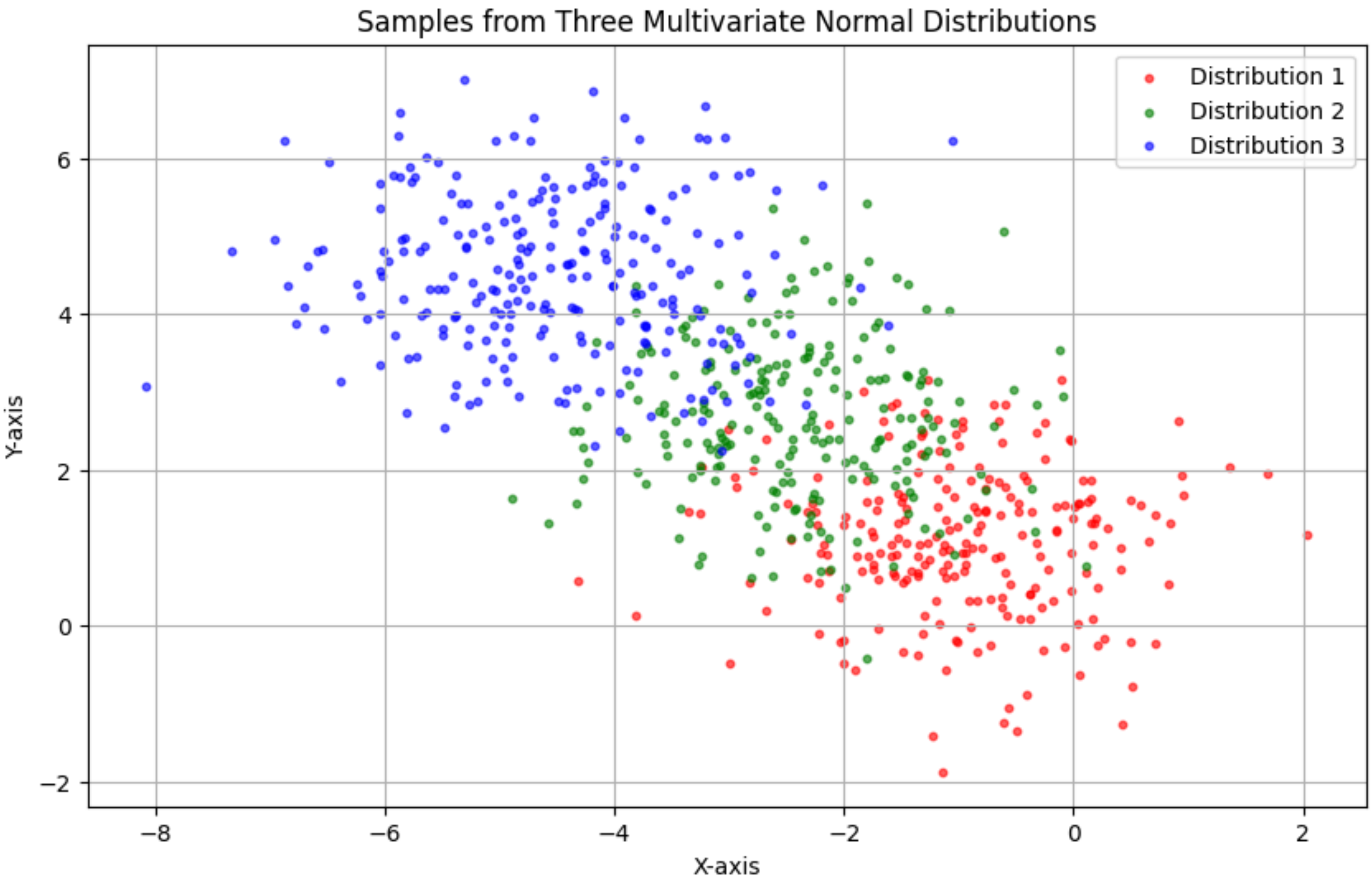
