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

SOLUTIONS

Probability and Statistics Theory

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

1

[3 points]

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?

ANSWER

PDFs should integrate to 1 over the domain of the function, which is $(0, 2)$ for $f(x)$. So we integrate and solve for a:

$$\int_0^2 ax^2 dx = 1$$

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

$$a = \frac{3}{8}$$

2

[3 points] 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

The cumulative distribution function, $F(x)$, of a PDF, $f(x)$, is the integral of the PDF from 0 to x. In this case it's in three parts. For $x \leq 0$, $F(x) = 0$; for $(x \geq 3)$, $F(x) = 1$. Otherwise, we need to calculate the integral:

$$\int_0^x \frac{1}{3} dz = \frac{1}{3} z \Big|_0^x = \frac{1}{3} x$$

Therefore, putting it all together, the CDF is:

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

3

[6 points] 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) Expected value:

$$\begin{aligned} E[x] &= \int_{-\infty}^{\infty} x f(x) dx \\ &= \int_0^3 x \left(\frac{1}{3} \right) dx \\ &= \frac{1}{6} x^2 \Big|_0^3 \\ &= \frac{9}{6} \\ &= \frac{3}{2} \end{aligned}$$

(b) Variance:

We start with the definition of variance, and we realize we already have one of the pieces for computing it, $E[X]$.

$$Var(X) = E[(X - E[X])^2] = E[x^2] - E[X]^2$$

Therefore, we only need to compute $E[X^2]$:

$$\begin{aligned} E[x^2] &= \int_{-\infty}^{\infty} x^2 f(x) dx \\ &= \int_0^3 x^2 \left(\frac{1}{3} \right) dx \\ &= \frac{1}{9} x^3 \Big|_0^3 \\ &= \frac{27}{9} \\ &= 3 \end{aligned}$$

Lastly, we compute the variance:

$$Var(X) = E[x^2] - E[X]^2 = 3 - \left(\frac{3}{2}\right) = \frac{3}{4}$$

4

[6 points] 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

$$E[X] = \frac{1}{N}\sum_i x_i = \frac{2+3+10-1-1}{5} = \frac{13}{5} = 2.6$$

(b) Variance

$$\begin{aligned}\$\$Var(X) &= E[(X - E[X])^2] \\ &= E[(X - (2.6)^2)] \\ &= \frac{1}{N}\sum_i (x_i - (2.6)^2) \\ &= \frac{(-0.6)^2 + (0.4)^2 + (7.4)^2 + (-3.6)^2 + (-3.6)^2}{5} \\ &= 16.24\end{aligned}$$

5

[8 points] 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) There are 26 letters in the English alphabet, and 10 possible numbers (0-9). In this case, different permutations are distinct, so "AJA2293" is distinct from "JAA2293". That is to say, order matters. In this case, the total number of different license plates is:

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

There are **175,760,000** distinct license plates that are possible.

(b) In this case we can consider this sampling without replacement (since one player cannot be in the same batting order twice). For the first position there are nine possibilities, for the second position there are 8 since we cannot include the player who was first also in the second. For the third position, we must exclude the previous two players from consideration, so there are 7, and so on...

$$9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 9! = 362,880$$

There are **362,880** possible batting orders.

(c) In this case we follow the same logic as in part (b), but we limit ourselves to 5 possible players in any given batting order:

$$9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 = \frac{9!}{(9-5)!} = 15,120$$

There are **15,120** possible batting orders.

(d) This is equivalent to asking how many combinations of 3 are possible from a set of 26. Here order does not matter. This is similar to part (c), except that we don't care about order, so we must divide by the number of ways of ordering each of the three members of the team:

One way of writing this is as follows:

$$\frac{26 \cdot 25 \cdot 24}{3!} = 2,600$$

Equivalently, this is written using the well-known formula:

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

There are **2,600** possible unique teams of 3 from a class of 26 students.

