

INF 493/792
Tópicos especiais III
Introdução à mineração de dados

Aula III: Encontrando objetos similares

Avisos

Leituras recomendadas no fim de cada aula
Impossível cobrir tudo nas aulas!

Piazza para tirar dúvidas

Lista 1 estará disponível nos próximos dias

Objetivo de hoje

Revisão do final da aula passada

O problema de encontrar objetos similares

KD-tree (espaços com baixa/média dimensão)

Hash de similaridade (espaços de alta dimensão)

Observação sobre a Distância do cosseno

$$d(x, y) = 1 - \frac{x \cdot y}{\|x\| \|y\|}$$

Sem o módulo!

- Alguns textos dizem que só deve ser usada para vetores cujas componentes são não negativas
- Alguns textos falam que pode variar entre 0 e 2

Métrica

Uma medida de dissimilaridade $\mathbf{d}(., .)$ é uma métrica se:

$$d(x, y) \geq 0$$

$$d(x, y) = d(y, x)$$

$$d(x, y) = 0 \iff x = y$$

$$d(x, y) \leq d(x, z) + d(z, y)$$

Propriedades: não-negatividade, simetria, isolamento, desigualdade triangular

Distância de Manhattan

Prove que esta é (ou não) uma métrica

5 minutos

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

Distância de Manhattan

E se eu falar que $|a + b| \leq |a| + |b|$, sendo **a** e **b** números reais

Distância Euclidiana

Prove que esta é (ou não) uma métrica

5 minutos

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Distância Euclidiana

E se eu der a desigualdade de Cauchy-Schawrz?

Distância Euclidiana ao quadrado

$$d(x, y) = \sum_{i=1}^n (x_i - y_i)^2$$

Distância euclidiana modificada

$$d(x, y) = \sqrt{\sum_{i=1}^{n-1} (x_i - y_i)^2}$$

Distância do cosseno

$$d(x, y) = 1 - \frac{x \cdot y}{\|x\| \|y\|}$$

$$x \cdot y = \sum_{i=1}^n x_i y_i \quad \|x\| = \sqrt{\sum_{i=1}^n x_i^2}$$

Notação

Em alguns textos:

dissimilaridade e distância são a mesma coisa

Em outros textos:

Distância e métrica são a mesma coisa

Problema

Dados um conjunto X de n pontos do espaço d -dimensional (real ou binário), um ponto de consulta q e uma função de distância d , como encontrar x (pertencente a X) que minimize $d(x, q)$?

Intuição: encontrar o ponto em X que seja o mais “próximo” de q

Aplicações/Motivação

Encontrar páginas Web similares (plágio ou repetição)

Clientes que compraram produtos similares

Imagens com características similares

Em classificação: regra do vizinho mais próximo!

Solução simples

Solução simples

Busca linear!

Custo **$O(nd)$**

Basicamente, única coisa conhecida para resultados exatos e com garantia

Aula de hoje, como fazer “melhor” que isso?

Quando d não é grande: kd-tree

Quando d é grande: hash de similaridade

KD-tree

Knowledge Discovery tree

Estrutura de dados que suporta **buscas intervalares** no espaço **d**-dimensional real

