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

Kate Coulter

Netid: kvc6

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

$$\int_0^2 ax^2 \, dx = 1$$
$$\frac{a}{3}x^3\Big|_0^2 = \frac{8}{3}a = 1$$
$$a = \frac{3}{8}$$

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

Note the definition for a CDF $F(x)$ given a PDF $f(x)$ is: $F_x(t) = P(x \leq t)$. Thus we know the following:

- If $x < 0$, then $f_x(t) = 0$ for all $t \leq x$ so that we have
$$F(x) = \int_{-\infty}^{0} f(t)\,dt = \int_{-\infty}^{0} 0\,dt = 0$$
- If $0 < x < 3$, then $f_x(t) = \frac{1}{3}$ for all $0 < t \leq x$ so that we have
$$F(x) = F(0) + \int_{0}^{x} \frac{1}{3}\,dt = 0 + \frac{1}{3}t\Big|_{t=0}^{x} = \frac{1}{3}x$$
- If $x \geq 3$, then $f_x(t) = 0$ for all $3 \leq t \leq x$ so that we have
$$F(x) = F(3) + \int_{3}^{x} f(t)\,dt = 1$$

Which results in the following CDF:

$$F(x) = \begin{cases} 0 & x \leq 0 \\ \frac{1}{3}x & 0 < x < 3 \\ 1 & x \geq 3 \end{cases}$$

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

(a) Using the definition of **expected value**,

$$\begin{aligned} \mathrm{E}(x) &= \int_{-\infty}^{\infty} xf(x)\,dx \\ &= \int_{-\infty}^{\infty} \frac{1}{3}x\,dx = \int_{0}^{3} \frac{1}{3}x\,dx \\ &= \frac{1}{6}x^2\Big|_{x=0}^{3} = \frac{1}{6}(3^2) - 0 \\ &= \frac{3}{2} \end{aligned}$$

(b) Using the definition of **variance**,

$$\begin{aligned} \mathrm{Var}(x) &= \int_{-\infty}^{\infty} [x - E(x)]^2 f(x)\,dx \\ &= \int_{0}^{3} \left(x - \frac{3}{2}\right)^2 \frac{1}{3}\,dx \\ &= \int_{0}^{3} \frac{1}{3}\left(x^2 - 3x + \frac{9}{4}\right) \\ &= \frac{1}{3}\left[\left(\frac{1}{3}x^3 - \frac{3}{2}x^2 + \frac{9}{4}x\right)\Big|_{x=0}^{3}\right] \\ &= \frac{9}{4} - \frac{9}{2} - \frac{9}{3} \\ &= \frac{3}{4} \end{aligned}$$

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
Note that for a finite list of discrete numbers, we will use the following definitions:

- The **mean**, $\bar{x}$, of that list is defined as:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

  where each data point is represented by x_i and the amount of data points in the list is represented by n. Note that the term **average** also can be used to refer to the mean value.

- The **median** of that list is an order statistic that gives the "middle" value, denoted as $\tilde{x}$. Specifically, when n is an odd number, $\tilde{x}$ is the value such that an equal number of samples are less than and greater than the value or, when n is an even number, $\tilde{x}$ is the average of the two central values.

- The **variance**, s^2, of that list is defined as:

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

Note that the variance s^2 defined above is not an unbiased estimator for the population variance σ^2. In order to obtain an unbiased estimator for σ^2, the following computation must be used:

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

We will now find the mean and variance values for the given discrete data vector.
(a)

$$\begin{aligned}\bar{x} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ &= \frac{1}{5}(2 + 3 + 10 + (-1) + (-1)) \\ &= \frac{13}{5} \\ \bar{x} &= 2.6\end{aligned}$$

(b)

$$
\begin{aligned}
s^2 &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \\
&= \frac{1}{5}\left[(2-2.6)^2 + (3-2.6)^2 + (10-2.6)^2 + (-1-2.6)^2 + (-1-2.6)^2\right] \\
&= \frac{0.36 + 0.16 + 54.76 + 12.96 + 12.96}{5} \\
&= \frac{81.2}{5} \\
s^2 &= 16.24
\end{aligned}
$$

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) Assuming repetition is allowed, so 26 choices for each letter position and 10 choices for each number position, which is given by:

$$26 \cdot 26 \cdot 26 \cdot 10 \cdot 10 \cdot 10 \cdot 10 = (26^3)(10^4) = 175,760,000$$

