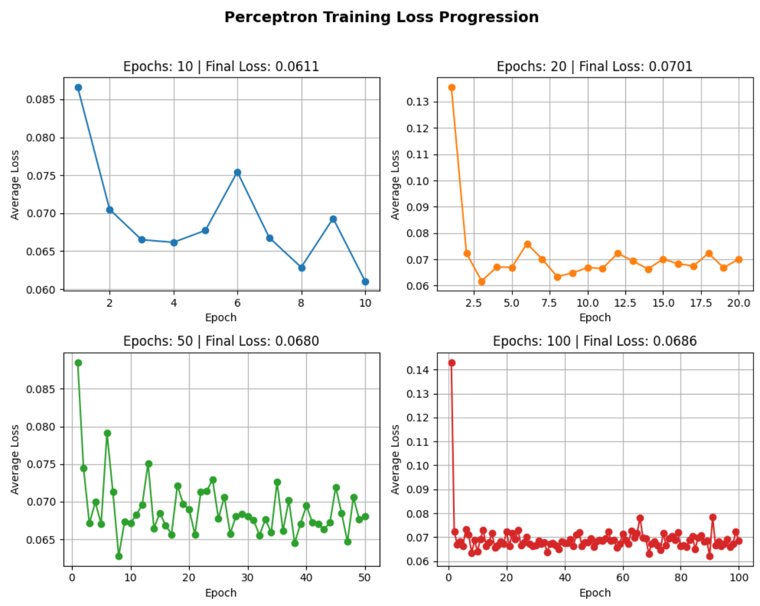

Figure 2: Loss progression plots

(d) Decision Boundary

Decision boundary:

$$w_1x_1 + w_2x_2 + b = 0 \;\Rightarrow\; x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}.$$

Numerical equations obtained from trained weights/biases:

- **Epochs 10:** $w = [0.1052,\ -0.0750],\ b = 0.0533 \Rightarrow\ x_2 = 1.401\,x_1 + 0.710$
- **Epochs 20:** $w = [0.1440,\ -0.1176],\ b = -0.1102 \Rightarrow\ x_2 = 1.225\,x_1 + 0.937$
- **Epochs 50:** $w = [0.0748,\ 0.0198],\ b = 0.0347 \Rightarrow\ x_2 = -3.786\,x_1 - 1.756$
- **Epochs 100:** $w = [0.0308,\ -0.1410],\ b = -0.0351 \Rightarrow\ x_2 = 0.219\,x_1 - 0.249$

Summary of Learned Parameters and Metrics

Epochs	**Slope**	**Intercept**	**Acc**	**Prec**	**F1**
10	1.40	0.71	0.86	0.80	0.87
20	1.22	0.94	0.87	0.93	0.86
50	-3.79	-1.76	0.81	0.74	0.84
100	0.22	-0.25	0.81	0.88	0.80

