Information Theory (UAI/500)

Lecture Notebook

Lecturer: Ivo Bukovsky

Information, Similarity of Data Behavior (Same Probability Spaces)

- Data in Sample Domain
- Data in Probability Domain (bins,histogram, probability)
- Relative Entropy (Kullback-Leibler divergence)

Notes and Notations

- Here I use "Domain" and "Space" interchangeably, as used in "Sample Domain/Space" and "Probability Domain/Space" (see Fig. 1-4)
- When speaking with statisticians, the words as "system" or "experiments" should be replaced by term "random process", and word "data collection" or "measurement" or "recordings" should be replaced by "sample selection" or "observation".

- It is common, e.g. also in [1], to denote X be a random variable and x be its value, so the notations would yield $p(X = x)$. Bellow, a scalar value of random variable X is denoted with its sample index as $x(k)$ as for time series, or just without the index (k) as for histograms, see the following text, codes, and graphs for Fig.1-Fig.3.
- Also, small p is usually used for values of probability mass function, while capital P is used for probability in theoretical derivations.

Therefore, notation for random variables, vs. matrix, vector (1-D array), or a scalar is as follows:

- X, Y ... random variables, capital italic (not bold),
- $\mathbf{X}, \mathbf{Y}$ ... matrix, bold capital, where particularly $\mathbf{X}$ can be for a matrix of row feature vectors as inputs to a neural network
- $\mathbf{x}, \mathbf{y}$ ... a vector (1-D array), small and bold, it can be either the whole set of values as 1-D array (data observation); in another way (a bit confusingly), letter "x" as $\mathbf{x}$ also denotes a feature vector as for a neural network; $\mathbf{y}$ is then network output or labels (targets).
- $x, y, x(k), y(k)$ ... scalars, small and italic, particular values, i.e., selected sample (value) of random variables X, Y

- Curly bracketes (braces) $\{\ \}$ for x such as $\{x\}$ usually denotes an array of values such as $\{x \text{ for } k = 1, 2, .., N\} = [x(1)\ ,\ x(2)\ ,\ \ldots\ x(N)]$. For the meaning of "ensemble", see [1] (subsection 2.1., p.22 and futher),
- p_x, p_y ... probabilities, i.e., probability mass functions of discrete random variables (e.g. as obtained from histograms, see Fig.1-Fig.3)
- P ... probability in general sense (in theoretical formulas)

Sample Domain, Histogram, and the Same Probability Space

Let's have two data observations (for simplicity two 1-D arrays) as follows

$$\mathbf{x} = [\{x(k) \text{ for } k = 1, 2, 3, \ldots, N_x\}] \text{ where } \ x \in \langle x_{min}, x_{max} \rangle,$$
$$\mathbf{y} = [\{y(k) \text{ for } k = 1, 2, 3, \ldots, N_y\}] \text{ where } \ y \in \langle y_{min}, y_{max} \rangle,$$
(1)

where $\mathbf{x}$ and $\mathbf{y}$ are observations of random variables X and Y, k stands for sample index and N_x, N_y denote the data lengths. So, $x(k)$ and $y(k)$ represent observed data from two different systems or these can be data colections (from two different recordings at the same system, at different times, or by two different sensors , etc...).

image-6.png

Fig. 1: Sample values $x(k)$ and $y(k)$ of random variables X and Y in their sample domains.

The (discrete) probability of x, i.e., probability mass function of discrete random variable x, can be here denoted as $p_x(x)$ that means a 1-D array as follows

$$\mathbf{p}_x = [p_x(x) \ \forall \ x] = [p_x(x_1)\ ,\ p_x(x_2)\ , \ldots ,\ p_x(x_i)\ , \ldots\ ,\ p_x(x_{n_{bins}})],$$
(2)

where x_i are bin centers for $i = 1, 2, \ldots, n_{bins}$, so $\mathbf{x}$ can be re-defined for histogram centers as follows

$$\mathbf{x} = [x_{min}\ ,\ x_{min} + \Delta x,\ x_{min} + 2\Delta x\ ,\ \ \ldots\ ,\ x_{max}] = [x_1\ ,\ x_2\ ,\ x_3\ , \ldots ,\ x_i\ , \ldots\ ,\ x_{n_{bins}}],$$
(3)

where Δx is the distance between bin centers as follows

$$\Delta x = \frac{x_{max} - x_{min}}{n_{bins}}$$
(4)

so

$$x_{i+1} = x_i + \Delta x$$
(5)

where i is the bin index of the histogram.

Similarly, the probability mass function of y can be then denoted as $p_y(y)$ where

$$y = [\ y_{min},\ \ y_{min} + \Delta y\ ,\ y_{min} + 2 \cdot \Delta y\ ,\ \ \ldots\ ,\ y_{max}] = [y_1\ ,\ y_2\ ,\ y_3\ , \ldots\ ,\ y_i\ , \ldots ,\ y_{n_{bins}}]$$

(6)

and

$$\mathbf{p}_y = [p_y(y)\, \forall\, y] = [p_y(y_1)\ ,\ p_y(y_2)\ , \ldots ,\ p_y(y_i)\ , \ldots\ ,\ p_y(y_{n_{bins}})],$$

(7)

where $\{y_i\ ;\ i = 1, 2, \ldots, n_{bins}\}$ are bin centers.

If individual data samples of two random variables X and Y have the same dimension, e.g. as for time series in (1) and (Fig.1), and if the numbers of bins are defined the same for X and Y, then probability mass functions of X and Y are defined on the **same probability space** (here via the same 1-D bin indexing $i = 1, 2, \ldots, n_{bins}$).

Similarly, the **same probability space** can be defined via the same numbers of bin centers of multidimensional probabilities (via multidim. histograms) for multi-dimensional random variables X and Y , i.e., where individual data samples are n-D points $\mathbf{x}(k) = [x_1(k)\ x_2(k)\ \ldots\ x_n(k)]$,$\mathbf{y}(k) = [y_1(k)\ y_2(k)\ \ldots\ y_n(k)]$, and all observed data samples are $\mathbf{X} = [\mathbf{x}(k=1)\ \mathbf{x}(k=2),\ \ldots,\ \mathbf{x}(k=N_x)]$, $\mathbf{Y} = [\mathbf{y}(k=1)\ \mathbf{y}(k=2),\ \ldots,\ \mathbf{y}(k=N_y)]$, while the numbers of samples may be different for X and Y, i.e., observed data are 2-D arrays (matrices) with dimensions $N_x \times n,\ N_y \times n$ where not necesarilly $N_x = N_y$. (just recall that for n-dimensional random variable X and thus for n-dimensional probability $p_x(X = \mathbf{x})$ it must hold that

$$\sum_{i_1=1}^{n_{bins,1}} \sum_{i_2=1}^{n_{bins,2}} \cdots \sum_{i_n=1}^{n_{nbins,n}} p_{\mathbf{x}}(i_1, i_2, \ldots, i_n) = 1\,,$$

(8)

where i_j is bin center indexing for dimensions $j = 1, 2, \ldots, n$ and for custom defined numbers of bins $n_{bins,j}$. If $n_{bins,j}$ are the same also for n-dimensional random variable Y, then X and Y have the **same probability space**.