(b) Total possible ways to arrange 9 players is given by:

$$9! = 362,880$$

(c) Total possible ways to choose 5 from 9 is given by:

$$\frac{n!}{(n-k)!} = \frac{9!}{(9-5)!} = \frac{9!}{4!} = 15,120$$

(d) Binomial coefficient problem, given by:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{26!}{3!(26-3)!} = \frac{26 \cdot 25 \cdot 24}{6} = 2,600$$

Linear Algebra

6 - part (a)

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 1. $\mathbf{AA} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 14 & 25 & 31 \\ 25 & 45 & 56 \\ 31 & 56 & 70 \end{bmatrix}$

2. $\mathbf{AA}^T = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 14 & 25 & 31 \\ 25 & 45 & 56 \\ 31 & 56 & 70 \end{bmatrix}$

3. $\mathbf{Ab} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}\begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} = \begin{bmatrix} 29 \\ 50 \\ 60 \end{bmatrix}$

4. $\mathbf{Ab}^T$ cannot be computed because in order for two matrices to be multiplied the number of columns of the first matrix must equal the number of rows of the second matrix. In this case, the first matrix, **A**, has 3 columns whereas the second matrix, $\mathbf{b^T} = [-1 \;\; 3 \;\; 8]$, only has 1 row.

5. $\mathbf{bA}$ cannot be computed due to the dimensions of the matrices as $\mathbf{b}$ has 1 column while $\mathbf{A}$ has 3 rows.

6. $\mathbf{b}^T\mathbf{A} = \begin{bmatrix} 1 & 3 & 8 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 29 & 50 & 60 \end{bmatrix}$

7. $\mathbf{bb}$ cannot be computed due to the dimensions of the matrices as $\mathbf{b}$ has 1 column while $\mathbf{b}$ has 3 rows.

8. $\mathbf{b}^T\mathbf{b} = \begin{bmatrix} 1 & 3 & 8 \end{bmatrix} \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} = 74$

9. $\mathbf{b}\mathbf{b}^T = \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} \begin{bmatrix} 1 & 3 & 8 \end{bmatrix} = \begin{bmatrix} 1 & -3 & -8 \\ -3 & 9 & 24 \\ -8 & 24 & 64 \end{bmatrix}$

10. $\mathbf{b} + \mathbf{c}^T$ cannot be computed due to the dimensions of the matrices not being equal, as $\mathbf{b}$ has dimensions $(3x1)$ and $\mathbf{c}^T$ has dimensions $(1x3)$.

11. $\mathbf{b}^T\mathbf{b}^T$ cannot be computed due to the dimensions of the matrices as $\mathbf{b}^T$ has 3 columns while $\mathbf{b}^T$ has 1 row.

12. $\mathbf{A}^{-1}\mathbf{b}$: to begin, we will need to find $\mathbf{A}^{-1}$. To find $\mathbf{A}^I$ we will use the Gauss-Jordan method on the following property:

$$[\ \mathbf{A}\ |\ \mathbf{I}\ ] = [\ \mathbf{I}\ |\ \mathbf{A}^{-1}\ ]$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 4 & 5 & 0 & 1 & 0 \\ 3 & 5 & 6 & 0 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 3 & 5 & 6 & 0 & 0 & 1 \\ 2 & 4 & 5 & 0 & 1 & 0 \\ 1 & 2 & 3 & 1 & 0 & 0 \end{array}\right]$$

We will transform this matrix into row echelon form using the following elementary row operations: 1. $R_1 = R_3$ 2. $R_2 = R_2 - \frac{2}{3}R_1$ 3. $R_3 = R_3 - \frac{1}{3}R_1$ 4. $R_3 = R_3 - \frac{1}{2}R_2$

$$\xrightarrow{reduced} \left[\begin{array}{ccc|ccc} 3 & 5 & 6 & 0 & 0 & 1 \\ 0 & 2/3 & 1 & 0 & 1 & -2/3 \\ 0 & 0 & 1/2 & 2 & -1 & 0 \end{array}\right]$$

