CAIM - Lab8
Locality Sensitive Hashing (LSH)

Víctor Vallejo Rives
Pau Núñez Amorós

14/01/2020

1 Task 1

1.1 Paràmetre `k`

El paràmetre `k` és el nombre de funcions de hash apilades per formar $x_i = (h_1(x), .., h_k(x))$. Hem de veure com varia:

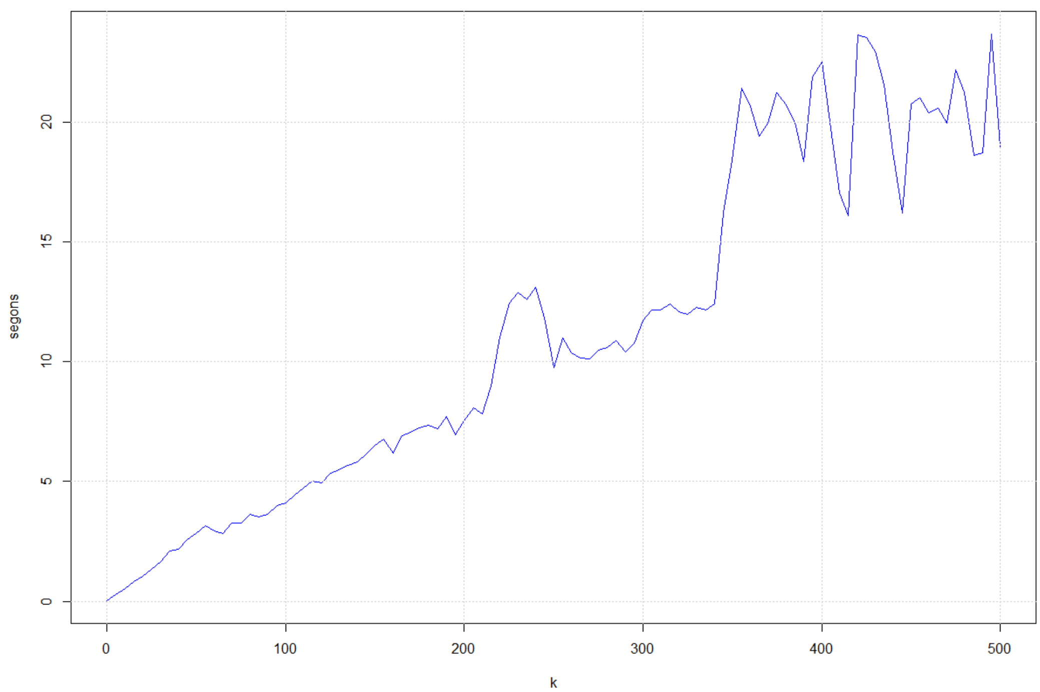
- el temps d'execució de l'algorisme `lsh`
- el nombre de candidats possibles

a mesura que incrementa el valor del paràmetre `k`. Per fer-ho hem usat un petit *script*:

```
$ for value in {0..500..5}; do python lsh.py -k $value >> task1_1.txt; done;
```

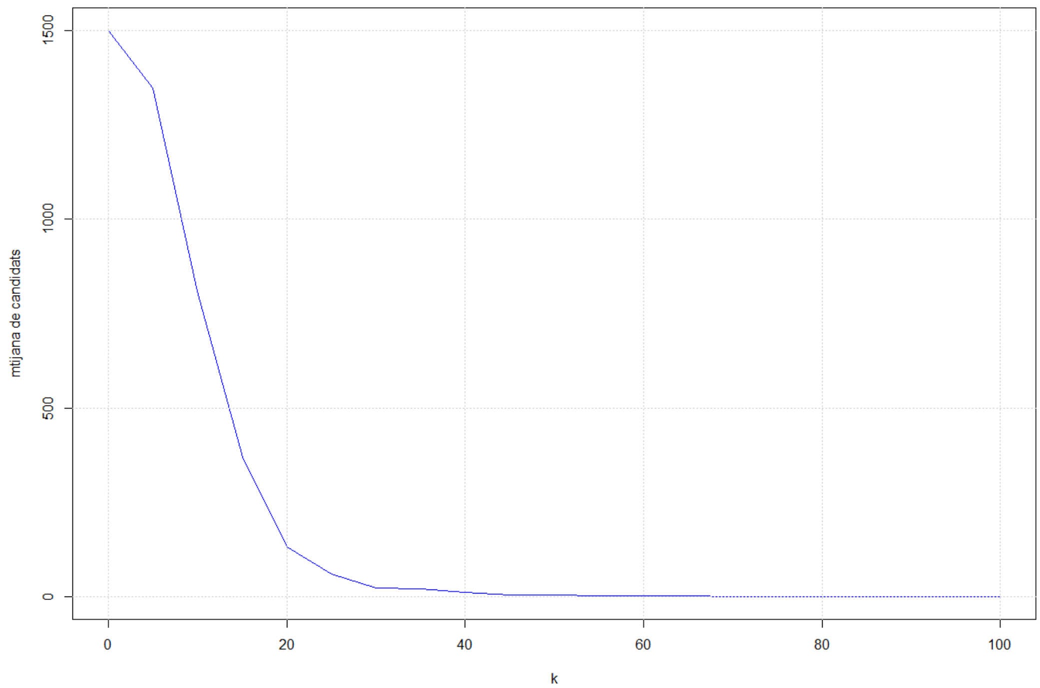
el qual ens genera parells de valors `(k, temps || mida)` (amb `steps` de 5 per la `k`) que després molt convenientment podem graficar. Com estem experimentant amb `k`, deixem `m` fixa al seu valor per defecte (`m = 5`).

