Assignment 1 - Probability, Linear Algebra, Programming, and Git

Joaquin Menendez

Netid: *jm622*

Probability and Statistics Theory

1

Let $f(x) = \begin{cases} 0 & x < 0 \\ \alpha x^2 & 0 \leq x \leq 2 \\ 0 & 2 < x \end{cases}$

For what value of α is $f(x)$ a valid probability density function?

Note: for all assignments, write out all equations and math for all assignments using markdown and LaTeX (https://tobi.oetiker.ch/lshort/lshort.pdf) and show all work

ANSWER

$$\int_0^2 ax^2 dx = \frac{ax^3}{3} = 1$$
$$\frac{a2^3}{3} - \frac{a0^3}{3} = 1$$
$$\frac{a2^3}{3} = 1$$
$$\frac{a8}{3} = 1$$
$$a = \frac{3}{8}$$

A valid probability funciton for f is going to given by a value of α equal to 3/8

2

What is the cumulative distribution function (CDF) that corresponds to the following probability distribution function? Please state the value of the CDF for all possible values of x.

$$f(x) = \begin{cases} \frac{1}{3} & 0 < x < 3 \\ 0 & \text{otherwise} \end{cases}$$

ANSWER

So, if we want to calculate the CDF for a value i of an uniform distribution we could say:
$\Omega - (X_i \cdot \kappa) =$ CDF complement (from the right)
$1 - (X_i \cdot \frac{1}{3}) =$ CDF complement (from the right)
$\frac{X_i}{3} = CDF$

$$f(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{3}x & 0 \leqslant x < 3 \\ 1 & x \geqslant 3 \end{cases}$$

3

For the probability distribution function for the random variable X,

$$f(x) = \begin{cases} \frac{1}{3} & 0 < x < 3 \\ 0 & \text{otherwise} \end{cases}$$

what is the (a) expected value and (b) variance of X. *Show all work*.

ANSWER

The probability density function is 1

(a) Expected value:

$E(X) = \int_a^b x\ f(x)\ dx$ where f(x) refers to the PDF
$\int_0^3 x\frac{1}{3}\ dx =$
$\frac{1}{3}\int_0^3 x\ dx =$
$\frac{1}{3}\int_0^3 \frac{1}{2}x^2 =$
$\frac{1}{3}\cdot(\frac{1}{2}3^2 - 0) =$
$\frac{1}{3}\cdot(\frac{1}{2}3^2 - 0) = \frac{3}{2}$

(b) Variance:
$Var(X) = E[X^2] - E[X]^2$, so using the theorem used above:
= $\int_0^3 \frac{1}{3}x^2\ dx - (\int_0^3 \frac{1}{3}x)^2$
= $\frac{1}{3}\int_0^3 x^2\ dx - (\frac{1}{3}\int_0^3 x)^2$
= $\frac{1}{3}\cdot\frac{1}{3}3^3 - (\frac{1}{3}\cdot(\frac{1}{2}3^2)^2$
= $3 - (\frac{3}{3})^2$
= $3 - \frac{9}{4}$
= 0.75

4

Consider the following table of data that provides the values of a discrete data vector $\mathbf{x}$ of samples from the random variable X, where each entry in $\mathbf{x}$ is given as x_i.

Table 1. Dataset N=5 observations

-	x_0	x_1	x_2	x_3	x_4
$\mathbf{x}$	2	3	10	-1	-1

What is the (a) mean, (b) variance, and the of the data?

Show all work. Your answer should include the definition of mean, median, and variance in the context of discrete data.

ANSWER

(a) Mean

$$\bar{X} = \frac{\sum_{i=0}^{n} x_i}{n}$$

$$\bar{X} = \frac{13}{5} = 2.6$$

(b) Variance

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{X})^2}{n}$$

$$\sigma^2 = \frac{\sum_{i=1}^{n=5} (x_i - 2.6)^2}{5}$$

In [6]:
```
a = np.array([2,3,10,-1,-1])
print('var = %f' %np.var(a))
```

```
var = 16.240000
```

The median is the value separating the higher half from the lower half of the data distribution. If we work with discrete distributions we can find the median by arranging all the numbers from smallest to greatest and picking the value on the exact middle, as long as the number of elements on the distribution is odd. In case of having even elements, the two number closest to the half will be summed and divided by two.

