Asmt 3: Distances and LSH

Turn in through Canvas by 2:45pm, then come to class:
Wednesday, February 5
100 points

Overview

In this assignment you will explore LSH and Euclidean distances.

You will use a data set for this assignment:

- `http://www.cs.utah.edu/~jeffp/teaching/cs5140/A3/R.csv`

It is recommended that you use LaTeX for this assignment (or other option that can properly digitally render math). If you do not, you may lose points if your assignment is difficult to read or hard to follow. Find a sample form in this directory: `http://www.cs.utah.edu/~jeffp/teaching/latex/`

1 Choosing r, b (35 points)

Consider computing an LSH using $t = 160$ hash functions. We want to find all object pairs which have Jaccard similarity above $\tau = .85$.

A: (15 points) Use the trick mentioned in class and the notes to estimate the best values of hash functions b within each of r bands to provide the S-curve

$$f(s) = 1 - (1 - s^b)^r$$

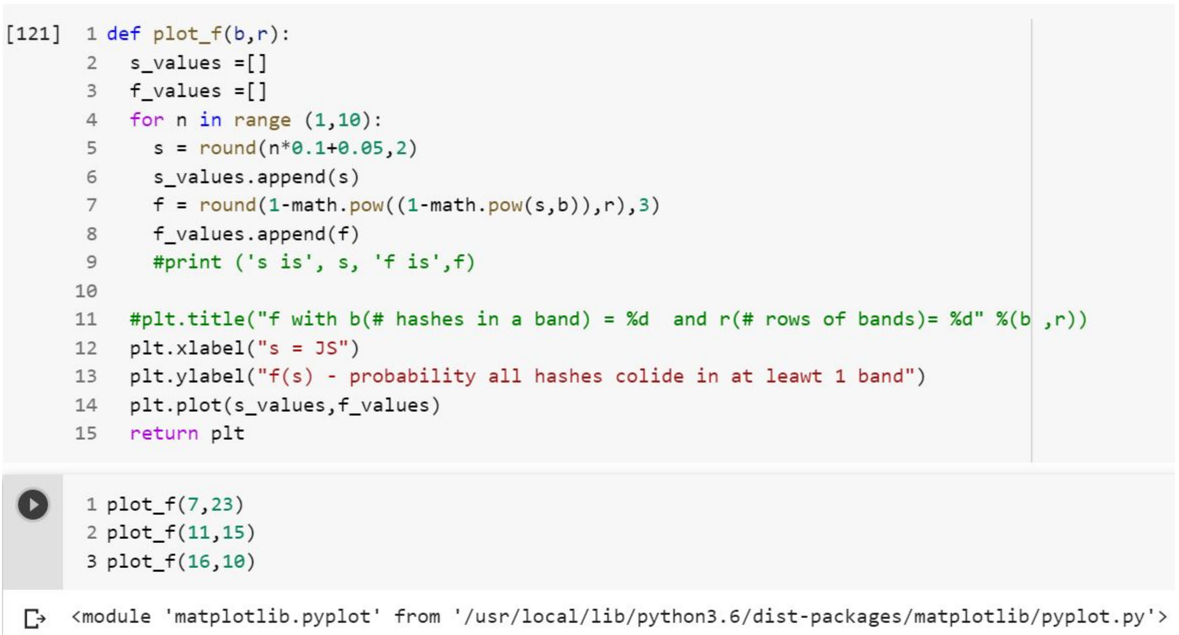
with good separation at τ. Report these values.

