Faculty of Technology, Policy and Management

SPM 4450 Fundamentals of Data Analytics

Assignment 2

Verifying The Curse of Dimensionality

Authors	:	Paraskevi Kokosia	4417062
		Bramka Arga Jafino	4516516
		Rhythima Shinde	4516389
Professor	:	Prof. Jan Van Der Berg	

Motivation

The curse of dimensionality is one of the key concerns in data analytics. It refers to “various phenomena that arise when analyzing and organizing data in high-dimensional spaces whose components are independently distributed (often with hundreds or thousands of dimensions) that do not occur in low-dimensional settings such as the physical space commonly modeled with just three dimensions (Hastie, Tibshirani, & Friedman, 2009).”

Through this assignment, we are trying to prove that such phenomenon exist, using simulations run over a certain data set in different dimensional spaces. A simulation analysis will be performed in high dimensional spaces ranging from 1 to 32 by using k-nearest neighbor predictions. Another simulation regarding the sample size and the different dimensional spaces will be executed. Conclusions regarding both experiments will be drawn at the end of this report.

Analysis

We begin by generating points in the p dimension cube using the function y = $e^{-a||x||}$ by selecting a value for a=3,

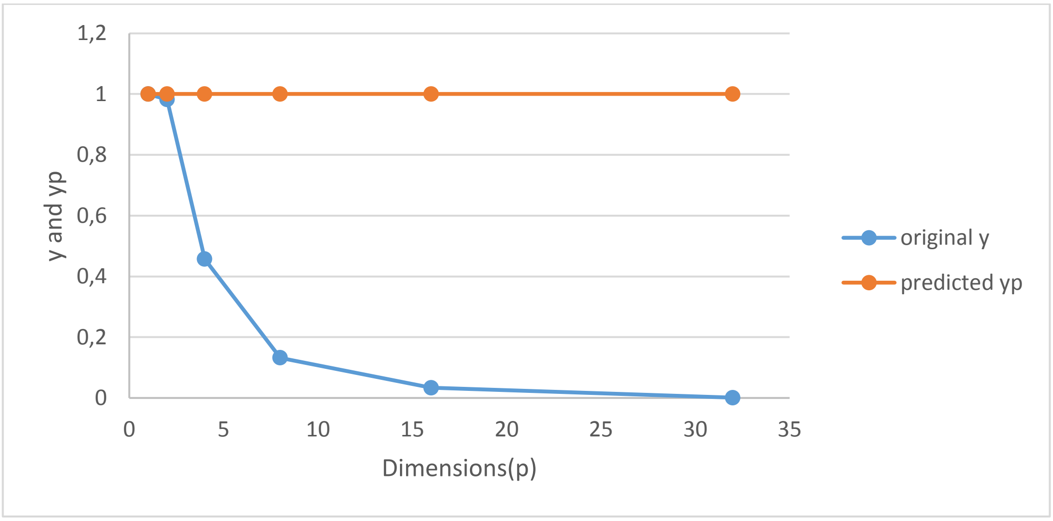
Where ||x||=$\sqrt{X1^2 + X2^2 \ldots\ldots Xp\text{^}2}$, where Xi (for i =1 to p) is the value of x in the ith dimension.

Then we have created a series of six data sets each one of 1000 points uniformly distributed for function y = $e^{-a||x||}$ for the suggested dimensions p= {1, 2, 4, 8, 16, 32}. We used the k nearest neighbor rule and more specifically for k=1 to predict y (y_p) again for x=0. In this approach, the distance between the nearest neighbors to x=0 was used to calculate the 1 nearest neighbor rule in p dimension hypercube.

In the following graph, the six predictions of the 1st nearest number for the suggested dimensions of p are shown (and the orange line depicts the original value of y).