Linear Algebra

6

[7 points] 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. $\mathbf{A}\mathbf{A}^T$
3. $\mathbf{A}\mathbf{b}$
4. $\mathbf{A}\mathbf{b}^T$
5. $\mathbf{b}\mathbf{A}$
6. $\mathbf{b}^T\mathbf{A}$
7. $\mathbf{b}\mathbf{b}$
8. $\mathbf{b}^T\mathbf{b}$
9. $\mathbf{b}\mathbf{b}^T$
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 "∘".

ANSWER

In [7]:
```
%matplotlib inline
import numpy as np

# Begin by creating vectors of each of the above matrices:
A = np.array([[1, 2, 3],[2, 4, 5],[3, 5, 6]])
b = np.array([-1, 3, 8])[np.newaxis].T
c = np.array([4, -3, 6])[np.newaxis].T
I = np.identity(3)

# 1. AA
print('1. AA: [3 x 3][3 x 3]\n{}\n'.format( A.dot(A) ))

# 2. AA^T
print('2. AA^T: [3 x 3][3 x 3]\n{}\n'.format( A.dot(A.T) ))

# 3. Ab
print('3. Ab: [3 x 3][3 x 1]\n{}\n'.format( A.dot(b) ))

# 4. Ab^T
print('4. Ab^T: [3 x 3][1 x 3]\n{}\n'.format( 'Shape mismatch - cannot be computed'

# 5. bA
print('5. bA: [3 x 1][3 x 3]\n{}\n'.format( 'Shape mismatch - cannot be computed' ))

# 6. b^TA
print('6. b^TA: [1 x 3][3 x 3]\n{}\n'.format( b.T.dot(A) ))

# 7. bb
print('7. bb: [3 x 1][3 x 1]\n{}\n'.format( 'Shape mismatch - cannot be computed' ))

# 8. b^T b
print('8. b^T b: [1 x 3][3 x 1]\n{}\n'.format( b.T.dot(b) ))

# 9. bb^T
print('9. bb^T: [3 x 1][1 x 3]\n{}\n'.format( b.dot(b.T) ))

# 10. b + c^T
print('10. b + c^T: [3 x 1] + [1 x 3]\n{}\n'.format( 'Shape mismatch - cannot be com

# 11. b^T b^T
print('11. b^T b^T: [3 x 1][3 x 1]\n{}\n'.format( 'Shape mismatch - cannot be comput

# 12. A^-1 b
print('12. A^-1 b: [3 x 3][3 x 1]\n{}\n'.format( np.linalg.inv(A).dot(b) ))

# 13. A o A
print('13. A o A: [3 x 3] o [3 x 3]\n{}\n'.format( A*A ))

# 14. b o c
print('14. b o c: [3 x 3] o [3 x 3]\n{}\n'.format( b*c ))
```

```
1. AA: [3 x 3][3 x 3]
[[14 25 31]
 [25 45 56]
 [31 56 70]]

2. AA^T: [3 x 3][3 x 3]
[[14 25 31]
 [25 45 56]
 [31 56 70]]
```

```
3. Ab: [3 x 3][3 x 1]
[[29]
 [50]
 [60]]

4. Ab^T: [3 x 3][1 x 3]
Shape mismatch - cannot be computed

5. bA: [3 x 1][3 x 3]
Shape mismatch - cannot be computed

6. b^TA: [1 x 3][3 x 3]
[[29 50 60]]

7. bb: [3 x 1][3 x 1]
Shape mismatch - cannot be computed

8. b^T b: [1 x 3][3 x 1]
[[74]]

9. bb^T: [3 x 1][1 x 3]
[[ 1 -3 -8]
 [-3  9 24]
 [-8 24 64]]

10. b + c^T: [3 x 1] + [1 x 3]
Shape mismatch - cannot be computed

11. b^T b^T: [3 x 1][3 x 1]
Shape mismatch - cannot be computed

12. A^-1 b: [3 x 3][3 x 1]
[[ 6.]
 [ 4.]
 [-5.]]

13. A o A: [3 x 3] o [3 x 3]
[[ 1  4  9]
 [ 4 16 25]
 [ 9 25 36]]

14. b o c: [3 x 3] o [3 x 3]
[[-4]
 [-9]
 [48]]
```

