Zaid Parvej Patel (zp2090):

CS3943 Mining Massive Data Sets

Locality Sensitive Hashing

1 LSH evaluation (20 points)

Evaluate the S curve, which you might recall we discussed in class is as follows: and for the following values of r and b: (show all your work):

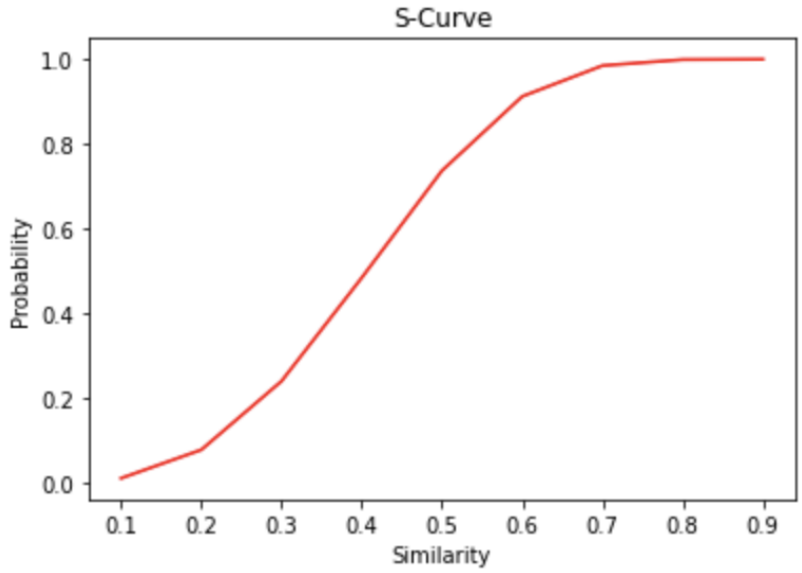
- r =3 and b = 10
- r =6 and b = 20
- r =5 and b = 50

Solution:

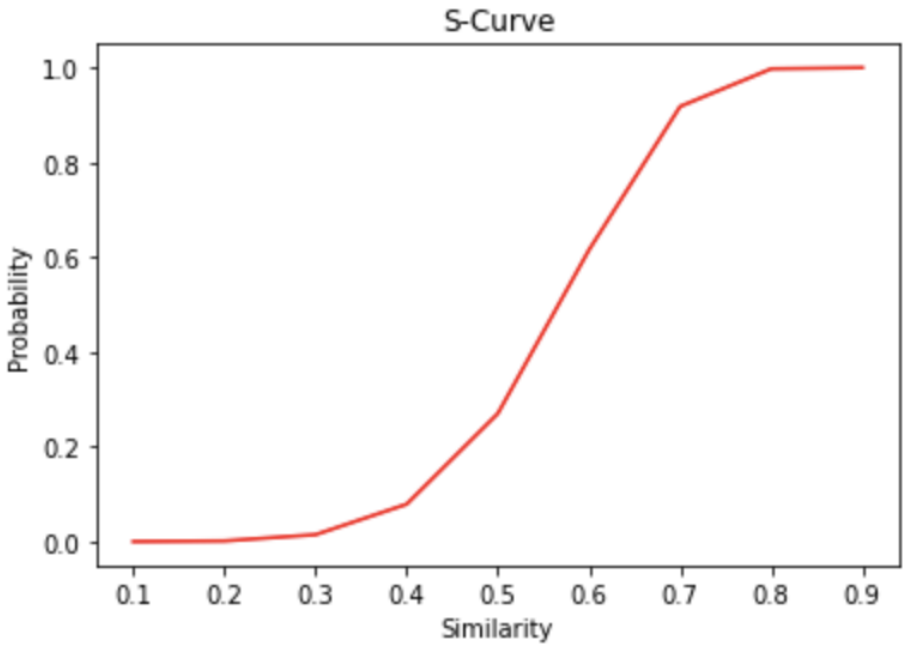
```
import numpy as np
import matplotlib.pyplot as plt

R = [3,6,5]
B = [10,20,50]

S = np.arange(0.1,1,0.1)
for r,b in zip(R,B):
 Pr = []
 for s in S:
   p = 1 - (1-(s)**r)**b
   Pr.append(p)
 print("Pr(r=",r,"& b=",b,") = ",Pr)

 plt.title("S-Curve")
 plt.xlabel("Similarity")
 plt.ylabel("Probability")
 plt.plot(S, Pr, color ="r")
 plt.show()
 print("\n")
```