La següent gràfica representa els resultats obtinguts:



Com podem observar, a mesura que augmentem el valor de `k` també ho fa el temps d'execució. Tot i la variabilitat, segueixen una relació lineal.

A l'hora de mesurar el nombre de candidats abtinguts en funció de `k`, hem decidit agafar un rang de {0 a 100} augmentant el seu valor en 5 cada cop.

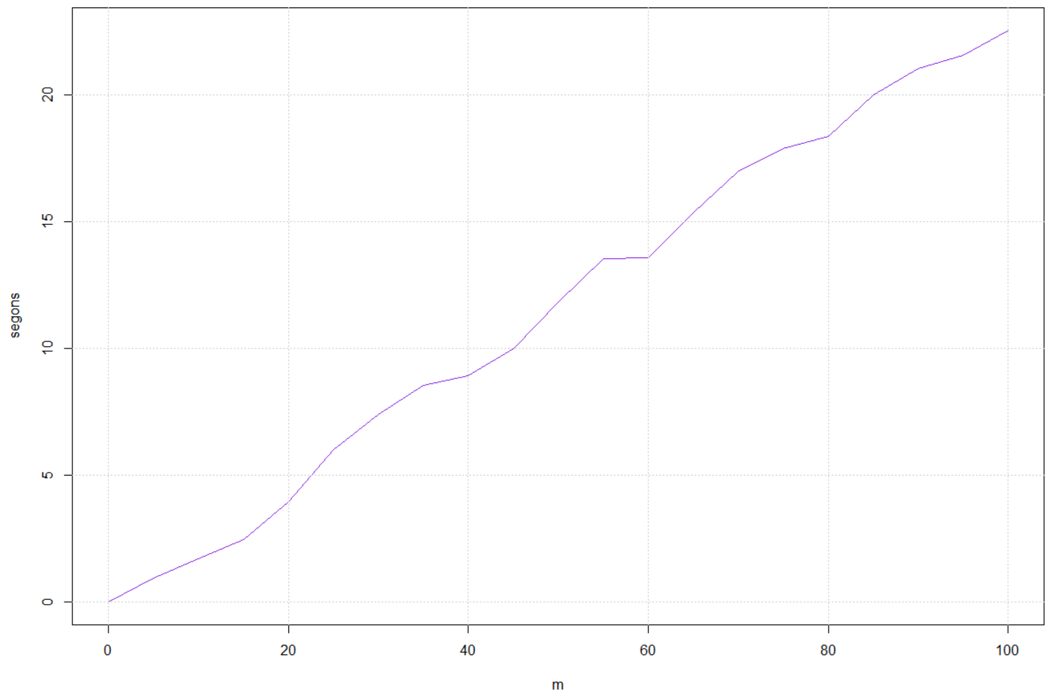


En aquest cas veiem que el nombre de candidats disminueix a mesura que augmentem el valor de `k`. I a diferencia que la gràfica anterior, no segueixen una relació de linealitat. El nombre de candidats baixa notablement més entre la k={0..20} que als seus altres valors, formant així una *Power Law*. També cal destacar que a partir de la {k=65} obtenim una mitjana de candidats molt propera a 0.

