Information Retrieval and Analysis
December 2020

Lab Session 9: Locality Sensitive Hashing

PROJECT REPORT

Elías Abad Rocamora
Victor Novelle Moriano
Barcelona, UPC - FIB & FME & ETSETB

Understanding LSH

1.1) Is the execution time linear with respect to K*M (O(K*M))?

In order to check this we have designed the following experiment:

1. First of all, we chose a set of values for K and M:

   $$K \in [1, 20] \text{ and } M \in [1, 20]$$

   Both sets of values are the same in order to check that none of both parameters has more influence in the execution time than the other

2. Then we ran the LSH execution for each combination of values of K and M and stored the result in a list of results for that value of K*M
3. After all the executions terminated, we averaged all the execution times with the same value of K*M

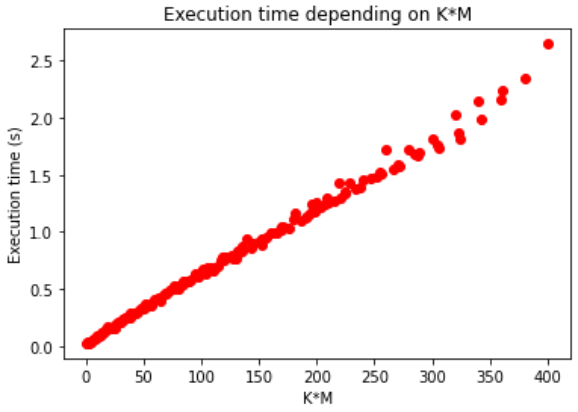

Figure 1: Experimental execution times in terms of K*M

As we can observe in **Figure 1**, the execution time is clearly linearly dependent with K*M.

1.2) What happens to the size of the candidate set when increasing k?

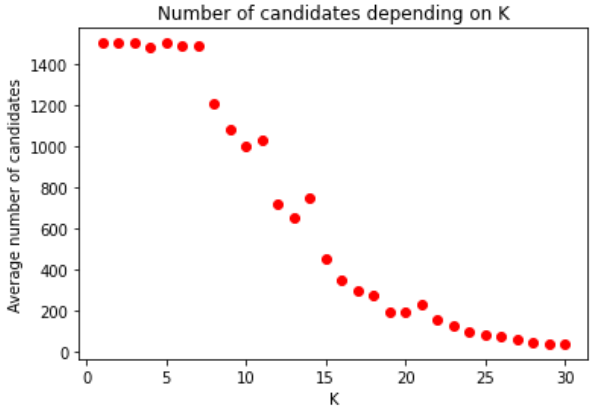

Figure 2: Evolution of the number of candidates with respect to K

The next step in the analysis of the LSH algorithm consisted of evaluating how variations in the *K value* affect the number of candidates for a given image. To perform this operation, the average number of candidates for the first 10 images was computed, performing *M* = 10 repetitions and setting $K \in [1, 30]$.

As can be seen in **Figure 2**, when k has a small value, the number of candidates is large, selecting almost all of the images as candidates. However, as the K value increases, the number of candidates decreases following a negative exponential function. This is due to the fact that all K-hashing functions have to provide the same value in order to classify two images on the same bin. The explanation of this behaviour is that with the stacking of the hash functions applied to our bit strings the probability of collision is reduced to S^K being S = % of bits that match between the two unary image representations.

1.3) What happens to the size of the candidate set when increasing M?

To follow up with our analysis of LSH, we will analyze how the value of M, that is the number of hashing repetitions, affects the number of matches obtained for an input image.

In order to do this analysis, we performed a very similar experiment to the one in **1.2**. The value of K was fixed to K = 10 and then for all the values of $M \in [1, 30]$, the first 10 images of the test dataset were hashed and the average number of candidates was computed.

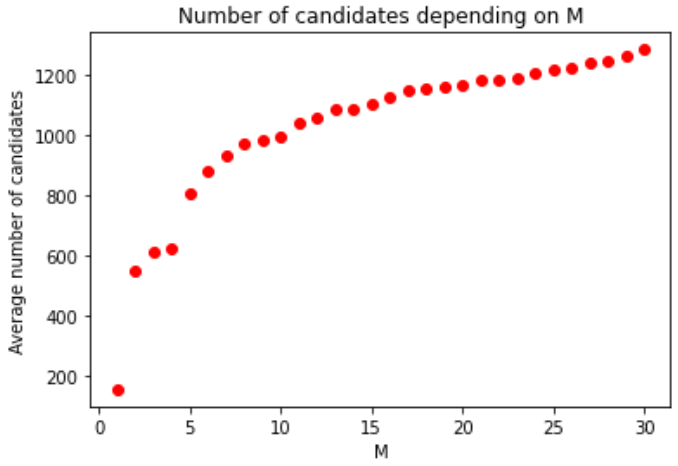

Figure 3: Evolution of the number of candidates with respect to M

As we can see in **Figure 3**, the number of candidates obtained increases logarithmically with M. This makes sense as two images have a match if they are in the same bin in at least one repetition of the hashing among the M repetitions. So, by increasing M, it is more likely that two images fall in the same bin at least once and therefore create a match.

1.4) Can you provide a function of K and M on the average number of candidates?