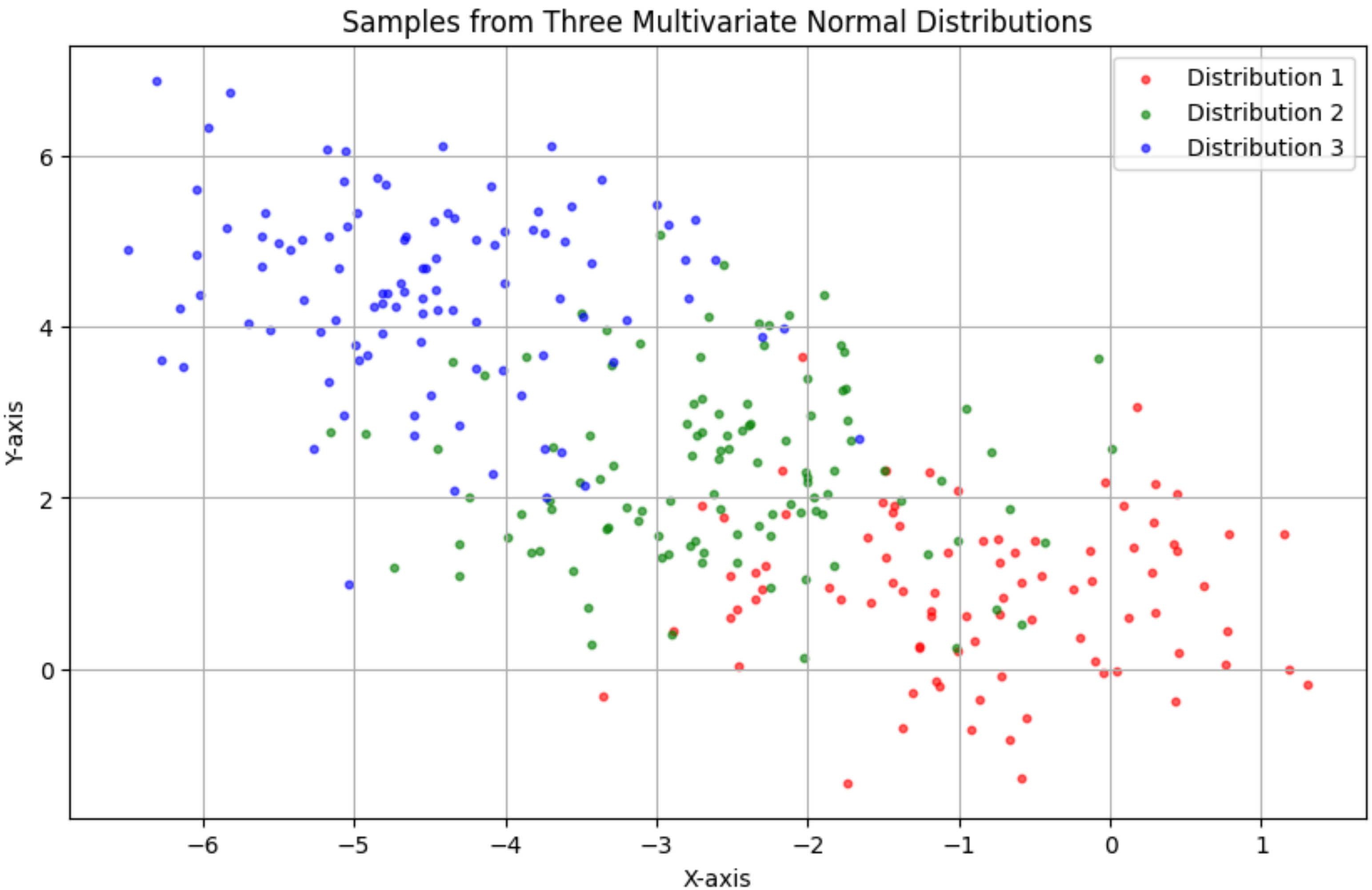
Question 1