5

Review of counting from probability theory.

(a) How many different 7-place license plates are possible if the first 3 places only contain letters and the last 4 only contain numbers?

(b) How many different batting orders are possible for a baseball team with 9 players?

(c) How many batting orders of 5 players are possible for a team with 9 players total?

(d) Let's assume this class has 26 students and we want to form project teams. How many unique teams of 3 are possible?

Hint: For each problem, determine if order matters, and if it should be calculated with or without replacement.

ANSWER

(a) Order not matter, the only important thing is the number of different plates.

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Possible N	26	26	26	10	10	10	10

$26 * 26 * 26 * 10 * 10 * 10 * 10 = 26^3 * 10^4 = 175760000$

(b) The order is important in this case. In the first position I have nine possible names, after I select one player I have eight possible names for the second position and so on.
$9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 9! = 362880$

(c) Order matters. I have only 5 positions to 9 players. So I have 9 possibilities to choose the first name, and so on until reach the 5th position. $\frac{n!}{k!}$ where n is my number of players and k are the available positions $9! - 5! = 9 * 8 * 7 * 6 = 3024$

(d) In this case the order of the teams does not seem to be important. This is a combinatory problem that seems to follow an

$C_{n,k} = \frac{n!}{k! * (n-k)!}$

$C_{26,3} = \frac{26!}{3!(26-3)!}$

Hint: For each problem, determine if order matters, and if it should be calculated with or without replacement.

In [56]:
```
print ('a : ',26**3 * 10**4)
print ('b : ',np.math.factorial(9))
print ('c : ',9*8*7*6)
print ('d : ',np.math.factorial(26)/(np.math.factorial(3)*np.math.factorial(23)))
```

```
a :  175760000
b :  362880
c :  3024
d :  2600.0
```

Linear Algebra

6

Matrix manipulations and multiplication. Machine learning involves working with many matrices, so this exercise will provide you with the opportunity to practice those skills.

Let $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 4 \\ -3 \\ 6 \end{bmatrix}$, and $\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Compute the following or indicate that it cannot be computed:

1. $\mathbf{AA}$
2. $\mathbf{AA}^T$
3. $\mathbf{Ab}$
4. $\mathbf{Ab}^T$
5. $\mathbf{bA}$
6. $\mathbf{b}^T\mathbf{A}$
7. $\mathbf{bb}$
8. $\mathbf{b}^T\mathbf{b}$
9. $\mathbf{bb}^T$
10. $\mathbf{b} + \mathbf{c}^T$
11. $\mathbf{b}^T\mathbf{b}^T$
12. $\mathbf{A}^{-1}\mathbf{b}$
13. $\mathbf{A} \circ \mathbf{A}$
14. $\mathbf{b} \circ \mathbf{c}$

Note: The element-wise (or Hadamard) product is the product of each element in one matrix with the corresponding element in another matrix, and is represented by the symbol "$\circ$".

ANSWER

In [57]:
```
A = np.matrix([(1,2,3),
               (2,4,5),
               (3,5,6)])

I = np.matrix([(1,0,0),
               (0,1,0),
               (0,0,1)])

b = np.array([-1,3,8])

c = np.array([4,-3,6])

# 1)
print('AA = \n', A*A)
```

```
AA =
 [[14 25 31]
 [25 45 56]
 [31 56 70]]
```

In [58]:
```
# 2)
print('AAt = \n', A*A.T) #The result is going to be tha same given the A matrix in symme
trical
```

```
AAt =
 [[14 25 31]
 [25 45 56]
 [31 56 70]]
```