Utilizada na prática em situações em que **d** não é muito grande

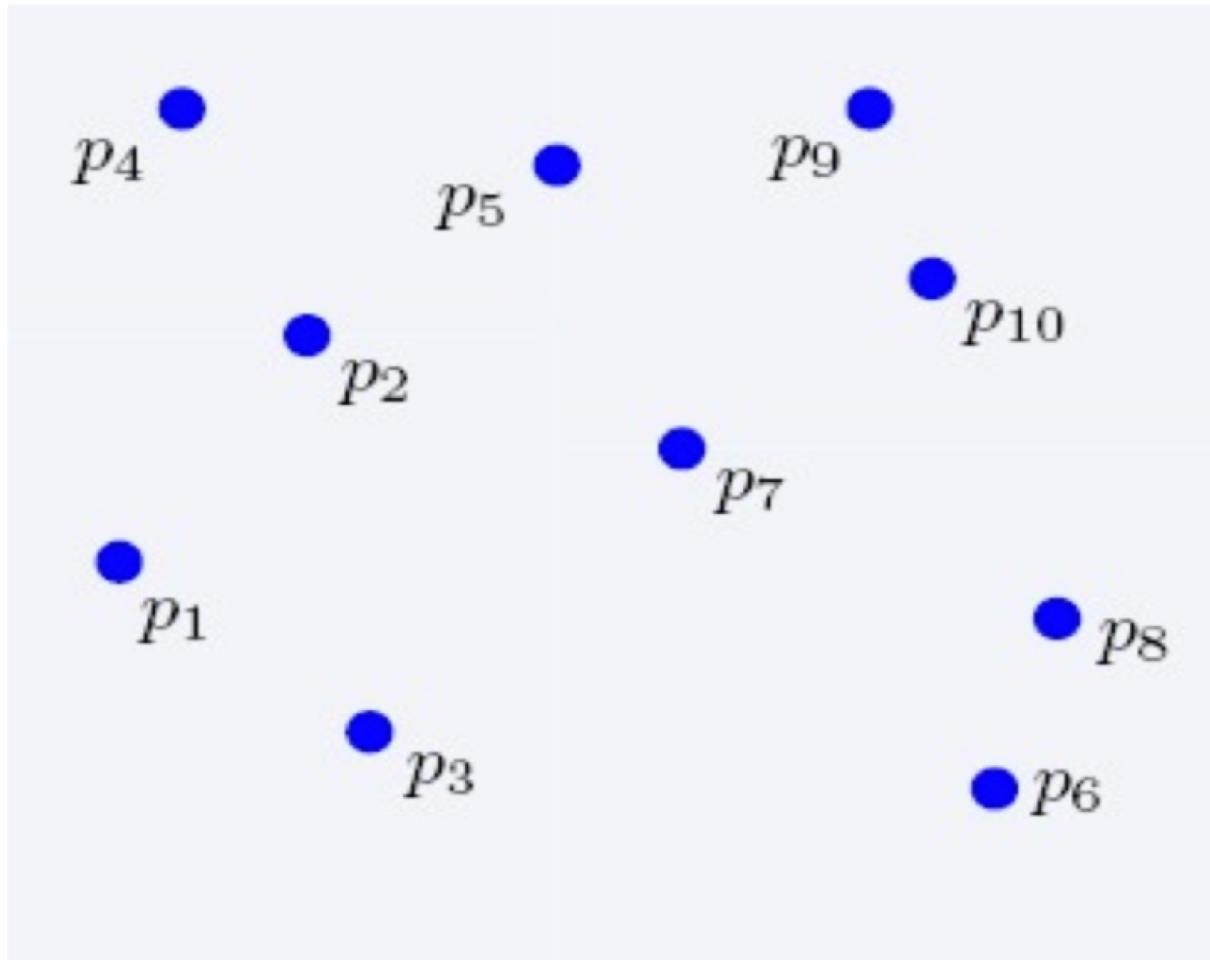
Não é a representação mais eficiente, mas é bem utilizada na prática