We will now transform our row echelon form matrix to its reduced form using the following operations: 5. $R_3 = 2R_3$ 6. $R_2 = R_2 - R_3$ 7. $R_1 = R_1 - 6R_3$ 8. $R_2 = \frac{3}{2}R_2$ 9. $R_1 = R_1 - 5R_2$ 10. $R_1 = \frac{1}{3}R_1$

$$\xrightarrow{reduced} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & -3 & 2 \\ 0 & 1 & 0 & -3 & 3 & -1 \\ 0 & 0 & 1 & 2 & -1 & 0 \end{array}\right]$$

Thus we know $\mathbf{A}^{-1} = \begin{bmatrix} 1 & -3 & 2 \\ -3 & 3 & -1 \\ 2 & -1 & 0 \end{bmatrix}$

$$\mathbf{A}^{-1}\mathbf{b} = \begin{bmatrix} 1 & -3 & 2 \\ -3 & 3 & -1 \\ 2 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 5 \end{bmatrix}$$

13. $\mathbf{A} \circ \mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \circ \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 9 \\ 4 & 16 & 25 \\ 9 & 25 & 36 \end{bmatrix}$

14. $\mathbf{b} \circ \mathbf{c} = \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} \begin{bmatrix} 4 \\ -3 \\ 6 \end{bmatrix} = \begin{bmatrix} -4 \\ -9 \\ 48 \end{bmatrix}$

6 - part (b)

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).

ANSWER

1. Calculate the eigenvalues and corresponding eigenvectors of matrix $\mathbf{A}$ above, from the last question.

As shown below, for matrix $\mathbf{A}$ we have eigenvalues $\lambda_1 = 11.3448$, $\lambda_2 = -0.5157$, and $\lambda_3 = 0.1709$ with corresponding eigenvectors $v_1 = (-0.328, -0.591, -0.737)$, $v_2 = (-0.737, -0.328, 0.591)$, and $v_3 = (0.591, -0.737, 0.328)$.

In [4]:
```
#1.
import numpy as np
import numpy.linalg as la
#create matrix array in python of A
A = np.array([[1,2,3],[2,4,5],[3,5,6]])
#get eigenvalues and eigenvectors for A
w = la.eig(A)[0]
v = la.eig(A)[1]
#print transpose of v so eigenvectors are in rows
print('Eigenvectors of A where each ith row represents the ith eigenvector: ',np.around(np.ma
trix.transpose(v), 4))
print('Eigenvalues of A where each ith entry corresponds to the ith eigenvector: ',np.around(
w, 4))
```

```
Eigenvectors of A where each ith row represents the ith eigenvector:  [[-0.328 -0.591 -0.73
7]
 [-0.737 -0.328  0.591]
 [ 0.591 -0.737  0.328]]
Eigenvalues of A where each ith entry corresponds to the ith eigenvector:  [11.3448 -0.5157
0.1709]
```

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: $\mathbf{AAv} = \lambda^2\mathbf{v}$

In [5]:
```
# 2 choose first eigenvalue and eigenvector from question 1
v0 = v[:,0]
lam0 = w[0]
#show that Av = lam*u
Av = np.dot(A,v0)
lamv = lam0*v0
print('Av = ', Av)
print('lamv = ', lamv)
Av == lamv
# note that Av == lamv does not compute as True currently due to
#rounding issues, so we will need to round our values as we did before
Av_rd = np.around(Av, 4)
lamv_rd = np.around(lamv, 4)
#to confirm now equal
print('Show that Av = lamv: ', Av_rd == lamv_rd)

#for higher orders show AAv = lam^2v
#will use same eigenvalue and eigenvector as before
AAv = np.dot(np.matmul(A,A),v0)
lamsqv = lam0**2 * v0
print('AAv = ', AAv)
print('lambda^2 v', lamsqv)
#look equal, but will have same problem as above
#with rounding errors, so will check bool after rounded
AAv_rd = np.around(AAv, 4)
lamsqv_rd = np.around(lamsqv, 4)
print('Show that relationship extends to higher orders: ', AAv_rd == lamsqv_rd)
```

```
Av =  [-3.72093206 -6.70488789 -8.36085845]
lamv =  [-3.72093206 -6.70488789 -8.36085845]
Show that Av = lamv:  [ True  True  True]
AAv =  [-42.2132832  -76.06570795 -94.85238636]
lambda^2 v [-42.2132832  -76.06570795 -94.85238636]
Show that relationship extends to higher orders:  [ True  True  True]
```

3. Show that the eigenvectors are orthogonal to one another (e.g. their inner product is zero). This is true for real, symmetric matrices.