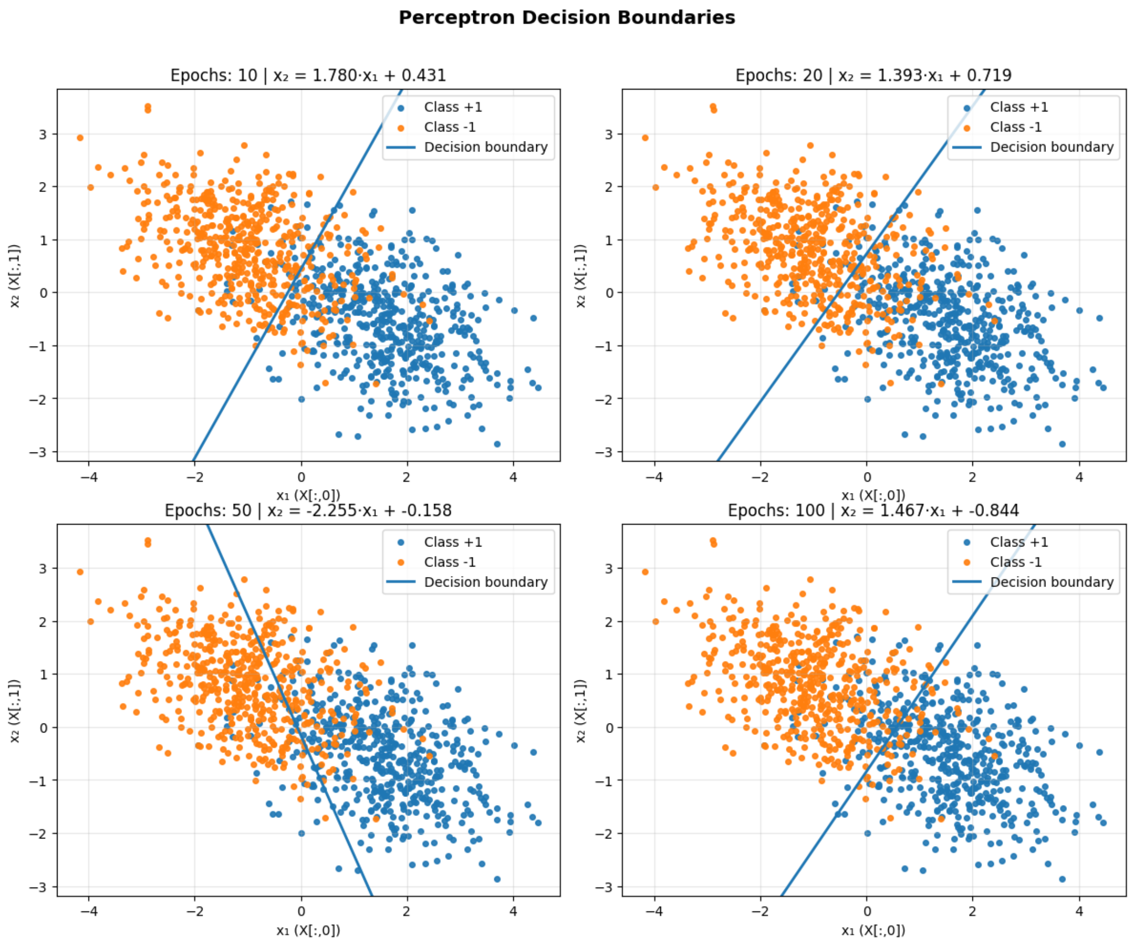

Figure 3: Decision boundary plots

(e) Sample Means and Covariances (by Class)

Class +1 (y=+1):

$$\hat{\mu}_1 = \begin{bmatrix} 1.5571 \\ -0.5572 \end{bmatrix}, \quad \hat{\Sigma}_1 = \begin{bmatrix} 1.1558 & -0.3629 \\ -0.3629 & 0.7724 \end{bmatrix}.$$

Class -1 (y=-1):

$$\hat{\mu}_2 = \begin{bmatrix} -1.0203 \\ 0.8172 \end{bmatrix}, \quad \hat{\Sigma}_2 = \begin{bmatrix} 1.1502 & -0.3775 \\ -0.3775 & 0.7371 \end{bmatrix}.$$

(f) Confusion Matrices

Layout:

$$\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$$

Epochs 10: $\begin{bmatrix} 470 & 30 \\ 90 & 410 \end{bmatrix}$, Acc=0.88, Prec=0.84, Rec=0.94, F1=0.89.
Epochs 20: $\begin{bmatrix} 477 & 23 \\ 119 & 381 \end{bmatrix}$, Acc=0.86, Prec=0.80, Rec=0.95, F1=0.87.
Epochs 50: $\begin{bmatrix} 459 & 41 \\ 135 & 365 \end{bmatrix}$, Acc=0.82, Prec=0.77, Rec=0.92, F1=0.84.
Epochs 100: $\begin{bmatrix} 415 & 85 \\ 37 & 463 \end{bmatrix}$, Acc=0.88, Prec=0.92, Rec=0.83, F1=0.87.

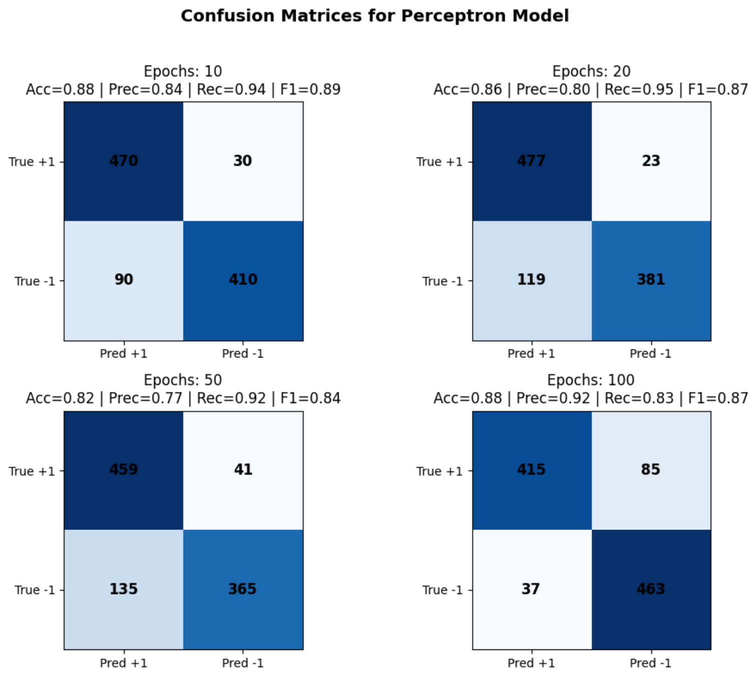

Figure 4: Confusion matricex plots for each epoch

(g) Performance Metrics (20 Epochs)

