Data Science: EN.553.436/636
Midterm 2

Name:
JHED:

Instructions

- **Write your name.**
- You will have 70 minutes to complete this exam.
- The exam has 3 problems, each having multiple parts.
- For short answers questions, a few words will usually suffice. No need to write an essay.
- You are allowed a single (front and back) note sheet.
  - You must turn in the note sheet along with your exam.
- No notes (other than your note sheet), books, calculators, phones, laptops, or any sort of electronic devices are allowed during the exam.
- Do not look turn the page before the exam begins. Doing so will result in a penalty.
- Good Luck!

Problem 1 (23 pts)

a) (4 pts)

Consider the binary AND function whose truth table is below.

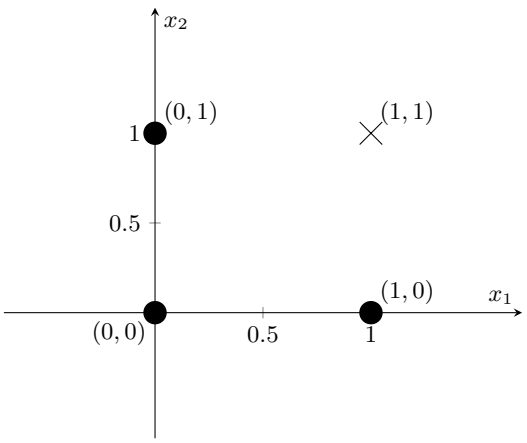
x_1	x_2	Class
0	0	0
0	1	0
1	0	0
1	1	1

In the space below construct an example of a decision tree that could be used to classify this 2-feature function. Include splitting conditions. You can consider the data (values of x_1, x_2) as either categorical or numeric. Be sure that the splitting functions for your tree are consistent with your assumption (i.e. $>, <, \geq, \leq$ for numeric or $=, \neq$ for categorical).

1.a) answer:

b) (3 pts)

Assume that the data was numeric (i.e. $\in \mathbb{R}$). Sketch and shade the decision boundaries of the tree from (a) on the plot below.

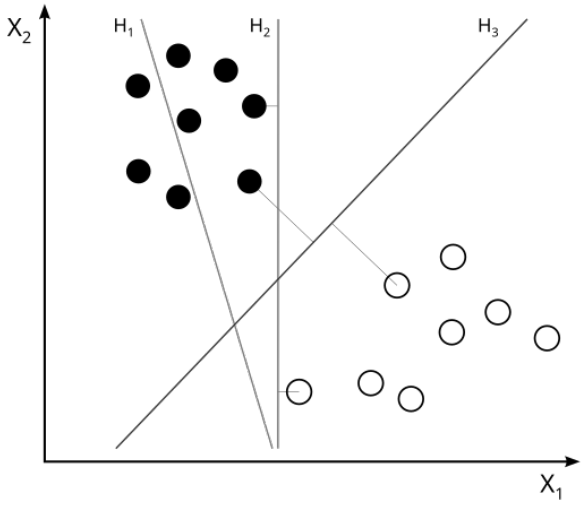


c) (3 pts)

Decision trees can have varying depths. Explain some of the pros and cons between different depths.

1.c answer:

d) (3 pts)



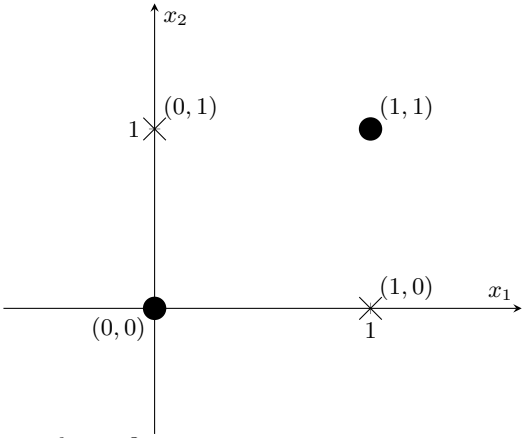
Which of the lines (H_1, H_2, H_3) above was most likely created by a Support Vector Machine classifier? Why?

1.d) answer:

e) (4 pts)

Consider the binary exclusive or function (XOR) plotted below and it's corresponding truth table.

x_1	x_2	Class
0	0	0
0	1	1
1	0	1
1	1	0



Would basic SVM work to classify this function? Why or why not?