7

[8 points] 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

In [62]:
```
import numpy as np

A = np.array([[1, 2, 3],[2, 4, 5],[3, 5, 6]])

# 1. Calculate the eigenvalues and corresponding eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)
print('1.)\n')
print('Eigenvalues:\n{}\n'.format(eigenvalues))
print('Eigenvectors:\n{}\n'.format(eigenvectors))

# 2. Verify that A * v = lambda * v
print('2.)\n')
vector = eigenvectors[:,0]
value  = eigenvalues[0]
Av = A.dot(vector)
lambdav = value * vector
print('Av = \n{}\n'.format(Av))
print('lambda v = \n{}\n'.format(lambdav))

# 3. Show that the eigenvectors are orthogonal to one another
#    This is shown by taking the inner products between vectors and demonstrating th
print('3.)\n')
combinations = [(0,1),(0,2),(1,2)]
for (i,comb) in enumerate(combinations):
    inner_product = np.inner(eigenvectors[:,comb[0]],eigenvectors[:,comb[1]])
    print('Inner product of eigenvector {} and {} = {:.4f}'.format(comb[0],comb[1],i
```

```
1.)

Eigenvalues:
[ 11.34481428  -0.51572947   0.17091519]

Eigenvectors:
[[-0.32798528 -0.73697623  0.59100905]
 [-0.59100905 -0.32798528 -0.73697623]
 [-0.73697623  0.59100905  0.32798528]]

2.)

Av =
[-3.72093206 -6.70488789 -8.36085845]

lambda v =
[-3.72093206 -6.70488789 -8.36085845]

3.)

Inner product of eigenvector 0 and 1 = -0.0000
Inner product of eigenvector 0 and 2 = -0.0000
Inner product of eigenvector 1 and 2 = -0.0000
```

Numerical Programming

8

[10 points] Loading data and gathering insights from a real dataset

Data. The data for this problem can be found in the `data` subfolder in the `assignments` folder on github (https://github.com/kylebradbury/ids705). The filename is `egrid2016.xlsx` . This dataset is the Environmental Protection Agency's (EPA) Emissions & Generation Resource Integrated Database (eGRID) (https://www.epa.gov/energy/emissions-generation-resource-integrated-database-egrid) containing information about all power plants in the United States, the amount of generation they produce, what fuel they use, the location of the plant, and many more quantities. We'll be using a subset of those data.

The fields we'll be using include:

field	description
SEQPLT16	eGRID2016 Plant file sequence number (the index)
PSTATABB	Plant state abbreviation
PNAME	Plant name
LAT	Plant latitude
LON	Plant longitude
PLPRMFL	Plant primary fuel
CAPFAC	Plant capacity factor
NAMEPCAP	Plant nameplate capacity (Megawatts MW)
PLNGENAN	Plant annual net generation (Megawatt-hours MWh)
PLCO2EQA	Plant annual CO2 equivalent emissions (tons)

For more details on the data, you can refer to the eGrid technical documents (https://www.epa.gov/sites/production/files/2018-02/documents/egrid2016_technicalsupportdocument_0.pdf). For example, you may want to review page 45 and the section "Plant Primary Fuel (PLPRMFL)", which gives the full names of the fuel types including WND for wind, NG for natural gas, BIT for Bituminous coal, etc.

There also are a couple of "gotchas" to watch out for with this dataset:

- The headers are on the second row and you'll want to ignore the first row (they're more detailed descriptions of the headers).
- NaN values represent blanks in the data. These will appear regularly in real-world data, so getting experience working with it will be important.

Your objective. For this dataset, your goal is answer the following questions about electricity generation in the United States:

(a) Which plant has generated the most energy (measured in MWh)?

(b) What is the name of the northern-most power plant in the United States?

(c) What is the state where the northern-most power plant in the United States is located?

(d) Plot a histogram of the amount of energy produced by each fuel for the plant.

(e) From the plot in (e), which fuel for generation produces the most energy (MWh) in the United States?

ANSWER

In [2]:
```
import pandas as pd

egrid = pd.read_excel('./data/egrid2016.xlsx', skiprows=[0])
```

(a) Which plant has generated the most energy (measured in MWh)?

