ML_HW5

Mohammad Javad Pesarakloo
810100103

July 17, 2024

Question 1

A

We denote mean of class one with m_1 and mean of class two with m_2.First we need to find m_1 and m_2:

$$m_1 = \begin{pmatrix}3\\6\end{pmatrix}, m_2 = \begin{pmatrix}8.4\\7.6\end{pmatrix}$$

Now we subtract means from their corresponding samples:

$$X_1 - m_1 = \{\begin{pmatrix}1\\-2.6\end{pmatrix}, \begin{pmatrix}-1\\0.4\end{pmatrix}, \begin{pmatrix}-1\\-0.6\end{pmatrix}, \begin{pmatrix}0\\2.4\end{pmatrix}, \begin{pmatrix}1\\0.4\end{pmatrix}\}$$

$$X_2 - m_2 = \{\begin{pmatrix}0.6\\2.4\end{pmatrix}, \begin{pmatrix}-2.4\\0.4\end{pmatrix}, \begin{pmatrix}0.6\\-2.6\end{pmatrix}, \begin{pmatrix}-0.4\\-0.6\end{pmatrix}, \begin{pmatrix}1.6\\0.4\end{pmatrix}\}$$

Now we can calculate S_1 and S_2:

$$S_1^2 = \begin{pmatrix}1 & -2.6\\-2.6 & 6.76\end{pmatrix} + \begin{pmatrix}1 & -0.4\\-0.4 & 0.16\end{pmatrix} + \begin{pmatrix}1 & 0.6\\0.6 & 0.36\end{pmatrix} + \begin{pmatrix}0 & 0\\0 & 5.76\end{pmatrix} + \begin{pmatrix}1 & 0.4\\0.4 & 0.16\end{pmatrix}$$

$$= \begin{pmatrix}4 & -2\\-2 & 13.2\end{pmatrix}$$

$$S_2^2 = \begin{pmatrix}0.36 & 1.44\\1.44 & 5.76\end{pmatrix} + \begin{pmatrix}5.76 & -0.96\\-0.96 & 0.16\end{pmatrix} + \begin{pmatrix}0.36 & -1.56\\-1.56 & 6.76\end{pmatrix} + \begin{pmatrix}0.16 & 0.24\\0.24 & 0.36\end{pmatrix} + \begin{pmatrix}2.56 & 0.64\\0.64 & 0.16\end{pmatrix}$$

$$= \begin{pmatrix}9.2 & -0.2\\-0.2 & 13.2\end{pmatrix}$$

And now S_w can be calculated:

$$S_w = \begin{pmatrix}4 & -2\\-2 & 13.2\end{pmatrix} + \begin{pmatrix}9.2 & -0.2\\-0.2 & 13.2\end{pmatrix} = \begin{pmatrix}13.2 & -2.2\\-2.2 & 26.4\end{pmatrix}$$

B

$$m_1 - m_2 = \begin{pmatrix} 3 & 3.6 \end{pmatrix} - \begin{pmatrix} 8.4 & 7.6 \end{pmatrix} = \begin{pmatrix} -5.4 & -4 \end{pmatrix}$$

$$S_B = \begin{pmatrix} -5 & -4 \end{pmatrix} \begin{pmatrix} -5.4 \\ -4 \end{pmatrix} = \begin{pmatrix} 29.16 & 21.6 \\ 21.6 & 16 \end{pmatrix}$$

C

$$A = S_w^{-1} S_B$$

$$S_w^{-1} = \frac{1}{343.64} \begin{pmatrix} 26.4 & 2.2 \\ 2.2 & 13.2 \end{pmatrix} = \begin{pmatrix} 0.076 & 0.006 \\ 0.006 & 0.038 \end{pmatrix}$$

$$\Rightarrow A = \begin{pmatrix} 2.34 & 1.73 \\ 0.99 & 0.73 \end{pmatrix}$$

$$\Rightarrow A - \lambda I \begin{pmatrix} 2.34 - \lambda & 1.73 \\ 0.93 & 0.73 - \lambda \end{pmatrix}$$

$$|A - \lambda I| = (\lambda - 2.34) \times (\lambda - 0.73) = 0$$

$$\Rightarrow \lambda^2 - 3.07\lambda - 0.004 = 0$$

$$\lambda_1 = -0.001, \lambda_2 = 3.07$$

Thus the greatest eigen value is **3.07**.

