Calculation of mutual information in data set with a uniformly random distribution.

Isak Ederlöv

April 21, 2023

1 Objectives

The objective of this exercise is to calculate the mutual information (MI) for a data set of uniform randomly distributed points in $\mathcal{R}^2$. According to a given estimation, $I(X,Y)^{estimated} \approx 0.15 \pm 0.02$, and this exercise will show if the generated data is constistent with this estimation.

The formula for mutual information is

$$MI = \sum_{i=1}^{N} \sum_{j=1}^{N} p(x,y) log(\frac{p(x,y)}{p(x)p(y)}) \tag{1}$$

Upon inspection of this equation, one may believe that the mutual information of two series of values will always equal zero, since the joint probability density function is by definition equal to the product of the marginal. The objective of this exercise is to investigate if, and to what degree, a limited data set may infer mutual information even when it might be unexpected.

2 Methodology

To begin, a data set of 300 points was generated with a uniform random distribution.

```
x = np.array([random.uniform(0, 1) for k in range(300)])
y = np.array([random.uniform(0, 1) for k in range(300)])
```

To calculate the mutual information, the terms of equation (1) are calculated in and stored in arrays. The equation is then applied, using the function *product* to generate indices corresponding to a nested for loop, or double sum operations. Since the calculations are to be repeated, the calculation is wrapped in a function.

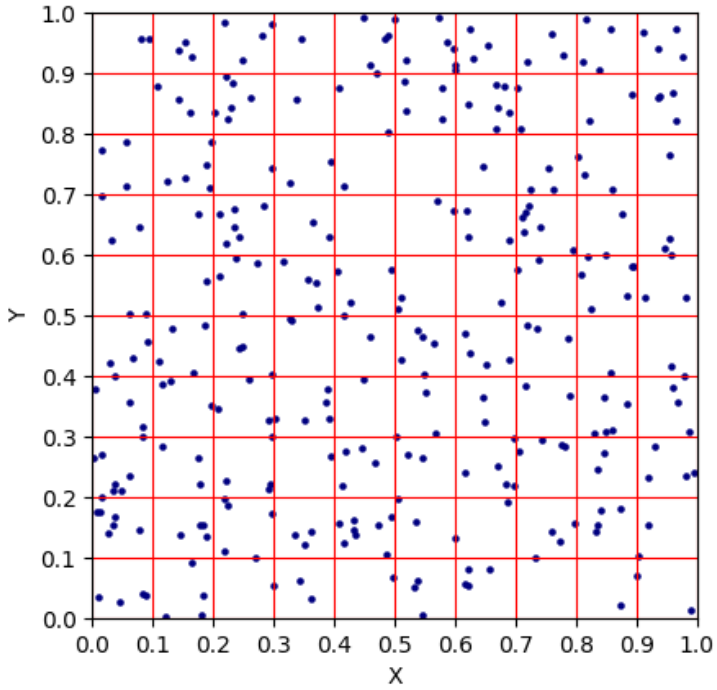
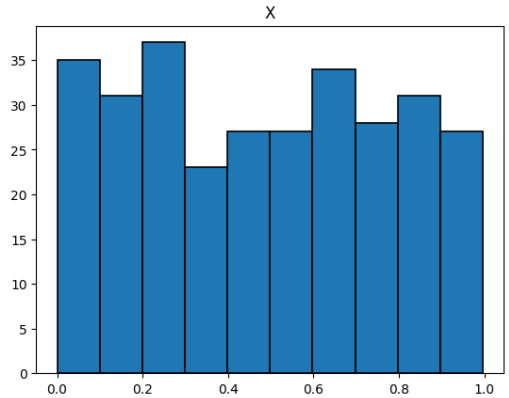


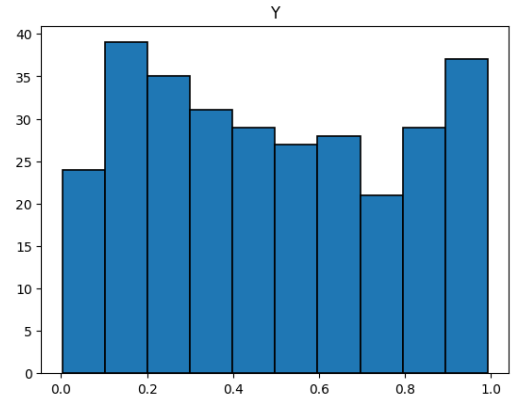
Figure 1: The generated data set divided into 100 bins.

```
def mutual(x, y, n_bins = 10) -> float:
    N = len(x)
    #2D histogram data because of automatic calculation of p(x,y)
    H, x_edges, y_edges = np.histogram2d(x, y, bins=n_bins)

    # flattened histograms equal to hist(x), hist(y)
    Px = np.sum(H, axis=1) / N
    Py = np.sum(H, axis=0) / N
    Pxy = H / N

    MI = 0
    # the product function is equivalent to a nested for loop
    for i, j in product(range(n_bins), range(n_bins)):
        if Pxy[i,j] > 0: #Safeguard to prevent log(0)
            MI += Pxy[i,j] * np.log(Pxy[i,j] / (Px[i] * Py[j]))
return MI
```

The variable H in this code snippet is here akin to an two-dimensional array of bins, as visualised in Figure 1. The variables *x_edges* and *y_edges* are not used, but are included to generate the right output parameters of the function *histogram2d*.