In [6]:
```
# 3
#to show the eigenvectors are orthogonal (or perpendicular) we
#show that each dot product is equal to zero

#create three eigenvectors labeled in the same way as part 1
v1 = v[:,0]
v2 = v[:,1]
v3 = v[:,2]
#make sure worked
print('Eigenvector 1: ', v1)
print('Eigenvector 2: ', v2)
print('Eigenvector 3: ', v3)
#dot product of eigenvector 1 and 2
print('v1 dot v2: ', np.dot(v1, v2))
print('v1 dot v3: ', np.dot(v1, v3))
print('v3 dot v2: ', np.dot(v3, v2))
#while these values are not zero, it is due to rounding issues
# as we saw in the other parts of this section, so will round to our
# previous standard of 4 decimal places, which gives the following equality comparison
print('v1 dot v2 == 0: ', np.around(np.dot(v1,v2))==0)
print('v1 dot v3 == 0: ', np.around(np.dot(v1,v3))==0)
print('v3 dot v2 == 0: ', np.around(np.dot(v3,v2))==0)
```

```
Eigenvector 1:  [-0.32798528 -0.59100905 -0.73697623]
Eigenvector 2:  [-0.73697623 -0.32798528  0.59100905]
Eigenvector 3:  [ 0.59100905 -0.73697623  0.32798528]
v1 dot v2:  -1.1102230246251565e-16
v1 dot v3:  -3.885780586188048e-16
v3 dot v2:  -5.828670879282072e-16
v1 dot v2 == 0:  True
v1 dot v3 == 0:  True
v3 dot v2 == 0:  True
```

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 [7]:
```
import numpy as np
import time

# Generate the random samples
N = 10000000
array = np.random.randn(N)

# Compute the sum of squares the non-vec way (using a for loop)
# first create new array containing squared values
sqarray = []
for i in array:
    sqarray.append(i**2)
# make sure that it worked
array[:3]
sqarray[:3]
# the values are now squared, so will sum these values
sqsum = 0
for i in sqarray:
    sqsum += i
#we will combine these two for loops into one so that its
#time can be accurately recorded
#time first method
start_nonvec = time.time()
sqsum = 0
for i in array:
    sqsum += i**2
run_nonvec = time.time() - start_nonvec
# round to 4 decimal places for readability
time_nonvec = np.around(sqsum, decimals = 4)
```

In [14]:
```
# Compute the sum of squares the vectorized way (using numpy)
time_vec = np.dot(array, array)
#time second method
start_vec = time.time()
time_vec = np.dot(array, array)
run_vec = time.time() - start_vec
# round for same reason
time_vec = np.around(time_vec, decimals=4)

#compare the two recorded times
howfast = run_nonvec / run_vec
#round
howfast = np.around(howfast, decimals=4)
howfast
# Print the results
print('Value using non-vectorized method:', time_nonvec)
print('Value using vectorized method:', time_vec)
print('Time [sec] (non-vectorized):', np.around(run_nonvec,4))
print('Time [sec] (vectorized):', np.around(run_vec,4))
print('The vectorized code is ', howfast, ' times faster than the vectorized code')
```

```
Value using non-vectorized method: 10004545.8828
Value using vectorized method: 10004545.8828
Time [sec] (non-vectorized): 6.1983
Time [sec] (vectorized): 0.0045
The vectorized code is  1367.0647  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 [16]:
```
import numpy as np
import time
import matplotlib.pyplot as plt
import matplotlib as mpl

mpl.rcParams['figure.figsize'] = (9,6)
plt.rcParams.update({'font.size': 13})