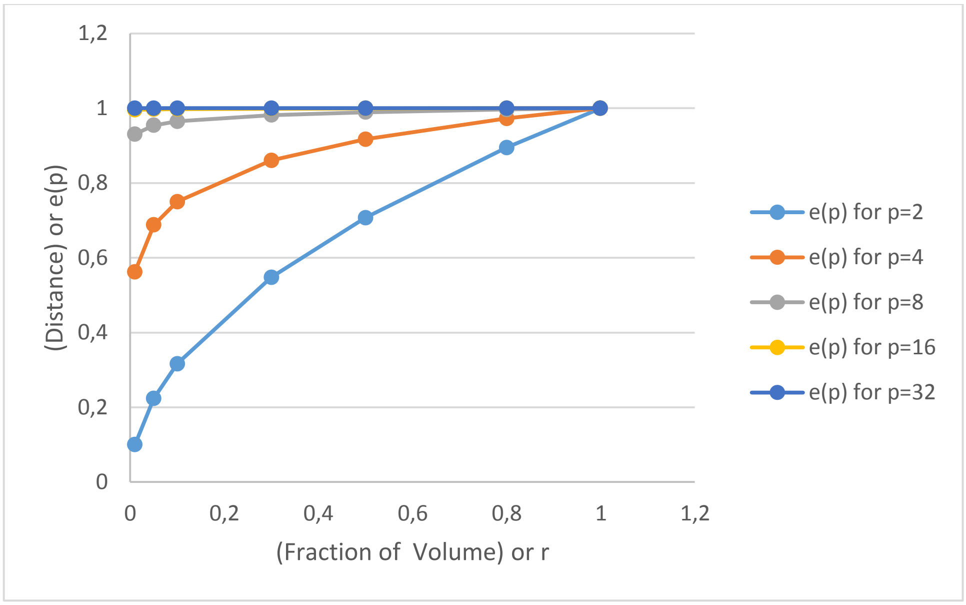
By predicted (y_p) and the value(y) in Figure 1 below, we mean the expected prediction error at x=0 for 1 nearest neighbor and the original value of y respectively.

Figure 1 Dimensions versus predicted value of y



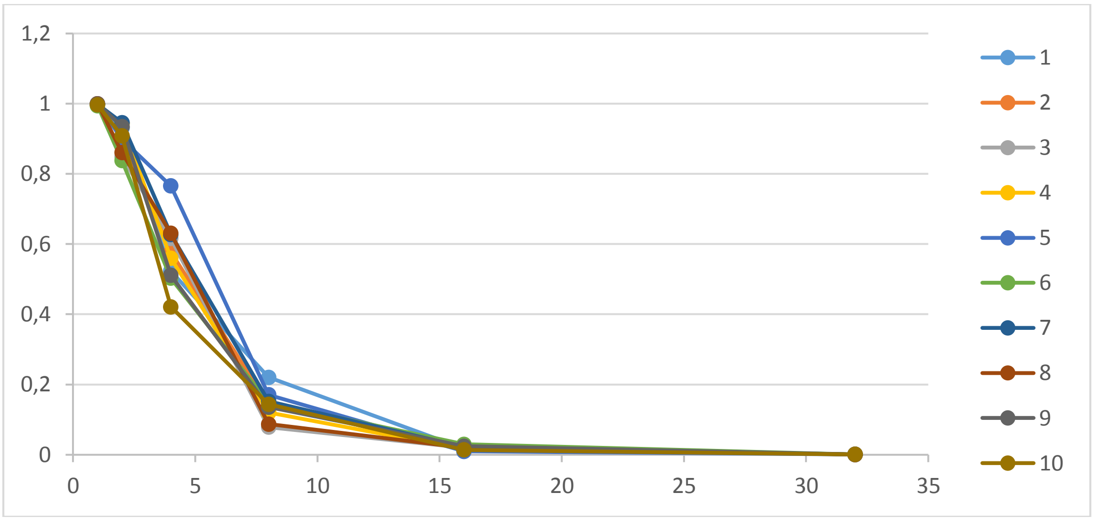
For hypercube with p dimensions the expected length equals $e_p\ (r) = r^{1/p}$ as given on slide 83 of ‘slides of lecture 1-2’ of the course. The plot of this formula, as it can be seen below in figure 2, shows that for covering a small amount of data of say even 5%, we almost need to cover the 100% of the coordinates of the hypercube of 32 dimensions.

Figure 2 Curse of dimensionality for hypercube with p dimensions



In order to analyse the sensitivity of the curse of dimensionality phenomenon with regard to the randomness of the data, we have repeated the experiment for 10 times. As can be seen in the figure 3 below, it can be drawn that the data generation process is not very sensitive since the predicted value y (yp) does not differ that much.

