Análisis Comparativo del Rendimiento Computacional en la Búsqueda de Vecinos Más Cercanos Utilizando KD-Tree y Fuerza Bruta en Espacios 3D

Roque Sosa, Owen Haziel
Facultad de Ciencias de la Computación
Universidad Nacional de San Agustín de Arequipa
Arequipa, Perú
oroque@unsa.edu.pe

***Abstract*—En este informe, evaluamos el rendimiento computacional de las operaciones fundamentales de un KD-tree en comparación con la fuerza bruta en la búsqueda de vecinos más cercanos en espacios 3D. Se analiza el costo de inserción en el KD-tree y se compara con el enfoque de fuerza bruta. Además, se investiga el impacto del parámetro *k* en los resultados obtenidos. Utilizando conjuntos de datos de prueba (`testX.zip`), medimos y comparamos los tiempos de ejecución, proporcionando una perspectiva sobre la eficiencia de cada enfoque en diferentes contextos.**

***Index Terms*—KD-Tree, Búsqueda de k-Vecinos Más Cercanos, Espacios 3D, Análisis de Rendimiento Computacional, Estructuras de Datos Geométricas.**

I. Introducción

En esta investigación, se aborda el análisis del rendimiento computacional de las operaciones fundamentales de un KD-tree en el contexto de búsqueda de vecinos más cercanos en un espacio tridimensional. Se compararán las implementaciones de búsqueda de vecinos más cercanos mediante fuerza bruta y utilizando un KD-tree. Además, se explorará el impacto del parámetro k (número de vecinos a buscar) en el rendimiento de estos algoritmos.

El estudio de estas estructuras de datos y algoritmos es esencial para comprender su aplicación en problemas de búsqueda eficiente en espacios multidimensionales, como el análisis de conjuntos de datos tridimensionales representativos del mundo real.

II. Marco Teórico

A. KD-Tree

Un KD-tree es una estructura de datos que organiza puntos en un espacio k-dimensional, facilitando búsquedas eficientes de vecinos cercanos y otras operaciones geométricas. La inserción en un KD-tree implica dividir recursivamente el espacio basado en las coordenadas de los puntos.

B. Método de los k Vecinos Más Cercanos

Dado un conjunto de puntos $P = \{p_1, \ldots, p_n\}$ en un espacio métrico X con una función de distancia d, que permite cierto procesamiento eficiente en P, el objetivo es abordar dos tipos de solicitudes:

1) Espacio Métrico: Un espacio métrico es un conjunto que posee una función de distancia asociada, definida sobre dicho conjunto. Esta función cumple propiedades atribuidas a la distancia, asegurando que para cualquier par de puntos en el conjunto, la distancia asignada por dicha función es válida.

2) Vecino Más Próximo: Se busca localizar el punto en P que está más próximo a $q \in X$.

3) Rango: Dado un punto $q \in X$ y $r > 0$, se buscan todos los puntos $p \in P$ que satisfacen $d(p, q) \leq r$.

Dado que la base de estos enfoques es el cálculo de distancias, el método más simple para buscar el vecino más cercano es conocido como "fuerza bruta" o "exhaustivo". Este método calcula todas las distancias entre un punto y los puntos en la muestra, asignando al conjunto de vecinos más cercanos aquellos cuya distancia es mínima.

Es importante destacar que los métodos basados en vecindad dependen fundamentalmente de la distancia y, en consecuencia, poseen características inherentes a esta, como la cercanía, lejanía y magnitud de longitud, entre otras. Además de ser utilizados para tareas de clasificación, los métodos basados en vecindad también encuentran aplicación en la agrupación de datos. Se pueden clasificar en dos grupos: métodos retardados (o lazy) y métodos no retardados (o eager), dependiendo de cómo se realiza el aprendizaje. [1]

En los métodos retardados como "k-vecinos-NN", cada vez que se va a clasificar un dato, en la fase de entrenamiento, se elabora un modelo específico para cada nuevo dato, y una vez que éste se clasifica sirve como un nuevo caso de entrenamiento para clasificar una nueva instancia. (Técnicas Bayesianas).

III. Análisis de los Métodos Implementados

A. Complejidad de construcción de un KD-Tree

La complejidad temporal de construir un árbol KD a partir de un conjunto de n puntos en un espacio k-dimensional depende del algoritmo de búsqueda de la mediana utilizado.