The following sketch and code with plots further explains the above notation.

image-8.png

Fig. 2: Histogram's i-th bin and the bin centre. (The bins define the probability space for x, basicaly by the number of bins n_{bins}.)

```
%matplotlib inline
from numpy import *
from matplotlib.pyplot import *
rcParams.update({'font.size': 12})
#======
def sr(a):  # for plots
    a=str(round(a,2))
    return(a)
#======
Nx=500
Ny=1000
```

```
n_bins=20
#----------
x=random.randn(Nx)/3+1
y=random.binomial(Ny, .01, size=Ny)*2-10  # y=0,1,2,...10000
#y=(y-mean(y))/std(y)  #z-scoring, data normalization

freq_x,x_edges=histogram(x,n_bins)  #
freq_y,y_edges=histogram(y,n_bins)  #

dx=(max(x)-min(x))/n_bins   # Delta x
dy=(max(y)-min(y))/n_bins   # Delta x

x_centers=[x_edges[i]+dx/2 for i in range(n_bins)]
y_centers=[y_edges[i]+dy/2 for i in range(n_bins)]


px=freq_x/Nx  # probabilities
py=freq_y/Ny


figure(figsize=(14,10))
subplots_adjust(hspace=0.8)
subplot(421);plot(x,label="x(k)");grid();ylabel("x(k)");xlabel("k [sample]");ylim(-1,4
title("(Measured) samples of $X$ in time (sample domain) \n $N_x=$"+str(Nx)+"$ \ , \ \
subplot(422);plot(y,label="y(k)");grid();ylabel("y(k)");xlabel("k [sample]");ylim(-30,
title("(Measured) samples of $Y$ in time (sample domain) \n $N_y=$"+str(Ny)+"$ \ , \ \
subplot(423);hist(x,bins=n_bins);grid();xlabel("x");ylabel("frequencies");title("histo
subplot(424);hist(y,bins=n_bins);grid();ylabel("frequencies");xlabel("y");title("histo
subplot(425);plot(x_centers,px,'-o');title("$p_x(x) \ , \ \sum p_x(x)=$"+str(sum(px)))
subplot(426);plot(y_centers,py,'-o');title("$p_y(y) \ , \ \sum p_y(y)=$"+str(sum(py)))
subplot(427);plot(arange(n_bins)+1,px,'-o');title("$p_x(i) \ , \ \sum p_x(i)=$"+str(su
subplot(428);plot(arange(n_bins)+1,py,'-o');title("$p_y(i) \ , \ \sum p_y(i)=$"+str(su
show()
```

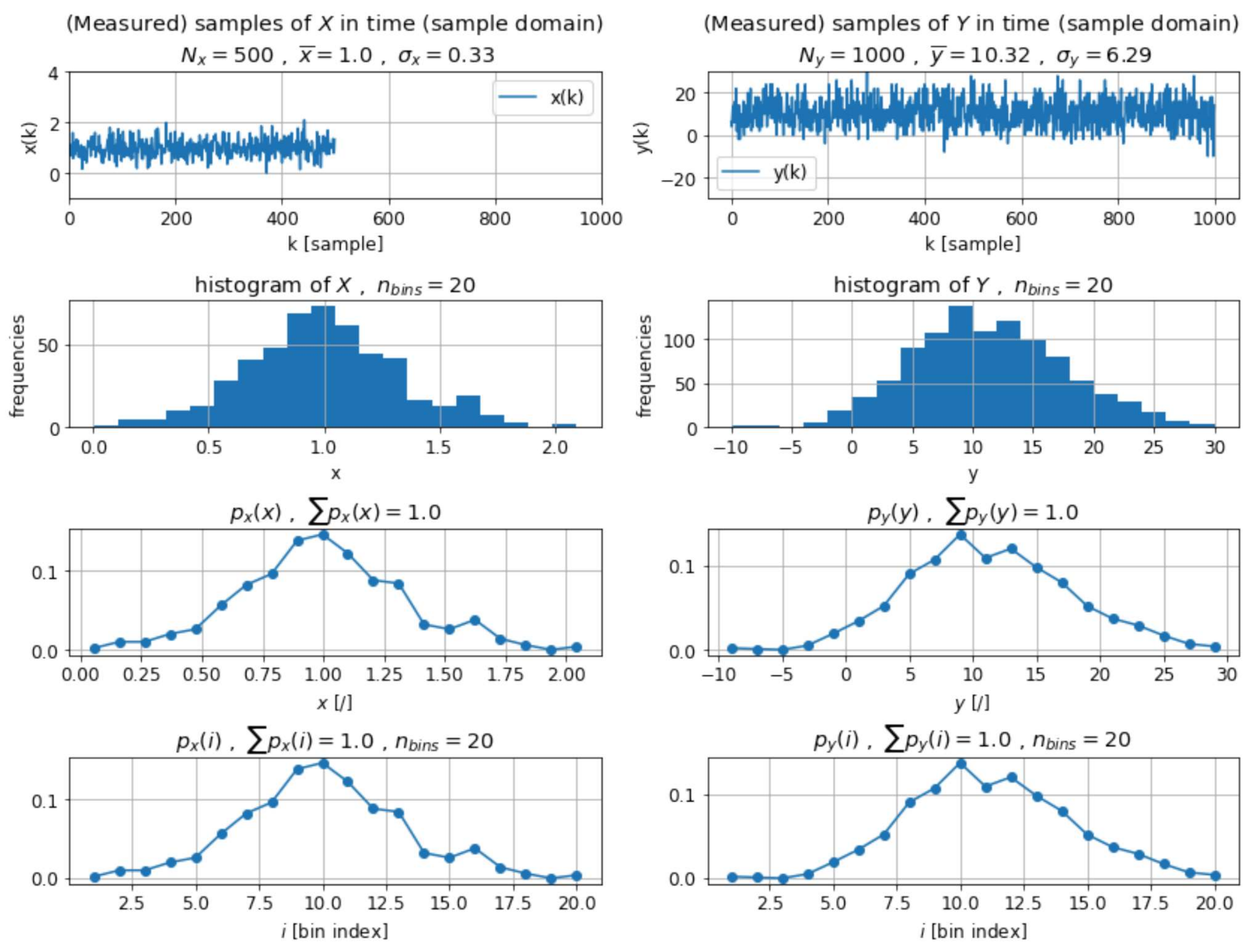


Fig. 3: Probabilities for sets x and y at the same probability space, i.e., $n_{bins,x} = n_{bins,y} = n_{bins}$.

We are now to compare the distributions of X and Y, i.e., to compare their probability mass functions $p_x(i)$ and $p_y(i)$ that can tell us how similar or distinct is behavior of these data regardless the length of data, and regardless their means and magnitudes.