# Initialize variables for this exerise
n = 2000
xvalues = np.linspace(-4,4,n)
yvalues = np.linspace(-4,4,n)
thresh = 0

# Nonvectorized implementation
t0 = time.time()
f = np.zeros((n,n))
f_thresholded = np.zeros((n,n))
for ix, x in enumerate(xvalues):
    for iy, y in enumerate(yvalues):
        f[ix,iy]             = x**2 - 2 * y**2
        f_thresholded[ix,iy] = f[ix,iy] > thresh
t1 = time.time()
time_nonvectorized = t1 - t0
```

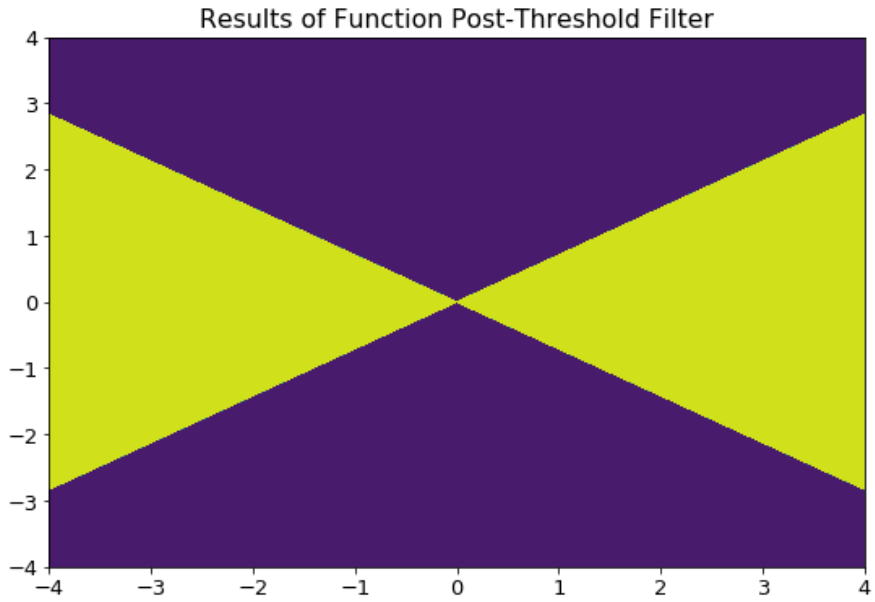
In [17]:
```
# Vectorized implementation
t0 = time.time()
xv, yv = np.meshgrid(xvalues, yvalues)
function = xv**2 - 2*yv**2
threshold = function > thresh
t1 = time.time()
time_vectorized = t1 - t0

# Print the time for each and the speed increase
howfast = time_nonvectorized / time_vectorized

print('Time [sec] (non-vectorized): ', np.around(time_nonvectorized,4))
print('Time [sec] (vectorized): ', np.around(time_vectorized))
print('The vectorized code is', np.around(howfast,4), 'times faster than the vectorized code'
)
```

```
Time [sec] (non-vectorized):  7.3517
Time [sec] (vectorized):  0.0
The vectorized code is 65.1402 times faster than the vectorized code
```

In [18]:
```
# Plot the result
plt.contourf(xvalues,yvalues,threshold)
plt.title('Results of Function Post-Threshold Filter')
plt.show()
```



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 $E[XY]$

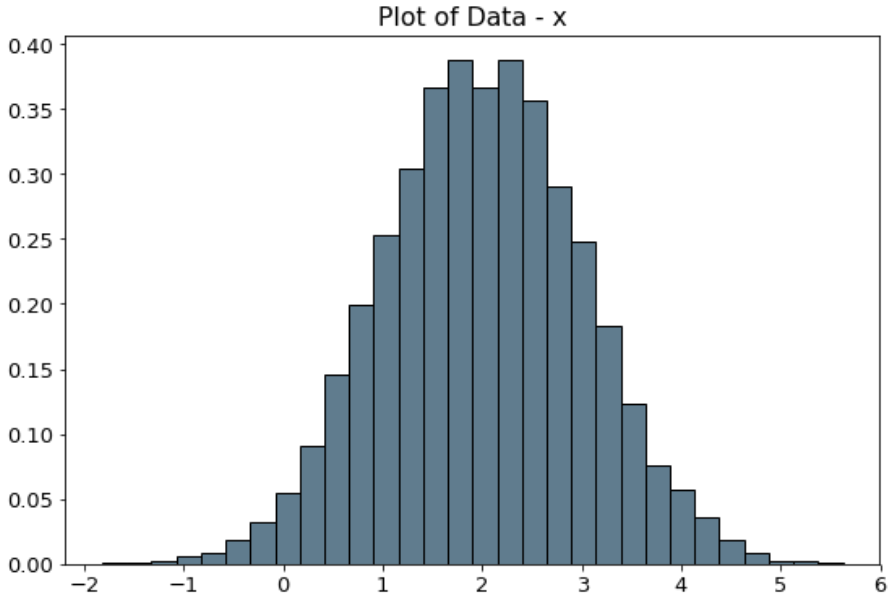
ANSWER

In [19]:
```
# 1.
# create x - vector of observations generated by provided criteria
mu = 2
sigma = 1
x = np.random.normal(mu, sigma, 10000)