2d kd-tree

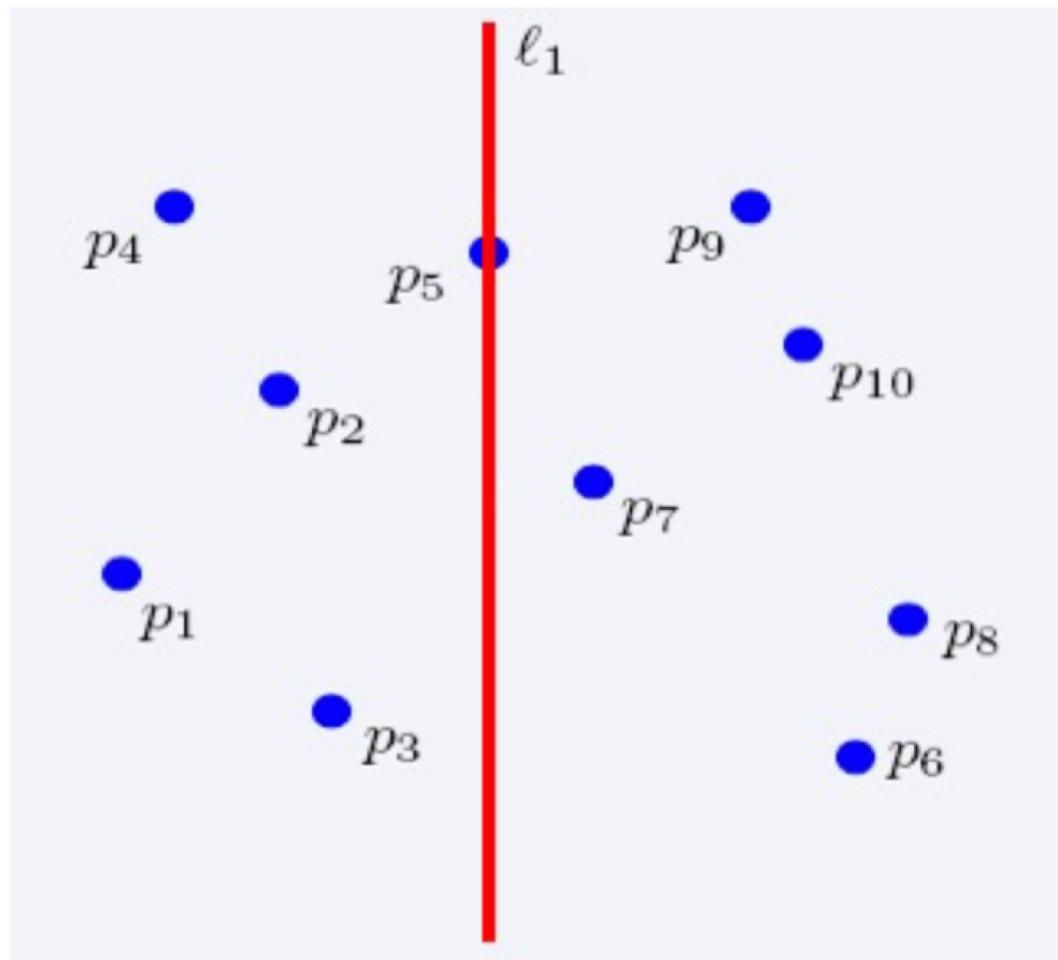
Construção

- Escolha **x** ou **y** (alternando)
- Encontre a mediana da coordenada; esse valor define uma linha vertical ou horizontal
- Repita o procedimento recursivamente em ambos os lados

