COMPSCI 371D Homework 2

Problem 0 (3 points)

Jihyeon Je (jj271), Tim Ho (th265), Rahul Prakash (rp221)

Part 1: High-Dimensional Neighborhoods

Problem 1.1 (Exam Style)

0-dimensional neighbors: $+++, ++-, +-+, +--, -++, -+-, --+, ---$

1-dimensional neighbors: $++0, +-0, -+0, --0, 0++, 0-+, +0+, -0+, 0+-, 0--, +0-, -0-$

2-dimensional neighbors: $+00, 0+0, 00+, 00-, -00, 0-0$

Total number of bins for 0-D neighbors: 8

Total number of bins for 1-D neighbors: 12

Total number of bins for 2-D neighbors: 6

Therefore, the total number of bins is 8+12+6 = 26.

There are a total of 8 0-D neighbors because there 8 bins that only touch on a single corner of B_{ijk}.

There are a total of 12 1-D neighbors because there 12 bins that only touch on a single edge of B_{ijk}.

There are a total of 6 2-D neighbors because there 6 bins that only touch on a single face of B_{ijk}.

Problem 1.2 (Exam Style)

$n(d,b) = \binom{d}{b} * 2^{d-b}$

Since there are d positions available and b corresponds to the total number of 0s per neighbor (0+-) string, $\binom{d}{b}$ accounts for all possible combinations of b 0s in d positions. This leaves d-b positions for nonzero(+ or -) entries, which can be filled in 2^{d-b} ways. This gives us $n(d,b) = \binom{d}{b} * 2^{d-b}$

$m(d) = 3^d - 1$

We want to generate a list of all possible neighbors for a given d. For each position within the 0+- notation there are three possible values either 0, +, or -. And there are d positions available in each string. Therefore the total number of neighbors is equal to 3^d. However, since we want to get rid of the case with all 0s (the bin $B_{i_1 \ldots i_d}$ itself), we subtract 1, which gives us $m(d) = 3^d - 1$.

d	$m(d)$	$n(d,0)$	$n(d,1)$	$n(d,2)$	$n(d,3)$	$n(d,4)$
1	2	2	–	–	–	–
2	8	4	4	–	–	–
3	26	8	12	6	–	–
4	80	16	32	24	8	–
5	242	32	80	80	40	10

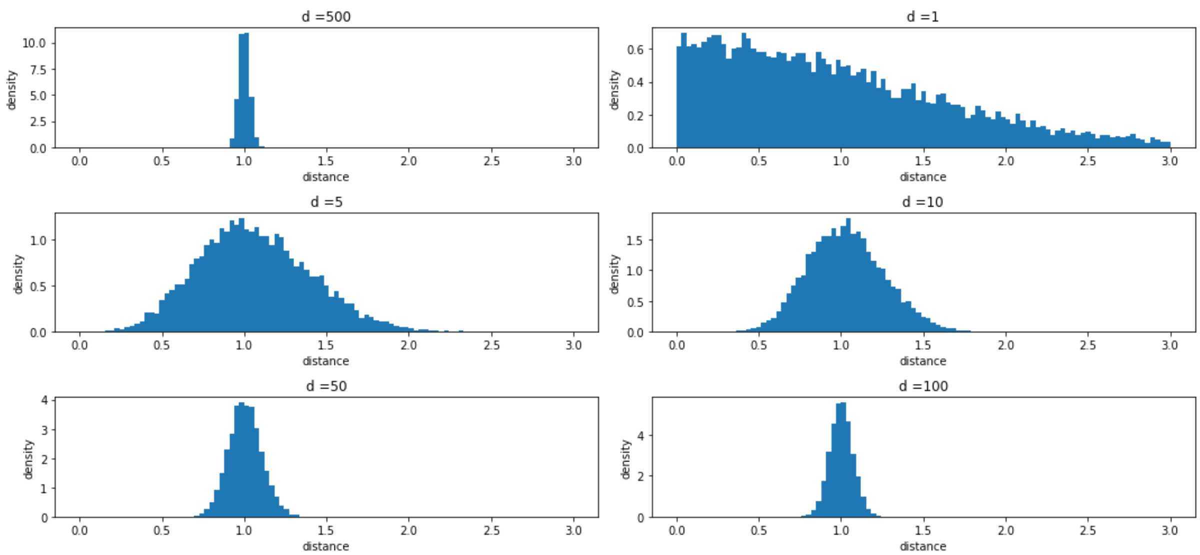
Part 2: Random Gaussian Vectors

Problem 2.1

In [1]:
```
import numpy as np
def samples(n,d):
    if d ==1:
        p = np.random.normal(0,np.sqrt(np.pi/2),(n,d))
    else:
        p = np.random.normal(0,1/np.sqrt(d-1),(n,d))
    return p
```

In [2]:
```
def distances(p):
    norm = []
    for i in range(0,len(p)):

        norm.append(np.linalg.norm(p[i]))
    return norm
```

In [3]:
```
import matplotlib.pyplot as plt
n = 10000
d = [1,5,10,50,100,500]
bins = np.linspace(0,3,100)
fig, axs = plt.subplots(3,2,figsize=(15,7))
for i, ax in enumerate(fig.axes):
    ax.hist(distances(samples(n,d[i-1])), bins = bins, density = True)
    ax.set_title(f'd ={d[i-1]}')
    ax.set_xlabel('distance')
    ax.set_ylabel('density')
fig.tight_layout()
```



As you increase d, the variance of the distance (norm) goes down thus more and more d-dimensional vectors will be clustered at the mean norm of 1.