In [3]:
```
egrid.sort_values(by='PLNGENAN', ascending=False)['PNAME'].iloc[0]
```

Out[3]:
```
'Palo Verde'
```

(b) What is the name of the northern-most power plant in the United States?

In [4]:
```
egrid.sort_values(by='LAT', ascending=False)['PNAME'].iloc[0]
```

Out[4]:
```
'Barrow'
```

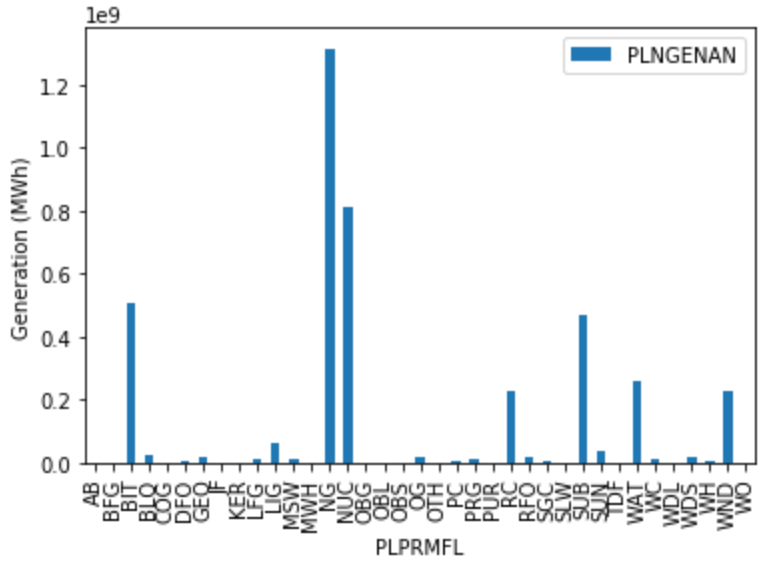
(c) What is the state where the northern-most power plant in the United States is located?

In [5]:
```
egrid.sort_values(by='LAT', ascending=False)['PSTATABB'].iloc[0]
```

Out[5]:
```
'AK'
```

(d) Plot a histogram of the amount of energy produced by each fuel for the plant. From this plot, which fuel for generation produces the most energy (MWh) in the United States?

In [8]:
```
import matplotlib.pyplot as plt

fuel_type_group = egrid.groupby('PLPRMFL').sum()
fuel_and_gen    = fuel_type_group[['PLNGENAN']]

fuel_and_gen.plot.bar()
plt.ylabel('Generation (MWh)')
plt.show()
```



(e) From the plot in (e), which fuel for generation produces the most energy (MWh) in the United States?

In [9]:
```
fuel_and_gen[fuel_and_gen['PLNGENAN'] == fuel_and_gen['PLNGENAN'].max()].index[0]
```

Out[9]:
```
'NG'
```

9

[8 points] 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 [3]:
```
import numpy as np
import time

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

# Compute the sum of squares the non-vectorized way (using a for loop)
t0 = time.time()
sum_of_squares = 0
for value in x:
    sum_of_squares += value**2
t1 = time.time()
time_nonvectorized = t1 - t0

# Compute the sum of squares the vectorized way (using numpy)
t0 = time.time()
sum_of_squares = np.dot(x,x)
t1 = time.time()
time_vectorized = t1 - t0

print('Time [sec] (non-vectorized): {:0.4f}'.format(time_nonvectorized))
print('Time [sec] (vectorized):     {:0.4f}'.format(time_vectorized))
print('The vectorized code is {:0.1f} times faster than the vectorized code'.format(
```

```
Time [sec] (non-vectorized): 4.6890
Time [sec] (vectorized):     0.0042
The vectorized code is 1125.3 times faster than the vectorized code
```

10

[10 points] 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` .