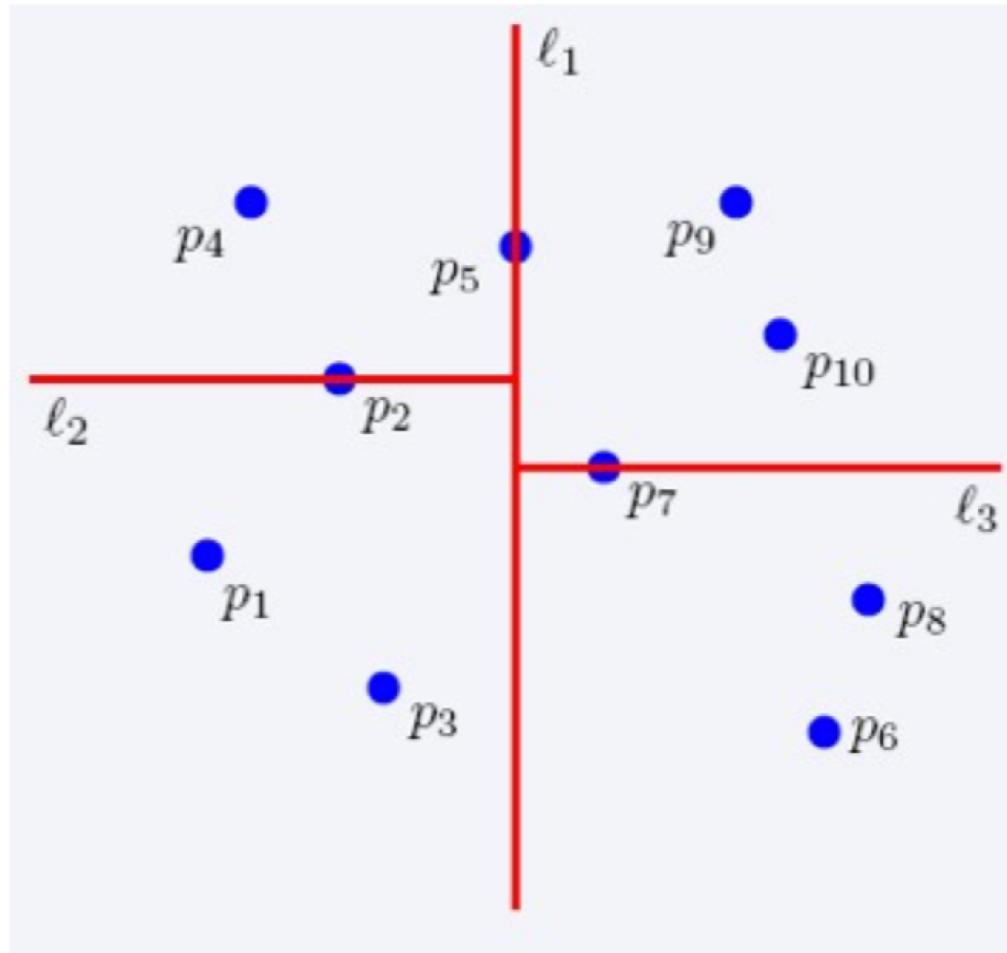
Construção – pontos iniciais



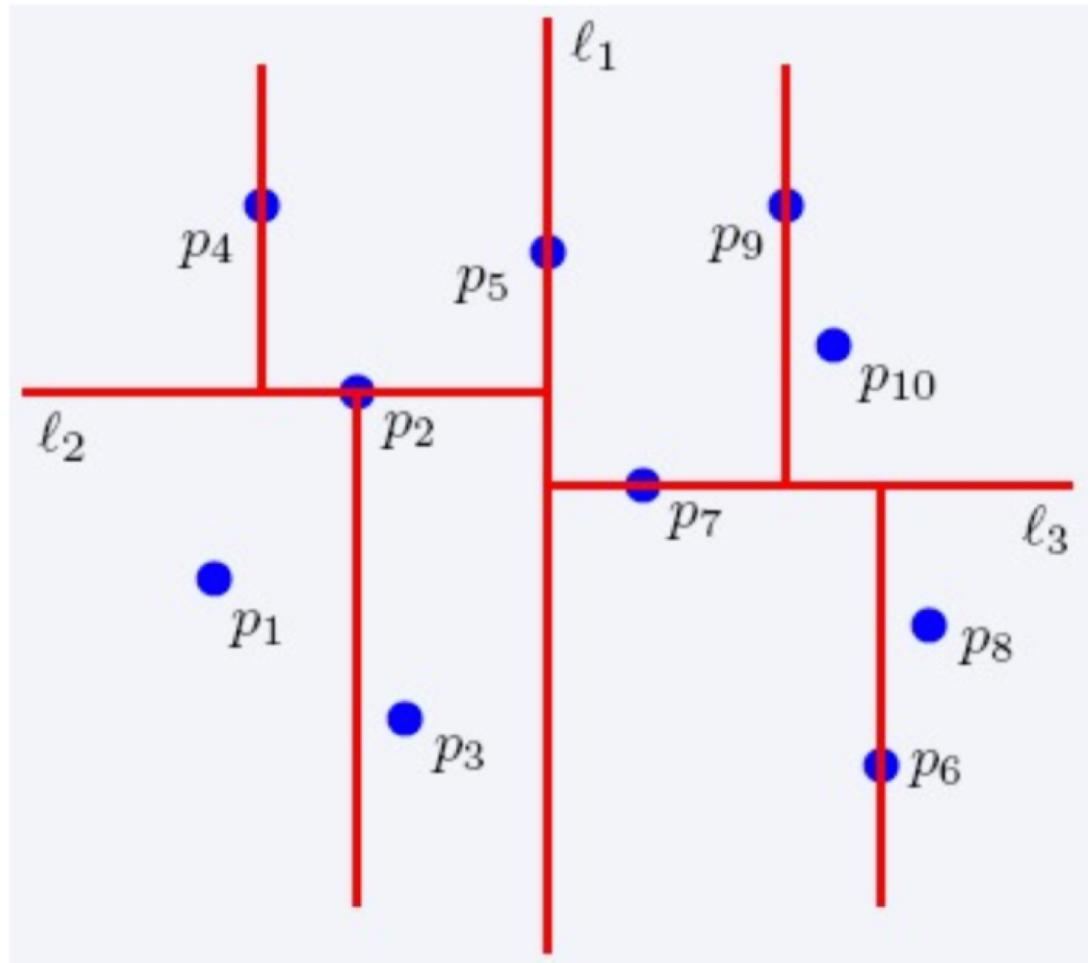