In [59]:
```
# 3)
print('Ab = \n', A.dot(b)) #is not going to be possible to multiplicate A*b given numpy
 assumes that b is an row vector despite is notated as a column vector.
# In other words a 3x3 * 1x3 is not a valid operation, but in reality we are multiplicat
ing a 3x3 * 3x1. To multiply in this case we need to use the .dot function or transform
 b into a matrix

print('or \n')
b = np.matrix([-1,3,8])
b=b.T #in order to make this matrix in a column vector
A*b
```

```
Ab =
 [[29 50 60]]
or
```

Out[59]:
```
matrix([[29],
        [50],
        [60]])
```

4) If we consider the vector swapping from column vector to a row vector given the transposition, we are not going to be able to multiply Ab^t (3x3 * 1x3)

5) If we consider the vector as a column vector, we are not going to be able to multiply bA. (3x1 * 3x3)

6) If we consider the vector swapping from column vector to a row vector given the transposition, we can multiply $b^t A$ (1x3 * 3x3)
$b^t A = [29, 50, 60]$

7) bb is not possible tu multiply given that is a 3 x1* 3x1 operation

In [60]:
```
# 8) bt.b Is possible. 1x3 * 3x1
print('bb = \n', b.T*b)
```

```
bb =
 [[74]]
```

In [61]:
```
# 9) b.bt is possile. 3x1 * 1x3
print('bbt = \n', b*b.T)
```

```
bbt =
 [[ 1 -3 -8]
 [-3  9 24]
 [-8 24 64]]
```

In [62]:
```
#10) b + ct. In the sum the orientation of vectors is not important
b =  np.array([-1,3,8])
b + c
```

Out[62]:
```
array([ 3,  0, 14])
```

11) $b_t b_t$ is not going to be possible $(1x3 * 1x3)$

In [63]:
```
# 12) A-1 . b
b = np.matrix([-1,3,8])
b=b.T #in order to make this matrix in a column vector
from scipy import linalg
linalg.inv(A)* b
```

Out[63]:
```
matrix([[ 6.],
        [ 4.],
        [-5.]])
```

In [64]:
```
# 13) A * A
np.multiply(A,A)
```

Out[64]:
```
matrix([[ 1,  4,  9],
        [ 4, 16, 25],
        [ 9, 25, 36]])
```

In [65]:
```
# 14) b * c
b = np.array([-1,3,8])
np.multiply(b,c)
```

Out[65]:
```
array([-4, -9, 48])
```

6

Eigenvectors and eigenvalues. Eigenvectors and eigenvalues are useful for some machine learning algorithms, but the concepts take time to solidly grasp. For an intuitive review of these concepts, explore this interactive website at Setosa.io (http://setosa.io/ev/eigenvectors-and-eigenvalues/). Also, the series of linear algebra videos by Grant Sanderson of 3Brown1Blue are excellent and can be viewed on youtube here (https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab).

1. Calculate the eigenvalues and corresponding eigenvectors of matrix $\mathbf{A}$ above, from the last question.
2. Choose one of the eigenvector/eigenvalue pairs, $\mathbf{v}$ and λ, and show that $\mathbf{Av} = \lambda\mathbf{v}$. Also show that this relationship extends to higher orders: $\mathbf{AAv} = \lambda^2\mathbf{v}$
3. Show that the eigenvectors are orthogonal to one another (e.g. their inner product is zero). This is true for real, symmetric matrices.

ANSWER

1) First step find eigenvalues of the A matrix.

[x] A should be square

In [66]:
```
print(np.linalg.eig(A))
w,v = np.linalg.eig(A)
v = v.T #trasposition makes easier the indexing
```

```
(array([11.34481428, -0.51572947,  0.17091519]), matrix([[-0.32798528, -0.73697623,  0.
59100905],
        [-0.59100905, -0.32798528, -0.73697623],
        [-0.73697623,  0.59100905,  0.32798528]]))
```

2)

In [67]:
```
eval0 = w[0]
evec0 = v[0]

eval1 = w[1]
evec1 = v[1]

eval2 = w[2]
evec2 = v[2]
```

In [68]:
```
A*evec0.T
```

Out[68]:
```
matrix([[-3.72093206],
        [-6.70488789],
        [-8.36085845]])
```