We can approximate b where f has the steepest slop fiven t and τ
$b \approx -log_\tau(t) = -log_{0.85}(160) \approx 31$
$r = \frac{t}{b} = \frac{160}{31} \approx 5$

If we have a budget to stay within $t = 160$ then we run a few tests for different b and r, $br <= t$ and look at $\tau = 0.85$ since we are only interested in finding anything with similarity above 0.85.
With $b = 11$ and $r = 15$, the probability of collision (at leat 1 band) is the high enough but yet the slop is sharp enough to have a good separation. So we'll pick those to be the optimal values. This is shown in the graph below (orange line)

if b is 7 ,r is 23 ,f(0.85) is 1.0
if b is 8 ,r is 20 ,f(0.85) is 0.998
if b is 9 ,r is 18 ,f(0.85) is 0.991
if b is 10 ,r is 16 ,f(0.85) is 0.97
if b is 11 ,r is 15 ,f(0.85) is 0.936
if b is 12 ,r is 13 ,f(0.85) is 0.864
if b is 13 ,r is 12 ,f(0.85) is 0.787
if b is 14 ,r is 11 ,f(0.85) is 0.697
if b is 15 ,r is 11 ,f(0.85) is 0.634
if b is 16 ,r is 10 ,f(0.85) is 0.538

if b is 17 ,r is 9 ,f(0.85) is 0.444
if b is 20 ,r is 8 ,f(0.85) is 0.271
if b is 25 ,r is 6 ,f(0.85) is 0.099
if b is 30 ,r is 5 ,f(0.85) is 0.038

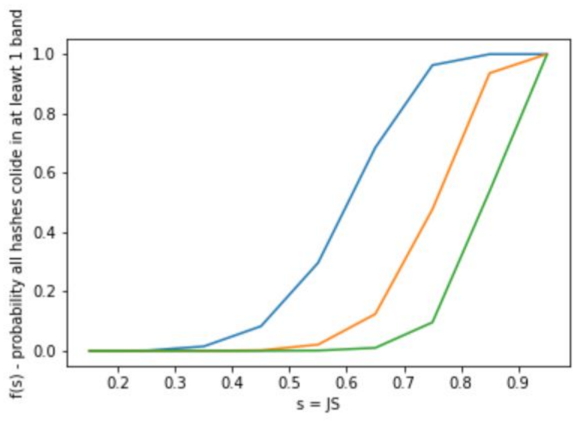


```
[121]
def plot_f(b,r):
  s_values =[]
  f_values =[]
  for n in range (1,10):
    s = round(n*0.1+0.05,2)
    s_values.append(s)
    f = round(1-math.pow((1-math.pow(s,b)),r),3)
    f_values.append(f)
    #print ('s is', s, 'f is',f)

  #plt.title("f with b(# hashes in a band) = %d  and r(# rows of bands)= %d" %(b ,r))
  plt.xlabel("s = JS")
  plt.ylabel("f(s) - probability all hashes colide in at leawt 1 band")
  plt.plot(s_values,f_values)
  return plt
```

```
plot_f(7,23)
plot_f(11,15)
plot_f(16,10)
```

```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.6/dist-packages/matplotlib/pyplot.py'>
```



B: (15 points) Consider the 4 objects A, B, C, D, with the following pair-wise similarities:

	A	B	C	D
A	1	0.77	0.25	0.33
B	0.77	1	0.20	0.55
C	0.25	0.20	1	0.91
D	0.33	0.55	0.91	1

Using your choice of r and b and $f(\cdot)$, what is the probability of each pair of the four objects for being estimated to having similarity greater that $\tau = 0.85$? Report 6 numbers.

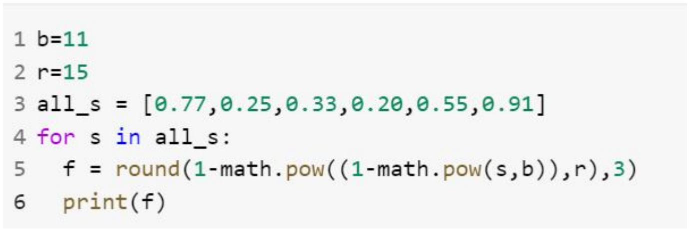
We'll use this formula to calculate the probability:

$$f(s) = 1 - (1 - s^b)^r$$

with s of the following values s = [0.77,0.25,0.33,0.20,0.55,0.91]

