Probability

MATH1900: Machine Learning

Location: http://people.sc.fsu.edu/~jburkardt/classes/ml_2019/probability/probability.pdf

A little more snow than we expected!

1 Analyzing a Random Process

At Michigan Tech, a lot of snow falls each winter. Since 1890, the students have measured and recorded the yearly totals. In a particular year, the actual amount of snow seems random, but over time, some patterns can be observed. The snowfall is an example of a random process; our interest is to try to describe the data usefully, and then perhaps to create a mathematical model that produces similar results.

We can read in the data using Python:

```
1  import numpy as np
2
3  data = np.loadtxt ( 'snow_data.txt' )
4  year = data[:,0]
5  snow = data[:,1]
6  m = len ( snow )
```

The simplest model of the snowfall replaces all the data by the average value. Also of interest will be the minimum and maximum values. Finally, we will also request something called the *standard deviation*:

```
1  snow_mean = np.mean ( snow )
2  snow_min = np.min ( snow )
3  snow_max = np.max ( snow )
4  snow_std = np.std ( snow )
```

The average tells us the "center" of the data. But we know the data varies a lot. The standard deviation tells us that "most" of the data is no more than one standard deviation from the average.

To verify this, let's make a `True/False` array that is `True` for each snow record that is in the interval `[mean-std,mean+std]`, count the number of `True` values, and convert to a percent:

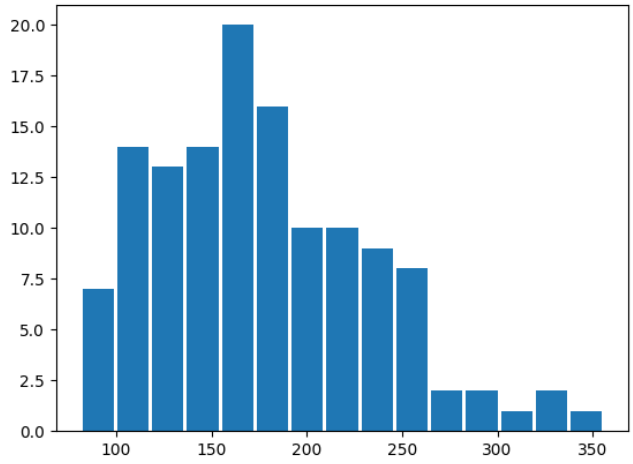
```
1  tf = (snow_mean - snow_std < snow) & (snow < snow_mean + snow_std )
2  print ( 100 * sum ( tf ) / m, 'percent of data within one std of mean.' )
```

[

Note that the mean is often symbolized by the Greek letter μ ("mu"); the standard deviation by σ ("sigma"). The square of the standard deviation is known as the *variance* and is symbolized by σ^2.

To picture the data, we will create a histogram, a sort of bar graph where each bar represents a numeric range. The height of the bar counts the number of times or relative frequency that a data item falls within its range.

```
import matplotlib.pyplot as plt
plt.hist ( snow, rwidth = 0.90, bins = 25 )
plt.show ( )
```



A histogram of the snowfall data

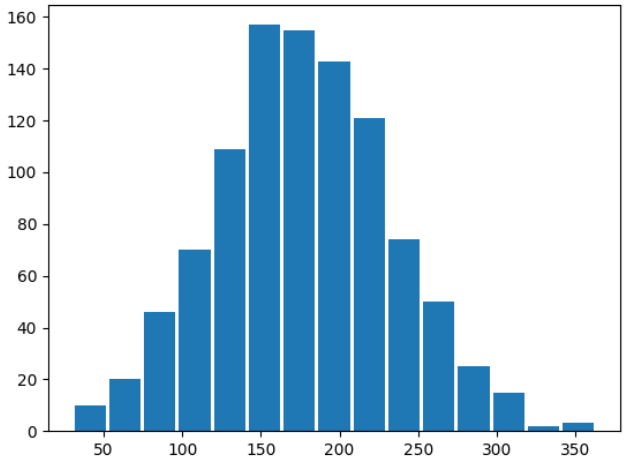
Using just the mean and standard deviation, we can try to create a simulation of the snowfall data. One simple model of a random process is known as the normal distribution. Variables modeled by the standard normal distribution have a mean of 0 and a standard deviation of 1, and can be computed by writing

```
x = np.random.normal ( )
```

