Laboratorio 01: La maldición de la dimensionalidad

Pablo Luis Carazas Barrios

30 Agosto 2023

Abstract

En el estudio de los datos de varias dimensiones, también conocidos como datos multidimensionales, ha aumentado en los últimos años. Esto nos ha traído nuevos desafíos, siendo uno de ellos la manera en que los organizamos. En el siguiente informe, evidenciaremos el problema de cómo aumenta el espacio entre valores dimensionales a medida que crecen las dimensiones a tomar en cuenta.

1 Introduction

El trabajo consiste en desarrollar un programa en C++ que genere puntos en *k-dimensiones* y posteriormente compare las distancias entre estos puntos. Las pruebas se llevaron a cabo utilizando 100 puntos en diversas dimensiones, específicamente 2, 10, 50, 100, 1000 y 2000.

2 Programa

El programa "Distancia entre puntos de *k-dimensiones*" abarca una estructura y una función, las cuales se detallarán a continuación.

2.1 Estructura punto

La estructura consta de 2 atributos y 1 método. El primer atributo es el *vector≤float≥ v* , el cual es el espacio donde almacenaremos los valores. Su tamaño variará dependiendo del segundo atributo, *int dimensiones*, que representa la cantidad de dimensiones que tendrá nuestro punto. Todo esto se crea y procesa mediante nuestro constructor. El constructor toma como parámetro *int _dimensiones* y luego se guarda en *dimensiones*, tambien se cambia el tamaño del vector a la cantidad de dimansiones.
El Método *void rellenar* lo que hace es mediante un *for* recorrer todos los espacios de *v* y rellenarlo de valores aleatorios entre el 0 y el 1, con 4 cifras decimal.

```
struct punto {
        punto(int _dimensiones) {
                v.resize(_dimensiones);
                dimensiones = _dimensiones;
                rellenar();
        }
        vector<float> v;
        int dimensiones = 0;
        void rellenar() {
                int tam = v.size();
                for (int i = 0; i < tam; i++) {
                        v[i] = rand() % 10001;
                        v[i] = v[i] / 10000;
                }
        }
};
```

2.2 Funcion distancia

La funcion es algo simple, es de tipo *float* ya que devuelve la distancia, toma como parametros variables de tipo *punto*, luego guarda la dimension de los puntos, hay una variable *float sum* en donde se guardara una suma cumulada, el bucle for iterara dependiendo de la cantidad de dimensiones y utiliza *distancia euclidiana.*

$$\sqrt{(x_1 - x_2) + (y_1 - y_2)}$$

, pero por partes, primero se guarda la suma de potencias de cada dimension, y recien al retornar el valor de la funcion se realiza la raiz cuadrada

```
float distancia(punto* v, punto* v2) {
        int t = v->dimensiones;
        float sum = 0;
        for (int i = 0; i < t; i++) {
                sum = sum + pow(v2->v[i] - v->v[i], 2);
        }
        return sqrt(sum);
}
```

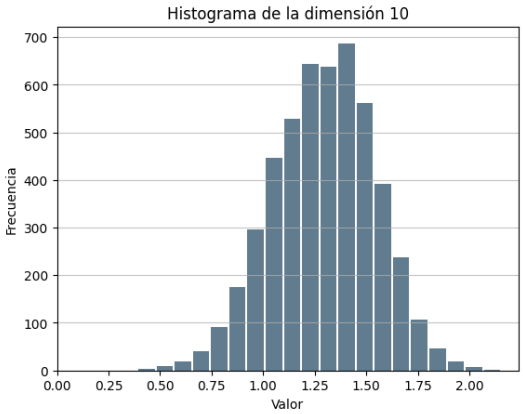
2.3 Main

En el main se declara *vector≤puntos*≥ puntos(100)* en el cual se almacenaran los 100 puntos requeridos, luego mediante el *for* se rellena los puntos en el vector, se hace de esta manera y no al crear el vector ya que al hacer eso solo se crea una copia del mismo punto 100 veces. En el siguiente bucle se realiza otro bucle y dentro se compara todos los puntos contra todos los puntos utilizando la funcion explicada anteriormente.

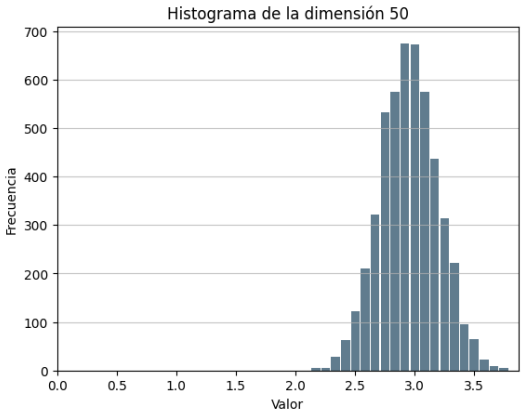
```
int main()
{
        vector<punto*> puntos(100);
        for (int i = 0; i < puntos.size(); i++) {
                puntos[i] = new punto(1000);
        cout << "distancias" << endl;
        for (int i = 0; i < puntos.size(); i++) {
                for (int j = i + 1; j < puntos.size(); j++) {
                        cout << distancia(puntos[i], puntos[j]) << endl;
                }
        }

}
```