As a first approach to model the number of candidates, we will define the probability of two images to have a match in terms of K,M and their similarity:

$$P(match(x,y|K,M)) = P(\exists\ i \ni [1,M] \mid h_i(x) = h_i(y)) = 1 - P(h_i(x) \neq h_i(y)\ \forall i \ni [1,M]) =$$
$$= (independence) = 1 - \prod_{i=1}^{M} P(h_i(x) \neq h_i(y)) = 1 - \prod_{i=1}^{M}(1 - P(h_i(x) = h_i(y))) =$$
$$= (P(h_i(x) = h_i(y)) = \prod_{j=1}^{K} s(x,y)) = 1 - \prod_{i=1}^{M}(1 - \prod_{j=1}^{K} s(x,y)) = 1 - (1 - s(x,y)^K)^M$$

If we define the match R.V. as:

$$match(x,y|K,M) \;=\; \{1 \; if \; \exists \; i \; \ni \; [1,M] \mid h_i(x) = h_i(y), \; 0 \; otherwise\} \;\sim\; B(1-(1-s(x,y)^K)^M)$$

If we suppose that the similarity with x among all images is constant and with value $sim(x)$, then the R.V. representing the number of matches of a certain image could be modelled as:

$$\#Matches(x|K,M) \;=\; E(\sum_{y \in Images} match(x,y)) \;\sim Bin(|Images|, 1-(1-sim(x)^K)^M) \;=$$
$$= \; |Images| \cdot 1-(1-sim(x)^K)^M$$

So, having defined our function for the number of matches, we can try some values of sim(x) and see how it fits our data. In order to do this and to accurately see the fit in a plot, we have repeated experiments **1.2** and **1.3** with our function and plotted the results. As in these experiments, the number of matches is averaged for different input values, we have to provide a value for the average similarity among every pair of images, and after some tries, we obtained these results with $mean(sim(x)) \;=\; 0.8$:

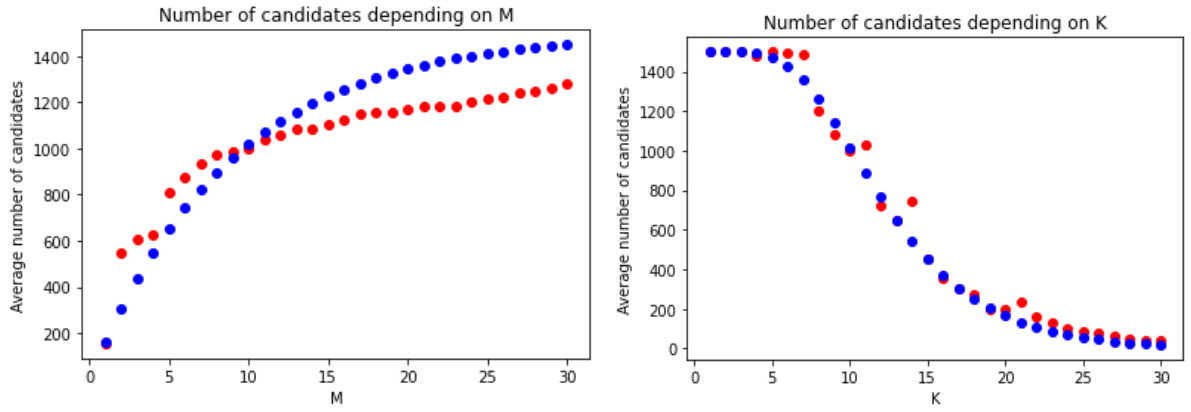

Figure 4: Experimental vs. theoretical average number of matches.

As we can observe, the function we have provided, although not being realistic to suppose that the similarity between images is constant along all the dataset, gives a very accurate approximation of the actual number of matches.

2. LSH vs. Brute Force

After analyzing the performance of the *LSH* depending on the values of its hyperparameters, we developed a brute force comparison algorithm to evaluate the difference of performance. To do so, we performed several executions of the *LSH* algorithm testing all possible K, M combinations using [1,5,15,40] as values.

		LSH			Brute Force	
K	**M**	**Hash creation**	**Avg. Cons. time**	**Avg. distance**	**Avg. Cons. time**	**Avg. distance**
1	1	0.0248	0.0834	52.7	0.14033	52.0
	5	0.0639	0.1562	52.0	0.1508	52.0
	15	0.2159	0.1479	52.0	0.1494	52.0
	40	0.5448	0.1565	52.0	0.1612	52.0
5	1	0.0429	0.0423	66.4	0.1495	52.0
	5	0.2318	0.1381	52.0	0.1522	52.0
	15	0.5446	0.1737	52.0	0.1729	52.0
	40	1.6919	0.1663	52.0	0.1675	52.0
15	1	0.1046	0.0121	83.6	0.1682	52.0
	5	0.5653	0.0404	52.0	0.1498	52.0
	15	1.57323	0.0558	52.0	0.1500	52.0
	40	3.8267	0.0975	52.0	0.1573	52.0
40	1	0.2473	0.0012	72.0	0.1748	52.0
	5	1.4319	0.0032	63.2	0.1631	52.0
	15	3.7920	0.0049	52.4	0.1316	52.0
	40	9.9785	0.0131	52.0	0.1582	52.0
Average		1.5550	0.0808	56.9	0.1561	52.0

Figure 5: Table of results of LSH and Brute Force when retrieving the closest image

We can see that for all cases, the average consulting time of the *LSH* algorithm is lower than the brute force algorithm. This was an expected result, as *LSH* only consults a subset of images (the ones that match in one of the *M* bins) whereas in brute force we always compare to the 1500 training images. Some people could argue that we should take into account the *LSH* hash creation on its time, however, it should be considered that this computation is performed offline once. Afterwards, all the comparison would use the previously created hash, being notably faster than the brute force (This difference will be more notorious as the number of images to compare to increases).

In relation to the *LSH* distance, it can be seen that, in most cases, the average Hamming distance to the best candidate found is pretty similar to the obtained by the *Brute Force Algorithm,* which is the smallest achievable.

Also, it can be seen that the biggest distances are associated with *M=1.* The decrease of the matching probability for using high *K* values is not compensated by performing several times the hashing process. Due to this, the best possible image may fall in a different bin if some of the *K* concatenated hash functions differ, not being able to find the best candidate.