CS 4140/6140: Data Mining HW 3

Novella Alvina - u1413401

February 2023

Part I

Choosing r, b (35 points)

Consider computing an LSH using t = 200 hash functions. We want to find all object pairs which have Jaccard similarity above $\tau = 0.75$.

1 Question A

Estimate the best values of hash functions b within each of r bands to provide the S-curve with good separation at τ:

```
# choosing r and b
t = 200
tau_js = 0.75

b = -math.log(t,tau_js)
r = t/b
b,r
```

Figure 1: Finding estimation of r and b by approximation

Result:
b = 18.417266398955398
r = 10.859374875054407
Hence the approximation value: 0.8785349341564557

```python
r = [1,2,4,5,8,10, 12.5, 200, 100, 50, 40, 25, 20, 16 ]
b = [200, 100, 50, 40, 25, 20, 16, 1,2,4,5,8,10, 12.5]

for i in range(len(r)):
    print("r: ", r[i], "| b: ", b[i], "| approx: ", 1/r[i] ** (1/b[i]))
```

Figure 2: Finding estimation of r and b by trial and error

Result:

```
r:  1 | b:  200 | approx:  1.0
r:  2 | b:  100 | approx:  0.9930924954370358
r:  4 | b:  50 | approx:  0.9726549474122855
r:  5 | b:  40 | approx:  0.9605627697295936
r:  8 | b:  25 | approx:  0.9201876506248752
r:  10 | b:  20 | approx:  0.8912509381337456
r:  12.5 | b:  16 | approx:  0.853971002857656
r:  200 | b:  1 | approx:  0.005
r:  100 | b:  2 | approx:  0.1
r:  50 | b:  4 | approx:  0.3760603093086394
r:  40 | b:  5 | approx:  0.4781762498950185
r:  25 | b:  8 | approx:  0.668740304976422
r:  20 | b:  10 | approx:  0.7411344491069477
r:  16 | b:  12.5 | approx:  0.8010698775896221
```

Based on this result, the closest one to 0.75 is r = 16 and b = 12.5. So we're going to use that approximation value for Q2.B

2 Question B

Considering the pair-wise similarities in the Question, use your choice of r and b and f $(\cdot)$ designed to find pairs of objects with similarity greater than τ : what is the probability, for each pair of the four objects, of being estimated as similar (i.e., similarity greater than $\tau = 0.75$)?

```python
D = {'AB': 0.77, 'AC': 0.25, 'AD': 0.33, 'BC': 0.2, 'BD': 0.55, 'CD': 0.91}
r = 16
b = 12.5
for i,s in enumerate(D.values()):
    D[list(D.keys())[i]] = 1 - (1-s**b)**r
D
```

Figure 3: Finding pairs of objects with similarity greater than τ

```
{'AB': 0.463034660558881,
 'AC': 4.7683705162171464e-07,
 'AD': 1.5329963128030144e-05,
 'BC': 2.930858888916532e-08,
 'BD': 0.00905323267301461,
 'CD': 0.9972107202823838}
```

Part II
Generating Random Directions

"generating random unit vector" in Section 4.6.4 in M4D

3 Question A

Describe how to generate a single random unit vector (chosen uniformly over from the space of all unit vectors $\mathbb{S}^{d-1}$) in d = 10 dimensions. To generate randomness, use only the operation u ← unif(0,1), which generates a uniform random variable between 0 and 1 (then other linear algebraic and trigonometric, etc operations are allowed). (This random uniform value can be called multiple times.)

Operation u ← unif(0,1) gives us 10 uniform random numbers as u1,u2,u3,...,u10. Using the Box-Muller transform, we can generate the two independent 1-Dimensional Gaussian random variable and got the random unit vector by

$$\vec{v} = [v1, v2, ..., v10, v11]/||v|| \tag{1}$$

```
[205] def random_unit_vector(dim):
        random_vector = []

        for d in range(int(dim/2)):

          u1 = random.uniform(0,1)
          u2 = random.uniform(0,1)

          y1 = math.sqrt(-2*np.log(u1))*math.cos(2*np.pi*u2)
          y2 = math.sqrt(-2*np.log(u1))*math.sin(2*np.pi*u2)

          random_vector.append(y1)
          random_vector.append(y2)

        unit_vector = random_vector/np.linalg.norm(random_vector)

        return unit_vector

      # random_unit_vector(dim)
```

Figure 4: generating random unit vectors

Result:

```
array([-0.05841863, -0.45604173, -0.21721037,  0.67171325,  0.04858661, 0.23626545,
0.10112409,  0.34509894, -0.19294789, -0.2559373 ])
```

4 Question B

Generate t = 200 unit vectors in Rd for d = 100. Plot of cdf of their pairwise dot products

```
t = 200
d = 100
t_unit_vectors = []
for i in range(t):
  t_unit_vectors.append(random_unit_vector(d))

comb = list(combinations(np.arange(len(t_unit_vectors)), 2))
pairwise_dotproduct = []
for c in comb:
  pairwise_dotproduct.append(np.dot(t_unit_vectors[c[0]], t_unit_vectors[c[1]]))


x = np.sort(pairwise_dotproduct)
y = np.arange(len(comb))/float(len(comb))
# ya = np.exp(-x**2)
# ya/=(ya*0.05).sum()
# y = np.cumsum(ya*0.05)
# plotting the line 2 points
plt.plot(x, y)

plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.title('pairwise dot products cdf')
plt.show()
```