Output:

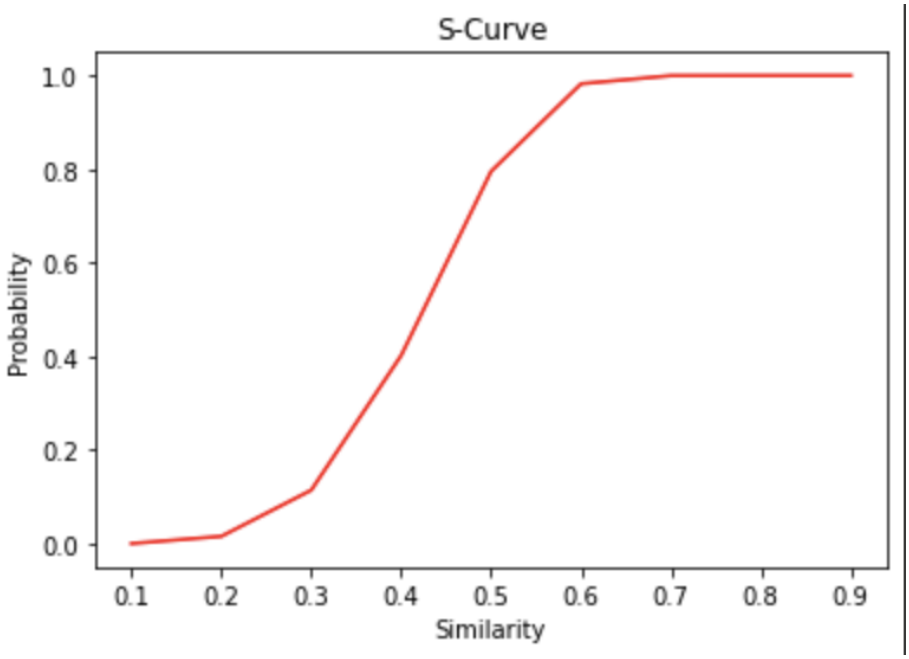
Pr(r= 3 & b= 10) = [0.009955119790251765, 0.07718058804273675, 0.23944889319887064, 0.4838707317677322, 0.7369244238361716, 0.9122674753991765, 0.985015105295655, 0.9992340538808936, 0.9999978635491371]



Pr(r= 6 & b= 20) = [1.9999810001669616e-05, 0.001279222058761964, 0.014479466504172311, 0.07880932311056232, 0.27018714400947597, 0.6154146360312677, 0.9181859965846744, 0.9977121251546806, 0.9999997398129465]



Pr(r= 5 & b= 50) = [0.0004998775195954597, 0.01587519984502117, 0.11453988231042189, 0.4022839522088044, 0.7955506304323648, 0.9825338277068608, 0.9998989958361557, 0.9999999976077777, 1.0]



2 LSH for approximate near neighbor search(30 points)

A dataset of images patches.csv, is provided. Each row in this dataset is a 20 × 20 image patch represented as a 400-dimensional vector. We will use the $L1$ distance metric on R400 to define similarity of images. We would like to compare the performance of LSH-based approximate near neighbor search with that of linear search (by linear search we mean comparing the query point z directly with every point x). You should use the code provided with the dataset for this task.

The included starter code in **`lsh.py`** marks all locations where you need to contribute code with `TODOs`. In particular, you will need to use the functions lsh_setup and lsh_search and implement your own linear search. The default parameters $L = 10$, $k = 24$ to lsh_setup work for this exercise, but feel free to use other parameter values as long as you explain the reason behind your parameter choice.