In [69]:
```
eval0*evec0.T
```

Out[69]:
```
matrix([[-3.72093206],
        [-6.70488789],
        [-8.36085845]])
```

In [70]:
```
np.allclose(A*evec0.T, eval0*evec0.T) # Comparing arrays element-wise in numpy is weird.

                                      # Despite that is obvious that the values are the
 same the comparision operator == returned false an all values
```

Out[70]:
```
True
```

Extending the relationship

In [71]:
```
A*A*evec0.T
```

Out[71]:
```
matrix([[-42.2132832 ],
        [-76.06570795],
        [-94.85238636]])
```

In [72]:
```
(eval0**2)*evec0.T
```

Out[72]:
```
matrix([[-42.2132832 ],
        [-76.06570795],
        [-94.85238636]])
```

In [73]:
```
np.allclose(A*A*evec0.T,(eval0**2)*evec0.T)
```

Out[73]:
```
True
```

3) Eigenvetors are orthogonal to each other

In [74]:
```
print('eigenvector 1 * eigenvector 2 : %.4f ' % (evec0 * evec1.T)[0])
```

```
eigenvector 1 * eigenvector 2 : -0.0000
```

In [75]:
```
print('eigenvector 1 * eigenvector 3 : %.4f ' % (evec0 * evec2.T)[0])
```

```
eigenvector 1 * eigenvector 3 : -0.0000
```

In [76]:
```
print('eigenvector 2 * eigenvector 3 : %.4f ' % (evec1 * evec2.T)[0])
```

```
eigenvector 2 * eigenvector 3 : -0.0000
```

Numerical Programming

7

Speed comparison between vectorized and non-vectorized code. Begin by creating an array of 10 million random numbers using the numpy random.randn module. Compute the sum of the squares first in a for loop, then using Numpy's `dot` module. Time how long it takes to compute each and report the results and report the output. How many times faster is the vectorized code than the for loop approach?

*Note: all code should be well commented, properly formatted, and your answers should be output using the `print()` function as follows (where the # represents your answers, to a reasonable precision):

```
Time [sec] (non-vectorized): ######

Time [sec] (vectorized):     ######

The vectorized code is ##### times faster than the vectorized code
```

ANSWER

In [39]:
```
import numpy as np
import time

# Generate the random samples
samples = np.random.rand(10**7)

# Compute the sum of squares the non-vectorized way (using a for loop)
# First I need to calculate the mean. (Ask if they want me to calculate also the mean in
 a for loop or in a vectorized way)
mean = sum(samples)/10**7

start = time.time()
sumsq_novec = 0
for i in samples:
    sumsq_novec += (i - mean)**2
end = time.time()
t_novec =(end - start)



# Compute the sum of squares the vectorized way (using numpy)

start = time.time()
sumsq_vec = sum((samples - mean)**2)
end = time.time()
t_vec =(end - start)



# Print the results
print('Time [sec] (non-vectorized): %f' % t_novec)

print('Time [sec] (vectorized): %f' % t_vec)

print('The vectorized code is '+ str(round(t_novec/t_vec,3)) + ' times faster than the v
ectorized code')
```

```
Time [sec] (non-vectorized): 5.329760
Time [sec] (vectorized): 1.648585
The vectorized code is 3.233 times faster than the vectorized code
```

8

One popular Agile development framework is Scrum (a paradigm recommended for data science projects). It emphasizes the continual evolution of code for projects, becoming progressively better, but starting with a quickly developed minimum viable product. This often means that code written early on is not optimized, and that's a good thing - it's best to get it to work first before optimizing. Imagine that you wrote the following code during a sprint towards getting an end-to-end system working. Vectorize the following code and show the difference in speed between the current implementation and a vectorized version.

The function below computes the function $f(x, y) = x^2 - 2y^2$ and determines whether this quantity is above or below a given threshold, `thresh=0`. This is done for $x, y \in \{-4, 4\}$, over a 2,000-by-2,000 grid covering that domain.