Notice that to have the same probability space for X and Y, both X and Y have to be of same-dimensionality (here are both 1-D arrays) while their length can be different; however, the number of bins must be the same.

Relative Entropy (Kullback-Leibler divergence)

Let's have two random variables X and Y whose values are denoted $x(k)$ and $y(k)$ in sample domain, and x_i and y_i. The shapes of their probability distributions can be roughly estimated from their histograms.

Let's denote their probability mass functions accordingly as follows

$$p_x = p_x(x_i) \;\text{ or }\; p_y = p_y(y_i)$$
(9)

or when using only the bin indexing i as follows

$$p_x = p_x(i) \;\text{ or }\; p_y = p_y(i)$$
(10)

where $i = 1, 2, 3, \dots, n_{bins}$.

Then, the information content of the random variable outcome (falling into i-th bin) is accordingly as follows

$$I_x(x_i) = log_2\left(\frac{1}{p_x(x_i)}\right) \;,\; I_y(y_i) = log_2\left(\frac{1}{p_y(y_i)}\right) \;\text{[bit]}$$
(11)

or analogically when using only i for the bin indexing as

$$I_x(i) = -log_2\big(p_x(i)\big) \;,\; I_y(i) = -log_2\big(p_y(i)\big) \;\text{[bit] where } i = 1, 2, \dots, n_{bins}$$
(12)

Gibbs' inequality and KL Divergence

image-2.png

Fig. 4: Probabilities for sets x and y at the same probability space, i.e., $n_{bins,x} = n_{bins,y} = n_{bins}$.

Because both $p_x, p_y \in \langle 0, 1\rangle$, it always holds for Shannon entropy that

$$\sum_{i=1}^{i=n_{bins}} p_x(i) \cdot I_x(i) \leq \sum_{i=1}^{i=n_{bins}} p_x(i) \cdot I_y(i)),$$
(13)

i.e.

$$0 \leq \sum_{i=1}^{i=n_{bins}} p_x(i) \cdot log_2\big(p_x(i)\big) - \sum_{i=1}^{i=n_{bins}} p_x(i) \cdot log_2\big(p_y(i)\big) = \dots = \sum_{i=1}^{i=n_{bins}} p_x(i) \cdot log_2\left(\frac{p_x(i)}{p_y(i)}\right) = D_K$$
[bit] ,

(14)

where D_{KL} is the **relative entropy** that is also called the **Kullback-Leibler divergence**, and $D_{KL} = 0$ only if the both distributions p_x and p_y are equal.

D_{KL} is practically useful,eventhough the KL divergence is not a true distance between two distributions because it is not symetric, i.e., $D_{KL}(p_x||p_y) \neq D_{KL}(p_y||p_x)$ and does not respect the tringular inequality, e.g., p.18 in [2].

Also notice, that p_x and p_y can be two differrent distributions, obtained from two different data sets, but they have to be defined at the same probability space (here both 1-D arrays and $i = 1, 2, \dots, n_{bins,x} = n_{bins,y}$).

For KL divergence, study also section 2.6 in [1] and related sections.

Example L-10

1. Generate 300 samples of 1-D random variable X, i.e., $X = x(k)$ where $k = 1, 2, \ldots, N_x = 300$ with normally distributed data with mean $\overline{x} = 50$ and standard deviation $\sigma_x = 10$, draw the data in a sample domain and draw the data in the probability domain (probability mass function $p_x(x)$) **[0.15 Points]**
2. Generate 500 samples of 1-D random variable Y, i.e., $Y = y(k)$ where $k = 1, 2, \ldots, N_y = 500$ (N_y...number of generated samples) with binomial distribution with probability parameter $p = 0.45$ as in [a], number of trials $n = 100$, e.g. use [a] , draw the data in a sample domain and draw the data in the probability domain (probability mass function $p_y(y)$. Notice, the parameter p as in [a] is not the same as $p_y(y)$ here **[0.15 Points]**
3. For the above 1-D random variables X and Y calculate the K-L divergence $D_{KL}(X||Y)$ **[0.2 Points]**
4. For the above 1-D random variables Y and X calculate the K-L divergence $D_{KL}(Y||X)$ **[0.2 Points]**
5. Keep the normally distributed random variable X from the above. Calculate and draw graph of $D_{KL}\left(X, Y(p)\right)$ where the binomial distribution parameter p for random variable Y varies as $p = [0.1, 0.2, 0.3, \ldots, 0.9]$, see Hint a) just bellow **[1.0 Point]**
6. Instead of $D_{KL}\left(X||Y(p)\right)$ as in the previous task, calculate and draw the Euclidean distance between the two distributions of p_x(X) and p_y(Y) for the generated data from the previous task, i.e. between observations normally distributed X and Y observations with binomial distribution of Y for $p = [0.1, 0.2, 0.3, \ldots, 0.9]$, see Hint b) **[0.5 Points]**
7. Here, investigate D_{KL} vs. the Euclidean distance as measures for similarity of two distributions. Repeat the above simulation experiments, i.e. task 5. and 6., at least 10-times, and observe (plot curves of) the behavior of D_{KL} vs. the Euclidean distances behavior (for at least 10 various generated datasets). What is your findings when comparing these two measures? **[2 Points]**

---

8. Generate 1000 samples of 2-dimensional random variable X with data points $\mathbf{x}(k) = [x_1(k), x_2(k)]$, $k = 1, 2, \ldots, N_x = 1000$ with normal distribution with zero mean and unit standard deviation (2 columns, each column has normally distributed values). Generate 2000 samples of 2-dimensional random variable Y with data points $\mathbf{y}(k) = [y_1(k), y_2(k)]$, $k = 1, 2, \ldots, N_y = 2000$ with minimum values 0 and maximum value 100 with uniform distribution [b] (2 columns, each column has uniformly distributed values):
   - plot each column of X observations, i.e., $x_1(k)$ and $x_2(k)$, in sample domain **[0.1 Points]**
   - plot each column of Y observations, i.e., $y_1(k)$ and $y_2(k)$, in sample domain **[0.1 Points]**
   - plot a point graph $x2 = x2(x1)$ $\forall k$ just to see the distribution of data points in 2-D **[0.1 Points]**

- plot a point graph $y2 = y2(y1)\ \forall k$ just to see the distribution of data points in 2-D **[0.1 Points]**
- Choose $n_{bins,1}$ and $n_{bins,2}$ and calcuate and show the data behavior in the probability space, i.e. plot $p_{\mathbf{x}}(\mathbf{x})$, (with some existing tool, via 2-D histogram,..., $\sum\limits_{\forall x_1}\sum\limits_{\forall x_2} p_{x_1,x_2}(x_1, x_2) \overset{!}{=} 1$) **[0.2 Points]**
- For the same $n_{bins,1}$ and $n_{bins,2}$ calcuate and show the data behavior in the probability space, i.e. plot $p_{\mathbf{y}}(\mathbf{y})$, (with some existing tool, via 2-D histogram,..., $\sum\limits_{\forall y_1}\sum\limits_{\forall y_2} p_{y_1,y_2}(y_1, y_2) \overset{!}{=} 1$) **[0.2 Points]**
- Show data behavior in the probability space, i.e. plot $p_{x,y}(x, y)$, (with some existing tool, 2-D histogram,...) **[0.2 Points]**

