

Estructuras de Datos

Grado de Ing. Informática

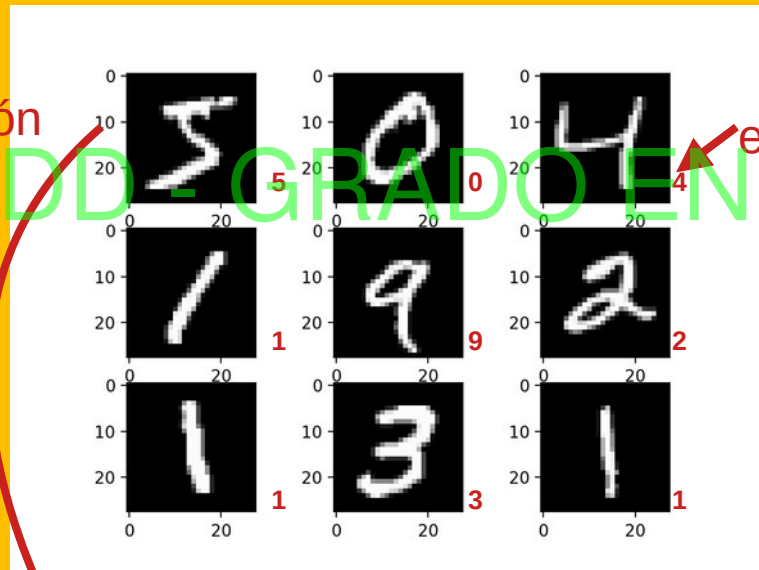
EEDD - GRADO EN ING. INFORMÁTICA - UCO

Árbol KD-TREE

Motivación

- El problema del vecino más cercano.

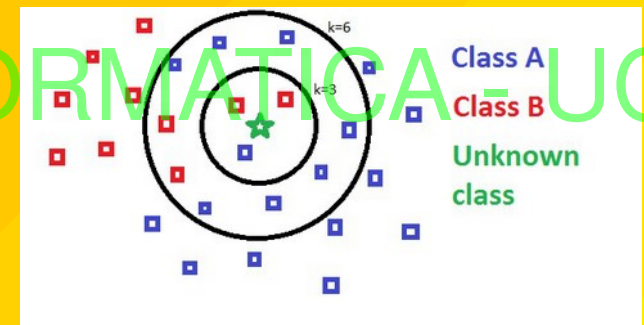
Dataset anotado



16x16 = 256 dimensiones
0010000100111001 ... 01100



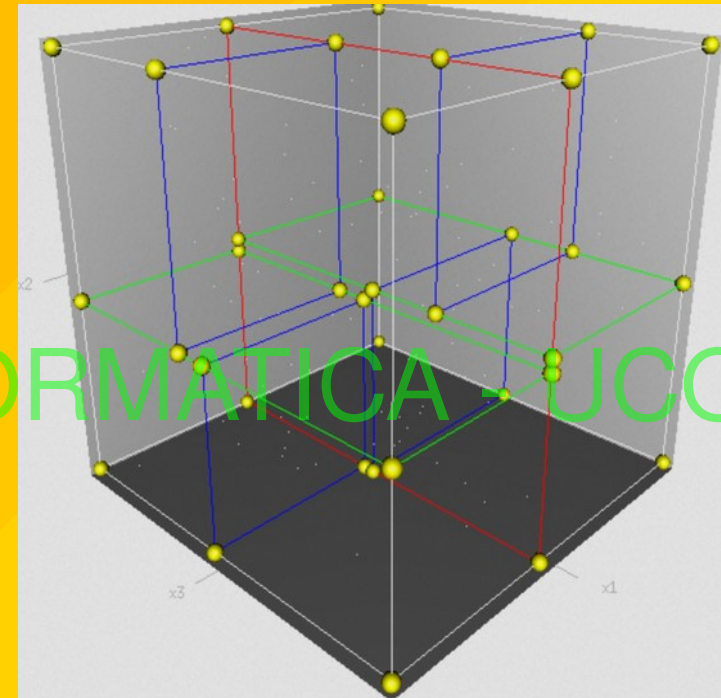
¿qué etiqueta
tendrá?



Usando un array de [patrones|
etiquetas] para representar el
Dataset, ¿qué complejidad tendría
buscar el vecino más cercano?
¿se podrá hacer mejor?