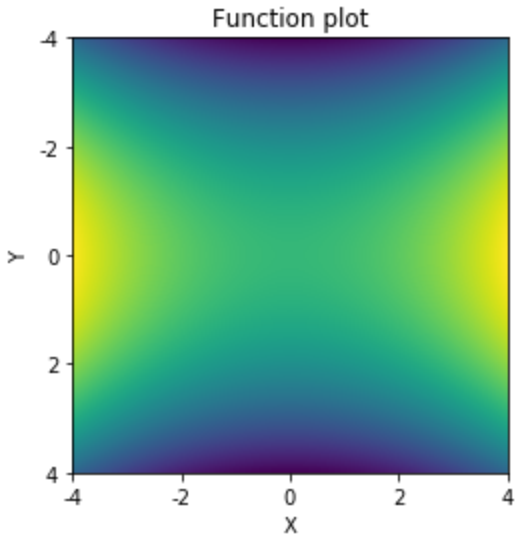
(a) Vectorize this code and demonstrate (as in the last exercise) the speed increase through vectorization and (b) plot the resulting data - both the function $f(x, y)$ and the thresholded output - using `imshow` (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.imshow.html?highlight=matplotlib%20pyplot%20imshow#matplotlib.pyplot.imshow) from `matplotlib`.

Hint: look at the `numpy` `meshgrid` (https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.meshgrid.html) documentation

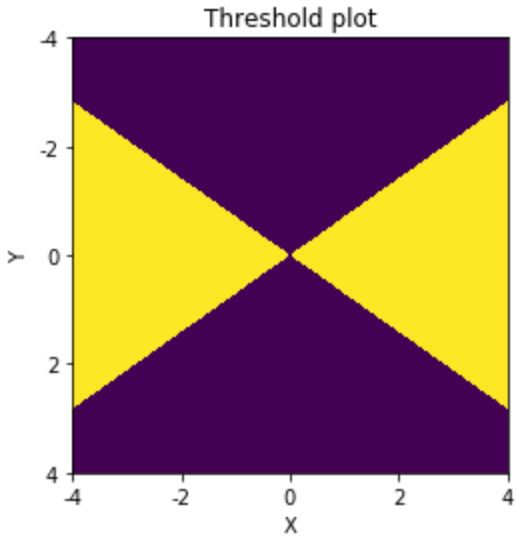
In [142]:
```
import numpy as np
import time
import matplotlib.pyplot as plt

# Initialize variables for this exerise

def f(x,y):
    return ((x**2) - (2*y**2))

thresh = np.zeros((2000, 2000))  # matrix of zeros for the non-vectorized implementation
threshv = np.zeros((2000, 2000)) # matrix of zeros for the vectorized implementation
xspace = np.linspace(-4, 4, 2000) #x vector with values from -4,4
yspace = np.linspace(-4, 4, 2000) #y vector with values from -4,4
xv,yv = np.meshgrid(xspace,yspace) # A N-D coordinate arrays for vectorized evaluations

# Nonvectorized implementation
start = time.time()
for y in range(0,2000):
    for x in range(0,2000):
        if f(xspace[x],yspace[y]) > 0:
            thresh[x][y] = 1
        elif f(xspace[x],yspace[y]) == 0:
            thresh[x][y] = 0
        elif f(xspace[x],yspace[y]) < 0:
            thresh[x][y] = -1

end = time.time()
t_novec = end - start

# Vectorized implementation
start_v= time.time()
values = f(xv,yv)
threshv[values > 0] = 1
threshv[values == 0] = 1
threshv[values < 0] = -1
end_v = time.time()
t_vec  = end_v - start_v

# Print the time for each and the speed increase
print('Time [sec] (non-vectorized): %f' % t_novec)

print('Time [sec] (vectorized): %f' % t_vec)

print('The vectorized code is '+ str(round(t_novec/t_vec,3)) + ' times faster than the v
ectorized code')
```

```
Time [sec] (non-vectorized): 28.518381
Time [sec] (vectorized): 0.146388
The vectorized code is 194.814 times faster than the vectorized code
```