- For each of the image patches in columns 100, 200, 300, . . . , 1000, find the top 3 near neighbors[1] (excluding the original patch itself) using both LSH and linear search. What is the average search time for LSH? What about for linear search?

- Assuming $\{z_j \mid 1 \le j \le 10\}$to be the set of image patches considered (i.e., z_j is the image patch in column 100j), to be the approximate near neighbors of z_j found using LSH to be the (true) top 3 near neighbors of z_j found using linear search, compute the following error measure:

  Plot the error value as a function of L (for $L = 10, 12, 14, \ldots, 20$, with $k = 24$). Similarly, plot the error value as a function of k (for $k = 16, 18, 20, 22, 24$ with $L = 10$). Briefly comment on the two plots (one sentence per plot would be sufficient).

- Finally, plot the top 10 near neighbors found[2] using the two methods (using the default $L = 10$, $k = 24$ or your alternative choice of parameter values for LSH) for the image patch in column 100, together with the image patch itself. You may find the function plot useful. How do they compare visually?

(i) Average search time for LSH and linear search

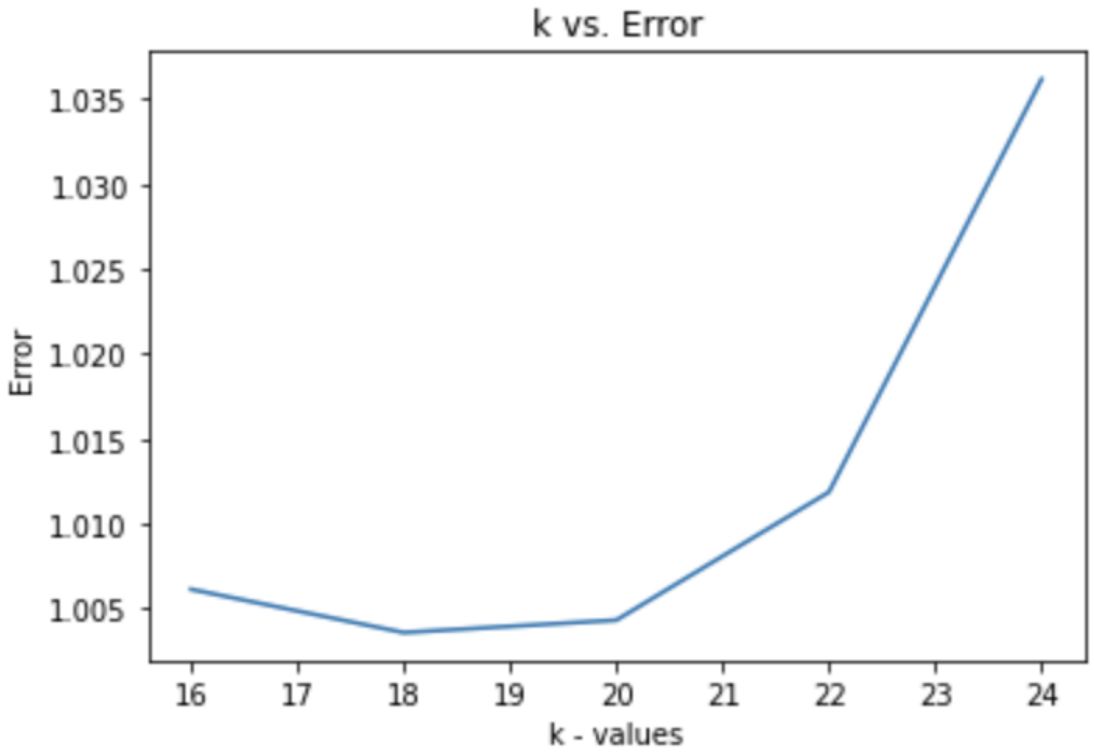
Solution:

The average runtime for LSH Search : 0.2868274927139282

The average runtime for Linear Search : 0.7164834499359131

(ii) Plot for error values vs L and error value vs K, and brief comments for each plot

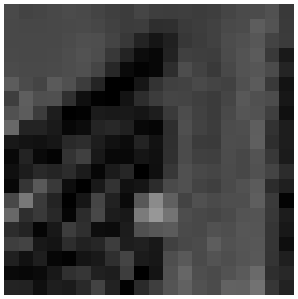
From the above plot, it can be observed that the Error value decreases for the increasing values of L. This concludes that error is inversely proportional to the value of L when k is constant.