3 Resultados

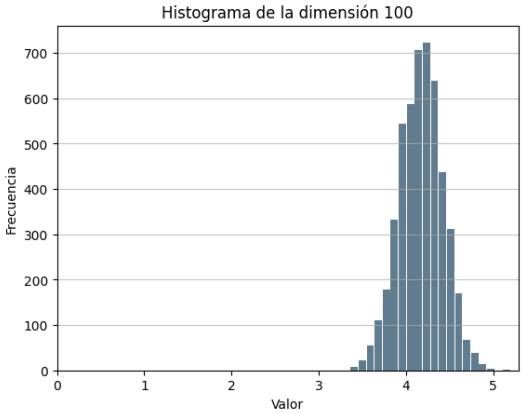
Después de realizar las pruebas con las dimensiones solicitadas, se obtuvieron los siguientes resultados, que ahora serán representados en forma de histogramas.



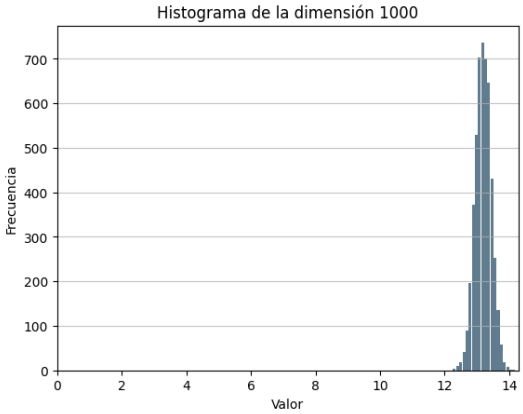
(a) 10 dimensiones



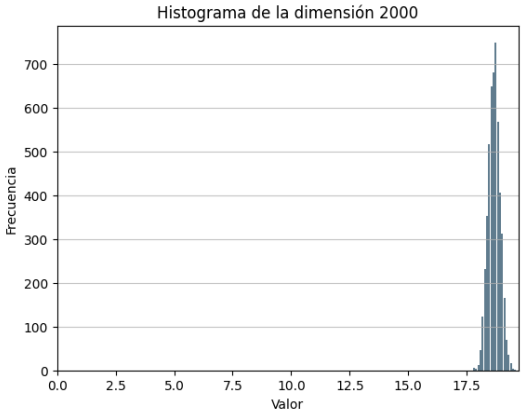
(b) 50 dimensiones



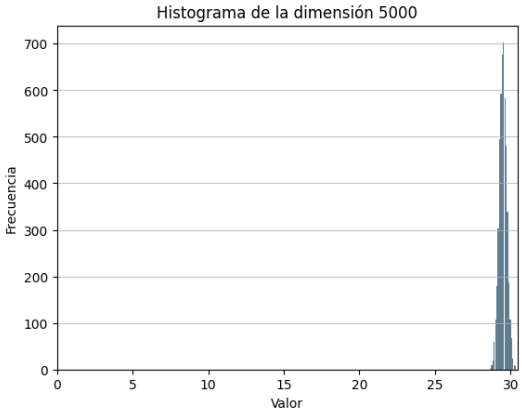
(c) 100 dimensiones



(d) 1000 dimensiones



(e) 2000 dimensiones



(f) 5000 dimensiones

Después de analizar los histogramas, se hace evidente un patrón: a medida que aumenta la cantidad de dimensiones de cada punto, las distancias entre ellos también aumentan de manera exponencial, esto tiene una explicación que se dará a continuación.

3.1 Análisis

Las imagenes evidencian que cada vez la distancia minima entre los puntos se aleja más del cero, esto tiene una explicación.
Primero tomemos en cuenta el caso de 2 dimensiones

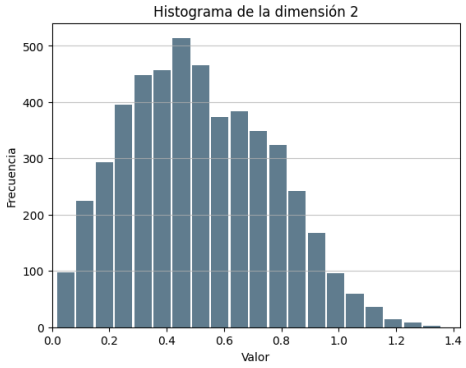


Figure 2: Histograma de 2 dimensiones

En este caso vemos como las distancias varian desde el 0 hasta 1.4, la explicación que le podemos atribuir es la misma formula de la distancia euclidania,

$$\sqrt{(x_1 - x_2) + (y_1 - y_2)}$$

, sabemos que el valor minimo que se puede tomar es de 0 y el valor máxmio que puede tomar un punto es 1, pero ¿de que nos sirve esta informacion? bueno, esto se puede explicar de manera grafica.