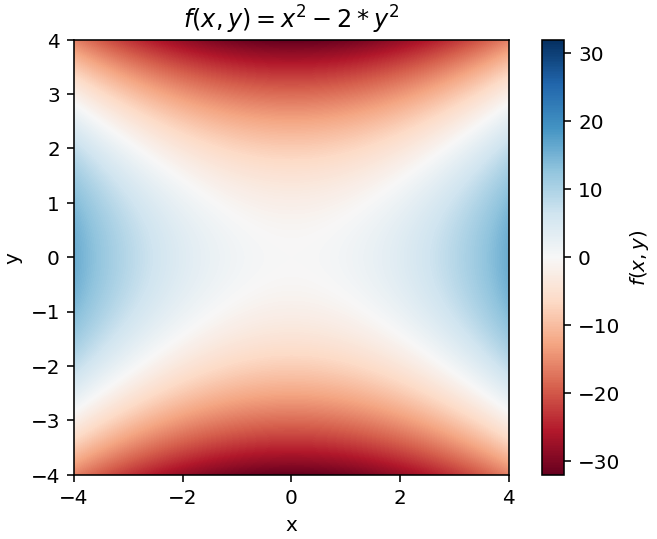
Hint: look at the `numpy` `meshgrid` *(https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.meshgrid.html) documentation*

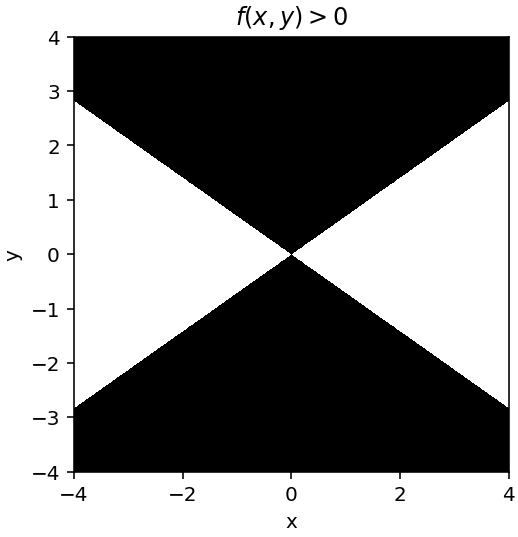
ANSWER

In [3]:

```
# This increases the resolution of the plots to make them cripser within the noteboo
%config InlineBackend.figure_format = 'retina'

import numpy as np
import time
import matplotlib.pyplot as plt

nvalues = 2000
xvalues = np.linspace(-4,4,nvalues)
yvalues = np.linspace(-4,4,nvalues)
thresh  = 0

# ---------------------------------------
# (a) Vectorization
# ---------------------------------------
# Nonvectorized implementation
t0 = time.time()
f = np.zeros((nvalues,nvalues))
f_thresholded = np.zeros((nvalues,nvalues))
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

# Vectorized implementation
t0 = time.time()
X,Y = np.meshgrid(xvalues,yvalues)
F = X**2 - 2 * Y**2
F_threshold = F > thresh
t1 = time.time()
time_vectorized = t1 - t0

# Vectorization performance results
print('Time [sec] (non-vectorized): {:0.4f}'.format(time_nonvectorized))
print('Time [sec] (vectorized):     {:0.4f}'.format(time_vectorized))
print('The vectorized code is {:0.1f} times faster than the vectorized code'.format(

# ---------------------------------------
# (b) Function plotting
# ---------------------------------------
# Plot the function f(x,y)
plot1 = plt.imshow(F, extent=(-4,4,-4,4), cmap='RdBu')
plt.xlabel('x')
plt.ylabel('y')
plt.title('$f(x,y) = x^2 - 2 * y^2$')
absmax = max([F.max(), abs(F.min())])
plt.gcf().colorbar(plot1, label='$f(x,y)$')
plot1.set_clim(-absmax,absmax)
plt.show()

# Plot the thresholded output f(x,y) > 0
plot2 = plt.imshow(F_threshold, extent=(-4,4,-4,4), cmap='gray')
plt.xlabel('x')
plt.ylabel('y')
plt.title('$f(x,y) > 0$')
#plt.gcf().colorbar(plot2, label='$f(x,y) > 0$')
plt.show()
```

```
Time [sec] (non-vectorized): 6.7477
Time [sec] (vectorized):     0.0435
The vectorized code is 155.2 times faster than the vectorized code
```