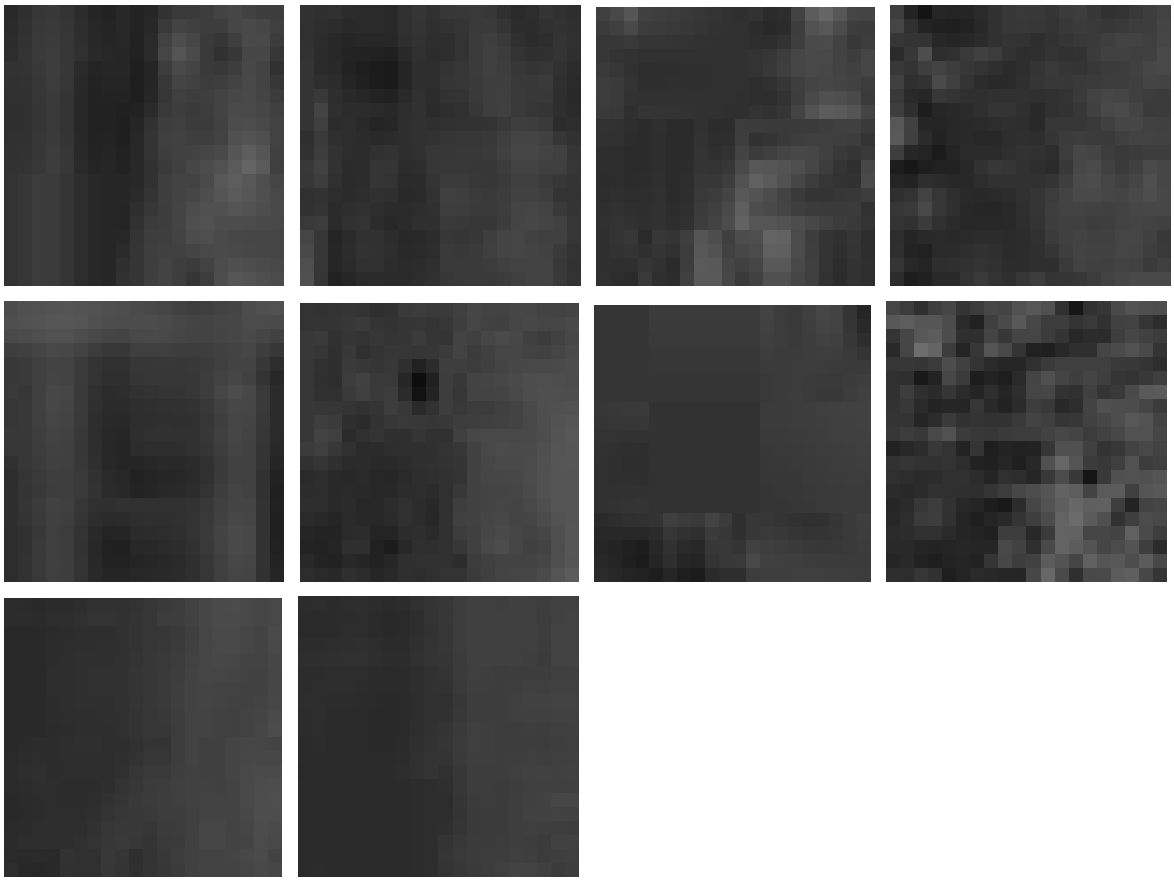
From the above plot, it can be observed that the Error value increases for the increasing values of k. This concludes that error is directly proportional to the value of k when L is constant.