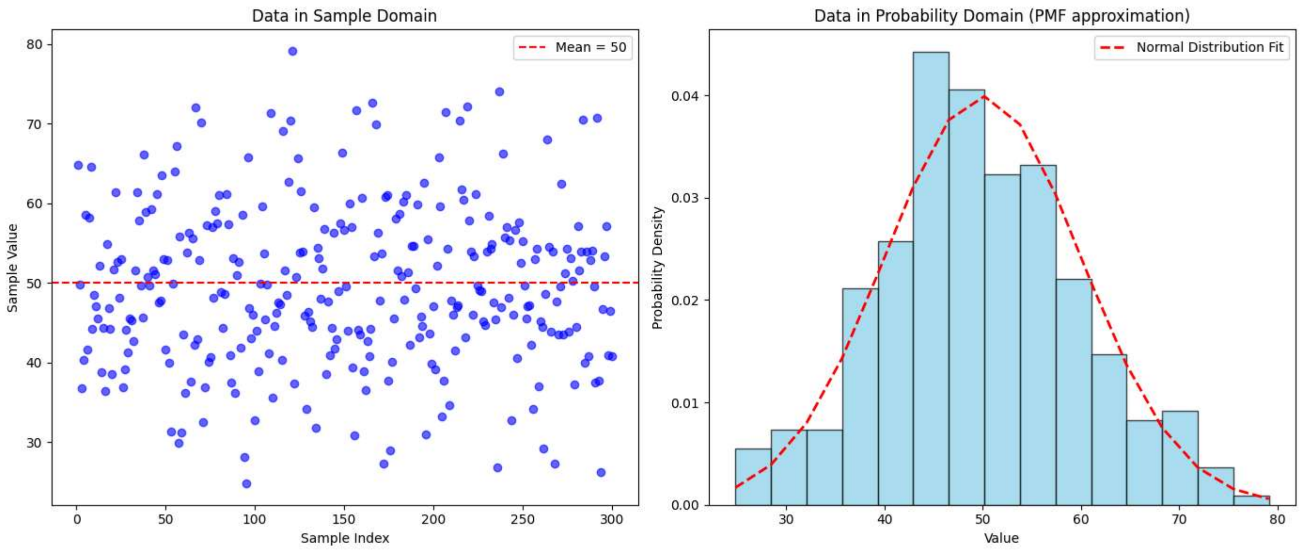
9. Calculate $D_{KL}(X||Y)$ **[2 Points]**
10. Calculate $D_{KL}(Y||X)$ **[1 Points]**

Hints:

a) Keep previously generated random variable X (300 smaples). Then you have to generate new 500 samples of binomially distributed data for every p=0.1, 0.2,...,0.9 , for the same number of trials n!.(Notice, n is here the parameter of binomial distribution, not a dimension of data points!) image-6.png b) image-5.png

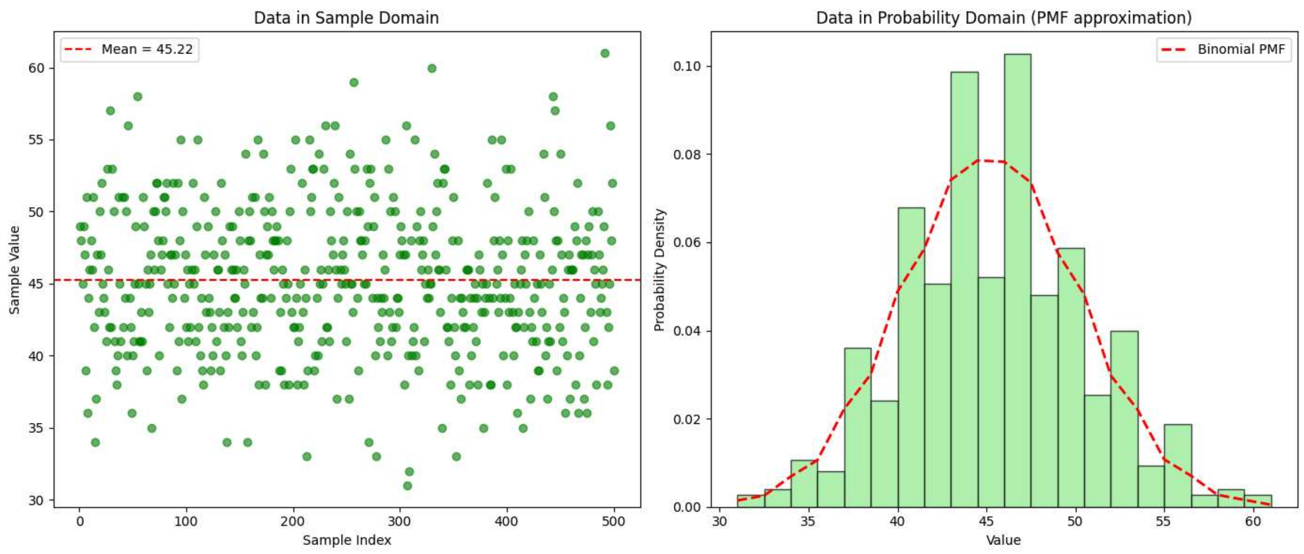
```
"""1) Generate 300 samples of 1-D random variable  X , i.e.,  X=x(k)  where  k=1,2,...
    and standard deviation  σx=10  , draw the data in a sample domain and draw the dat

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Parameters
mean = 50
std_dev = 10
Nx = 300

# Generate samples
samples_X = np.random.normal(mean, std_dev, Nx)

# data in the sample domain
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.scatter(range(1, Nx + 1), samples_X, color='b', alpha=0.6)
plt.axhline(mean, color='r', linestyle='--', label=f'Mean = {mean}')
plt.xlabel('Sample Index')
plt.ylabel('Sample Value')
plt.title('Data in Sample Domain')
plt.legend()

# data in the probability domain (approximating the PMF with a histogram)
```

```
plt.subplot(1, 2, 2)
count, bins, ignored = plt.hist(samples_X, bins=15, density=True, color='skyblue', edg
plt.plot(bins, norm.pdf(bins, mean, std_dev), 'r--', linewidth=2, label='Normal Distri
plt.xlabel('Value')
plt.ylabel('Probability Density')
plt.title('Data in Probability Domain (PMF approximation)')
plt.legend()

plt.tight_layout()
plt.show()