In [143]:
```
# Plot the result
plt.title('Function plot')
plt.xlabel('X')
plt.ylabel('Y')
plt.xticks(np.linspace(0,2000,5),['-4','-2','0','2','4'])
plt.yticks(np.linspace(0,2000,5),['-4','-2','0','2','4'])
val = plt.imshow(values)
```



In [144]:
```
plt.title('Threshold plot')
plt.xlabel('X')
plt.ylabel('Y')
plt.xticks(np.linspace(0,2000,5),['-4','-2','0','2','4'])
plt.yticks(np.linspace(0,2000,5),['-4','-2','0','2','4'])
image = plt.imshow(threshv)
```



9

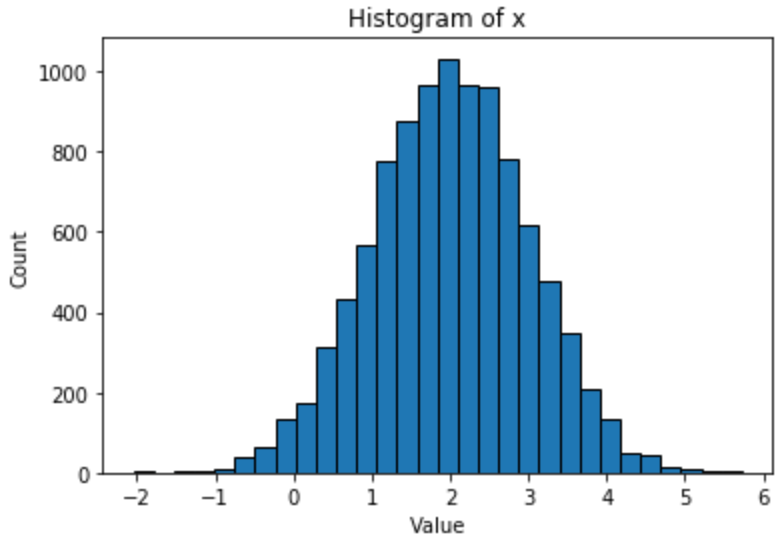
This exercise will walk through some basic numerical programming exercises.

1. Synthesize $n = 10^4$ normally distributed data points with mean $\mu = 2$ and a standard deviation of $\sigma = 1$. Call these observations from a random variable X, and call the vector of observations that you generate, $\mathbf{x}$.
2. Calculate the mean and standard deviation of $\mathbf{x}$ to validate (1) and provide the result to a precision of four significant figures.
3. Plot a histogram of the data in $\mathbf{x}$ with 30 bins
4. What is the 90th percentile of $\mathbf{x}$? The 90th percentile is the value below which 90% of observations can be found.
5. What is the 99th percentile of $\mathbf{x}$?
6. Now synthesize $n = 10^4$ normally distributed data points with mean $\mu = 0$ and a standard deviation of $\sigma = 3$. Call these observations from a random variable Y, and call the vector of observations that you generate, $\mathbf{y}$.
7. Plot the histogram of the data in $\mathbf{y}$ on a (new) plot with the histogram of $\mathbf{x}$, so that both histograms can be seen and compared.
8. Using the observations from $\mathbf{x}$ and $\mathbf{y}$, estimate $E[XY]$