(a) Histogram of x series distribution



(b) Histogram of y series distribution

With this abstraction, regenerating the experiment to calculate the distribution of MI is now possible.

```
MI_values = np.empty(300)
for i in range(300):
    x_exp = np.array([random.uniform(0, 1) for k in range(300)])
    y_exp = np.array([random.uniform(0, 1) for k in range(300)])
    MI_values[i] = mutual(x_exp,y_exp)

plt.hist(MI_values, 10, edgecolor='black', linewidth=1.2)
plt.xlabel("Observed mutual information")
plt.ylabel("Counts")
```

A sense of how the amount of points affect the MI would be useful for discussion and understanding of the concept. All of the previous steps are thus refactored and calculated for a wide variety of n.

```
def different_n(n):

    MI_values = np.empty(100)
    for i in range(100):
        x_exp = np.array([random.uniform(0, 1) for k in range(n)])
        y_exp = np.array([random.uniform(0, 1) for k in range(n)])
        MI_values[i] = mutual(x_exp,y_exp)
    return np.mean(MI_values)

x_exp = np.linspace(50, 5000, 451)
y_exp = np.empty(len(x_exp))

for i, n in enumerate(x_exp):
    y_exp[i] = different_n(int(n))
```

3 Results

Figure 3 shows that the average MI is approximately 0.15, with the majority of the data within ± 0.02, which is consistent with the given estimate.

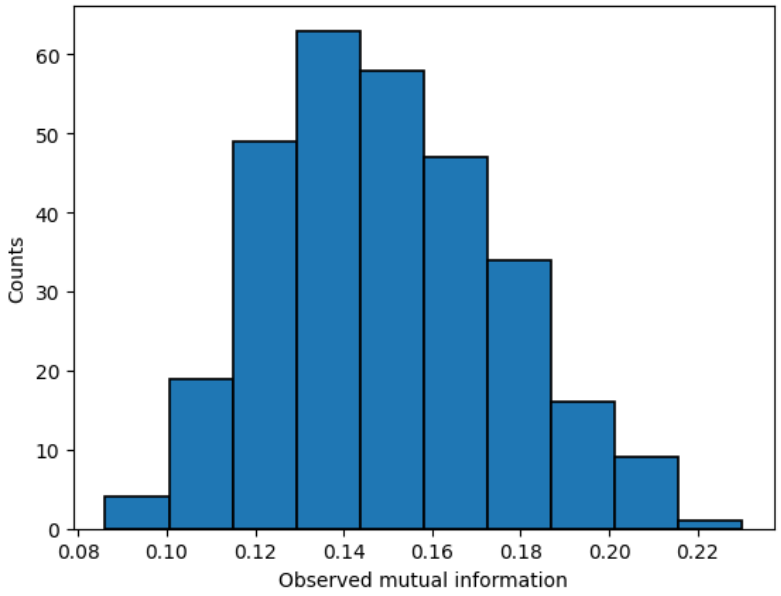


Figure 3: Distribution of calculated MI during 300 instances of regenerated data set.

4 Discussion

When we compute the mutual information, we are essentially comparing the joint probability distribution to the product of the marginal probability distributions. Even if the two variables are independent, there may still be some values of x and y where the observed joint probability distribution is different from the product of the observed marginal probability distributions. This can happen due to random sampling variation, especially with a small number of samples.

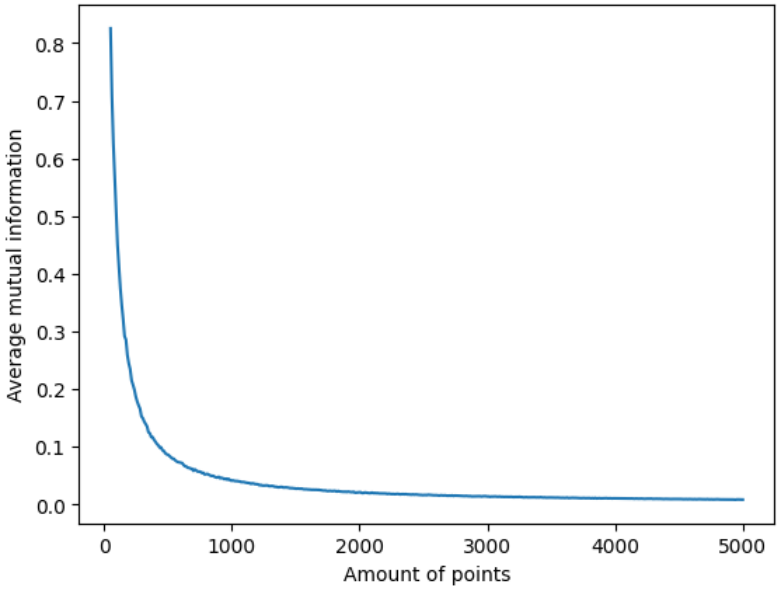


Figure 4: Average MI for different amount of points generated.

Therefore, a small positive mutual information value between two independent variables does not necessarily imply that they are dependent. It could simply be due to random variation in the joint probability distribution. However, if the mutual information value is significantly larger than zero, it could suggest some dependence between the variables.

The results visualised in Figure 4 are expected and intuitive for the reasons previously explained. With more points generated, the points become more likely to be evenly spread, and thus display less mutual information. On a quick inspection of Figure 4, the curve appears similar to one of the form $\frac{a}{bx}$. It would be interesting to investigate further why this is the case and make and understand a model of this behaviour.

5 Summary

This exercise shows that mutual information can be present in data sets where there mathematically shouldn't be. It also shows that the calculated mutual information decreases as the size of the data set increases.