Question 2

A

Model selection is the process of selecting among many candidate models for a modeling problem while model assessment is the process of estimating the model's prediction error (generalization error) over an independent new data sample.The reason for which we use model selection is that each model have multiple hyper parameters that can affect the performance of the model;in this process we find the best hyper parameters to get the best modeling result.Also this process can be accross different types of machine learning models such as SVM, Logistic Regression, etc.

B

In process of model selection, we need sufficient data while in most cases, we don't have this amount of data. Instead, there are two main classes of techniques to approximate the ideal case of model selection:

1. **Probabilistic Measures** : Choose a model via in-sample error and complexity.There are four major probabilistic measures:
   - Akaike Information Criterion (AIC)
   - Bayesian Information Criterion (BIC)
   - Minimum Description Length (MDL)
   - Structural Risk Minimization (SRM)
2. **Resampling Methods** : Choose a model via estimated out-of-sample error.In this mothod, we seek to estimate the performance of the model by splitting the training dataset into sub train and test sets.This process may be repeated multiple times and the mean performance accross each trail is reported.There exists three common resampling methods:
   - Random train-test splits.
   - Cross-Validation (k-fold, LOOCV, etc.)
   - Bootstrap

C

When number of data samples is not sufficient, it is known that training error is optimistically biased and therefore is not a good basis for choosing a model.The performance can be penalized based on how optimistic the training error is believed to be. This is typically achieved using algorithm-specific methods, often linear, that penalize the score based on the complexity of the model explained in the previous subsections.

Question 3

$$\theta = \{\lambda_1, \lambda_2, \alpha\}$$

First we define hidden variable z such that:

$$z_i = \begin{cases} 0 & x_i comes from the first component \\ 1 & otherwise \end{cases}$$

$$p(x_i, z_i|\theta) = (\alpha\lambda_1 e^{-\lambda_1 x_i})^{1-z_i}((1-\alpha)\lambda_2 e^{-\lambda_2 x_i})^{z_i}$$

$$\xrightarrow{\log} \log p(x_i, z_i|\theta) = (1-z_i)(\log\alpha + \log\lambda_1 - \lambda_1 x_i) + z_i(\log(1-\alpha) + \log\lambda_2 - \lambda_2 x_i)$$

$$\log\_complete\_likelihood = \sum_{i=1}^{n}(1-z_i)(\log\alpha+\log\lambda_1-\lambda_1 x_i)+z_i(\log(1-\alpha)+\log\lambda_2-\lambda_2 x_i)$$

$$Q(\theta,\theta^t) = E[log\_complete\_likelihood|\theta]$$

$$= \sum_{i=1}^{n}(1-E[z_i])(\log\alpha+\log\lambda_1-\lambda_1 x_i)+E[z_i](\log(1-\alpha)+\log\lambda_2-\lambda_2 x_i)$$

$$\Rightarrow E[z_i] = p(z_i=1|x_i,\theta^t) = \frac{p(x_i|z_i=1,\theta^t)p(z_i=1|\theta^t)}{p(x_i|\theta^t)}$$

$$= \frac{\lambda_2^t e^{-\lambda_2^t x_i}(1-\alpha^t)}{\alpha^t\lambda_1^t e^{-\lambda_1^t x_i}+(1-\alpha^t)\lambda_2^t e^{-\lambda_2^t x_i}}$$

$$= \gamma_i^t$$

Now that we have the expected likelihood, we can find the updating rule for parameters:

$$\frac{\partial Q}{\partial\alpha} = \sum_{i=1}^{n}(1-\gamma_i^t)\frac{1}{\alpha}-\frac{\gamma_i^t}{1-\alpha} = 0$$

$$\Rightarrow (1-\alpha)\sum_{i=1}^{n}(1-\gamma_i^t) = \alpha\sum_{i=1}^{n}\gamma_i^t$$

$$\Rightarrow \alpha = 1-\frac{\sum_{i=1}^{n}\gamma_i^t}{n}$$

$$\frac{\partial Q}{\partial\lambda_1} = \sum_{i=1}^{n}(1-\gamma_i^t)(\frac{1}{\lambda_1}-x_i) = 0$$

$$\Rightarrow \lambda_1 = \frac{n-\sum_{i=1}^{n}\gamma_i^t}{\sum_{i=1}^{n}(1-\gamma_i^t)x_i}$$