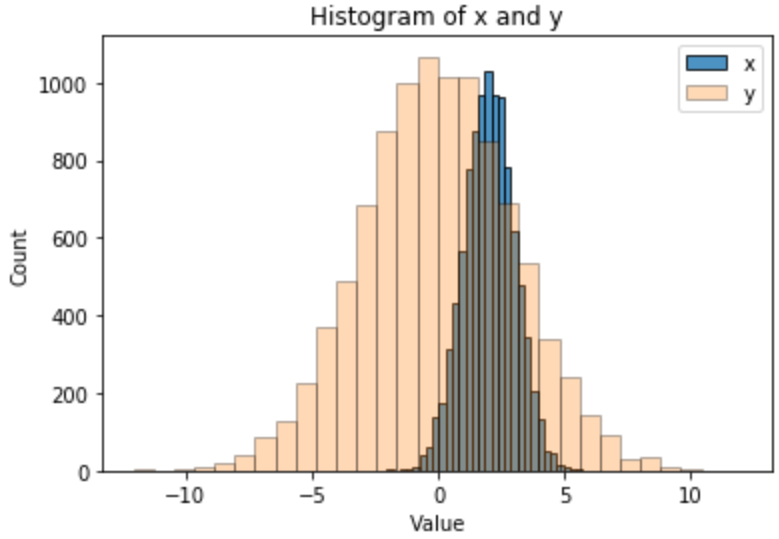
ANSWER

In [13]:
```
# 1
x = np.random.normal(2,1,10**4)
# 2
print('Mean: %.4f' % x.mean())
print('SD: %.4f' % x.std())
# 3
plt.title('Histogram of x')
plt.xlabel('Value')
plt.ylabel('Count')
plt.hist(x,bins=30,ec='black')
plt.show()
# 4
print('Percentile 90: %.4f'% np.percentile(x,90))
# 5
print('Percentile 99: %.4f'% np.percentile(x,99))
# 6
y = np.random.normal(0,3,10**4)
# 7
plt.title('Histogram of x and y')
plt.xlabel('Value')
plt.ylabel('Count')
plt.hist(x,bins=30,alpha=0.8,label= 'x',ec='black')
plt.hist(y,bins=30,alpha=0.3,label= 'y',ec='black')
plt.legend(loc='upper right')
plt.show()
# 8
```

```
Mean: 1.9975
SD: 0.9961
```




```
Percentile 90: 3.2826
Percentile 99: 4.2644
```



9.8) Given that both distributions are independent the Expected value could be calculated as $E|X| * E|Y|$. Given that the expected value of a normal distribution with $\sigma = 0$ is 0. Then $E|XY|$ would be ideally equal to 0.

But, given that the values here are beeing randomly generated we could aproximate to this value as $\frac{1}{1000} = \sum_{i=0}^{n=999} X_i * Y_i$

10

Estimate the integral of the function $f(x)$ on the interval $0 \leq x < 2.5$ assuming we only know the following points from f:

Table 1. Dataset containing n=5 observations

x_i	**0.0**	**0.5**	**1.0**	**1.5**	**2.0**
y_i	6	7	8	4	1

ANSWER We can use the Riemann sum aproximation (https://en.wikipedia.org/wiki/Riemann_sum) to solve this problem. We would start from the left and we would try to recreate the area of the function using rectangules using the known points.

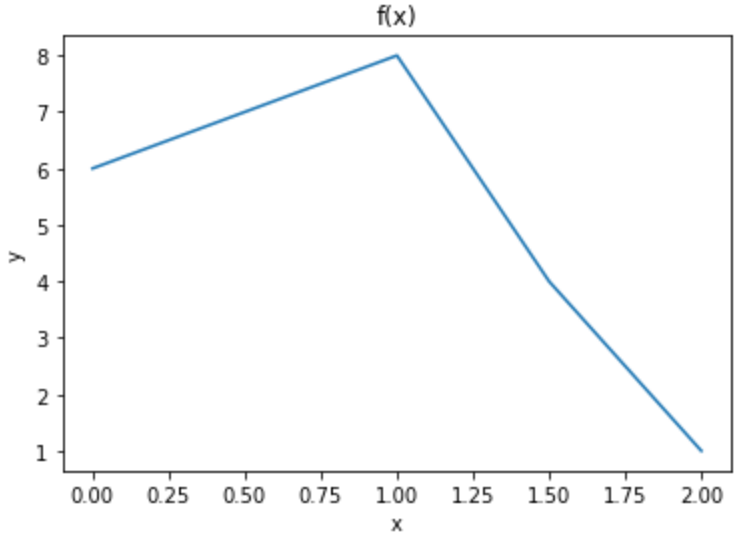
The interval [a, b] is therefore divided into n subintervals, each of length $\Delta x = \frac{b-a}{n}$