1.2 Paràmetre m

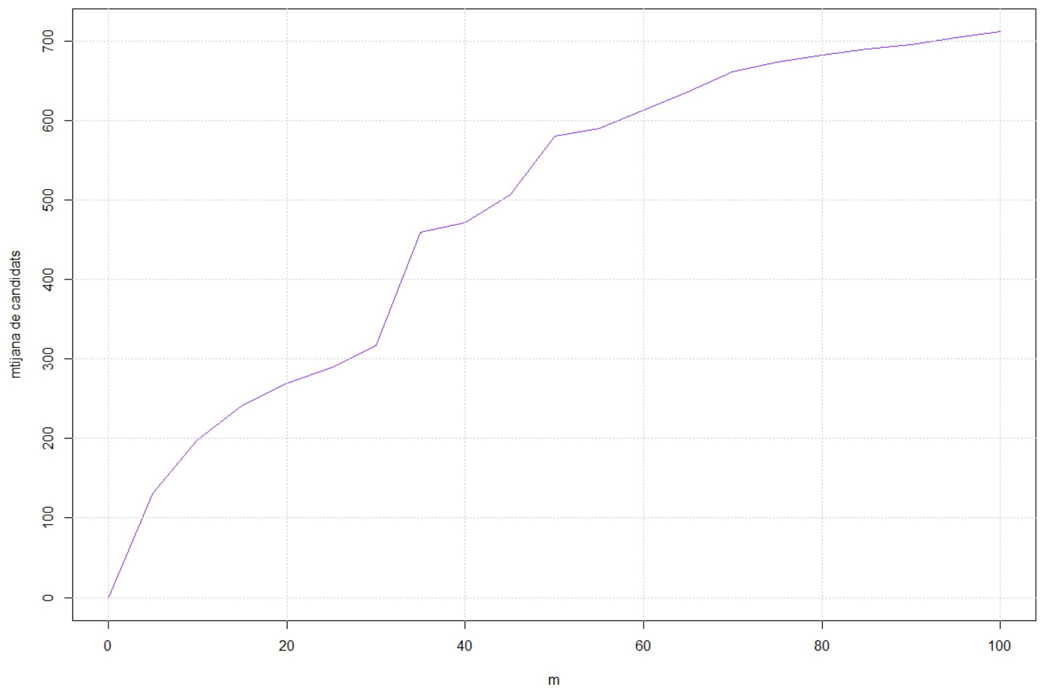
En aquest apartat hem de mesurar el mateix que a l'anterior, però aquest cop per a la variable `m`. Aquest paràmetre és el que determina el nombre de vegades que s'apliquen les `k` funcions de hash. Per tal de dur a terme aquests càlculs hem deixat el valor per defecte de (`k = 20`).

La següent gràfica representa els resultats obtinguts:



Com podem observar, de nou el temps d'execució augmenta a mesura que ho fa la variable que estem modificant. En aquest cas es veu de forma encara més clara el fet que les dues variables estàn relacionades linealment.

Seguidament procedim a fer la segona comparació amb el nombre de candidats. De nou hem agafat valors de `m` de {0 a 100} augmentant-la en 5 cada vegada.



Els resultats obtinguts en aquest cas son totalment contraris als anteriors fets amb `k`, ja que ara el nombre de candidats augmenta a la vegada que ho fa `m`. És a dir que quantes més iteracions s'executen, més candidats es troben, com ens diu la intuició. La gràfica de nou mostra una *Power Law*, però en aquest cas és creixent.

2 Task 2

Per tal de fer aquest apartat, ens demanen que implementem una funció que calculi la *distància* entre dues imatges, una que faci una busqueda de força bruta i una altra que donada una imatge busqui el seu veí més proper. L'implementació d'aquestes funcions no ha suposat un problema gaire gran com a tal, però a l'hora de mesurar quant de temps trigaven en executar-se sí.

Ens demanen que comparem el *LSH* amb l'algorisme de *Força Bruta*, tant el temps que triguen en executar-se com el veí resultant obtingut. Hem utilitzat la llibreria `time` per tal de fer les mesures de temps. A primer cop d'ull, ens vam trobar que la suma de temps d'execució dels dos algoritmes diferia molt del temps total que requeria l'execució del `main`. En alguns casos ambdós algorismes sumaven un temps menor a 2 segons, però el `main` necessitava més de 20 per executar-se. En un principi ens va semblar que teniem algun error a l'hora de mesurar els temps dels algorismes, ja que la diferència era massa gran. Després de fer algunes proves i mesures vam veure que els càlculs estaven bé, i que tot aquest temps extra que necessitava el `main` provenia de les inicialitzacions.

Seguidament podem veure una taula amb el temps d'execució de cada algorisme, la diferència (distància) entre el veí triat pels dos algoritmes i per últim els casos en els que *LSH* no ha sigut capaç de trobar cap veí. Hem mesurat aquests termes per diverses combinacions de `k` i `m`.

	k=1 m=1	k=5 m=1	k=10 m=1	k=15 m=1	k=1 m=5	k=1 m=10	k=1 m=15	k=5 m=5	k=10 m=10	k=15 m=15	k=50 m=10	k=10 m=50	k=50 m=50
Brute Force	0.73	0.75	0.77	0.74	0.77	0.75	0.75	0.77	0.80	0.77	0.79	0.76	0.77
LSH	0.42	0.22	0.08	0.06	0.81	0.83	0.75	0.71	0.54	0.25	0.00	0.69	0.01
Difference	7	144	269	316	0	0	0	0	0	0	108	0	0
Not-Founds	-	-	-	-	-	-	-	-	-	-	1	-	-