```



```
"""2) Generate 500 samples of 1-D random variable  Y , i.e.,  Y=y(k)  where  k=1,2,...
    with probability parameter  p=0.45  as in [a], number of trials  n=100 , e.g. use
    (probability mass function  py(y) . Notice, the parameter  p  as in [a] is not the

# Parameters
p = 0.45
n = 100
Ny = 500  # Number of samples

# Generate samples
```

```
samples_Y = np.random.binomial(n, p, Ny)

# data in the sample domain
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.scatter(range(1, Ny + 1), samples_Y, color='g', alpha=0.6)
plt.axhline(np.mean(samples_Y), color='r', linestyle='--', label=f'Mean = {np.mean(sam
plt.xlabel('Sample Index')
plt.ylabel('Sample Value')
plt.title('Data in Sample Domain')
plt.legend()

# data in the probability domain (approximating the PMF with a histogram)
plt.subplot(1, 2, 2)
count, bins, ignored = plt.hist(samples_Y, bins=20, density=True, color='lightgreen',
plt.plot(bins, binom.pmf(bins.astype(int), n, p), 'r--', linewidth=2, label='Binomial
plt.xlabel('Value')
plt.ylabel('Probability Density')
plt.title('Data in Probability Domain (PMF approximation)')
plt.legend()

plt.tight_layout()
plt.show()

```



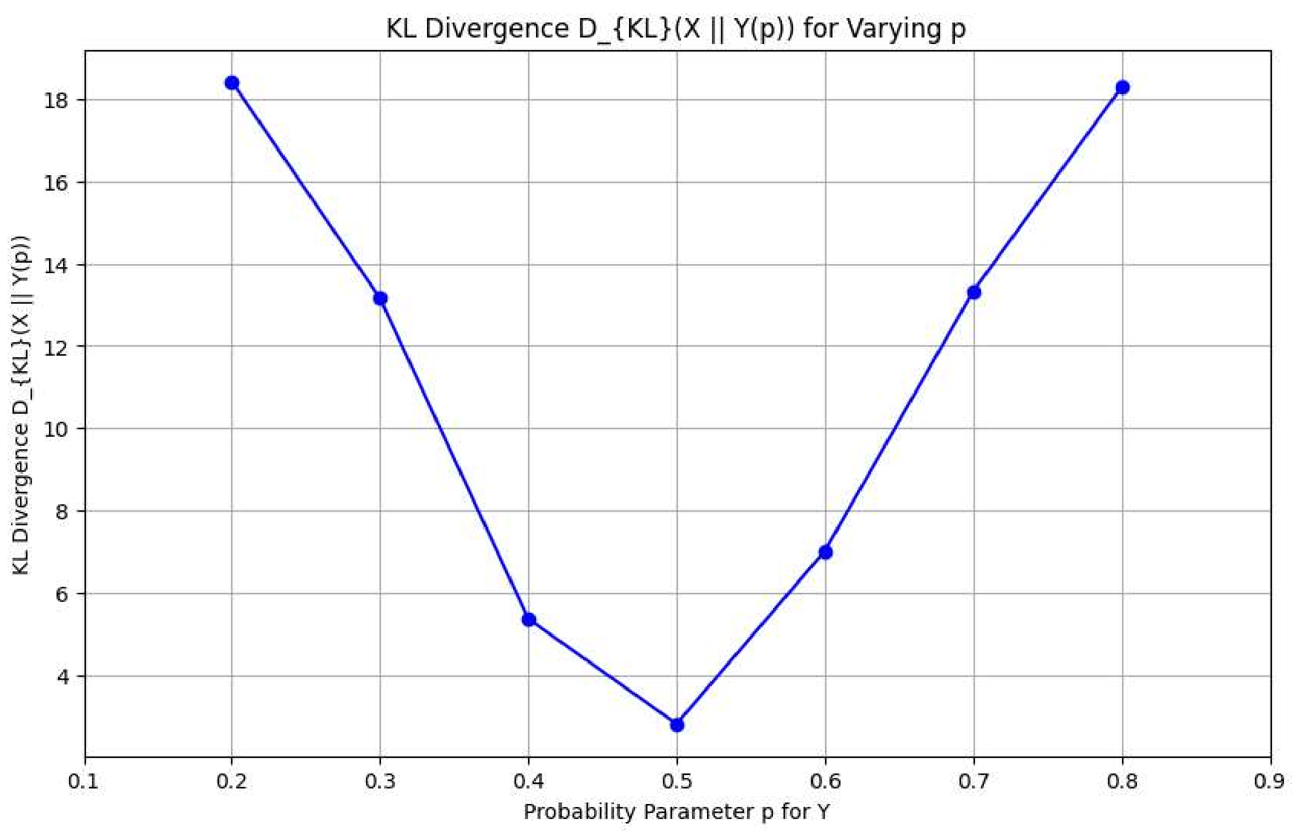
```
"""3)For the above 1-D random variables  X  and  Y  calculate the K-L divergence  DKL(

# range and bins for PMF estimation
xmin = min(min(samples_X), min(samples_Y))
xmax = max(max(samples_X), max(samples_Y))
bins = np.arange(xmin, xmax + 1)  # Use integer bins for clarity

# Calculate empirical PMFs for X and Y
hist_X, _ = np.histogram(samples_X, bins=bins, density=True)
hist_Y, _ = np.histogram(samples_Y, bins=bins, density=True)

# Avoid division by zero by adding a small epsilon to histograms
epsilon = 1e-10
hist_X = hist_X + epsilon
hist_Y = hist_Y + epsilon

# Calculate KL divergence
kl_divergence = entropy(hist_X, hist_Y)

print("KL Divergence D_KL(X || Y):", kl_divergence)

```

```
KL Divergence D_KL(X || Y): 2.654946954826417
```

```
# 4) For the above 1-D random variables  Y  and  X  calculate the K-L divergence  DKL(

kl_divergence = entropy(hist_Y, hist_X)

print("KL Divergence D_KL(Y || X):", kl_divergence)
```