Probability of each pair for being estimated to having similarity greater than $\tau = 0.85$ is:

AB: 0.581
AC: 0.0
AD: 0.0
BC: 0.0
BD: 0.021
CD: 0.999



```
b=11
r=15
all_s = [0.77,0.25,0.33,0.20,0.55,0.91]
for s in all_s:
  f = round(1-math.pow((1-math.pow(s,b)),r),3)
  print(f)
```

2 Generating Random Directions (30 points)

A: (10 points) Describe how to generate a single random unit vector in $d = 10$ dimensions using only the operation $u \leftarrow \text{unif}(0, 1)$ which generates a uniform random variable between 0 and 1. *(This can be called multiple times.)*

This can be done by generate 2 uniform random numbers $u_1, u_2 \in [0, 1]$ in d=10 dimension, then we can generate two independent Gaussian random variables using the Box-Muller transform): $g = \sqrt{-2ln(u_1)}cos(2\pi u_2)$. Then we divide by its norm which results in a unit vector of length 1.

```
#n is dimension, in this case n=10
def rand_unit_vector(n):
  u1 = np.random.random(n)
  u2 = np.random.random(n)
  r_squared = -2*np.log(u1)
  r = np.sqrt(r_squared)
  theta = 2*np.pi*u2
  g = r*np.cos(theta)
  g = g/np.linalg.norm(g)
  return g
```

B: (20 points) Generate $t = 160$ unit vectors in $\mathbb{R}^d$ for $d = 100$. Plot of cdf of their pairwise dot products (yes, you need to calculate $\binom{t}{2}$ dot products).

We generate 160 unit vectors in 100 dimension, then calculate their pairwise dot product

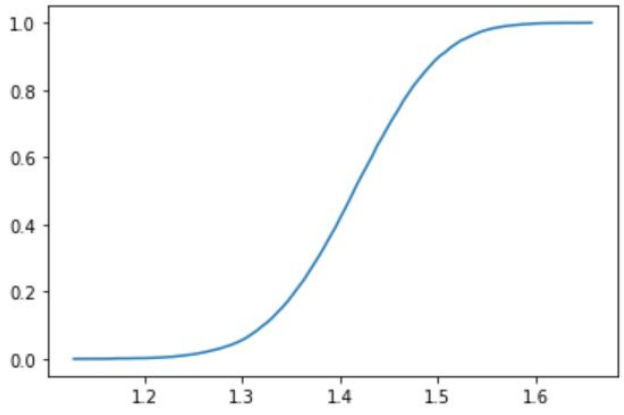
```python
#generate single random unit vector
#n is number of dimension
def rand_unit_vector(n):
  u1 = np.random.random(n)
  u2 = np.random.random(n)
  r_squared = -2*np.log(u1)
  r = np.sqrt(r_squared)
  theta = 2*np.pi*u2
  g = r*np.cos(theta)
  g = g/np.linalg.norm(g)
  return g
```

```python
def plot_cdf(prod_vecs):
    num_bins = 100
    counts, bin_edges = np.histogram (prod_vecs, bins=num_bins)
    cdf = np.cumsum (counts)
    plt.plot (bin_edges[1:], cdf/cdf[-1])
    return plt
```

```python
#generate random unit vectors and calculate its pairwise dot product
t = 160 # number of vecctors
n = 100 # number of dimension
vectors =[]
for i in range(t):
  vectors.append(rand_unit_vector(n))
y= scipy.spatial.distance.pdist(vectors, 'euclidean')

#plot CDF
plot_cdf(y)
```

```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.6/dist-packages/
```