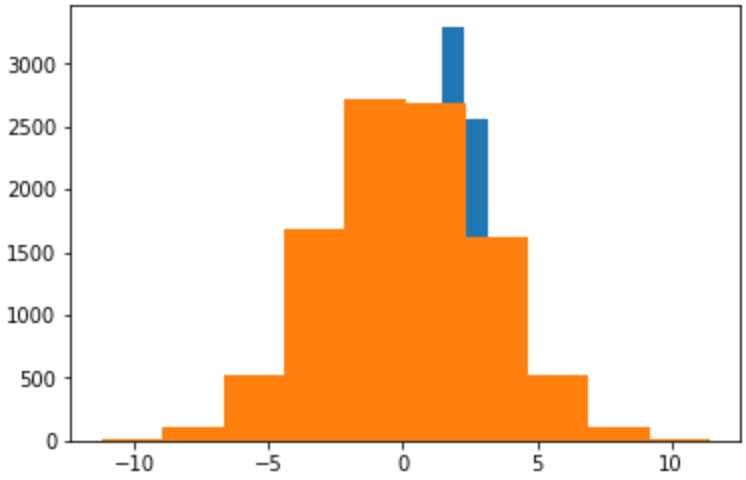
11

[10 points] 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. Create a new figure and plot the histogram of the data in $\mathbf{y}$ on the same axes 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 [55]:
```
n = 10000
s1 = np.random.randn(n)+2
s2 = np.random.randn(n)*3

#print(np.round(s1))
#print(np.round(s2))
d = False
plt.hist(s1,density=d)
plt.hist(s2,density=d)
plt.show()
```

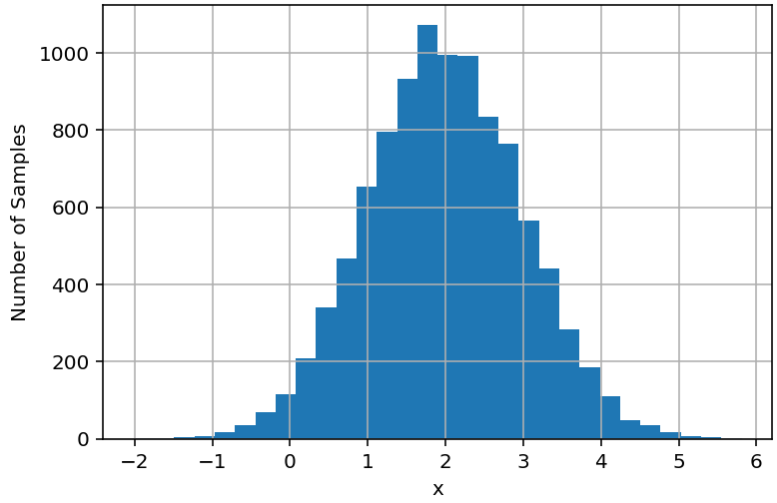


In [90]:

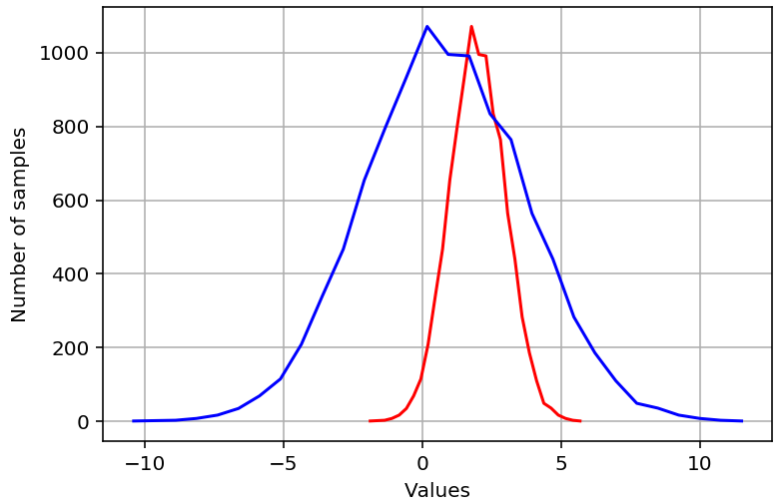
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(12) # This ensures that random numbers will be reproducible