Figure 3 10 simulation results for dimension (p) versus predicted y (yp)



In order to keep *error(p)* approximately the same while increasing p, a growing number of sample points is needed. For this reason, we have calculated the prediction error err_p for sample sizes of 10, 100, 500, 1000 for increasing dimensions. The prediction error was calculated as $error_p = 1 - y_0{}^p$. The graphs below illustrate the prediction error in relation to the suggested dimensions and the functions of dimension with respect to the sample size accordingly.

Figure 4 Error for varying sample sizes (dotted line shows the logarithmic fit curves)

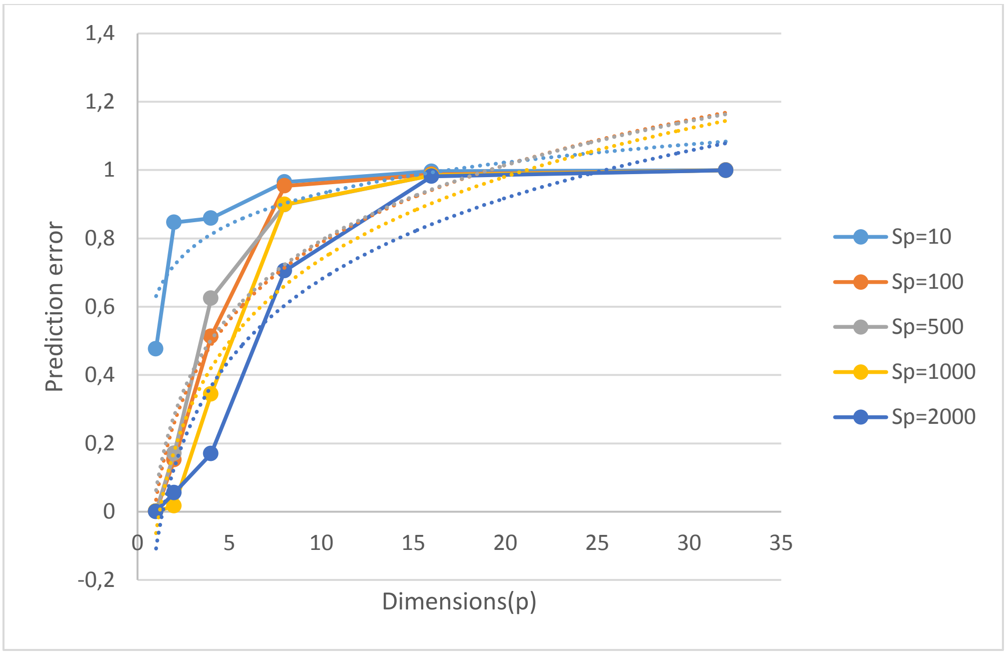
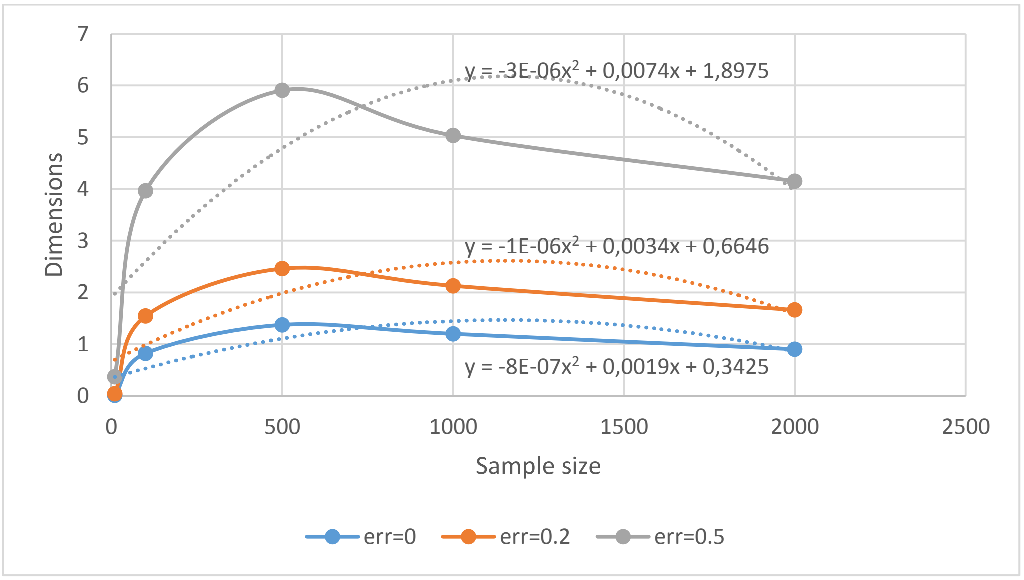


