Sapienza University of Rome

Big Data Computing First Homework

Andrea Fioraldi 1692419

November 4, 2019

Exercise 1

A signature matrix M has m rows (and so m functions), divided in b bands of r rows each. Let $sim(X,Y) = s$. We know that $Pr[h_\pi(X) = h_\pi(Y)] = s$ (i.e. a signature is in line with another signature in the row with probability s) in a row thanks to the property of minhash.
We want that the two documents X and Y hashes to the same bucket for at least one band if have similarity $s \geq \theta_1$ in which θ_1 is our similarity upper treshold. We want that the two documents X and Y hashes to the same bucket for no bands if have similarity $s < \theta_2$ in which θ_2 is our similarity lower treshold.
We now denote a generic treshold t.
The probability that all the rows in a band are equal is t^r. The inverse, the probability that at least one row in a band is different form each other is $1 - t^r$.
The probability that the signatures in all rows of each band are unequal is $(1 - t^r)^b$ and the inverse, the probability that at least one band has equal signatures in all rows is $1 - (1 - t^r)^b$.
Given similar documents (i.e $s \geq \theta_1$) the probability to not hash them to the same bucket for any band and so there is a *false negative* is $(1 - \theta_1^r)^b$. In the same way, given two not similar documents (i.e. $s < \theta_2$) the probability that they hashes to the same bucket for at least one band and so there is a *false positive* is $1 - (1 - \theta_2^r)^b$.

So, being the two θ functions of b and r, we can build a system of equations that describe how b and r to have at most the probability p_1 to have a false negative and the probability p_2 to have a false positive.

$$\begin{cases} (1-\theta_1^r)^b < p_1 \\ 1-(1-\theta_2^r)^b < p_2 \end{cases}$$

Assuming $\theta_1 = 0.7$, $\theta_2 = 0.5$ and $p_1 = p_2 = 0.01$ we can identify and approximated minimum m that meets the previous exposed requirements.

$$\begin{array}{lll} & \min b * r & s.t. \\ (1) & (1-0.7^r)^b < 0.01 & \\ (2) & 1-(1-0.5^r)^b < 0.01 & \end{array}$$

To solve this non linear optimization problem, I firstly tried to approximate it using Taylor expansion untile the third power of the *exp* and *log* functions and then solve the approximated problem with Microsfot Z3. I report the produced script.

```
from z3 import *

b = Int("b")
r = Int("r")

log = lambda x: ( (x-1) - (1/2)*(x-1)**2 + (1/3)*(x-1)**3 )

rbfn = lambda a: ( 1 + b*log( 1 - 1 + r*log(a) +1/2*r**2*(log(a))**2 +
    1/6*r**3*(log(a))**3 ) +1/2*b**2*(log( 1 - 1 + r*log(a) +1/2*r**2*(log
    (a))**2 + 1/6*r**3*(log(a))**3 ))**2 + 1/6*b**3*(log( 1 - 1 + r*log(a)
     +1/2*r**2*(log(a))**2 + 1/6*r**3*(log(a))**3 ))**3 )

o = Optimize()
o.add( 1 - rbfn(0.5) <= 0.01 )
o.add( rbfn(0.7) <= 0.01 )
o.add(r >= 0)
o.add(b >= 0)

h = o.minimize(b*r)
print(o.check())
```

Unfortunatly, Z3 was not able to solve this optimization problem (the result of the script is `unsat`) and I took the dirty way, a bruteforce with SageMath.

```
from sage . all import *

for i in xrange (1, 2**32) :
        f = divisors(i)
        for b in f:
                r = i // b
                if (1 - pow(1 - pow(0.5, r), b)) <= 0.01 and pow((1 - pow
                    (0.7, r)), b) <= 0.01:
                        print(r, b)
                        exit(0)
```

The found values are $r = 19$ and $b = 4038$.

1 Exercise 2

We know that $||X - Y||^2 = \sum_{j=1}^{d}(x_j - y_j)^2$. Te expected distance is $\mathbb{E}[||X - Y||^2] = \sum_{j=1}^{d} \mathbb{E}[(x_j - y_j)^2] = \sum_{j=1}^{i-1} \mathbb{E}[(x_j - y_j)^2] + \mathbb{E}[(x_i - y_i)^2] + \sum_{j=i+1}^{d} \mathbb{E}[(x_j - y_j)^2]$.
As seen at lesson, $\mathbb{E}[(x - y)^2] = \int 01 \int 01(x - y)dxdy = \frac{1}{6}$.
For the i-th dimension, we have that:

$$\mathbb{E}[(x_i - y_i)^2] = \begin{cases} \mathbb{E}[(a - a)^2] = 0 & \text{if X and Y are in the same cluster} \\ \mathbb{E}[(a - b)^2] = (a - b)^2 & \text{if X and Y are in different clusters} \end{cases}$$

The expected distance when X and Y are in the same cluster is:

$$\sum_{j=1}^{i-1} \frac{1}{6} + 0 + \sum_{j=i+1}^{d} \frac{1}{6} = \frac{d-1}{6}$$

The expected distance when X and Y are in different clusters:

$$\sum_{j=1}^{i-1} \frac{1}{6} + (a - b)^2 + \sum_{j=i+1}^{d} \frac{1}{6} = \frac{d-1}{6} + (a - b)^2$$

A clustering algorithm based on the distance can fail when the expected distance $\frac{d-1}{6}$ cannot be easily distinguished from $\frac{d-1}{6} + (a - b)^2$.
This happens when $d \to \infty$ or $(a - b) \to 0$ or both. Intuitively, in addition to the curse of dimensionality (with an increasing d the probability that two points have a distance less then the averga distance decrease), less is the contribute of $(a - b)^2$, less effective is the clustering.
When i is a-priori known and $|a - b|$ is not too small, a good strategy to cluster the points is to project all the points to such dimension. All the points of a cluster will collapse to the same coordinate and so, after the projection, the clustering is trivial. We can of course apply a clustering algorithm like k-means to the result of the projection, but it is overpowered and a simple one-pass is enough.

This strategy is, however, not feasible in the general case. Without knowing the i-th dimension this cannot be applied and the individuation of such dimension is not a trivial task. We can use a method like PCA to identify it, but remains an hard task when d is very huge.