

A=(24.21; 0)

B=(0;225.24)

C=(-14.33; 0)

P(Angulo<=90°) = 0,75 = P1

D1 = 90°

D2 = 135°

P(Angulo>=135°) = 0,4375

1. Para conseguir b y r debo realizar un grid search con los valores de D1, D2, P1, y P2; siendo la familia que cumple con esto:

H(90°, 135°, 0.75, 0.4375)

Y r y b pueden ser conseguidos buscando que P1 sea al menos 0,75 y P2 a lo sumo 0,4375 con la siguiente fórmula:

P1 = 1-(1-(1-d1/180°)^r)^b

P2 = 1-(1-(1-d2/180°)^r)^b

Y aplicándole grid search para ver que p2 sea menor o igual a 0,4375, y que p1 sea mayor o igual a 0,75 será:

R=1

B=2

1. La distancia coseno sale tomando hiperplanos aleatorios para la cantidad r\*b de hashes que necesito; de esta forma, cada hiperplano define un minhash distinto. Si el resultado del producto escalar entre el vector es menor o igual a 0 entonces el minhash será -1, mientras que si el producto escalar otorga un numero mayor a 0 entonces el minhash es 1.

Para que A colisione con B y B con C, pero no A con C, entonces necesito que B y C coincidan en una familia de Hashing, y B y C en otra.

Un ejemplo fácil puede ser que al haber un único minhash por familia, entonces ese minhash se multiplica por un escalar y se le hace % cant.buckets.

Entonces, queda:

A.H1 = 24.21\*H1x + 0

B.H1 = 0+225.24\*H1y

C.H1 = -14.33\*H1x + 0

Si por ejemplo H1x y H1y son positivas, entonces el minhash de A y B se establece en 1 por ser un resultado mayor a 0, y el de C en -1 por ser un resultado menor a 0.

Entonces, a la hora de aplicar la familia de hashing, el resultado para A y B será:

Si uso constante = constante2 = 2, y cant.buckets = 3

(Constante \* 1 + Constante2) % cant.buckets -> Irán a parar al mismo bucket. (Al (2+2)%3 = 1)

Y para C:

(Constante \* -1 + Constante2) % cant.buckets -> Ira a parar al (-2+2)%3 = 0

Por lo tanto en la primer familia habrá colisión entre A y B. Ahora para el otro minhash debo repetir los pasos con la misma cantidad de buckets, buscando ahora que A sea el que hashee distinto que B y C, e incluso distinto que C en la anterior familia. Por lo tanto, A tiene que parar al 2 o al 1, B puede ir a parar al 0 o a 2 si es que C va a parar al 2; y C puede ir a parar a 0 o 2.

Entonces:

Defino constante = 2, constante2 = 1, y cant buckets era 3:

A.H2 = 24.21\*H2x + 0

B.H2 = 0+225.24\*H2y

C.H2 = -14.33\*H2x + 0

Con H2x negativo, y H2y positivo obtengo que B y C hashean a la misma posición por ser ambos resultados del minhash 1, y A hashea con resultado -1. Entonces la familia queda:

Para A:

(2\*-1+1)%3 = 2

Para B y C:

(2\*1+1)%3 =0

Entonces finalmente la tabla de hashing de LSH queda:

|  |
| --- |
| {C,B} |
| {A,B} |
| {A} |

Donde claramente se ve que B colisiona con A y C, pero A nunca colisiona con C

1. Q = (-5, -3).

Para encontrar los elementos similares al Query debo pasarlo por el LSH, y aplicarle las mismas funciones que a A,B, y C. De esta manera, las posiciones donde se almacene en la tabla serán porque coincidió con los r minhashes en alguna de las b familias posibles. Entonces:

MH1(Q) = (-5, -3) \* (1,1) = -8 => MH1(Q) = -1 por ser < 0

MH2(Q) = (-5, -3) \* (-1,1) = 2 => MH1(Q) = 1 por ser > 0

Entonces ahora aplicando las familias, la primer familia da:

H1(MH1(Q)) = (-1\*2+2)%3 = 0

Por lo tanto obtenemos la primer posición de la tabla, que serán los elementos {C,B}

Y para la segunda familia da:

H2(MH2(Q)) = (2\*1+1)%3 = 0

Lo que nos da los mismos elementos que antes.

Por lo tanto, los elementos con los cuales el elemento Query es similar son: {C,B}