3 Angular Hashed Approximation (35 points)

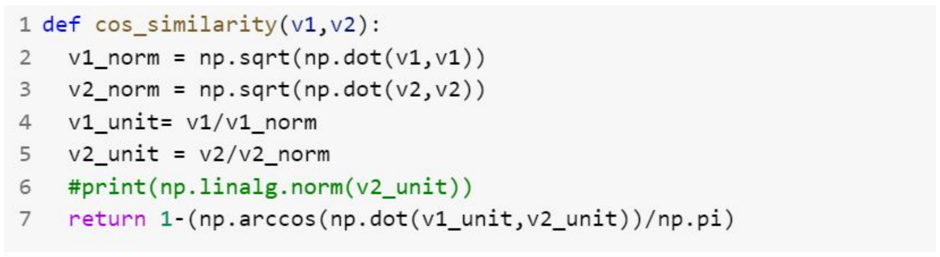
Consider the $n = 500$ data points in $\mathbb{R}^d$ for $d = 100$ in data set R, given at the top. We will use the angular similarity, between two vectors $a, b \in \mathbb{R}^d$:

$$\mathsf{s}_{\text{ang}}(a, b) = 1 - \frac{1}{\pi} \arccos(\langle \bar{a}, \bar{b} \rangle)$$

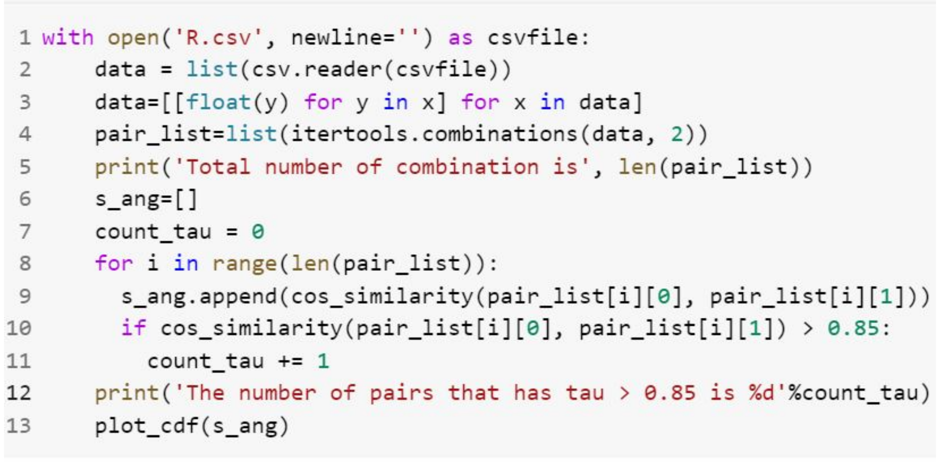
If a, b are not unit vectors (e.g., in $\mathbb{S}^{d-1}$), then we convert them to $\bar{a} = a/\|a\|_2$ and $\bar{b} = b/\|b\|_2$. The definition of $\mathsf{s}_{\text{ang}}(a,b)$ assumes that the input are unit vectors, and it takes a value between 0 and 1, with as usual 1 meaning most similar.

A: (15 points) Compute all pairs of dot products *(Yes, compute $\binom{n}{2}$ values)*, and plot a cdf of their angular similarities. Report the number with angular similarity more than $\tau = 0.85$.

The number of pairs that has $\tau > 0.85 = 39283$
We first create the function to calculate for angular or cosin similarity. First compute the norm of each vector (square of its own dot product), then to get unit vector we normalize by dividing by its length, then use this function to get the similarity: $\mathsf{s}_{\text{ang}}(a,b) = 1 - \frac{1}{\pi}\arccos(\langle \bar{a}, \bar{b}\rangle)$
Then we input data, make it into a list, convert the data into float for calculation and get $\binom{500}{2}$ pairwise combination, then we calculate the similarity for each pair