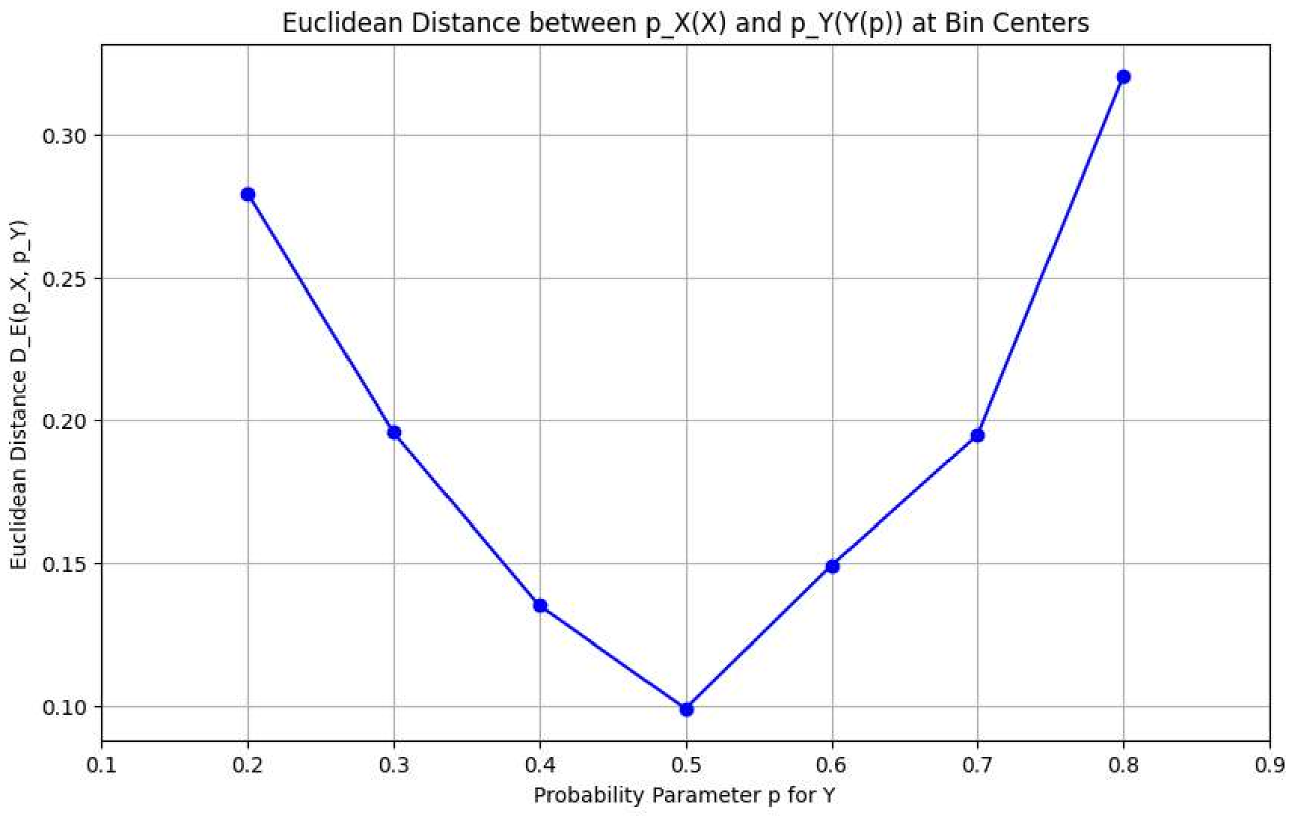
```
KL Divergence D_KL(Y || X): 0.47053169029992703
```

```
""" 5) Keep the normally distributed random variable  X  from the above.
      Calculate and draw graph of  DKL(X,Y(p))  where the binomial distribution parame

p_values = np.arange(0.1, 1.0, 0.1)
kl_divergences = []

# Calculate KL divergence D_KL(X || Y(p)) for varying p
n_Y = 100  # Number of trials for Y
for p in p_values:
    samples_Y = np.random.binomial(n_Y, p, 500)  # Generate 500 samples for each p

    # Calculate empirical PMFs for X and Y
    hist_X, bins_X = np.histogram(samples_X, bins=30, density=True)
    hist_Y, _ = np.histogram(samples_Y, bins=bins_X, density=True)

    epsilon = 1e-10
    hist_X += epsilon
    hist_Y += epsilon

    # Calculate KL divergence
    kl_divergence = entropy(hist_X, hist_Y)
    kl_divergences.append(kl_divergence)

# Plotting the KL divergence against p
plt.figure(figsize=(10, 6))
plt.plot(p_values, kl_divergences, marker='o', linestyle='-', color='b')
plt.title('KL Divergence D_{KL}(X || Y(p)) for Varying p')
plt.xlabel('Probability Parameter p for Y')
plt.ylabel('KL Divergence D_{KL}(X || Y(p))')
plt.xticks(p_values)
plt.grid()
plt.show()

```



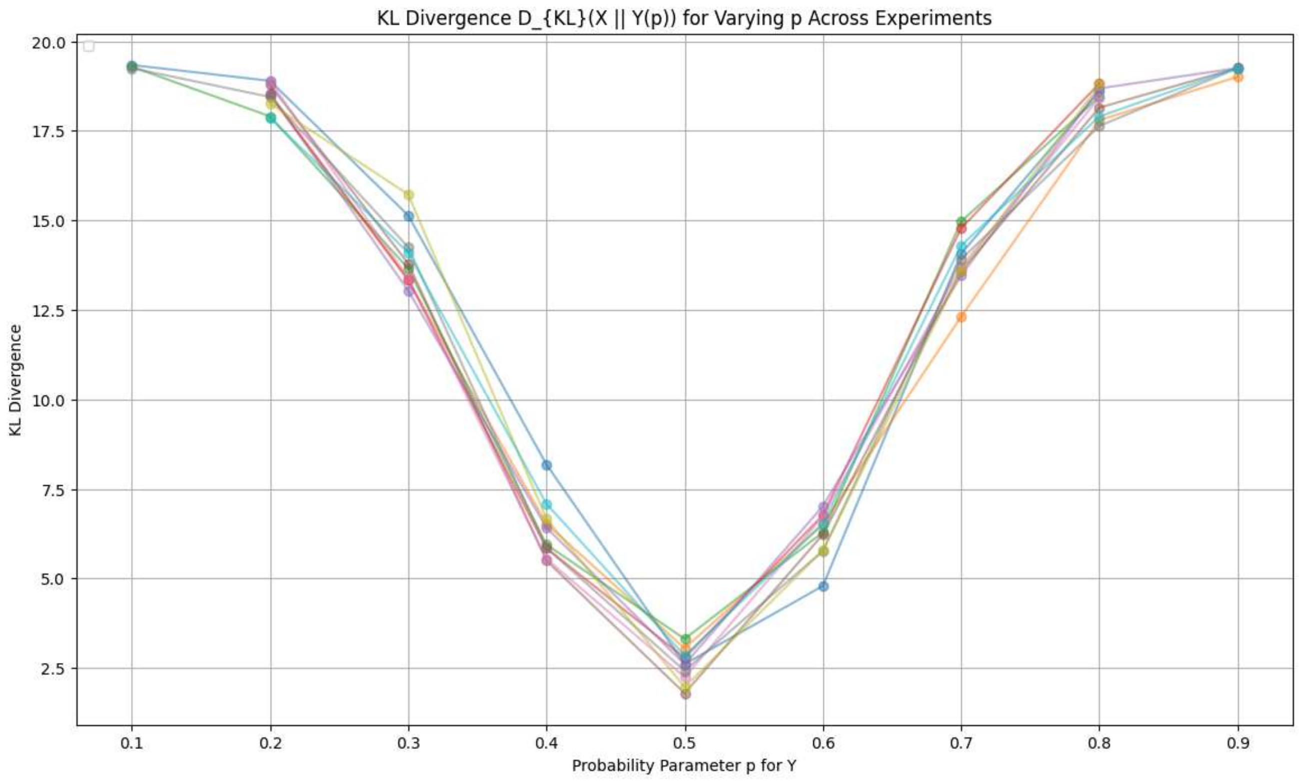
```
""" 6) Instead of  DKL(X||Y(p))  as in the previous task, calculate and draw the Eucli
    for the generated data from the previous task, i.e. between observations normally
    for  p=[0.1,0.2,0.3,...,0.9] """



import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, binom

# Parameters for the normal distribution X
mean_X, std_dev_X = 50, 10
Nx = 300  # Number of samples for X

# Generate samples for X
samples_X = np.random.normal(mean_X, std_dev_X, Nx)

# Euclidean distance values for different p
p_values = np.arange(0.1, 1.0, 0.1)
euclidean_distances = []

# Calculate Euclidean distance for varying p
n_Y = 100  # Number of trials for Y
for p in p_values:
```