But if we wish to sample a normal distribution with a given mean and standard deviation, and we want `n` such samples, we simply include this information in the call. Using statistics from our snowfall data:

```
import numpy as np
snow_mean = 176.70
snow_std = 55.95
n = 1000
snow_sim = np.random.normal ( snow_mean, snow_std, n )
```

Here is a histogram of our simulated snowfalls:



A histogram of the simulated snowfall data

Although our simulation looks good in the center, we have gotten many more low snowfalls than are seen in the actual records. This suggests that the behavior of the normal random numbers may come close, but does not fully capture what's happening in this real life example.

2 Correlated Variables: Cold Hands, Warm Heart

The saying *cold hands, warm heart* suggests that there is a simple relationship between two seemingly independent things we can measure. We can even imagine that the colder the hands, the warmer the heart, and that there is some kind of linear relationship going on here. Because one thing goes down as the other goes up; we say the temperatures of hand and heart are **correlated**, but in a negative sense. In contrast, the saying *no pain, no gain* suggests that pain and gain must increase or decrease together; this suggests they are positively **correlated** variables.

In mathematics, the relationship between two quantities can be completely explained by a linear function. In electronics, if a conductor has a resistance of R, then the voltage V and current I are related by

$$V = I * R$$

Assuming we know R already, this suggests a relationship so strong that if we know V, we don't have to measure I; its value is completely controlled by the formula.

In life, many variables are related more weakly. Knowing that a child is 7 years old, you can make a good guess as to the child's weight. Knowing that a city is 500 miles away, you can make a good guess as to how many hours it would take you to drive there. You would not expect these estimates to be exact, but you would expect to be able to predict a reasonable value using a simple linear formula.

Suppose we want to determine to what extent two quantities X and Y have a linear relationship. There are two common ways to measure the strength of this relationship, the **covariance** $C(x, y)$ and the **correlation coefficient**, $f(x, y)$.

Suppose we have n measurements of X and Y, which we stored in the variables x and y. The first thing we need to do is subtract off the means, μx and μy. Then the covariance is

$$C(x, y) = \frac{\sum_{i=1}^{n}(x_i - \mu(x))(y_i - \mu(y))}{n}$$

You may also see covariance defined with a divisor of $(n-1)$ *instead of* n*. This minor difference arises when considering data sampled from a larger, unknown set.* The correlation coefficient divides by the standard deviations of the two variables:

$$r(x,y) = \frac{\sum_{i=1}^{n}(x_i - \mu(x))(y_i - \mu(y))}{n\,\sigma(x)\sigma(y)}$$

The advantage of the using the correlation coefficient is that the resulting value has a very simple interpretation. The value of $r(x,y)$ will always be a number between -1 and 1, with meaning:

$$r(x,y) = \begin{cases} \text{near -1, strongly anticorrelated} \\ \text{near -1/2, moderately anticorrelated} \\ \text{near 0, not related} \\ \text{near +1/2, moderately related} \\ \text{near +1, strongly related} \end{cases}$$

Here, "anticorrelated" means that as one variable goes up, the other goes down. "Moderately correlated" means that the relationship can only be partially explained by a linear formula. Not related at all means that the two variables seem to have no relationship.

3 Example: Some simple numeric lists

To get a feeling for what correlation is telling you, let's look at some tiny datasets. Suppose that we have $n = 3$ observations of some quantities at times $t = 1, 2, and 3$. Using the formula, we can compute the correlation of the value of t with each variable:

```
t     1    2    3        r(t,t)  =  1
y1   10   20   30        r(t,y1) =  1
y2  100  200  300        r(t,y2) =  1
y3  100  120  140        r(t,y3) =  1
y4   50   45   40        r(t,y4) = -1
y5    2.4  4.7  5.8      r(t,y5) =  0.98
y6    4    0    4        r(t,y6) =  0.0
y7    1   -2    1        r(t,y7) =  0.0
y8    np.random.rand(3)  r(t,y8) = -0.27  (depends on results!)
```

In cases of `t` and `y1` through `y3`, an exact linear relationship is easy to see. For `y4` the relationship is also linear, but with a negative correlation: $y4 = 55 - 5 * t$. For `y5`, the relationship $y5 = 2 * t$ is a good approximation. For the remaining cases, the correlation coefficient indicates that a linear relationship explains little or none of the behavior of the data.