1.1, 1.2, 1.3

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

def genRandNormal(n_samples):
    mean1 = [-1, 1]
    cov1 = [[1, 0], [0, 1]]

    mean2 = [-2.5, 2.5]
    cov2 = [[1, 0], [0, 1]]

    mean3 = [-4.5, 4.5]
    cov3 = [[1, 0], [0, 1]]

    allSamples = []
    labels = []
    samples_type1 = []
    samples_type2 = []
    samples_type3 = []
    sampleTypeVec = np.random.randint(1, 4, n_samples)

    for sampleType in sampleTypeVec:
        if sampleType == 1:
            sample = np.random.multivariate_normal(mean1, cov1)
            samples_type1.append(sample)
            labels.append(1)
        elif sampleType == 2:
            sample = np.random.multivariate_normal(mean2, cov2)
            samples_type2.append(sample)
            labels.append(2)
        elif sampleType == 3:
            sample = np.random.multivariate_normal(mean3, cov3)
            samples_type3.append(sample)
```

```
            labels.append(3)

        allSamples.append(sample)


    samples_type1 = np.array(samples_type1)
    samples_type2 = np.array(samples_type2)
    samples_type3 = np.array(samples_type3)
    allSamples = np.array(allSamples)
    labels = np.array(labels)

    return allSamples, labels, [samples_type1, samples_type2, samples_type3]

def plotSamples(subSamples):
    plt.figure(figsize=(10, 6))

    if subSamples[0].size > 0:
        plt.scatter(subSamples[0][:, 0], subSamples[0][:, 1], s=10, alpha=0.6, label='Distribution 1', color='r')
    if subSamples[1].size > 0:
        plt.scatter(subSamples[1][:, 0], subSamples[1][:, 1], s=10, alpha=0.6, label='Distribution 2', color='g')
    if subSamples[2].size > 0:
        plt.scatter(subSamples[2][:, 0], subSamples[2][:, 1], s=10, alpha=0.6, label='Distribution 3', color='b')

    plt.title('Samples from Three Multivariate Normal Distributions')
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.legend()
    plt.grid(True)
    plt.show()

# Generate train and test samples
trainSamples, trainLabels, trainSubSamples = genRandNormal(700)
plotSamples(trainSubSamples)

testSamples, testLabels, testSubSamples = genRandNormal(300)
plotSamples(testSubSamples)
```


Samples from Three Multivariate Normal Distributions
Distribution 1
Distribution 2
Distribution 3
Y-axis
X-axis
6
4
2
0
−2
−8
−6
−4
−2
0
2



1.4

```
In [4]: from sklearn.neighbors import KNeighborsClassifier

        def KNNClassifier(n_neighbors, trainSamples, trainLabels, testSamples, testLabels):
            # Fit KNN classifier
            KNN = KNeighborsClassifier(n_neighbors)
```

```
    KNN.fit(trainSamples, trainLabels)

    # Predict using KNN classifier
    preds = KNN.predict(testSamples)

    # Calculate error rate
    return np.mean(preds != testLabels)

# Calculate error rate on train set
accuracy_train = KNNClassifier(1, trainSamples, trainLabels, trainSamples, trainLabels)
print(f'Error rate on train set: {accuracy_train * 100:.2f}%')

# Calculate error rate on test set
accuracy_test = KNNClassifier(1, trainSamples, trainLabels, testSamples, testLabels)
print(f'Error rate on test set: {accuracy_test * 100:.2f}%')
```

```
Error rate on train set: 0.00%
Error rate on test set: 21.00%
```

KNN with k=1 achieves 0% error rate on the training set due to overfitting, but this does not generalize well to the test set, resulting in lower accuracy.

1.5

In [5]:

```
k_values = range(1, 21)
train_errors = []
test_errors = []

for k in k_values:
    train_error = KNNClassifier(k, trainSamples, trainLabels, trainSamples, trainLabels)
    test_error = KNNClassifier(k, trainSamples, trainLabels, testSamples, testLabels)

    train_errors.append(train_error)
    test_errors.append(test_error)

# Plotting train and test errors rate
plt.figure(figsize=(10, 6))
plt.plot(k_values, train_errors, marker='o', label='Train Error')
plt.plot(k_values, test_errors, marker='o', label='Test Error')
plt.title('Train and Test Errors vs. k')
```