3.1.1 Dimensiones

Vivimos en un universo de 3 dimensiones por lo que se nos hace facil visualizar menores dimensiones, en un espacio de 1 dimension

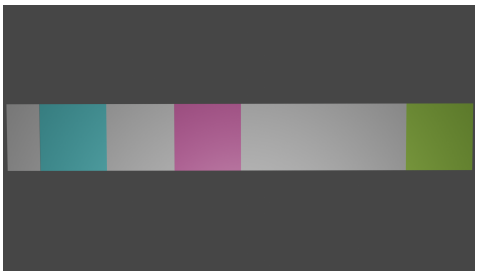


Figure 3: Visualizacion de 1 dimension

se puede aplicar la distancia euclidiana pero solo con una diferencia $\sqrt{(x_1 - x_2)}$, los puntos (en este caso cuadrados) solo se pueden desplazar en una dirección, para un espacio de 2 dimensiones

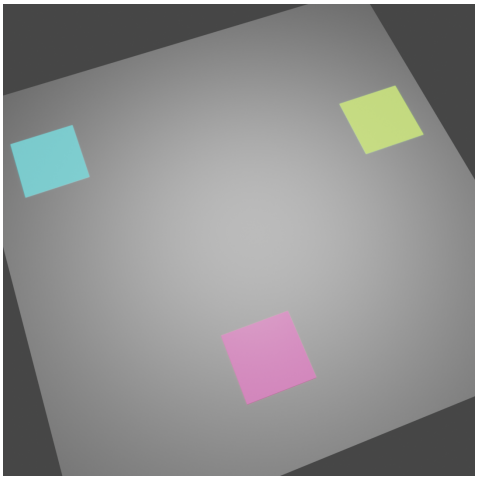


Figure 4: Visualizacion de 2 dimension

se sigue aplicando la distancia euclidiana pero tomando en cuenta una posibilidad más de desplazamiento $\sqrt{(x_1 - x_2) + (y_1 - y_2)}$, y así sucesivamente para siguientes dimensiones.

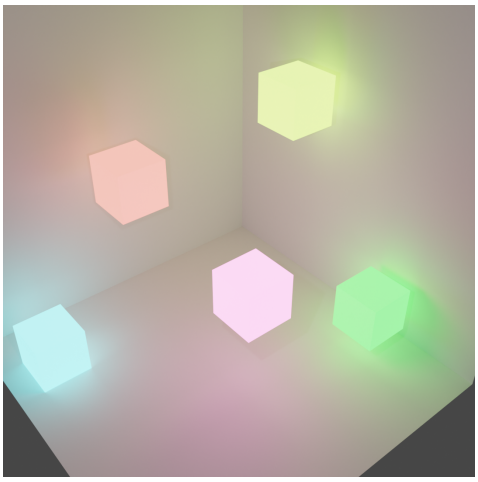


Figure 5: Visualizacion de 3 dimension

Entonces, retomando el caso de las 2 dimensiones explicado en la imagen 2, la mayor distancia que se alcanza es de 1.4, esto se debe a que tomando el caso que fuese una sola dimensión, la distancia maxima que pueden tomar los puntos es la diferencia de el mayor valor que puede tomar un punto y el menos valor $1 - 0 = 1$, ahora para 2 dimensiones usamos la distancia euclidiana con la mayor distancia posible que pueden tomar los puntos $\sqrt{(1) + (1)}$ siendo esto 1.4142 el cual es el tope que muestra el histograma, y esa curva que genera el grafico es debido a la distribución gaussiana de los valores de los punto.
Tomando en cuenta que tenemos 100 puntos de *k dimensiones*, y cada dimension puede tomar valores entre el 0 y el 1 con 4 cifras decimal, esto se puede denotar de la siguiente manera

$$P(\text{Todas las dimensiones de un punto sean 1}) = 0.0001^k$$

y como son 100 puntos

$$P(\text{Todas las k dimensiones son 1 en los 100 puntos}) = (0.0001^k)^100 = 0.0001^{k*100}$$

con esta formula tambien se nos da la expicacion el porque se separa del 0, ya que la probabilidad de que todos los puntos tengan la misma posición es muy baja ya que se aplicaria la misma formula de probabilidad anterior

4 Conclusiones

Con este ejercicio pudimos concluir varias cosas, a menor sea la cantidad de dimensiones es mayor la probabilidad de proximidad de los puntos, $P(\text{Todas las k dimensiones son x posicion en los 100 puntos}) = (0.0001^k)^k = 0.0001^{k*100}$, la formula hayada anteriormente aplica para cualquier punto que se quiera analizar. Al saber que al aumentar la cantidad de dimensiones la probabilidad que colisionen 2 puntos es muy bajo, lo cual es un gran problema a la hora de organizar datos.
Actualmente todo podemos decir que es un dato multidimensiones, cada cosa, ser u objeto es un elemento multidimensional, y es un gran problema su organización, el hallar una coincidencia, colision de puntos, es muy baja tomando en cuenta las dimensiones por analizar y las posibilidades que posee cada dimension. Con todo lo visto podemos decir que el estudio de datos multidimensionales es una realidad dificil, con grandes resultados cuando se encuentren nuevas maneras eficientes de organizar datos con millones de dimensiones, el estudio de estos datos traerá beneficios computacionales y buenos resultados, pero su organización seguirá siendo una "maldicion".

5 Enlaces

Github con el codigo fuente e informe