Instead of doing the arithmetic ourselves, we can use the **numpy** function `C=np.corrcoef(x,y)` to compute the correlation between two sets of data, or the command `C=np.corrcoef(A)` to compute the correlation between the rows of the data matrix `A`.

The output `C=np.corrcoef(x,y)` will be a 2×2 matrix:

```
r(x,x)  r(x,y)
r(y,x)  r(y,y)
```

so that the entry `C[0,1]` is the only "interesting" result, the value of $r(x,y)$.

In the case of a data matrix, the output `C=np.corrcoef(A)` will be an $m \times m$ matrix comparing the rows:

```
r(row0,row0)   r(row0,row1)   ... r(row0,rowm-1)
r(row1,row0)   r(row1,row1)   ... r(row1,rowm-1)
     ...            ...       ...       ...
r(rowm-1,row0) r(rowm-1,row1) ... r(rowm-1,rowm-1)
```

Given a correlation matrix like this, any off-diagonal entries near to +1 or -1 are revealing two variables that are closely related.

4 Example: Pets and Owners

A survey of dog owners has recorded the weight in pounds of the dog and the owner, the yearly pay of the owner, and the size of the owner's yard in square feet. The data in *dog_data.txt* is:

```
Dog   Owner      Pay    Yard
---   -----  -------  - ----
 89     180    80000    8000
 55     147    95000    4500
110     125    70000   15000
 60     210    60000    7000
 12     153   145000     200
 16     105   110000    5000
 40     220    95000    8000
 23     173   120000    2500
 76     162   105000    9000
 14     137   152000    1000
```

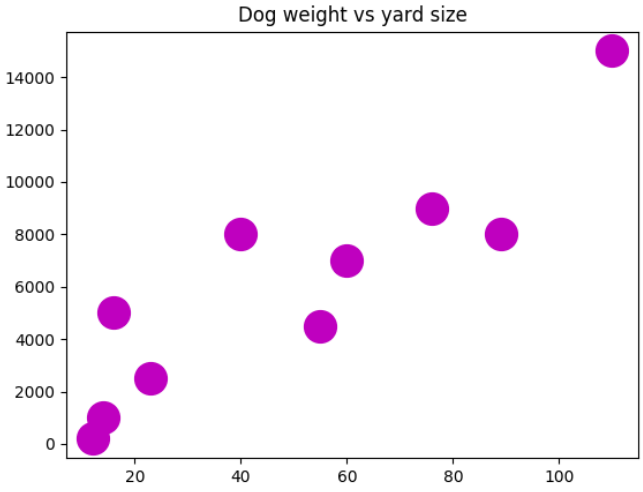
Is there a correlation between the weight of the dog, and any of the other variables?

```
import numpy as np
data = np.loadtxt ( 'dog_data.txt' )

dog = data[:,0]
dog_mean = np.mean ( dog )
dog_std = np.std ( dog )

own = data[:,1]
own_mean = np.mean ( own )
own_std = np.std ( own )

pay = data[:,2]
pay_mean = np.mean ( pay )
pay_std = np.std ( pay )

yard = data[:,3]
yard_mean = np.mean ( yard )
yard_std = np.std ( yard )

n = len ( dog )

r = np.dot ( dog - dog_mean, own - own_mean ) / dog_std / own_std / n
print ( 'dog weight/owner_weight correlation is ', r )
r = np.dot ( dog - dog_mean, pay-pay_mean ) / dog_std / pay_std / n
print ( 'dog weight/owner_pay correlation is ', r )
r = np.dot ( dog - dog_mean, yard - yard_mean ) / dog_std / yard_std / n
print ( 'dog weight/yard_size correlation is ', r )
```