1.e) answer:

f) (6 pts)

Describe the main differences in the assumptions between Naive Bayes, Linear Discriminant Analysis, and Quadratic Discriminant Analysis.

1.f) answer:

Problem 2 (11 pts)

a) (4 pts)

Recall homework 7, one of the problems had you fill in a function to calculate euclidean distance for use with KMeans. Why are we interested in euclidean distance with regards to the KMeans algorithm?

2.a) answer:

b) (3 pts)

Why is train-test splitting not necessary for clustering?

2.b) answer:

c) (4 pts)

Relative to KMeans, GMMs are generally considered the more "flexible" algorithm. Explain why this is.

2.c) answer:

Problem 3 (16 pts)

Bob is trying to create a program that interacts with websites. He finds that there is a new type of CAPTCHA which asks you to identify different clusters within a figure. Bob wants to come up with an algorithm to do the verification for him. To do so he collects some pictures from the verification and converts them into point cloud data. The code to load the data and the plotted data is below.

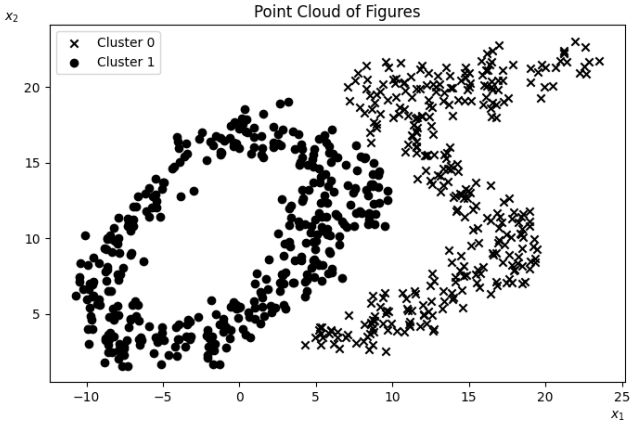
```
data = np.loadtxt("figures.csv", delimiter=",")

x = data[:, :2] # The data is a set of points in a 2-dimensional plane.
y = data[:, 2] # The label indicates which cluster the point belongs to.

print(data.shape)
print(x.shape)
print(y.shape)

plot(....) # For the sake of space we will not include every line for plotting
```

```
(711, 3)
(711, 2)
(711,)
```



a) (12 pts)

Bob now decides to try out spectral embedding. His attempt his below. Unfortunately, he is awful about naming functions and often forgets their purpose. Based on the code below explain the purpose of *function1, function2, function3*. Describe what each function is designed to accomplish.

```
def rbf(a, b, l):
    return np.exp(-np.sum((a - b) ** 2) / (2 * l ** 2))

def function1(x, l):
    n = len(x)
    aff_matrix = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            aff_matrix[i, j] = rbf(x[i], x[j], l)
    return aff_matrix

def function2(aff_matrix):
    laplacian = np.copy(aff_matrix)
    np.fill_diagonal(laplacian, 0)
    w = np.sum(laplacian, axis=1)
    laplacian = np.diag(w) - laplacian
    laplacian = laplacian / w.reshape(-1, 1)
    return laplacian

def function3(laplacian):
    eigen_value, eigen_vector = np.linalg.eig(laplacian)
    return eigen_value, eigen_vector
```

3.a) answer:

function1:

function2:

function3:

b) (4 pts)

Recall the Laplacian matrix we compute in the section, $L = D - A$ where D represents the degree of the nodes and A is the adjacency of a unweighted graph (A_{ij} is either 0 or 1). Here we compute the Laplacian for general weighted graph. Please select (circle) the correct formula the Laplacian according to the function defined above.

A. $L_{ij} = \begin{cases} -A_{ij} & i \neq j, \\ \sum_{k=1,k\neq i}^{n} A_{ik} & i = j. \end{cases}$

B. $L_{ij} = \begin{cases} -\frac{A_{ij}}{\sum_{k=1,k\neq i}^{n} A_{ik}} & i \neq j, \\ 1 & i = j. \end{cases}$

C. $L_{ij} = \begin{cases} A_{ij} & i \neq j, \\ -\sum_{k=1,k\neq i}^{n} A_{ik} & i = j. \end{cases}$

D. $L_{ij} = \begin{cases} \frac{A_{ij}}{\sum_{k=1,k\neq i}^{n} A_{ik}} & i \neq j, \\ -1 & i = j. \end{cases}$