# 2.
# calculate the mean and std deviation for our sample to 4 decimal places
print('2. Calculate the mean and standard deviation values of x')
print('mean = ', np.around(np.mean(x), decimals=4))
print('standard deviation = ', np.around(np.std(x), decimals=4))
print('While these values are not exactly the same as the population distribution statistics,
 we can still validate our work thus far.')
```

```
2. Calculate the mean and standard deviation values of x
mean =  2.0006
standard deviation =  0.9973
While these values are not exactly the same as the population distribution statistics, we ca
n still validate our work thus far.
```

In [20]:
```
# 3.
n, bins, patches = plt.hist(x, 30, density=True, facecolor='#607c8e', edgecolor='black', line
width=1)
plt.title('Plot of Data - x')
plt.show()
```



In [21]:
```
# 4
#find the 90th percentile
per90 = np.around(np.percentile(x, 90), decimals=4)
print('4. The 90th percentile of x is =', per90)
```

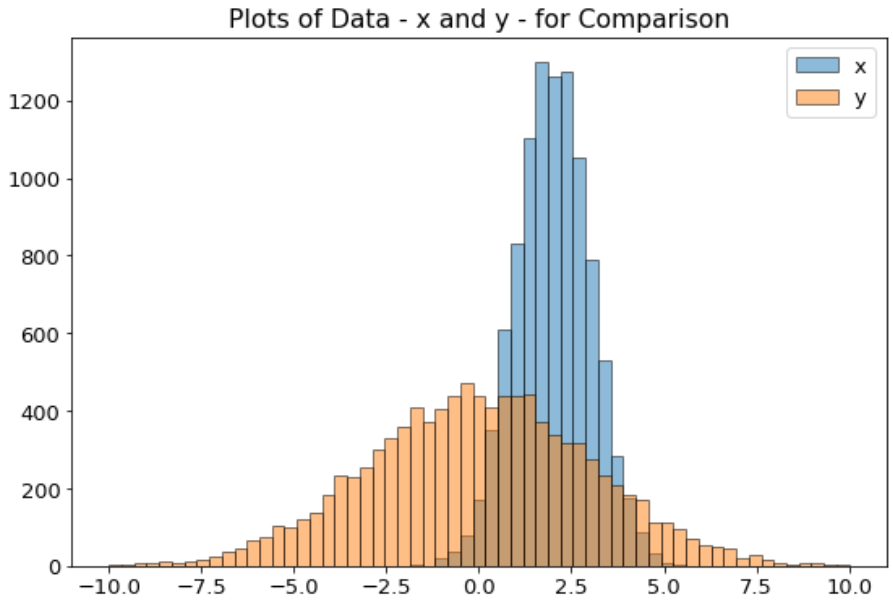
```
4. The 90th percentile of x is = 3.2787
```

In [22]:
```
# 5
#find the 99th percentile
per99 = np.around(np.percentile(x, 99), decimals=4)
print('5. The 90th percentile of x is =', per99)
```

```
5. The 90th percentile of x is = 4.3444
```

In [23]:
```
# 6.
# create y - vector of observations generated by provided criteria
mu = 0
sigma = 3
y = np.random.normal(mu, sigma, 10000)

# 7.
bins = np.linspace(-10, 10, 60)
plt.hist(x, bins, alpha=0.5, label='x', edgecolor='black')
plt.hist(y, bins, alpha=0.5, label='y', edgecolor='black')
plt.legend(loc='upper right')
plt.title('Plots of Data - x and y - for Comparison')
plt.show()
```



In [24]:
```
# 8.
# for normally distributions, we know the expected value is equal to the mean value of that v
ariable
# for random, independent observations, we know the joint expected value E(XY) is found by mu
ltiplying the expected values of each variable
# which gives us E(XY) = E(X)E(Y), calculated below:
expect_x = np.mean(x)
expect_y = np.mean(y)
expect_xy = np.around(expect_x*expect_y, decimals=4)
print('8. E(XY) = E(X) E(Y) =', expect_xy)
```