For the 10-epoch model: Accuracy=**0.88**, Precision=**0.84**, Recall=**0.94**, F_1=**0.89**.
For the 20-epoch model: Accuracy=**0.87**, Precision=**0.93**, Recall=**0.79**, F_1=**0.86**.
For the 50-epoch model: Accuracy=**0.82**, Precision=**0.77**, Recall=**0.92**, F_1=**0.84**.
For the 100-epoch model: Accuracy=**0.88**, Precision=**0.92**, Recall=**0.83**, F_1=**0.87**.

(h) Discussion and Efficiency Evaluation

The perceptron achieved the best trade-off at **20 epochs** with accuracy 0.87 and F1 score 0.86. The slope and intercept changes over training show the boundary adjusting directionally as the algorithm converges. My deduction from this result is that lower epoch numbers may cause the optimization end prematurely and the higher epoch rates can cause oscillations in the loss rate. Whether or not the loss rate being equal to the lower values of this oscillations or to the higher ones cannot be predicted simply, so keeping the epoch number in a reasonable rate actually improves the converged loss rate.

The model correctly captures the linear separability trend between the two Gaussian clusters. However, because the Gaussians overlap, 100% accuracy is impossible; errors concentrate in the intersection region.

Considering the theoretical Bayes error (around 10–15% for such overlap), achieving 81–87% accuracy indicates the model performs **close to optimally for a linear classifier**. It learns effectively, showing that the algorithm is **efficient enough and behaves correctly**, even though it will not outperform more advanced probabilistic or kernel-based models.

Bayes Error Explanation. For any two probabilistic classes $P(x|C_1)$ and $P(x|C_2)$ with prior probabilities $P(C_1)$ and $P(C_2)$, the minimum possible classification error is given by the *Bayes error rate*:

$$P_e = \int \min\left[P(C_1|x),\, P(C_2|x)\right] dx.$$

This represents the theoretical limit of how well any classifier—even an ideal one—can separate the two distributions. Because the Gaussian distributions in this experiment overlap significantly, the Bayes error is nonzero (approximately 10–15%). Therefore, the perceptron's achieved accuracy (up to 87%) is very close to the theoretical maximum achievable for this dataset, demonstrating that it is indeed a reasonable and efficient model for linearly separable approximation. Although it is still better to evaluate this result relative to the task which is tried to be accomplished. For example an accuracy rate of 87% is not reasonable in the medical field, but it may be accepted in a spam mail detection task.

The handwritten solutions are included at the end of the report. The GitHub Link and CoLab link for this homework are in the underlined relative texts.

Ata Ataş
040230771
[illegible]

b) Dağılımlar arası örtüşme sağlayabilmek için mean1=[1,5 -0,5] seçtim
mean2=[-1,0 0,8]
ve varyansları da var1=1,2; var2=0,8 şeklinde farklı seçerek perceptron optimizasyonu için daha karmaşık bir ortam yaratmak istedim.
Kovaryansi -0,4 alarak iki sınıf arası negatif bir ilişki yarattım.

$$\text{cov} = \begin{bmatrix} 1,2 & -0,4 \\ -0,4 & 0,8 \end{bmatrix}$$

c) Epoch sayısı 10'dan 20'ye giderken loss azalıyor. Bunun sebebini daha fazla iterasyon sayesinde gradyanı minimum noktasına çekmenin daha kolaylaşması olarak verebiliriz. Ama 20 Epoch üstünde loss'un artmasını da mevcut learning rate ile fazla iterasyon sayısının gradyan hesabında osilasyonlara sebep olması olarak açıklayabiliriz.

e) class 1 mean = [1,56 -0,56] cov: $\begin{bmatrix} 1,58 & -0,36 \\ -0,36 & 0,77 \end{bmatrix}$

class 2 mean = [-1,020 0,82] cov: $\begin{bmatrix} 1,15 & -0,37 \\ -0,37 & 0,74 \end{bmatrix}$

h) Standart bayes öğreticisinde maksimum verimliliği %86'ya yakın bulurken %87 perceptron accuracy yeterliliği gösterecek olmakla beraber gauss dağılımına göre iyi sayılır.