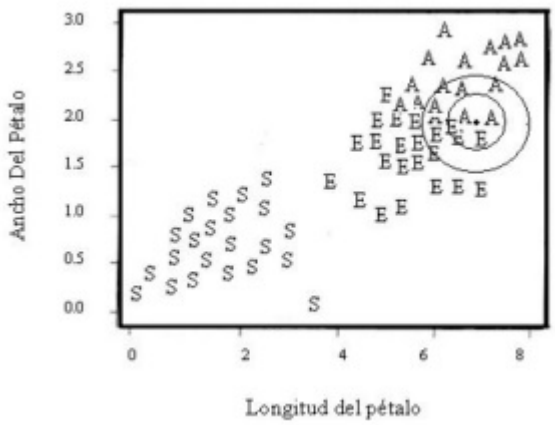

Fig. 1. Clasificación mediante Vecino más próximo [2]

Si se utiliza un algoritmo de mediana de medianas más eficiente, la complejidad total es $O(n \log n)$, en promedio. Esto se debe a que, en cada nivel del árbol, se elige un nuevo hiperplano para dividir el conjunto de puntos, lo que requiere un tiempo $O(n)$. Como el árbol es binario, el número de niveles del árbol es $O(\log n)$. Por tanto, la complejidad temporal total de la construcción del árbol K-D es $O(n \log n)$. Sin embargo, en el peor de los casos, el tiempo de construcción puede ser $O(n^2)$, lo que puede ocurrir si el árbol está muy desequilibrado.

Si, en cambio, se utiliza un algoritmo de ordenación clásico con complejidad $O(n \log n)$ para hallar la mediana, como en este caso; la complejidad total pasa a ser $O(n \log^2 n)$.

B. Complejidad de inserción en KD Tree

La inserción en un árbol KD funciona exactamente igual que en un árbol binario con el ajuste de que seguimos la rama según la coordenada de división de cada nodo y colocamos el nodo como hoja en la posición correcta con respecto a la coordenada de división correcta al final. correcta al final.

La operación de inserción en un KD-tree tiene una complejidad temporal de $O(\log n)$, donde n es el número de nodos en el árbol. Esto se debe a que, en cada nivel del árbol, la búsqueda desciende por la rama adecuada basándose en la coordenada de división. Como el árbol KD se equilibra automáticamente durante la inserción, el número de niveles es proporcional a $\log n$. Por lo tanto, la complejidad temporal de la inserción es eficiente y permite mantener un rendimiento logarítmico incluso en conjuntos de datos grandes.

C. Complejidad de la búsqueda de los k-Vecinos más Cercanos

Definiendo previamente los siguientes parámetros:

- n: número de puntos del conjunto de datos de entrenamiento
- d: dimensionalidad de los datos
- k: número de vecinos que tenemos en cuenta para la votación

a) Usando fuerza bruta: El algoritmo consiste en una simple búsqueda exhaustiva, que en pseudocódigo es:

```
Recorre todos los puntos k veces:
   1. Calcula la distancia entre la muestra
       actual del clasificador y puntos de
       entrenamiento, recuerde el indice del
       elemento con la distancia mas pequenha
       (ignorar los puntos previamente
       seleccionados)
   2. Anhada la clase del indice encontrado al
        contador
Devuelve la clase con mas votos como
    prediccion
```

Se trata de una estructura de bucle anidado, en la que el bucle exterior tarda k pasos y el bucle interior tarda n pasos. El tercer punto es $O(1)$ y el cuarto es $O(\#$ de clases), por lo que son más pequeños. Además, tenemos que tener en cuenta el número de dimensiones d, ya que más direcciones significan vectores más largos para calcular las distancias. Por tanto, la complejidad temporal es $O(n \cdot k \cdot d)$.

En cuanto a la complejidad espacial, necesitamos un vector pequeño para contar los votos de cada clase. Casi siempre es muy pequeño y es fijo, por lo que podemos tratarlo como una complejidad espacial $O(1)$.

b) Usando un KD Tree: La estructura de árbol k-d permite la operación de "consulta de los k puntos vecinos más cercanos", esencial para kNN. El método básico implica realizar la consulta k veces, eliminando el punto encontrado cada vez, con una complejidad de $O(k \cdot \log(n))$. Sin embargo, debido a que el árbol k-d ya recorta espacio durante su construcción, después de una única consulta, se conoce aproximadamente la región donde buscar. Las implementaciones prácticas del árbol k-d permiten consultar k vecinos a la vez, con una complejidad de $O(\sqrt{n} + k)$, lo cual es altamente eficiente para dimensiones mayores, comunes en el aprendizaje automático.

D. Incremento del parámetro 'n'

Este es el problema descrito como la maldición de la dimensionalidad. La complejidad de un problema se incrementa al aumentar la dimensión de las variables involucradas. Al aumentar la dimensión, el espacio está cada vez más vacío complicando cualquier proceso de inferencia. Al aumentar la dimensión aumenta su volumen y la densidad de la variable aleatoria disminuye. En espacios con muchas dimensiones, el método del vecino más próximo puede presentar algunos problemas, debido a que la vecindad se hace muy grande.
Por ejemplo, para el caso de 5000 puntos distribuidos uniformemente en un hípercubo unitario, se desea aplicar el método del vecino más próximo, considerando además que el punto de referencia está en el origen. En una dimensión, se recorrería en promedio una distancia de 5/5000 = 0.001 para tener a los 5 vecinos más próximos. En dos dimensiones, se recorrería en promedio una distancia de 0.001 para tener un cuadrado que tuviera 0.001 de volumen. En d dimensiones se recorrería 0.0011/d (media geométrica). [1]

IV. Resultados

A. Tiempo de ejecución del método de inserción

Se muestra a continuación los distintos tiempos de ejecución de inserción en base a un KD balanceado o no, con las muestras de datos provistas.