(iii) Plot of 10 nearest neighbors found by the two methods(also include the original image) and brief comparison

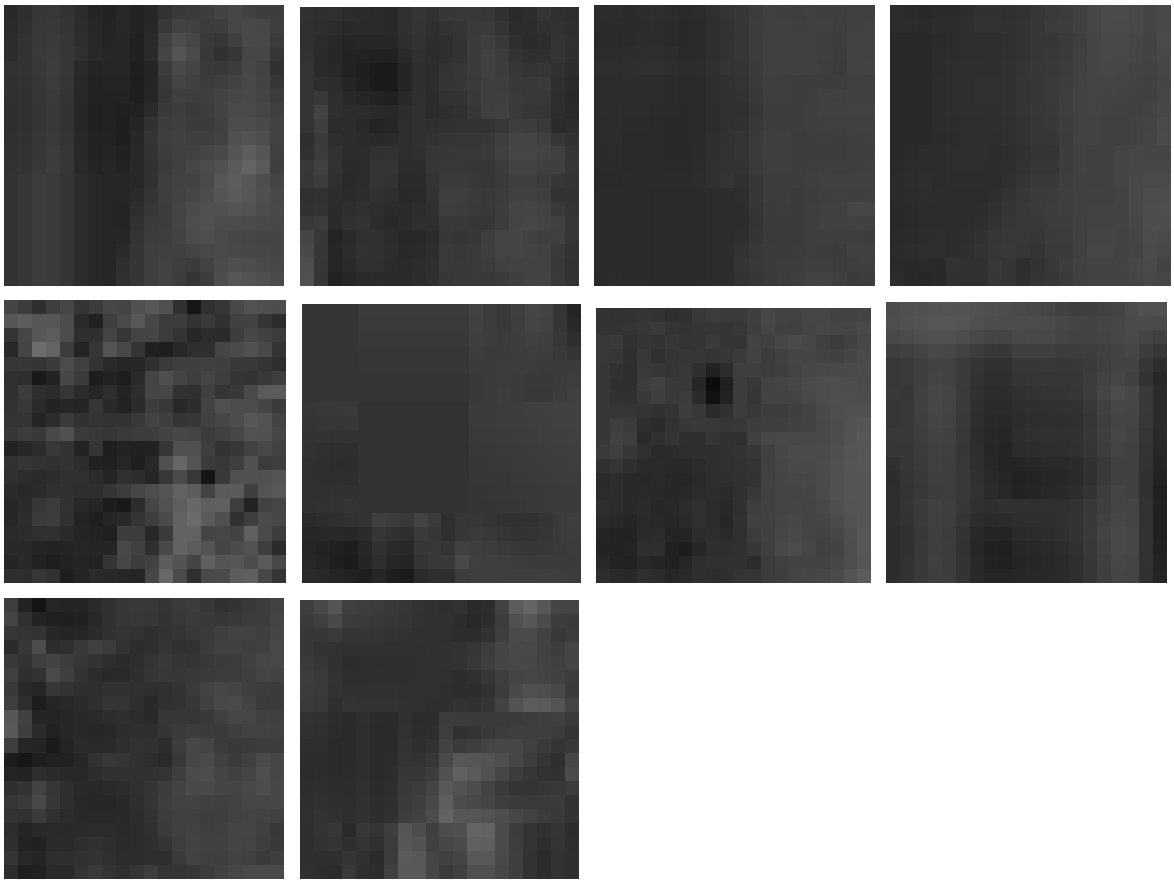
Original Image:



Nearest neighbors using LSH:



Nearest Neighbors using Linear Search:



From the above images, it can be seen that the nearest neighbors obtained using LSH as well as Linear Search are the same and they provide similar results. The only difference which is observed that the execution time of LSH is comparatively lesser (3 times) than Linear Search.

(iv) Code for all the above.

The code for the above results is executed in the link to the google colab notebook:

https://colab.research.google.com/drive/1-Ua3kza_mvDShvP16_CWIBWUWunRNc2v?authuser=1#scrollTo=HvDT7jMqoxXp