To get a feeling for the results of this analysis, we can make plots of the dog weight versus owner weight, pay, and yardsize:

```
import matplotlib.pyplot as plt

```

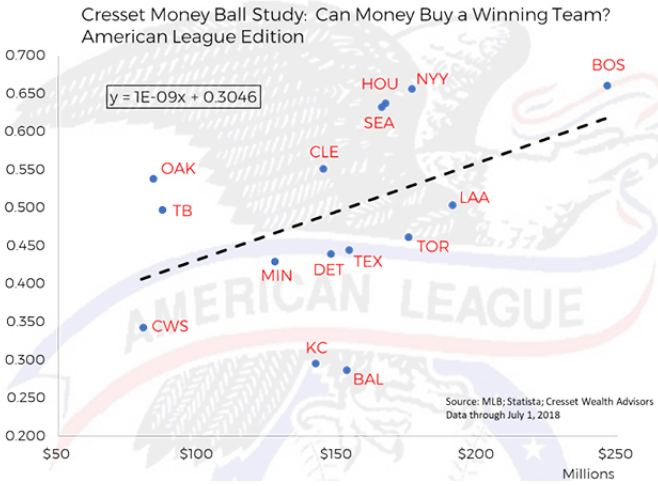
```
plt.plot ( dog, own, 'bo' )
plt.title ( 'Dog weight vs owner weight' )
plt.show ( )

plt.plot ( dog, pay, 'ro' )
plt.title ( 'Dog weight vs owner pay' )
plt.show ( )

plt.plot ( dog, yard, 'mo' )
plt.title ( 'Dog weight vs yard size' )
plt.show ( )
```



Dog weight positively correlates with yard size

5 Example: Baseball wins and payrolls

It is often argued that a baseball team can buy a championship, that a team with a high payroll has bought the best players and will win the most games. Can we estimate whether there is a relationship?



Total payroll and winning percentage seem to be correlated

The file *moneyball.txt* contains a list of winning percentages and total payroll for teams over the first half of a season. We can compute the correlation between payroll and winning percentage by commands like this:

```
import numpy as np
data = np.loadtxt ( 'moneyball_data.txt' )

wins = data[:,0]
wins_mean = np.mean ( wins )
wins_std = np.std ( wins )

pay = data[:,1]
pay_mean = np.mean ( pay )
pay_std = np.std ( pay )

n = len ( wins )

r = np.dot ( wins - wins_mean, pay - pay_mean ) / wins_std / pay_std / n
print ( 'payroll/winning correlation is ', r )
```

which results in a value of $r \approx 0.48$, which says that there is a positive correlation between the two quantities: more money and more wins seem to go together. Because the value of r is not extremely close to 1, we can also conclude that there are many underlying factors that are involved.

6 Computing Assignment #2

Let's take some data from recent house sales, in which 9 separate variables were measured. The most interesting variable is the final selling price. We'd like to see which other variables are most strongly related to the price. These will be variables whose correlation coefficient is closer to -1 or +1 than to 0.

The file *homes_data.txt* records:

- 0: **sel**, the selling price ($)
- 1: **ask**, asking price ($)
- 2: **liv**, living area (square feet)
- 3: **rms**, rooms (#)
- 4: **bed**, bedrooms (#)
- 5: **bat**, bathrooms (#)
- 6: **age**, age (years)
- 7: **lot**, lot size (acres)
- 8: **tax**, taxes ($)

Write a Python program called *hw2.py* which

1. reads `data`, the information in the file *homes_data.txt*;
2. copies columns 0, 6, and 7 of `data` as variables `sel`, `age` and `lot`;
3. computes and prints the correlation between `sel` and `age`;
4. computes and prints the correlation between `sel` and `lot`;
5. computes and prints the 9×9 matrix of **all** correlations by the commands `C=corrcoef(data.transpose())` and `print(C)`.

Email a copy of your program to Dr Schneier **mhs64@pitt.edu** before Friday, 13 September.

For your consideration: You may notice, in the matrix of correlations `C`, that some entries are 1, others are very close to 1. This tells you that these pairs of data are strongly related. Think about why this is so.

A negative value suggests that increasing one variable tends to decrease the other. Think about why this might make sense.