Figure 5 Function of dimension with respect to sample size (dotted line shows the best fit polynomial curves)



From the simulation we found that:

$$Sp = a \pm \sqrt{(b + cp)}$$

Where a, b, and c are constant, S_p is the sample size and p is the number of dimensions

For example, for error=0,

a= 1.87e3, b=1.84e6, c=1.25e6 (here, me6 means m X 10^6)

So,

$$Sp = 1.87e3 \pm \sqrt{(1.84e6 + 1.25e6 * p)}$$

Conclusions

According to the curse of dimensionality: more are the number of dimensions, lower is prediction accuracy. In this case, the effects are seen as 'nearest points' prediction' in high dimensions are generally far away from the original values/ the starting point as mentioned in the figure 1. From the simulations executed and tested above with the 1 nearest neighbour rule,we conclude that predictive power in high dimensional spaces is low as shown in Figure 1. For small dimensions the nearest neighbour prediction is close to x= 0 for x=0 as the value of the nearest neighbour is less. But as the dimensions increase, the nearest neighbour gets farther away and thus the error in prediction of the values of y increases (since the nearest points are far away). Thus as dimension increases, the nearest neighbours stays away from the target point and higher bias errors are incurred and variance remains almost constant with increasing dimension.

We found also that the predictions are similar to the result of the hypercube $e_p(r) = r^{1/p}$. It can be imagined "as the side-length of the sub-cube needed to capture a fraction r of the volume of the data, for different dimensions p (Hastie, Tibshirani, & Friedman, 2009)." The Figure 2 helps understand this curse of dimensionality which shows that to cover as small as 5% of data, we need to cover almost 100% of range of each dimension/coordinate in high dimension space (e.g. p=32).

Regarding the sensitivity (as shown in Figure 3), after performing the simulation 10 times, we concluded that the data generation process is not very sensitive since the predicted value y (y_p) does not differ that much.

From Figure 4 and 5, conclusions can be drawn for changes in prediction error with the changing sample size and the dimensions. The first conclusion from Figure 4 is that for the same dimension increasing sample size reduces the prediction error. Figure 5 concludes that the best fit function of sample size in terms of dimensions (while keeping constant error), turns out like a square root function as shown in equation 1 and 2. This may not be a generalised function and as seen from Figure 5, it changes significantly with the error value.

Bibliography

Hastie T., T. R. (2009). *The elements of statistical learning--data mining, inference, and prediction.* (2nd ed.). New York: Springer.

Appendix

The code written in python for generating the values and simulations is given below:

```python
import numpy as np
import matplotlib.pyplot as plt
import csv
import math

ydata=[]
z=[]
a = 3
n=[10,100,500,1000,3000]
dimensions = [1,2,4,8,16,32]
for ind in range (len(n)):
    print "n=",n[ind]
    for index in range(len(dimensions)):
        i=0
        #while i<10
        y2=[]
        print "Dimension=", dimensions[index]
        #while i<10:
        xdata = np.random.uniform(0,1, size=(n[ind],dimensions[index]))
```

```
 #### 2. calculating the ||X||#####
z2 = xdata**2 #square the data file
z3 = z2.sum(axis=1) #sum the squared data file for each row
zeuclid = [x**(0.5) for x in z3] #calculate the euclidean distance


y1 = [math.exp( x * (-a) ) for x in zeuclid] #### 3. calculating the y####
#ydata.append(y1)
#y = np.array(y)
dataset = np.array(zip(xdata,ydata))
#b = open('x_1.csv', 'w')
#m = csv.writer(b)
#m.writerows(xdata)
#m.writerows(y)
m=min(zeuclid)
y3=math.exp( m * (-a))
y2.append(y3)
#i+=1
print y2
```