```
plt.xlabel('k')
plt.ylabel('Error Rate')
plt.xticks(k_values)
plt.legend()
plt.grid(True)
plt.show()
```

Test error in k-NN classifiers typically decreases initially as k increases from small values, but beyond an optimal point, further increasing k can lead to higher test error due to underfitting.

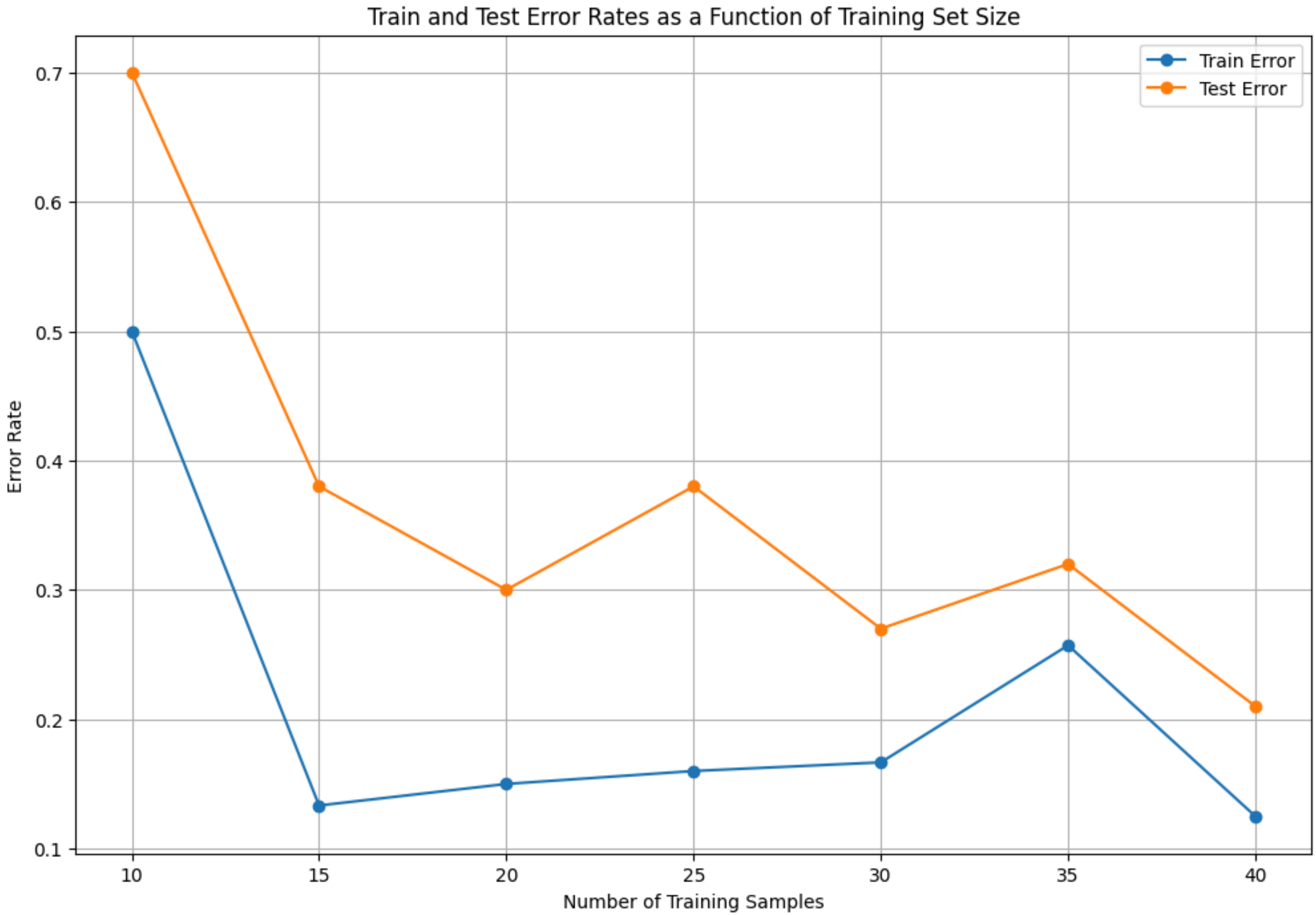
For instance, in cases where k is too large, the model may generalize poorly due to oversmoothing, ignoring local patterns and leading to increased bias and higher test error.

1.6

- Decrease in Error Rates: As the sizes of the training and test sets increase, we expect the error rates to decrease. This is because a larger training set usually provides a better representation of the data distribution, improving the model's performance.
- Impact of k: Using k = 10 in the KNN algorithm may introduce some bias, leading to better generalization. This means the model might not overfit the training data as much as with smaller values of k, but it could also smooth out some variations in the data.
- Variation in Graph Behavior: Due to the bias introduced by k = 10, there may be variations in the error rates across different training sets. This variability is expected because different random samples might lead to slight differences in model performance.

In [7]:
```
# Generate a fixed test set of size 100
testSamples, testLabels, _ = genRandNormal(100)