KD-Tree

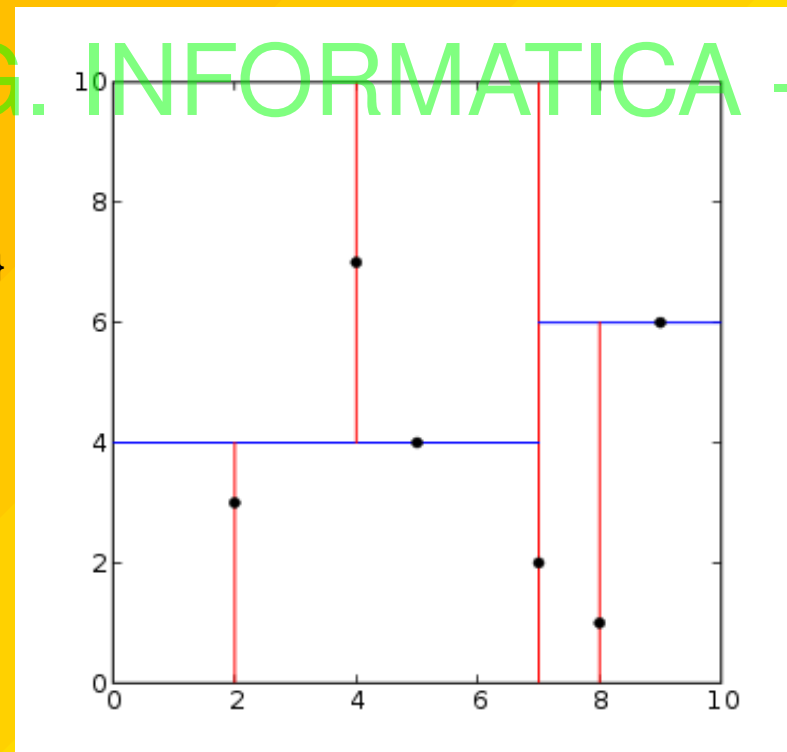
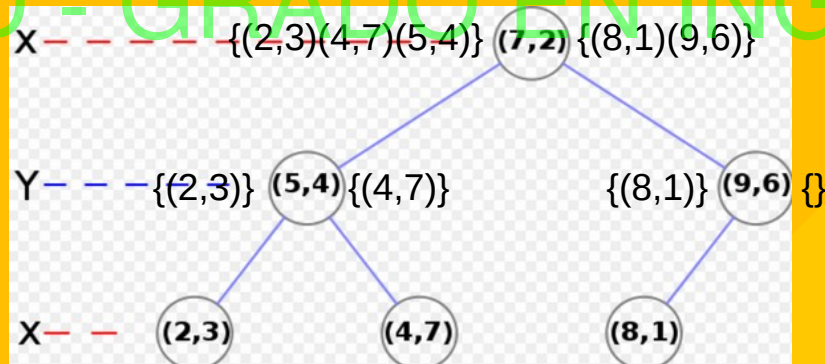
- Representan una partición de un espacio de K Dimensiones.
- Permiten optimizar la búsqueda de:
 - Los K patrones más cercanos a otro dado.
 - Todos los patrones en un radio dado alrededor de otro punto.



División del espacio por bi-particiones.

KDTree

- Construcción: ejemplo.
 - Dataset: $[(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)]$



KDTree

- Construcción: Algoritmo canónico.

```
Algorithm make_kdtree (ds:DArray[Point],
                      depth:Integer):Binarytree[Point]
Var
  kdtree:BinaryTree[Point]
  axis:Integer
Begin
  kdtree ← BinaryTree.create()
  If ds.size()>0 Then
    axis ← depth mod Point.dimensions();
    sort_axis(ds, axis)
    median ← ds[ds.size()//2]
    kdtree.createRoot(median)
    kdtree.set_left(make_kdtree(ds[0:ds.size()//2], depth+1);
    kdtree.set_right(make_kdtree(ds[ds.size()//2+1:], depth+1);
  End-if
  Return kdtree
End.                                     O ( ? )
```