```
8. E(XY) = E(X) E(Y) = 0.0466
```

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:

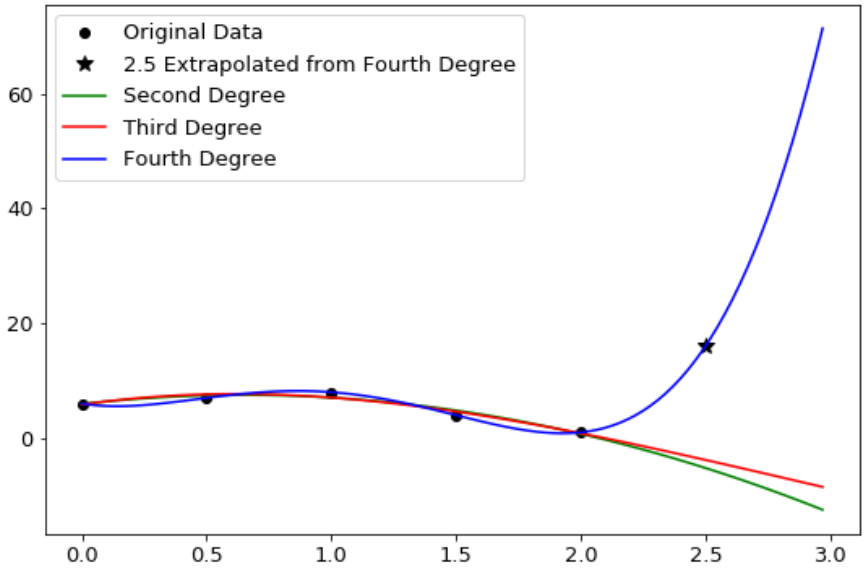
Table 1. Dataset containing n=5 observations

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

ANSWER

To estimate the integral of a discrete data set, we can use different rules like the Trapezoidal rule, but in this case because we are also extrapolating a data point for $x = 2.5$, we will estimate a function that fits the data and integrate.

In [74]:
```
#first input the data we have
import scipy.integrate as integrate
x = np.array([0,.5,1,1.5,2])
y = np.array([6,7,8,4,1])
#want the degree to be small enough that doesn't overfit
#but large enough that it seems to accurately fit the dat
z2 = np.polyfit(x, y, 2)
f2 = np.poly1d(z2)
z3 = np.polyfit(x, y, 3)
f3 = np.poly1d(z3)
z4 = np.polyfit(x, y, 4)
f4 = np.poly1d(z4)
xp = np.linspace(0,3,num=100,endpoint=False)
plt.plot(x,y,'ko', label='Original Data')
plt.plot(2.5,f4(2.5),'k*', markersize=10, label='2.5 Extrapolated from Fourth Degree')
plt.plot(xp,f2(xp),'g-', label='Second Degree')
plt.plot(xp,f3(xp),'r-', label='Third Degree')
plt.plot(xp,f4(xp),'b-', label='Fourth Degree')
plt.legend()
plt.show()
```



From this plot, it seems like the fourth degree polynomial is the best fit without getting into overfitting problems. This function is given by:

$$f(x) = 7.333x^4 + 28.67x^3 + 30.17x^2 - 6.833x + 6$$

which we find by using the values corresponding with the variable f4 in the above code. We will integrate as usual for the given range to approximate the integral value, which we will call k:

$$\begin{aligned} k &= \int_0^{2.5} 7.333x^4 + 28.67x^3 + 30.17x^2 - 6.833x + 6f(x)\,dx \\ &= \left(1.4666x^5 + 7.1675x^4 + 10.0567x^3 - 3.4165x^2 + 6x\Big|_{x=0}^{2.5}\right) \\ &= 573.985 \end{aligned}$$

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, **Kate Coulter**, 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

(a) https://github.com/katecoulter/machine-learning-course (https://github.com/katecoulter/machine-learning-course)
(b) https://katecoulter.github.io/machine-learning-course/ (https://katecoulter.github.io/machine-learning-course/)

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

https://github.com/katecoulter/machine-learning-course/blob/master/HDI%20Indicators%20-%202018.ipynb
(https://github.com/katecoulter/machine-learning-course/blob/master/HDI%20Indicators%20-%202018.ipynb)