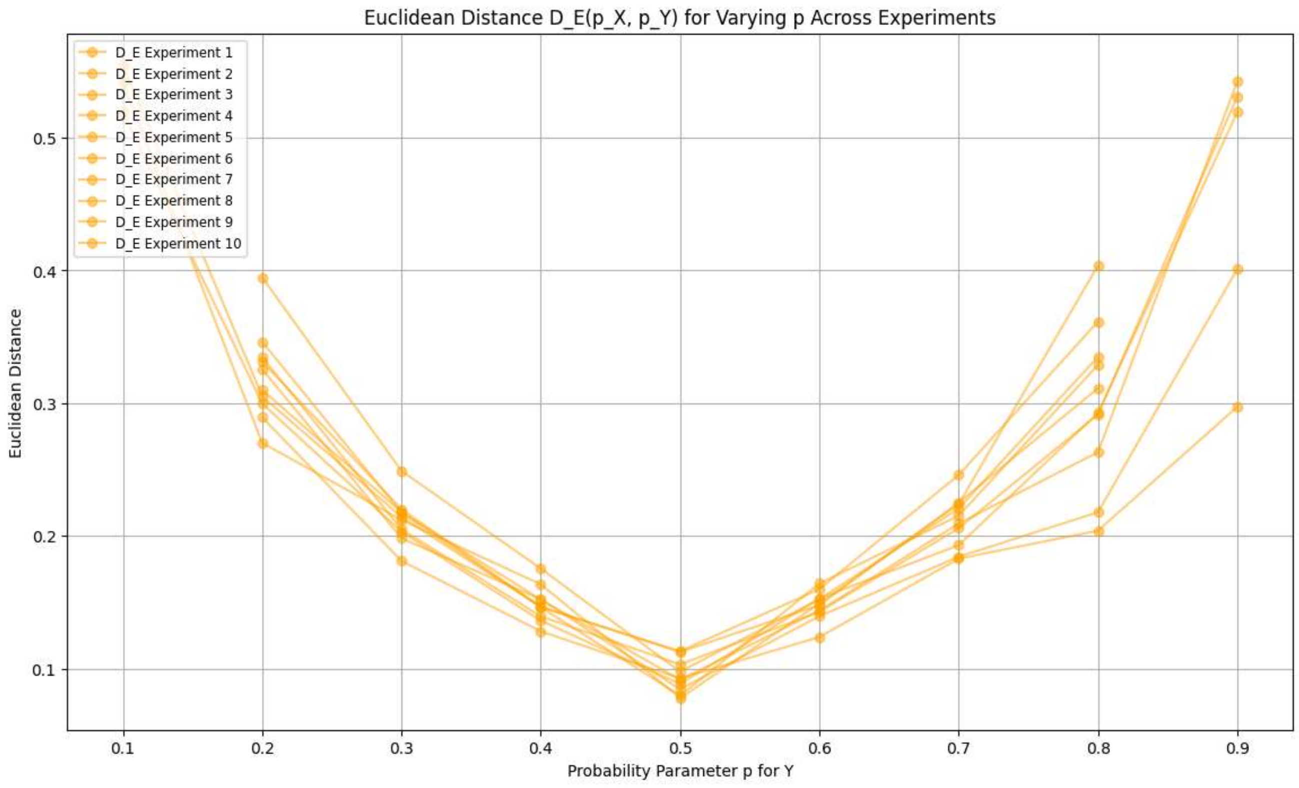
```python
    # Generate 500 samples for the binomial distribution Y
    samples_Y = np.random.binomial(n_Y, p, 500)

    # Calculate empirical PMFs for X and Y
    hist_X, bins_X = np.histogram(samples_X, bins=30, density=True)
    hist_Y, bins_Y = np.histogram(samples_Y, bins=bins_X, density=True)

    # Calculate bin centers
    bin_centers_X = 0.5 * (bins_X[:-1] + bins_X[1:])
    bin_centers_Y = 0.5 * (bins_Y[:-1] + bins_Y[1:])

    # Ensure the histograms are the same length for comparison
    if len(hist_X) > len(hist_Y):
        hist_Y = np.pad(hist_Y, (0, len(hist_X) - len(hist_Y)), 'constant')
    elif len(hist_X) < len(hist_Y):
        hist_X = np.pad(hist_X, (0, len(hist_Y) - len(hist_X)), 'constant')

    #  Compute the Euclidean distance based on bin centers
    # Interpolate Y's histogram to the X's bin centers
    hist_Y_interp = np.interp(bin_centers_X, bin_centers_Y, hist_Y, left=0, right=0)

    # Calculate the Euclidean distance
    euclidean_distance = np.sqrt(np.sum((hist_X - hist_Y_interp) ** 2))
    euclidean_distances.append(euclidean_distance)

# Plotting the Euclidean distance against p
plt.figure(figsize=(10, 6))
plt.plot(p_values, euclidean_distances, marker='o', linestyle='-', color='b')
plt.title('Euclidean Distance between p_X(X) and p_Y(Y(p)) at Bin Centers')
plt.xlabel('Probability Parameter p for Y')
plt.ylabel('Euclidean Distance D_E(p_X, p_Y)')
plt.xticks(p_values)
plt.grid()
plt.show()