KDE-Tree

- Búsqueda del vecino más cercano.

```

Algorithm find_NN(t:KDTree, p:Point, depth:Integer): Point
Prec: Not t.is_empty()
Var
  axis:Integer //The axis to inspect.
  curr, curr2: Point //The best up to date.
Begin
  axis ← depth mod p.dim() //Selected the axis to split.
  curr ← t.item()
  curr2 ← curr
  If p[axis] ≤ curr[axis] And Not t.left().is_empty() Then
    curr2 ← find_NN(t.left(), p, depth+1)
  Else If p[axis] > curr[axis] And Not t.right().is_empty() Then
    curr2 ← find_NN(t.right(), p, depth+1)
  End-If
  If dist(p, curr2) < dist(p, curr) Then
    curr ← curr2
  If abs(p[axis] - t.item()[axis]) < dist(curr, p) Then
    If p[axis] > t.item()[axis] And Not t.left().is_empty() Then
      curr2 ← find_NN(t.left(), p, depth+1)
    Else If Not t.right().is_empty() Then
      curr2 ← find_NN(t.right(), p, depth+1)
    End-If
    If dist(curr2, p) < dist(curr, p) Then
      curr ← curr2
    End-If
  Return curr
End.
  
```

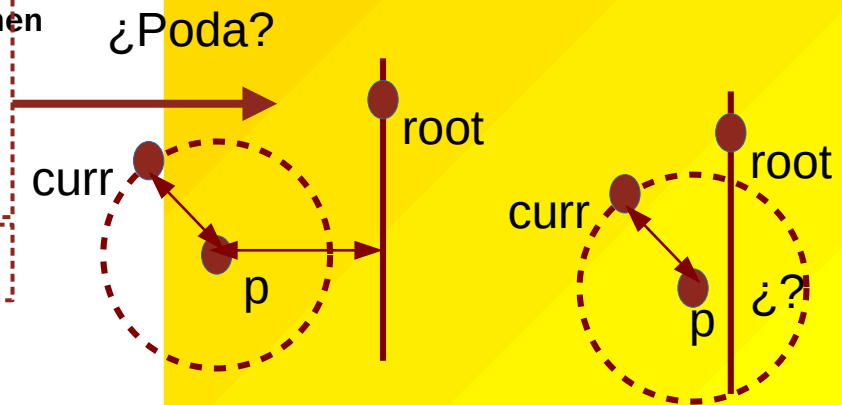
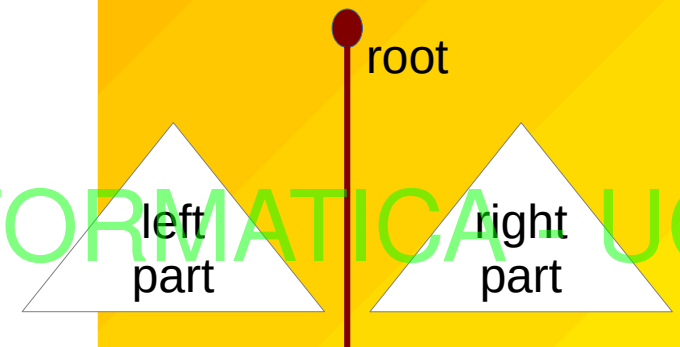
Paso 1

Paso 2

Paso 3

Paso 4

$O(?)$



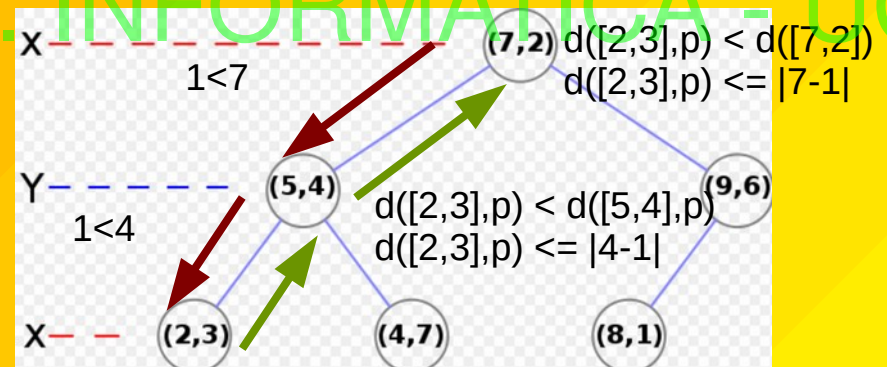
KDE-Tree

- Ejemplo (distancia L1).
 - El 1-NN de $p:[1,1]$

Nivel.Paso

- 0.1 $c=[7,2]$, Como $1 < 7$ descender hijo izq.
- 1.1 $c=[5,4]$, Como $1 < 4$ descender hijo izq.
- 2.1 $c=[2,3]$, Hoja.
- 1.2 $c=[5,4], c2[2,3]$, $L1(c2,p) < L1(c,p)$, $c \leftarrow c2$
- 1.3 $c=[2,3]$, $|1-4| \geq L1(c,p)$. Poda.
- 0.2 $c=[7,2], c2[2,3]$, $L1(c2,p) < L1(c,p)$, $c \leftarrow c2$.
- 0.3 $c=[2,3]$, $|1-7| \geq L1(c,p)$. Poda.