So in this case $\Delta x = \frac{2.5-0}{5} = 0.5$

The Riemann sum could be defined as: $S = \sum_{i=1}^{n-1} f(x_i)\,\Delta x_i$

In [77]:
```
x = (0.0 , 0.5 , 1.0 , 1.5 , 2.0 )
y = (6 , 7 , 8 , 4 , 1 )
interval = 0.5
S = 0
for i in range(0,len(x)):
    S += y[i]*interval

plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('f(x)')
print ('The estimate integral of the function f(x) is = %.1f' %S)
```

```
The estimate integral of the function f(x) is = 13.0
```



Version Control via Git

11

Complete the Atlassian Git tutorial (https://www.atlassian.com/git/tutorials/what-is-version-control), specifically the following sections. Try each concept that's presented. For this tutorial, instead of using BitBucket, use Github. Create a github account here if you don't already have one: https://github.com/ (https://github.com/)

1. What is version control (https://www.atlassian.com/git/tutorials/what-is-version-control)
2. What is Git (https://www.atlassian.com/git/tutorials/what-is-git)
3. Install Git (https://www.atlassian.com/git/tutorials/install-git)
4. Setting up a repository (https://www.atlassian.com/git/tutorials/install-git)
5. Saving changes (https://www.atlassian.com/git/tutorials/saving-changes)
6. Inspecting a repository (https://www.atlassian.com/git/tutorials/inspecting-a-repository)
7. Undoing changes (https://www.atlassian.com/git/tutorials/undoing-changes)
8. Rewriting history (https://www.atlassian.com/git/tutorials/rewriting-history)
9. Syncing (https://www.atlassian.com/git/tutorials/syncing)
10. Making a pull request (https://www.atlassian.com/git/tutorials/making-a-pull-request)
11. Using branches (https://www.atlassian.com/git/tutorials/using-branches)
12. Comparing workflows (https://www.atlassian.com/git/tutorials/comparing-workflows)

For your answer, affirm that you either completed the tutorial or have previous experience with all of the concepts above. Do this by typing your name below and selecting the situation that applies from the two options in brackets.

ANSWER

I, Joaquin Menendez, affirm that I have completed the above tutorial

12

Using Github to create a static HTML website:

1. Create a branch in your `machine-learning-course` repo called "gh-pages" and checkout that branch (this will provide an example of how to create a simple static website using Github Pages (https://pages.github.com/))
2. Create a file called "index.html" with the contents "Hello World" and add, commit, and push it to that branch.
3. Submit the following: (a) a link to your github repository and (b) a link to your new "Hello World" website. The latter should be at the address https://[USERNAME].github.io/ECE590-assignment0 (https://[USERNAME].github.io/ECE590-assignment0) (where [USERNAME] is your github username).

ANSWER

'Hello World' Webpage (https://joaquinmenendez.github.io/ECE590-assignment0/)

Github repository (https://github.com/joaquinmenendez/joaquinmenendez.github.io)

Exploratory Data Analysis

13

Here you'll bring together some of the individual skills that you demonstrated above and create a Jupyter notebook based blog post on data analysis.

1. Find a dataset that interests you and relates to a question or problem that you find intriguing
2. Using a Jupyter notebook, describe the dataset, the source of the data, and the reason the dataset was of interest.
3. Check the data and see if they need to be cleaned: are there missing values? Are there clearly erroneous values? Do two tables need to be merged together? Clean the data so it can be visualized.
4. Plot the data, demonstrating interesting features that you discover. Are there any relationships between variables that were surprising or patterns that emerged? Please exercise creativity and curiosity in your plots.
5. What insights are you able to take away from exploring the data? Is there a reason why analyzing the dataset you chose is particularly interesting or important? Summarize this as if your target audience was the readership of a major news organization - boil down your findings in a way that is accessible, but still accurate.
6. Create a public repository on your github account titled "machine-learning-course". In it, create a readme file that contains the heading "ECE590: Introductory Machine Learning for Data Science". Add, commit, and push that Jupyter notebook to the master branch. Provide the link to the that post here.

ANSWER

Exploratory Data Analysis - Admission to graduate progams (https://github.com/joaquinmenendez/machine-learning-course)