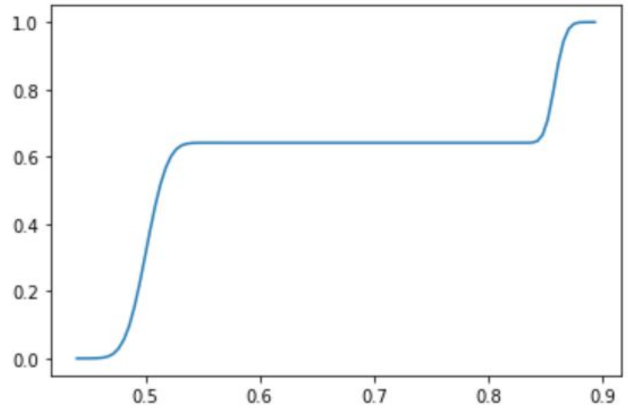


```python
def cos_similarity(v1,v2):
  v1_norm = np.sqrt(np.dot(v1,v1))
  v2_norm = np.sqrt(np.dot(v2,v2))
  v1_unit= v1/v1_norm
  v2_unit = v2/v2_norm
  #print(np.linalg.norm(v2_unit))
  return 1-(np.arccos(np.dot(v1_unit,v2_unit))/np.pi)
```



```python
with open('R.csv', newline='') as csvfile:
    data = list(csv.reader(csvfile))
    data=[[float(y) for y in x] for x in data]
    pair_list=list(itertools.combinations(data, 2))
    print('Total number of combination is', len(pair_list))
    s_ang=[]
    count_tau = 0
    for i in range(len(pair_list)):
      s_ang.append(cos_similarity(pair_list[i][0], pair_list[i][1]))
      if cos_similarity(pair_list[i][0], pair_list[i][1]) > 0.85:
        count_tau += 1
    print('The number of pairs that has tau > 0.85 is %d'%count_tau)
    plot_cdf(s_ang)
```

```
Total number of combination is 124750
The number of pairs that has tau > 0.85 is 39283
```



B: (20 points) Now compute the dot products and angular similarities among $\binom{t}{2}$ pairs of the t random unit vectors from Q2.B. Again plot the cdf, and report the number with angular similarity above $\tau = 0.85$.

The number of pairs that has $\tau > 0.85$ is 0

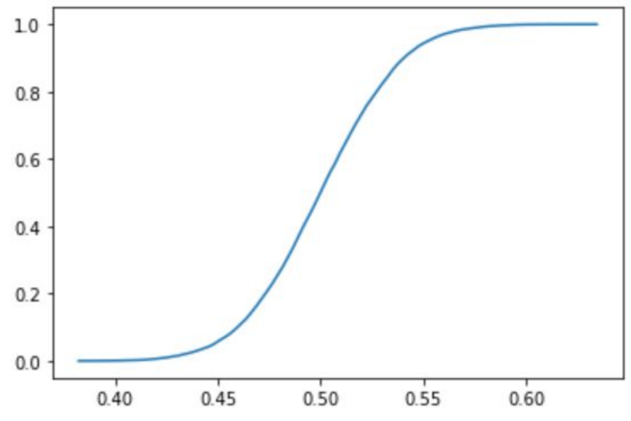
```python
#generate random unit vectors
t = 160 # number of vecctors
n = 100 # number of dimension
vectors =[]
for i in range(t):
  vectors.append(rand_unit_vector(n))

#create list of pairwise combination
pair_list=list(itertools.combinations(vectors, 2))
print('Total number of combination is', len(pair_list))

s_ang=[]
count_tau = 0
for i in range(len(pair_list)):
  #calculate the pairwise cosine similarity
  s_ang.append(cos_similarity(pair_list[i][0], pair_list[i][1]))
  #count if tau for the pairwise similarity is >0.85
  if cos_similarity(pair_list[i][0], pair_list[i][1]) > 0.85:
    count_tau += 1
print('The number of pairs that has tau > 0.85 is %d'%count_tau)

#plot CDF
plot_cdf(s_ang)
```

```
Total number of combination is 12720
The number of pairs that has tau > 0.85 is 0
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.6/dist-package
```



4 Bonus (3 points)

Implement the banding scheme with your choice of r, b, using your $t = 160$ random vectors, to estimate the pairs with similarity above $\tau = 0.85$ in the data set R. Report the fraction found above $\tau = 0.85$. Compare the runtime of this approach versus a brute force search.