Resultado $c=[2,3]$



Candidato: [2,3]

El vecino más cercano es: **[2,3]**

* $L1([1,3], [2,5]) = |1-2| + |3-5| = 3$

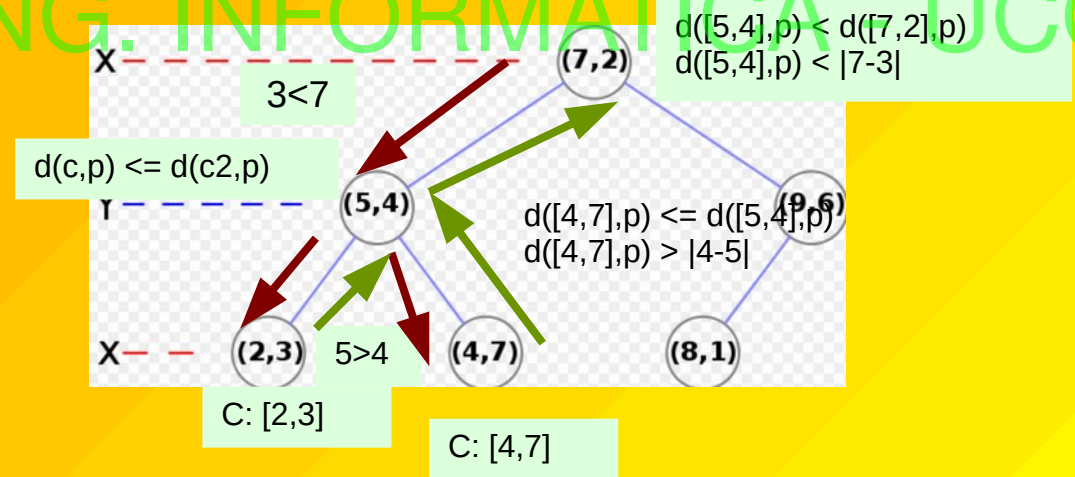
KDE-Tree

- Ejemplo (distancia L1).
 - El 1-NN de $p:[3,5]$

Nivel.Paso

- 0.1 $c=[7,2]$, Como $3 < 7$ descender hijo izq.
- 1.1 $c=[5,4]$, Como $5 > 4$ descender hijo der.
- 2.1 $c=[4,7]$, Hoja.
- 1.2 $c=[5,4]$, $c2=[4,7]$, $L1(c,p) \leq L1(c2,p)$
- 1.3 $c=[5,4]$, $|5-4| < L1(c,p)$, desc. hijo izq.
- 2.1 $c=[2,3]$, Hoja
- 1.4 $c=[5,4]$, $c2=[2,3]$, $L1(c,p) \leq L1(c2,p)$.
- 0.2 $c=[7,2]$, $c2=[5,4]$, $L1(c,p) > L1(c2,p)$, $c < -c2$
- 0.3 $c=[5,4]$, $|3-7| \geq L1(c,p)$, Poda.

Resultado $c=[5,4]$



El vecino más cercano es: **[5,4]**

KD-Tree

- Resumiendo.
 - Representa una bipartición de una dimensión del espacio en cada nivel.
 - Su construcción puede ser $O(N \log N)$.
 - Si utilizamos la mediana como pivote, tendremos un árbol perfectamente equilibrado.
 - La localización es en promedio $\log(N)$ pero puede degenerar en $O(N)$ si el número de dimensiones es muy alto.
 - Será mejor que una búsqueda exhaustiva si $N \gg 2^k$

KDE-Tree

- Lecturas recomendadas.
 - KDTree: en.wikipedia.org/wiki/K-d_tree

EEDD - GRADO EN ING. INFORMÁTICA - UCO