```



```python
""" 7) Here, investigate  DKL  vs. the Euclidean distance as measures for similarity o
  i.e. task 5. and 6., at least 10-times, and observe (plot curves of) the behavior of
  (for at least 10 various generated datasets). What is your findings when comparing t

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, binom, entropy

# Parameters for the normal distribution X
mean_X, std_dev_X = 50, 10
Nx = 300  # Number of samples for X
n_Y = 100  # Number of trials for Y
num_experiments = 10  # Number of datasets to generate

# Probability parameter values for Y
p_values = np.arange(0.1, 1.0, 0.1)

# To store results
kl_results = []
euclidean_results = []

# Perform the experiments
for _ in range(num_experiments):
    samples_X = np.random.normal(mean_X, std_dev_X, Nx)  # Samples for X
    kl_distances = []
    euclidean_distances = []

    for p in p_values:
        samples_Y = np.random.binomial(n_Y, p, 500)  # Generate samples for Y

        # Calculate empirical PMFs for X and Y
        hist_X, bins_X = np.histogram(samples_X, bins=30, density=True)
        hist_Y, bins_Y = np.histogram(samples_Y, bins=bins_X, density=True)

        # Calculate bin centers
        bin_centers_X = 0.5 * (bins_X[:-1] + bins_X[1:])
        bin_centers_Y = 0.5 * (bins_Y[:-1] + bins_Y[1:])

        # Interpolate Y's histogram to the X's bin centers
        hist_Y_interp = np.interp(bin_centers_X, bin_centers_Y, hist_Y, left=0, right=

        # Avoid division by zero for KL divergence calculation
        epsilon = 1e-10
        hist_X += epsilon
        hist_Y_interp += epsilon

        # Calculate KL divergence
        kl_distance = entropy(hist_X, hist_Y_interp)
        kl_distances.append(kl_distance)

        # Calculate the Euclidean distance
        euclidean_distance = np.sqrt(np.sum((hist_X - hist_Y_interp) ** 2))
        euclidean_distances.append(euclidean_distance)

    kl_results.append(kl_distances)
    euclidean_results.append(euclidean_distances)

# Convert results to arrays for easier plotting
kl_results = np.array(kl_results)
euclidean_results = np.array(euclidean_results)

# Plotting
plt.figure(figsize=(14, 8))

# Plot KL divergence results
for i in range(num_experiments):
    plt.plot(p_values, kl_results[i], marker='o', linestyle='-', alpha=0.5)

plt.title('KL Divergence D_{KL}(X || Y(p)) for Varying p Across Experiments')
plt.xlabel('Probability Parameter p for Y')
plt.ylabel('KL Divergence')
plt.grid()
plt.legend(loc='upper left', fontsize='small')
plt.show()

plt.figure(figsize=(14, 8))

# Plot Euclidean distance results
for i in range(num_experiments):
    plt.plot(p_values, euclidean_results[i], marker='o', linestyle='-', alpha=0.5, col

plt.title('Euclidean Distance D_E(p_X, p_Y) for Varying p Across Experiments')
plt.xlabel('Probability Parameter p for Y')
plt.ylabel('Euclidean Distance')
plt.grid()
plt.legend(loc='upper left', fontsize='small')
plt.show()

```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note that a
```





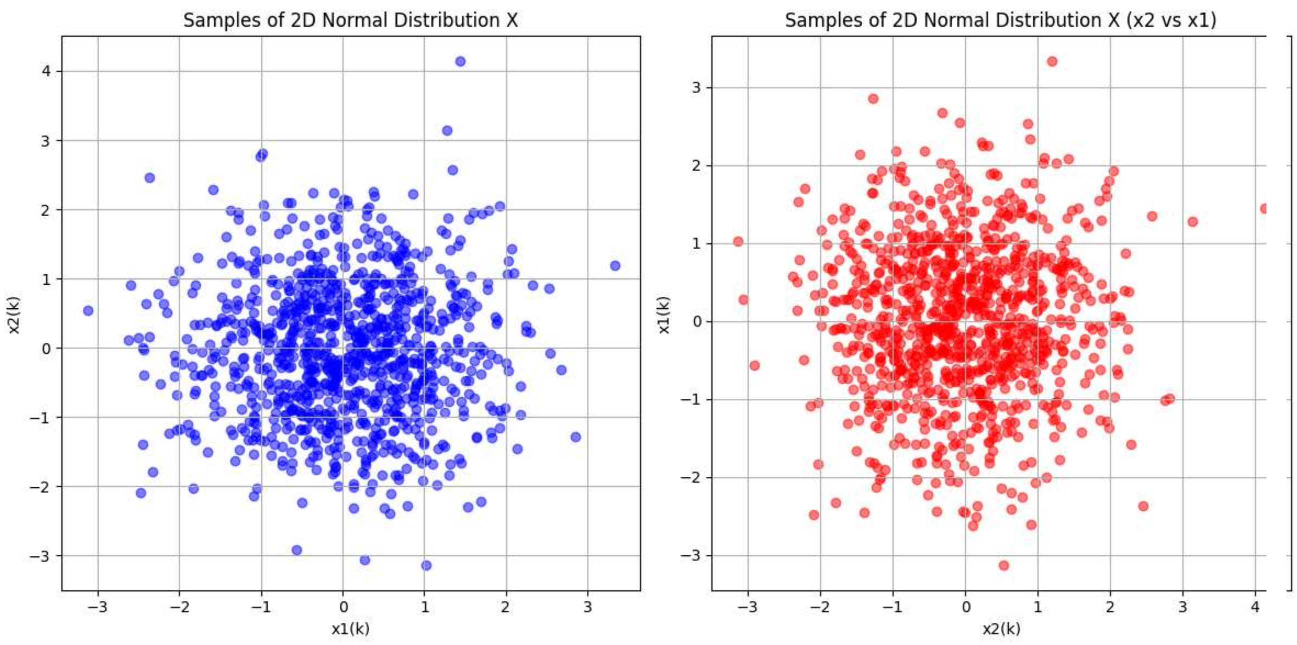
Observe the KL divergence and Euclidean distance vary with different values of p.

KL Divergence is sensitive to the tails of the distributions. It may increase sharply when p leads to a significant change in the distribution shape of Y(p).

Euclidean Distance can be affected by the distribution of mass across the bins. It might not capture the difference in distribution tails as effectively as KL divergence.

```
""" 8)Generate 1000 samples of 2-dimensional random variable  X  with data points  x(
     and unit standard deviation (2 columns, each column has normally distributed val
     y(k)=[y1(k),y2(k)] , k=1,2,...,Ny=2000  with minimum values 0 and maximum value
    - plot each column of  X  observations, i.e.,  x1(k)  and  x2(k) , in sample doma

import numpy as np
import matplotlib.pyplot as plt

# Parameters for the normal distribution X
mean_X = 0
std_dev_X = 1
Nx = 1000  # Number of samples for X

# Generate samples for the 2-dimensional normal distribution X
samples_X = np.random.normal(mean_X, std_dev_X, (Nx, 2))

# Parameters for the uniform distribution Y
Ny = 2000  # Number of samples for Y
min_Y = 0
max_Y = 100

# Generate samples for the 2-dimensional uniform distribution Y
samples_Y = np.random.uniform(min_Y, max_Y, (Ny, 2))

```

```
# Plotting the samples of X
plt.figure(figsize=(12, 6))

# Plot x1(k) vs x2(k)
plt.subplot(1, 2, 1)
plt.scatter(samples_X[:, 0], samples_X[:, 1], alpha=0.5, color='blue')
plt.title('Samples of 2D Normal Distribution X')
plt.xlabel('x1(k)')
plt.ylabel('x2(k)')
plt.grid(True)

# Plot x2(k) vs x1(k)
plt.subplot(1, 2, 2)
plt.scatter(samples_X[:, 1], samples_X[:, 0], alpha=0.5, color='red')
plt.title('Samples of 2D Normal Distribution X (x2 vs x1)')
plt.xlabel('x2(k)')
plt.ylabel('x1(k)')
plt.grid(True)

plt.tight_layout()
plt.show()

```


Samples of 2D Normal Distribution X
x1(k)
x2(k)
Samples of 2D Normal Distribution X (x2 vs x1)
x2(k)
x1(k)