Figure 5: Generate a single random unit vector in d = 10 dimensions

Result:

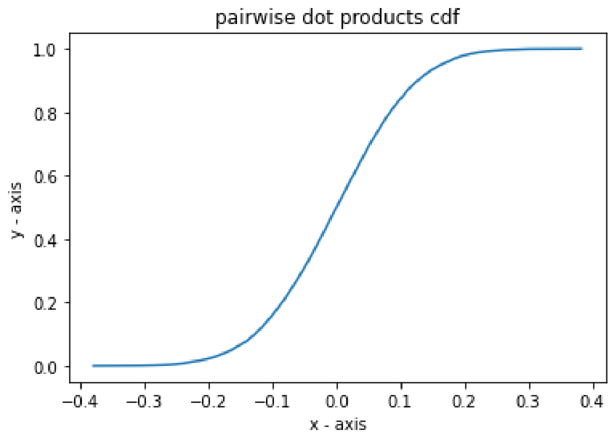

Figure 6: Generate a single random unit vector in d = 10 dimensions result

Part III
Angular Hashed Approximation

n = 500 data points in $\mathbb{R}^d$ for d = 100 in data set R, given in Canvas. We will use the angular similarity, between two vectors $a, b \in \mathbb{R}^d$ based on 4.6.4 in M4D

5 Question A

Compute all pairs of dot products and plot a cdf of their angular 2 similarities. Report the number with angular similarity more than $\tau = 0.75$.

Result:

```
[208] n = 500
      d = 100
      tau_js = 0.75

      def angular_similarities(a, b):
          anorm = np.linalg.norm(a)
          bnorm = np.linalg.norm(b)
          dot_product = np.dot(a,b)/(anorm * bnorm)
          return 1- np.arccos(dot_product)/np.pi
```

Figure 7: Angular Similarity

```
comb = list(combinations(np.arange(len(dataset)), 2))
dotproduct_angular_similarities = []
count = 0
for c in comb:
    angular_similarity = angular_similarities(dataset[c[0]], dataset[c[1]])
    dotproduct_angular_similarities.append(angular_similarity)
    if angular_similarity > 0.75:
      count += 1

x = np.sort(dotproduct_angular_similarities)
y = np.arange(len(comb))/float(len(comb))

plt.plot(x, y)

plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.title('angular similarities dot products cdf')
plt.show()
```

Figure 8: Compute all pairs of dot products and plot a cdf of their angular 2 similarities

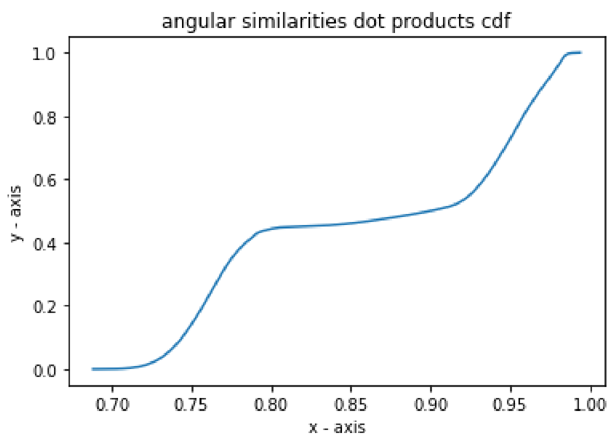

Figure 9: CDF of their angular 2 similarities of dot product

6 Question B

Now compute the angular similarities among t combination 2 pairs of the t random unit vectors from 2 Q2.B. Again plot the cdf, and report the number with angular similarity above $\tau = 0.75$.

```
comb = list(combinations(np.arange(len(t_unit_vectors)), 2))
dotproduct_angular_similarities_t = []
count = 0
for c in comb:
    angular_similarity = angular_similarities(t_unit_vectors[c[0]], t_unit_vectors[c[1]])
    dotproduct_angular_similarities_t.append(angular_similarity)
    if angular_similarity > 0.75:
      count += 1

x = np.sort(dotproduct_angular_similarities_t)
y = np.arange(len(comb))/float(len(comb))

plt.plot(x, y)

plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.title('angular similarities dot products using unit vectors from Q2.B cdf')
plt.show()
```

Figure 10: compute the angular similarities among t combination 2 pairs of the t random unit vectors from 2 Q2.B.

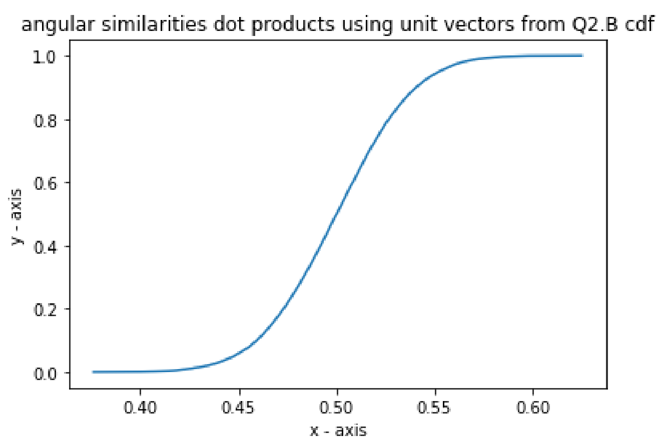

Figure 11: CDF of their angular 2 similarities ofamong t combination 2 pairs of the t random unit vectors from 2 Q2.B.