# List of training sizes
train_sizes = [10,15, 20, 25, 30, 35, 40]
test_errors = []
train_errors = []

for size in train_sizes:
    sampledTrainSamples, sampledTrainLabels, _ = genRandNormal(size)
    train_error = KNNClassifier(10, sampledTrainSamples, sampledTrainLabels, sampledTrainSamples, sampledTrainLabels)
    train_errors.append(train_error)
    test_error = KNNClassifier(10, sampledTrainSamples, sampledTrainLabels, testSamples, testLabels)
    test_errors.append(test_error)

plt.figure(figsize=(12, 8))
plt.plot(train_sizes, train_errors, label='Train Error', marker='o')
plt.plot(train_sizes, test_errors, label='Test Error', marker='o')
```

```
plt.xlabel('Number of Training Samples')
plt.ylabel('Error Rate')
plt.title('Train and Test Error Rates as a Function of Training Set Size')
plt.legend()
plt.grid(True)
plt.show()
```



As we can see the graph behaves as we expected.

1.7

As stated in 1.6 we expected some variation in the plot between sets sizes and execution intervals, due to k=10 which contributes to bias in the classification process and different seize data sets we can't predict correctly the plot for the next 5 - 10 execution intervals mainly caused by the randomness in generating the data set as magnified by the small set size. Although randomness is involved we can expect and in fact see that the trend in error rate is present in all plots generated for all set sizes, the error rate starts high for the smaller set sizes and decreases as the set sizes increase.

1.8

We can suggest a variation for the standard known classification process that takes into consideration the neighbors distance:

$$w(x, x^*) = \frac{1}{\|x^* - x\|_2}$$

where x^* is the vector of the sample that we want to classify, and x is a sample from the training set.

The output of the classifier is the label with the highest "weighted vote" sum among the K nearest neighbors, where the distance is measured using the Euclidean distance.

$$y^* = \underset{\text{label}}{\arg\max} \sum_{(x,y)\in N_k} \frac{w(x^*, x)}{\sum_{i=1}^{n} w(x^*, x)} \cdot \mathbb{I}(\text{label} = y)$$