# 1. Synthesize data x
print('1.)')
x = np.random.randn(10000) + 2
print('Examples from x: {}'.format(x[0:5]))

# 2. Calculate the mean and standard deviation
print('\n2.)')
print('x_mean = {:.4}'.format(x.mean()))
print('x_std  = {:.4}'.format(x.std()))

# 3. Plot a historgram of x with 30 bins
print('\n3.)')
plt.hist(x,bins=30)
plt.xlabel('x')
plt.ylabel('Number of Samples')
plt.grid('on')
plt.show()

# 4. Compute the 90th percentile of x
print('\n4.)')
percentile90 = np.percentile(x,90)
print('90th percentile = {:.4}'.format(percentile90))

# 5. Compute the 99th percentile of x
print('\n5.)')
percentile99 = np.percentile(x,99)
print('90th percentile = {:.4}'.format(percentile99))

# 6. Synthesize the vector y
print('\n6.)')
y = 3*np.random.randn(10000)
print('Examples from y: {}'.format(y[0:5]))

# 7. Plot the histogram of the data in x and y on one plot and compare
print('\n7.)')
histx, binsx = np.histogram(x, bins=30)
histy, binsy = np.histogram(y, bins=30)
centerx = (binsx[:-1] + binsx[1:]) / 2
centery = (binsy[:-1] + binsy[1:]) / 2
plt.plot(centerx,histx,'r-',label='x')
plt.plot(centery,histx,'b-',label='y')
plt.xlabel('Values')
plt.ylabel('Number of samples')
plt.grid('on')
plt.show()

# 8. Estimate E[XY]
# Since X and Y are independent, this is very simple: E[XY] = E[X]E[Y]
print('\n8.)')
print('E[XY] = E[X]E[Y] = {:.4}'.format(x.mean() * y.mean()))
```

```
1.)
Examples from x: [ 2.47298583  1.31857412  2.2424395   0.29926437  2.75314283]

2.)
x_mean = 1.988
```

```
x_std  = 0.9943

3.)
```




```
4.)
90th percentile = 3.267

5.)
90th percentile = 4.319

6.)
Examples from y: [ 4.04422192  3.85535993  0.99711177  1.55658765 -3.88588689]

7.)
```




```
8.)
E[XY] = E[X]E[Y] = 0.01241
```

Version Control via Git

12

[1 point] You will need to use Git to submit assignments and in the course projects and is generally a version control and collaboration tool. You can even use some Git repositories (e.g. Github) as hosts for website, such as with the course website (https://kylebradbury.github.io/ids705/index.html).

Complete the Atlassian Git tutorial (https://www.atlassian.com/git/tutorials/what-is-version-control), specifically the following listed sections. Try each concept that's presented. For this tutorial, instead of using BitBucket as your remote repository host, you may use your preferred platform such as Github (https://github.com/) or Duke's Gitlab (https://gitlab.oit.duke.edu/users/sign_in).

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)

I also have created two videos on the topic to help you understand some of these concepts: Git basics (https://www.youtube.com/watch?v=fBCwfoBr2ng) and a step-by-step tutorial (https://www.youtube.com/watch?v=nH7qJHx-h5s).

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, [**your name here**], affirm that I have [**completed the above tutorial / I have previous experience that covers all the content in this tutorial**]*

ANSWER

Exploratory Data Analysis

13

[20 points] 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.

Here your analysis will evaluated based on:

1. Data cleaning: did you look for and work to resolve issues in the data?
2. Quality of data exploration: did you provide plots demonstrating interesting aspects of the data?
3. Interpretation: Did you clearly explain your insights? Restating the data, alone, is not interpretation.
4. Professionalism: Was this work done in a way that exhibits professionalism through clarity, organization, high quality figures and plots, and meaningful descriptions?

ANSWER