Part 3: Linear Separability and Voronoi Diagrams

Problem 3.1 (Exam Style)

V = $\{a_1 x_1 + b = 0$, where $a_1 = 1, b \in (-5, -2]\}$

Problem 3.2 (Exam Style)

H = $\oslash$, an empty set since no such line exists.

Problem 3.3 (Exam Style)

line 1: pq, $x_1 - x_2 + 1$

line 2: qr, $x_1 + x_2 - 5$

line 3: ru, $x_1 - 5$

line 4: pu, $3x_1 - 5x_2 + 5$

Problem 3.4 (Exam Style)

a) There are a total of 5 edges

b) (2,2), (3,1)

c)

Line 1: $rs, x - 2y - 1 = 0$

Line 2: $pr, x + y - 4 = 0$

Line 3: $pq, y - 2 = 0$

Line 4: $qr, x - 2 = 0$

Line 5: $ps, x - 3 = 0$

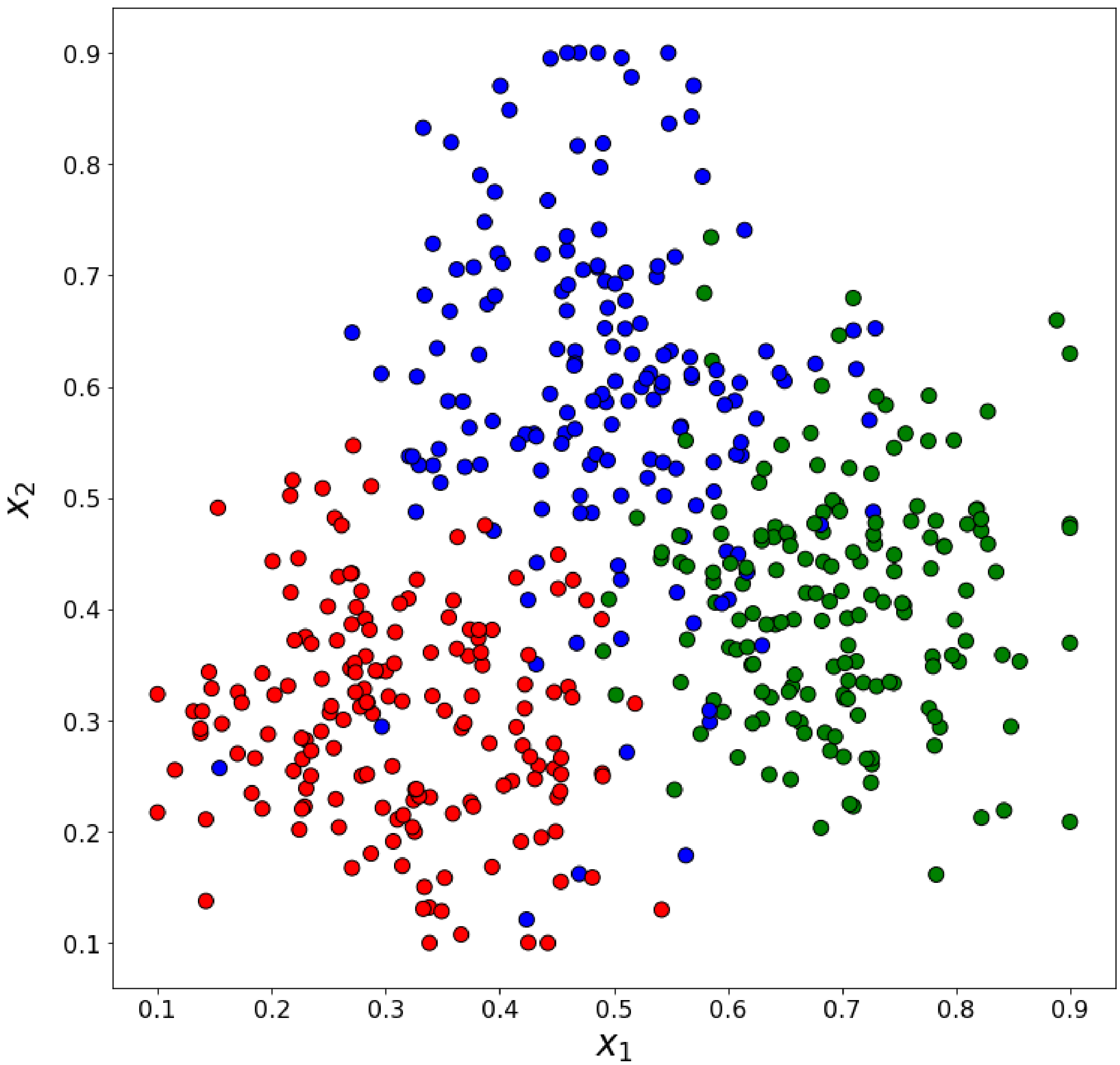
Part 4: Nearest Neighbors

In [4]:
```
from urllib.request import urlretrieve
from os import path as osp


def retrieve(file_name, semester='fall21', course='371d', homework=2):
    if osp.exists(file_name):
        print('Using previously downloaded file {}'.format(file_name))
    else:
        fmt = 'https://www2.cs.duke.edu/courses/{}/compsci{}/homework/{}
/{}'
        url = fmt.format(semester, course, homework, file_name)
        urlretrieve(url, file_name)
        print('Downloaded file {}'.format(file_name))
```

In [5]:
```
from matplotlib import pyplot as plt
from matplotlib.lines import Line2D


def decorate(x_label=None, y_label=None, title=None,
             line_width=3, font_size=24):
    for child in plt.gca().get_children():
        if isinstance(child, Line2D):
            plt.setp(child, linewidth=line_width)
    legend_handles, _ = plt.gca().get_legend_handles_labels()
    if len(legend_handles):
        plt.legend(fontsize=font_size)
    if x_label is not None:
        plt.xlabel(x_label, fontsize=font_size, labelpad=5)
    if y_label is not None:
        plt.ylabel(y_label, fontsize=font_size, labelpad=15)
    plt.xticks(fontsize=font_size//1.5)
    plt.yticks(fontsize=font_size//1.5)
    if title is not None:
        plt.title(title, fontsize=font_size)
```

In [6]:
```
import pickle
from matplotlib.colors import ListedColormap



file_name = 'data.pkl'
retrieve(file_name)