$$\text{y}^* \text{ is the label for } \text{x}^*$$

שאלה 3

סעיף ו':

נתון $S=\{(x_i,y_i)\}_{i=1}^{m}$

צ"ל: א. לכל $w_1,w_2\in\mathbb{R}^d$ קיים $w\in\mathbb{R}^d$ כך ש: $p_1=p,\ p_2=1-p$

ב. לכל $w\in\mathbb{R}^d$ קיימים $w_1,w_2\in\mathbb{R}^d$ כך ש: $p_1=p,\ p_2=1-p$

יהיו w_1,w_2, נבחר $w=w_1-w_2$, אזי מתקיים:

$$p=P_w(y_i=1\mid x_i)=\frac{e^{w^Tx_i}}{1+e^{w^Tx_i}}\overset{\text{הצבת } w}{=}\frac{e^{w_1^Tx_i-w_2^Tx_i}}{1+e^{w_1^Tx_i-w_2^Tx_i}}=\frac{\frac{e^{w_1^Tx_i}}{e^{w_2^Tx_i}}}{1+\frac{e^{w_1^Tx_i}}{e^{w_2^Tx_i}}}=$$

$$=\frac{\frac{e^{w_1^Tx_i}}{\cancel{e^{w_2^Tx_i}}}}{\frac{e^{w_2^Tx_i}}{\cancel{e^{w_2^Tx_i}}}+\frac{e^{w_1^Tx_i}}{\cancel{e^{w_2^Tx_i}}}}=\frac{e^{w_1^Tx_i}}{e^{w_2^Tx_i}+e^{w_1^Tx_i}}=\frac{e^{w_1^Tx_i}}{\sum_{j=1}^{2}e^{w_j^Tx_i}}=p_1$$

$$1-p=P_w(y_i=0\mid x_i)=1-\frac{e^{w^Tx_i}}{1+e^{w^Tx_i}}=\frac{1+\cancel{e^{w^Tx_i}}-\cancel{e^{w^Tx_i}}}{1+e^{w^Tx_i}}=\frac{1}{1+e^{w^Tx_i}}\overset{\text{הצבת } w}{=}$$

$$=\frac{1}{1+e^{w_1^Tx_i-w_2^Tx_i}}=\frac{1}{1+\frac{e^{w_1^Tx_i}}{e^{w_2^Tx_i}}}=\frac{\frac{e^{w_2^Tx_i}}{\cancel{e^{w_2^Tx_i}}}}{\frac{e^{w_2^Tx_i}}{\cancel{e^{w_2^Tx_i}}}+\frac{e^{w_1^Tx_i}}{\cancel{e^{w_2^Tx_i}}}}=\frac{e^{w_2^Tx_i}}{e^{w_2^Tx_i}+e^{w_1^Tx_i}}=$$

$$=\frac{e^{w_2^Tx_i}}{\sum_{j=1}^{2}e^{w_j^Tx_i}}=p_2$$

כלומר, קיבלנו כי לכל $w_1,w_2\in\mathbb{R}^d$ קיים $w\in\mathbb{R}^d$ כך ש: $p_1=p,\ p_2=1-p$

כנדרש ■ א'

יהי W, נגדיר $W_1=W$, $W_2=0$, ואזי מתקיים:

$$p_1 = P_w(y_i=1|x_i) = \frac{e^{w_1^T x_i}}{\sum_{j=1}^{2} e^{w_j^T x_i}} = \frac{e^{w_1^T x_i}}{e^{w_1^T x_i}+e^{w_2^T x_i}} \underset{\text{הצבת } w_1,w_2}{=} \frac{e^{w^T x_i}}{e^{w^T x_i}+e^{\bar{0}^T x_i}} = \frac{1}{1+e^{w^T x_i}} = p$$

$$p_2 = P_w(y_i=0|x_i) = \frac{e^{w^T x_i}}{\sum_{j=1}^{2} e^{w_j^T x_i}} = \frac{e^{w_1^T x_i}}{e^{w_1^T x_i}+e^{w_2^T x_i}} \underset{\text{הצבת } w_1,w_2}{=} \frac{e^{w^T x_i}}{e^{w^T x_i}+e^{\bar{0}^T x_i}} =$$

$$= \frac{1}{1+e^{w^T x_i}} = \frac{1+e^{w^T x_i}-e^{w^T x_i}}{1+e^{w^T x_i}} = 1-\frac{e^{w^T x_i}}{1+e^{w^T x_i}} = 1-p$$

כלומר, קיבלנו כי לכל $W\in\mathbb{R}^d$ קיימים $w_1,w_2\in\mathbb{R}^d$ כך ש:
$p_1=p$, $p_2=1-p$ כנדרש ■ מ'

סעיף ב:

נמצא את בעיית השוואות המשתנים השקולה לבעיית האופטימיזציה:

$$w^* = \underset{w}{\operatorname{argmax}} \left\{ \sum_{i=1}^{m} \log \left(P_w \left(Y_i = y \mid x_i \right) \right) \right\}$$

לפי סעיף 1, מתקיים: $P_w(Y_i = k \mid x_i) = \frac{e^{w_k^T x_i}}{\sum_{j=1}^{k} e^{w_j^T x_i}} = P_k$ לכן, על מנת

למצוא k וקטורי משקולות מתאימים עלינו לפתור את בעיית האופטימיזציה

$$\underset{k \in \{1,\dots,K\}}{\operatorname{argmax}} \left(\frac{e^{w_k^T x_i}}{\sum_{j=1}^{k} e^{w_j^T x_i}} \right)$$

כלומר מציאת אנ"מ מתאימים.

log הוא פונקציה מונוטונית עולה, ולכן שקול למציאת הערך שממקסם את log הפונקציה:

$$\log\left(P(Y_i = k \mid x_i)\right) = \log \left(\frac{e^{w_k^T x_i}}{\sum_{j=1}^{k} e^{w_j^T x_i}} \right) = \log\left(e^{w^T x_i}\right) - \log \left(\sum_{j=1}^{k} e^{w_j^T x_i} \right)$$

$$= w_k^T x_i - \log \left(\sum_{j=1}^{k} e^{w_j^T x_i} \right)$$

פונקציה זו מייצגת דוגמה יחידה מסט האימון. נרצה לחשב עבור m דוגמאות:

$$\mathcal{L}_S(w) = \sum_{i=1}^{m} \left(w^T x_i - \log \left(\sum_{j=1}^{k} e^{w_j^T x_i} \right) \right)$$

נגזור ונשווה ל-0 עבור כל אחת מ-k המשוואות:

$$\forall\, k = 1, \dots, K: \frac{\partial \mathcal{L}_S(w)}{\partial w_k} = \sum_{i=1}^{m} 1\left[Y_i = k\right] x_i - \sum_{i=1}^{m} \left(\frac{e^{w_k^T x_i}}{\sum_{j=1}^{k} e^{w_j^T x_i}} \right) x_i = 0$$

$$\Rightarrow \sum_{i=1}^{m} 1\left[Y_i = k\right] x_i = \sum_{i=1}^{m} \left(\frac{e^{w_k^T x_i}}{\sum_{j=1}^{k} e^{w_j^T x_i}} \right) x_i :$$

בעיית השוואת המשתנים השקולה לבעיית האופטימיזציה

סעיף 3:

a. קיימים 3 וקטורי משקולת w_1, w_2, w_3 מכיוון שברצוננו למצוא 3 על-מישורים מפרידים כל אחד מ- Y הלייבלים, כאשר נבחר שקיימת התפלגות מולטינומית על 3 על-המישורים כך ש:

$$P_w(y_i = k \mid x_i) = \frac{e^{w_k^T x_i}}{\sum_{j=1}^{k} e^{w_j^T x_i}} = P_{ik}, \quad k \in \{1,2,3\}$$

b. מתקיים כי $x \in \mathbb{R}^2$ בעוד $w_i \in \mathbb{R}^3$ מכיוון שאנו מוסיפים לוקטור w_i כניסה נוספת המייצגת את גורם ההטיה (bias)

c. נתון

$$w_1 = (8, -2.5, 2)$$
$$w_2 = (2, 0.5, -1.5)$$
$$w_3 = (-10, 2, -0.5)$$

לכל x_i, $i=1,2,3$, נוסיף כניסה x_0 כפוקטור עם ערך 1 על מנת לייצג את ההטיה.
מתקיים:

$x_1 = (1,1,8)$:

$$\sum_{j=1}^{3} e^{w_j^T x_1} = \underbrace{e^{8\cdot1+(-2.5)\cdot1+2\cdot8}}_{e^{w_1^T x_1}} + \underbrace{e^{2\cdot1+0.5\cdot1+(-1.5)\cdot8}}_{e^{w_2^T x_1}} + \underbrace{e^{-10\cdot1+2\cdot1+(-0.5)\cdot8}}_{e^{w_3^T x_1}} =$$

$$= e^{21.5} + e^{-9.5} + e^{-12}$$

ולכן:

$$P_w(y_1 = 0 \mid x_1) = \frac{e^{w_1^T x_1}}{\sum_{j=1}^{3} e^{w_j^T x_1}} = \frac{e^{21.5}}{e^{21.5}+e^{-9.5}+e^{-12}} \approx 1$$

$$P_w(y_1 = 1 \mid x_1) = \frac{e^{w_1^T x_1}}{\sum_{j=1}^{3} e^{w_j^T x_1}} = \frac{e^{-9.5}}{e^{21.5}+e^{-9.5}+e^{-12}} \approx 0$$

$$P_w(y_1 = 2 \mid x_1) = \frac{e^{w_1^T x_1}}{\sum_{j=1}^{3} e^{w_j^T x_1}} = \frac{e^{-12}}{e^{21.5}+e^{-9.5}+e^{-12}} \approx 0$$

$$\Rightarrow y_1 = 0$$

$x_2 = (1, 6, -2):$

$$\sum_{j=1}^{3} e^{W_j^T x_2} = \underbrace{e^{8\cdot 1 + (-2.5)\cdot 6 + 2\cdot(-2)}}_{e^{W_1^T x_2}} + \underbrace{e^{2\cdot 1 + 0.5\cdot 6 + (-1.5)\cdot(-2)}}_{e^{W_2^T x_2}} + \underbrace{e^{-10\cdot 1 + 2\cdot 6 + (-0.5)\cdot(-2)}}_{e^{W_3^T x_2}} =$$

$$= e^{-11} + e^{8} + e^{3}$$

$$p_W(y_2 = 0 \mid x_2) = \frac{e^{W_1^T x_2}}{\sum_{j=1}^{3} e^{W_j^T x_2}} = \frac{e^{-11}}{e^{-11} + e^{8} + e^{3}} \approx 0$$

$$p_W(y_2 = 1 \mid x_2) = \frac{e^{W_1^T x_2}}{\sum_{j=1}^{3} e^{W_j^T x_2}} = \frac{e^{8}}{e^{-11} + e^{8} + e^{3}} \approx 1$$

$$p_W(y_2 = \ \mid x_2) = \frac{e^{W_1^T x_2}}{\sum_{j=1}^{3} e^{W_j^T x_2}} = \frac{e^{3}}{e^{-11} + e^{8} + e^{3}} \approx 0$$

$\Rightarrow y_2 = 1$

$x_3 = (1, 12, 4)$:

$$\sum_{j=1}^{3} e^{w_j^T x_3} = \underbrace{e^{8\cdot 1+(-2.5)\cdot 12+2\cdot 4}}_{e^{w_1^T x_3}} + \underbrace{e^{2\cdot 1+0.5\cdot 12+(-1.5)\cdot 4}}_{e^{w_2^T x_3}} + \underbrace{e^{-10\cdot 1+2\cdot 12+(-0.5)\cdot 4}}_{e^{w_3^T x_3}} =$$

$$= e^{-14} + e^{2} + e^{12}$$

$$p_w(y_3 = 0 \mid x_3) = \frac{e^{w_1^T x_3}}{\sum_{j=1}^{3} e^{w_j^T x_3}} = \frac{e^{-14}}{e^{-14}+e^{2}+e^{12}} \approx 0$$

$$p_w(y_3 = 1 \mid x_3) = \frac{e^{w_1^T x_3}}{\sum_{j=1}^{3} e^{w_j^T x_3}} = \frac{e^{2}}{e^{-14}+e^{2}+e^{12}} \approx 0$$

$$p_w(y_3 = 2 \mid x_3) = \frac{e^{w_1^T x_3}}{\sum_{j=1}^{3} e^{w_j^T x_3}} = \frac{e^{12}}{e^{-14}+e^{2}+e^{12}} \approx 1$$

$\Rightarrow y_3 = 2$