TABLE I
TIEMPOS DE EJECUCIÓN: INSERT

Métodos insert	**Tamaños de muestra (ms)**		
	1000.csv	***10000.csv***	***20000.csv***
No Balanceado	1.398	127.510	540.939
Balanceado	3.237	42.065	86.068

Asimismo, se presenta también el tiempo de ejecución obtenido de las funciones de búsqueda de n-Vecinos más Cercanos (se hizo uso del dataset mas es)

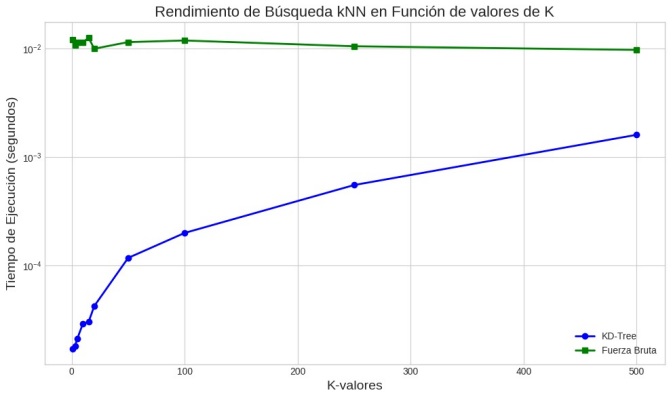

Fig. 2. Tiempos de ejecución en Fuerza Bruta

V. Discusión

En este trabajo se implementó la inserción balanceada en un KD Tree a traves del uso del algoritmo de busqueda de la mediana en un conjunto de puntos. Tal como se mencionó anteriormente; el costo computacional de este método es susceptible a mejoras dependiendo del algoritmo implementado. En esta oportunidad se implementó un algoritmo clásico (enfoque sencillo). Mas este puede mejorarse, y pasar de tener un costo de $O(n \cdot \log(n))$ a $O(n)$, usando un algoritmo más eficiente (Búsqueda de mediana usando Ramificación y Poda).

VI. Conclusiones

En este estudio, hemos explorado a fondo la aplicación y rendimiento de la estructura de datos KD-tree en el contexto de la búsqueda de vecinos más cercanos (kNN). La inserción eficiente en el árbol KD, con una complejidad temporal de $O(\log n)$, destaca como un componente clave para su utilidad en aplicaciones de kNN. La capacidad del árbol KD para adaptarse y equilibrarse automáticamente durante la inserción garantiza un rendimiento logarítmico, incluso en conjuntos de datos considerables.

Además, hemos examinado la fase de predicción, donde la estructura de árbol k-d demuestra su eficacia al permitir la búsqueda de los k puntos vecinos más cercanos con una complejidad $O(\sqrt{n} + k)$. Este enfoque ofrece un balance óptimo entre eficiencia y precisión, especialmente en dimensiones superiores comunes en el aprendizaje automático.

Comparando con el método de fuerza bruta, la implementación del árbol KD se destaca significativamente en términos de eficiencia computacional. Los resultados obtenidos a partir de conjuntos de datos de prueba respaldan la eficacia de KD-tree, subrayando su capacidad para optimizar la búsqueda en espacios multidimensionales.

En resumen, el árbol KD emerge como una herramienta valiosa para problemas de búsqueda de vecinos más cercanos, proporcionando soluciones eficientes y escalables para conjuntos de datos de diversos tamaños y dimensionalidades. Su aplicación en el aprendizaje automático y otras áreas afines promete contribuir de manera significativa a la mejora de los algoritmos basados en kNN.

References

[1] Quezada, N. (2017). "K-vecinos más próximos en una aplicación de clasificación y predicción en el Poder Judicial del Perú". [Tesis de maestría, Universidad Nacional Mayor de San Marcos, Facultad de Ciencias Matemáticas, Unidad de Posgrado]. Repositorio institucional Cybertesis UNMSM.

[2] Rodríguez, J., Rojas E., Franco, R. "Clasificación de datos usando el método k-nn". Colombia. I+D investigación y Desarrollo.

[3] C. G. Cambronero y I. G. Moreno, "ALGORITMOS DE APRENDIZAJE: KNN y KMEANS [Inteligencia en Redes de Telecomuncicación]", Ujaen.es. [En línea]