with open(file_name, 'rb') as file:
    data = pickle.load(file)

plt.figure(figsize=(12, 12))
plt.scatter(data['x'][:, 0], data['x'][:, 1], marker='o',
            s=100, edgecolor='k', c=data['y'],
            cmap=ListedColormap(('r', 'g', 'b')))
decorate(x_label='$x_1$', y_label='$x_2$')
plt.show()
```

```
Using previously downloaded file data.pkl
```



In [7]:
```
from sklearn.neighbors import NearestNeighbors


def neighbors(query_points, training_points, k):
    nn = NearestNeighbors(n_neighbors=k)
    nn.fit(training_points)
    indices = nn.kneighbors(query_points, return_distance=False)
    return indices
```

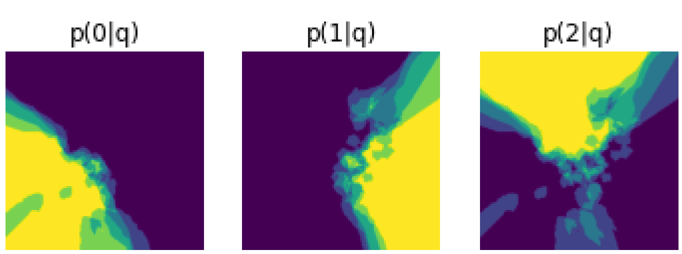
In [8]:
```
def knn(query_points, training_samples, k, summary):
    indices = neighbors(query_points, training_samples['x'], k)
    return summary(training_samples['y'][indices])
```

Problem 4.1

In [9]:
```
import numpy as np

n_grid = 200
grid_points = np.linspace(0., 1., n_grid)
xx, yy = np.meshgrid(grid_points, grid_points)
queries = np.stack((xx.flatten(), yy.flatten()), axis=1)
```

In [10]:
```
def p_label(ys):
    k = 5
    ps = []
    for i in range (0,ys.shape[0]):
        den = [len(np.where(ys[i]==0)[0])/k,len(np.where(ys[i]==1)[0])/k
,len(np.where(ys[i]==2)[0])/k]
        ps.append(den)

    return ps
```

In [11]:
```
prob = knn(queries, data, k=5, summary = p_label)
prob = np.asarray(prob)


for i in range(0,3):
    plt.subplot(1,3,i+1)
    plt.imshow(np.reshape(prob[:,i], (n_grid,n_grid)), cmap=plt.cm.virid
is, origin='lower')
    plt.title(f'p({i}|q)')
    plt.axis('off')
```



Problem 4.2

In [12]:
```
def majority(ys):
    p = np.argmax(p_label(ys),axis=1)
    return p
```

In [13]:
```
import matplotlib

k = [1,5,25]

for i in range(0,3):

    plt.subplot(1,3,i+1)
    prob = np.asarray(knn(queries, data, k=k[i], summary = majority))
    plt.imshow(np.reshape(prob, (n_grid,n_grid)), cmap=matplotlib.colors
.ListedColormap(('r','g','b')), origin='lower')
    plt.title(f'k={k[i]}')
    plt.axis('off')
```