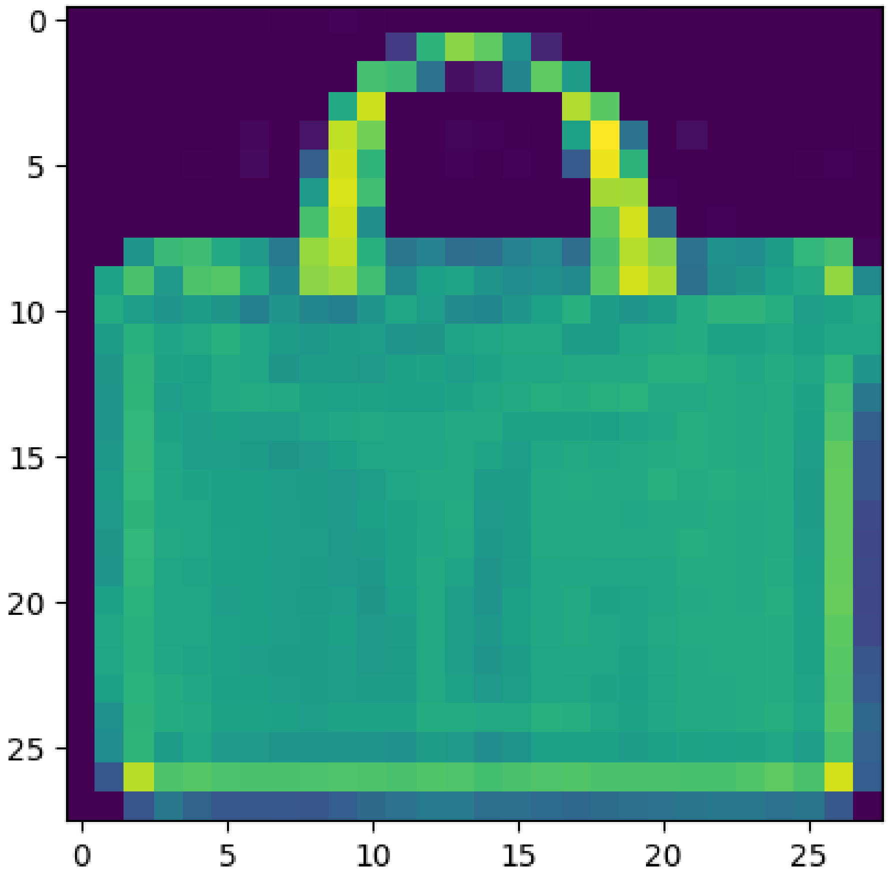
And with the same methodology, we have:

$$\lambda_2 = \frac{\sum_{i=1}^{n}\gamma_i^t}{\sum_{i=1}^{n}\gamma_i^t x_i}$$

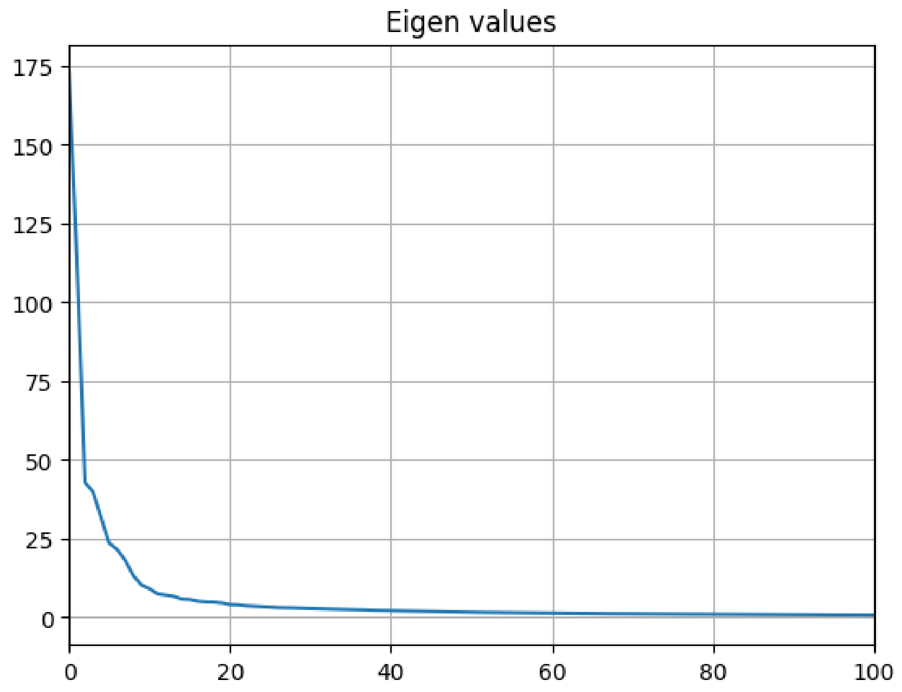
Question 4

Question 5

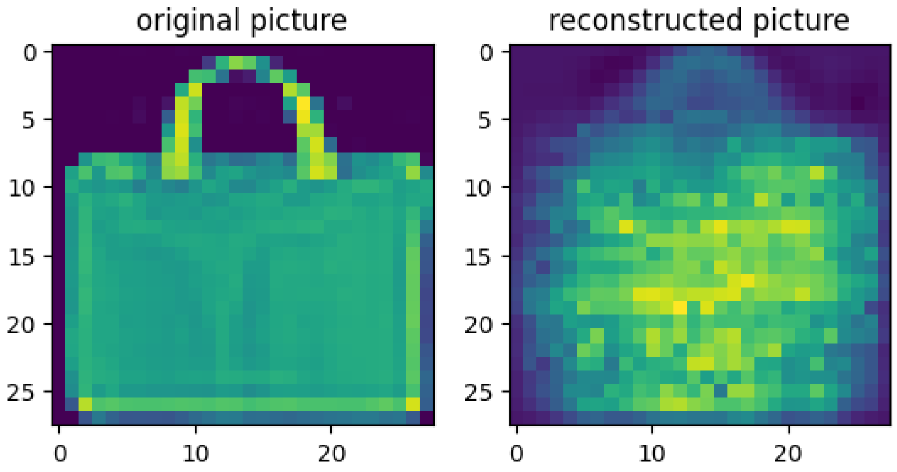
Codes related to this question are stored in Q5.ipynb notebook file.As suggested in the description, selecting a random index and visualizing its picture, we have:



Now in order to standardize the images to make them feedable to PCA models, we can use **StandardScalar**.The covariance matrix can also be calculated using the **np.cov** built-in function of numpy.In the next step, eigen values and their corresponding eigen vectors can also be calculated using the **np.linalg.eig** built-in function of numpy.Visualizing the eigen values, we get the following figure:

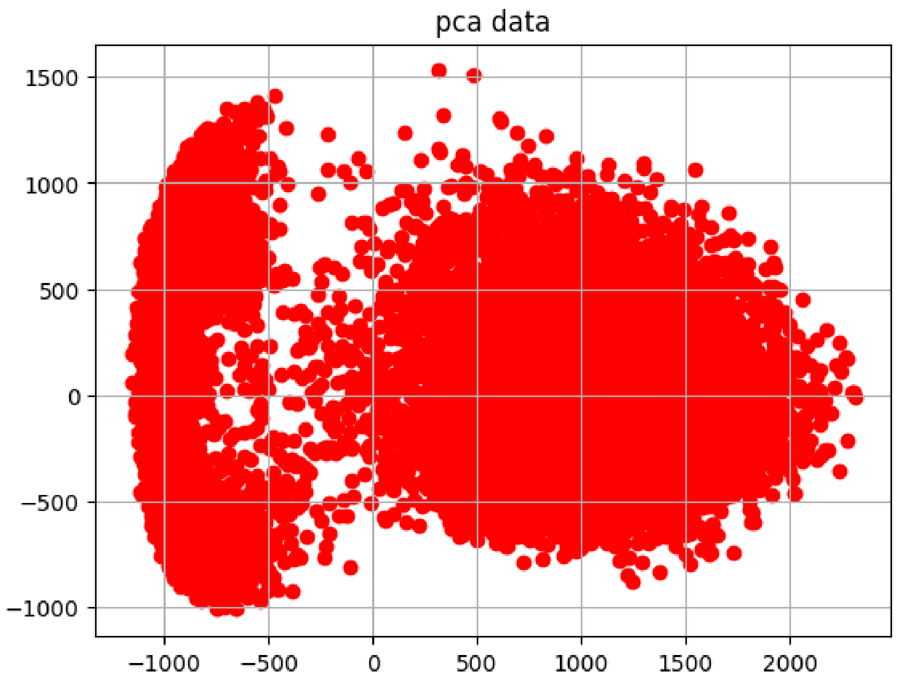


As we see in the above figure, most of the variance is held in the first 25 components;thus we can compress the pictures using these components. Then by reconstructing the picture and comparing it with the original one, we have the following figure:

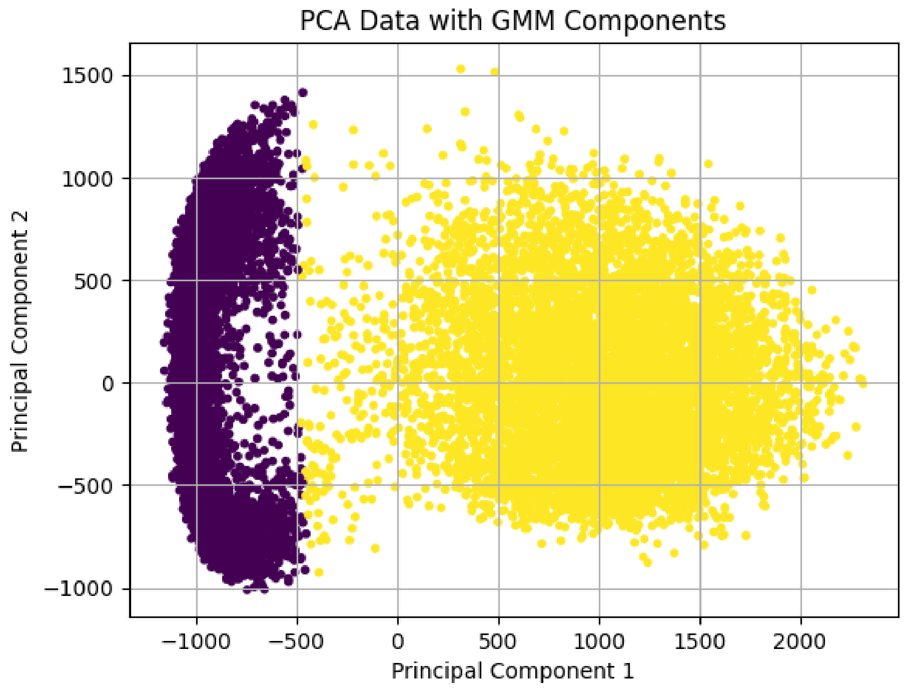


Question 6

Codes related to this question are stored in Q6.ipynb notebook file.At first, we can load data and extract the samples related to 0 and 1.Then we can concatenate these two classes and apply a principal component analysis on them to be able to visualze them;we have the following figure:

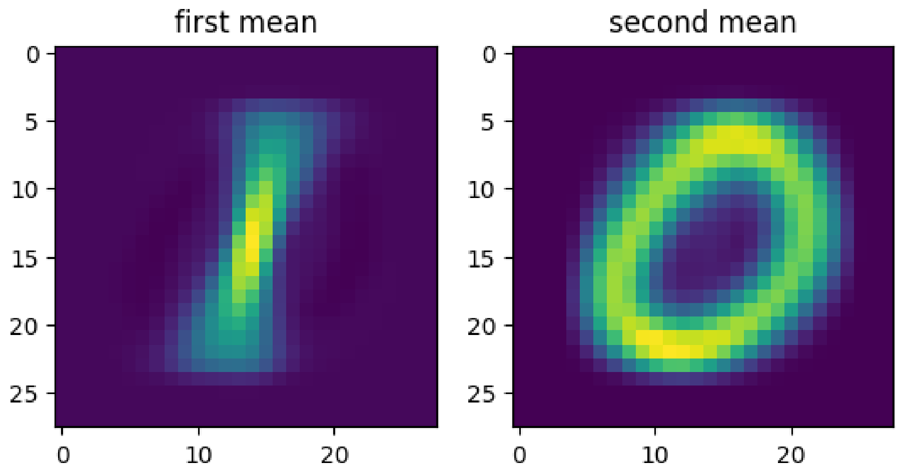


Now we can fit a gaussian mixture model with 2 components which gives us the following figure:

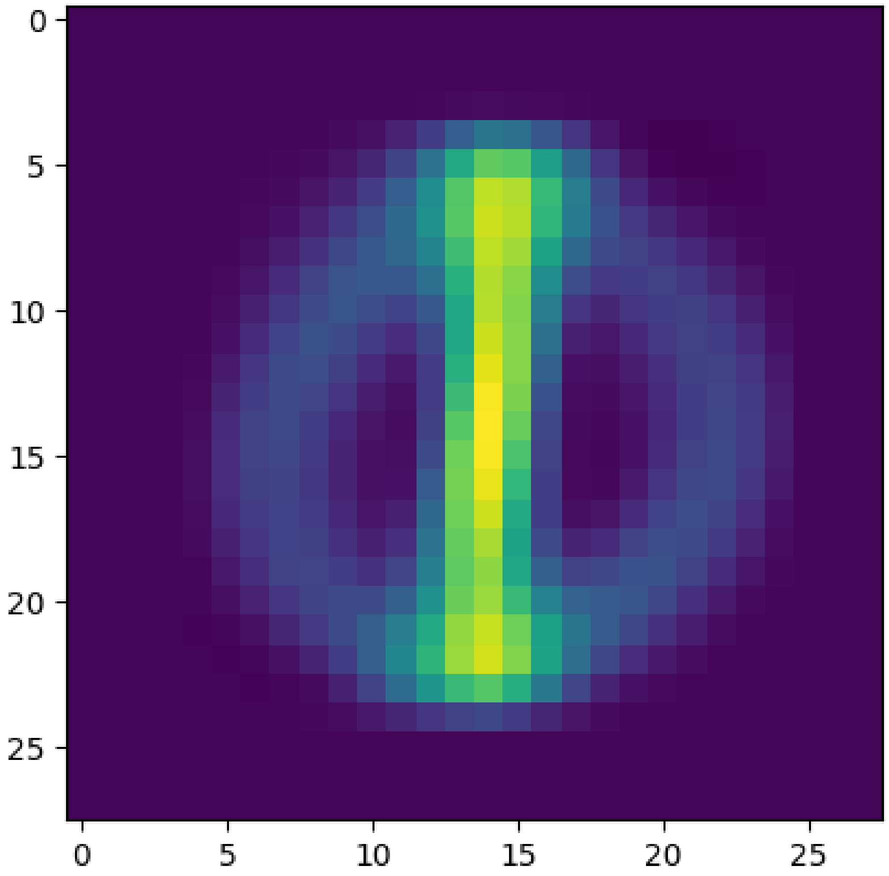


As asked in the description, difference between mean of two components is : **1915.67**.

Now inverse transforming the data to the original dimensions, we expect that means have high resemblance to components 0 and 1 which we see in the following figure:



Now if we find the samples whose probability of belonging to each component have low differnce, we expect that points to be similar to both classes.Doing this analysis, we get the following figure:



Now we are asked to analyze the differnce between means of components for all different pairs of classes.using the following code, we found the highest and the shortest differnce possbile:

```
    max_distance = 0
    min_distance = float('inf')
    max_pair = None
    min_pair = None

    for class1, class2 in combinations(range(10), 2):
        data_class1 = X[Y == class1]
        data_class2 = X[Y == class2]
        data = np.concatenate([data_class1, data_class2])

        pca = PCA(n_components=2)
        pca_data = pca.fit_transform(data)

        gmm = GaussianMixture(n_components=2)
        gmm.fit(pca_data)
        distance = np.linalg.norm(gmm.means_[0] - gmm.means_
            [1])

        if distance > max_distance:
            max_distance = distance
            max_pair = (class1, class2)

        if distance < min_distance:
            min_distance = distance
            min_pair = (class1, class2)

    print(f'Pair with highest distance: {max_pair}, Distance
        : {max_distance}')
    print(f'Pair with shortest distance: {min_pair},
        Distance: {min_distance}')
```

which gives the following output:

1. Pair with highest distance: (0, 1), Distance: 1915.66

2. Pair with shortest distance: (4, 9), Distance: 979.84

The above results make perfect sense because we expect the distance to be maximized when two classes do not have high resemblance like classes 0 and 1.On the other hand, 4 and 9 look so much like each other which is why we see the shortest distance between these two.