Mediana eixo x = p5



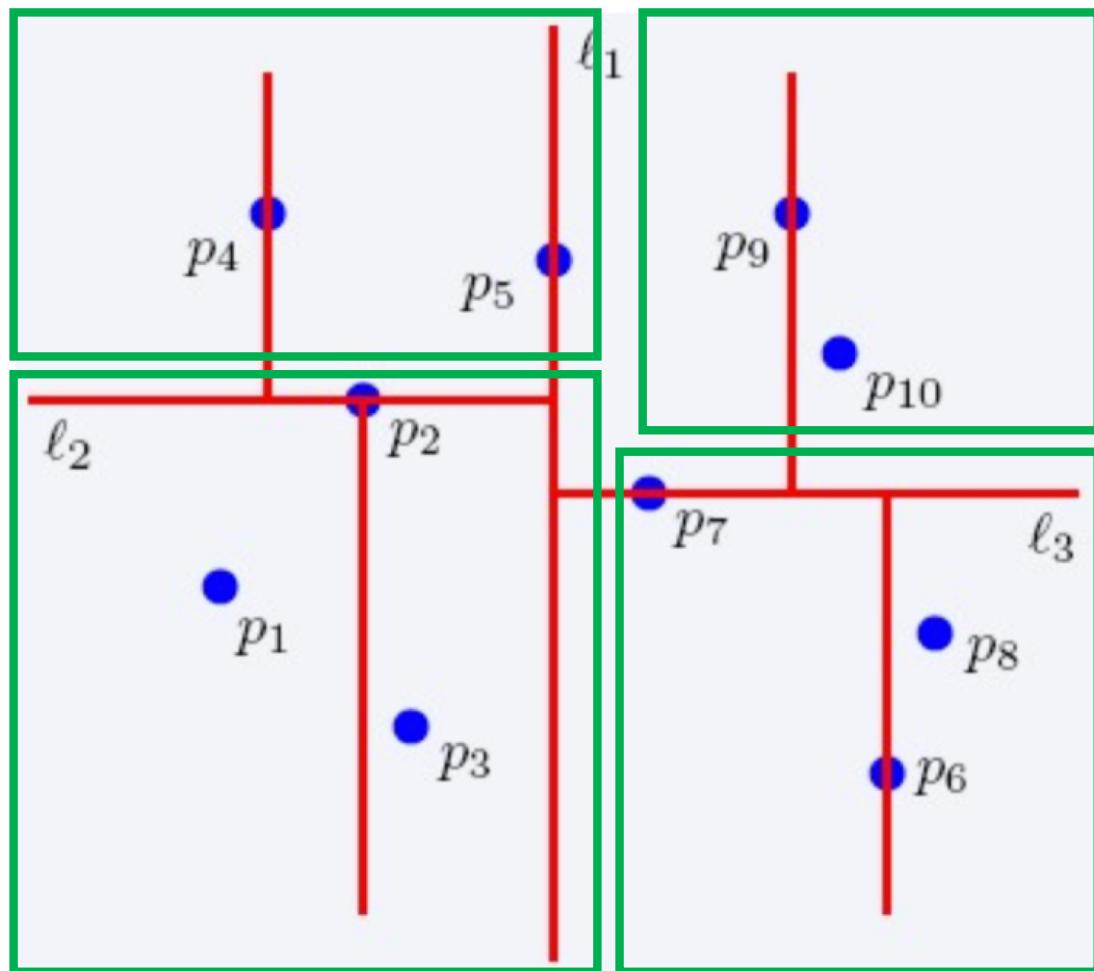
Medianas eixo $y = p_2$ e p_7



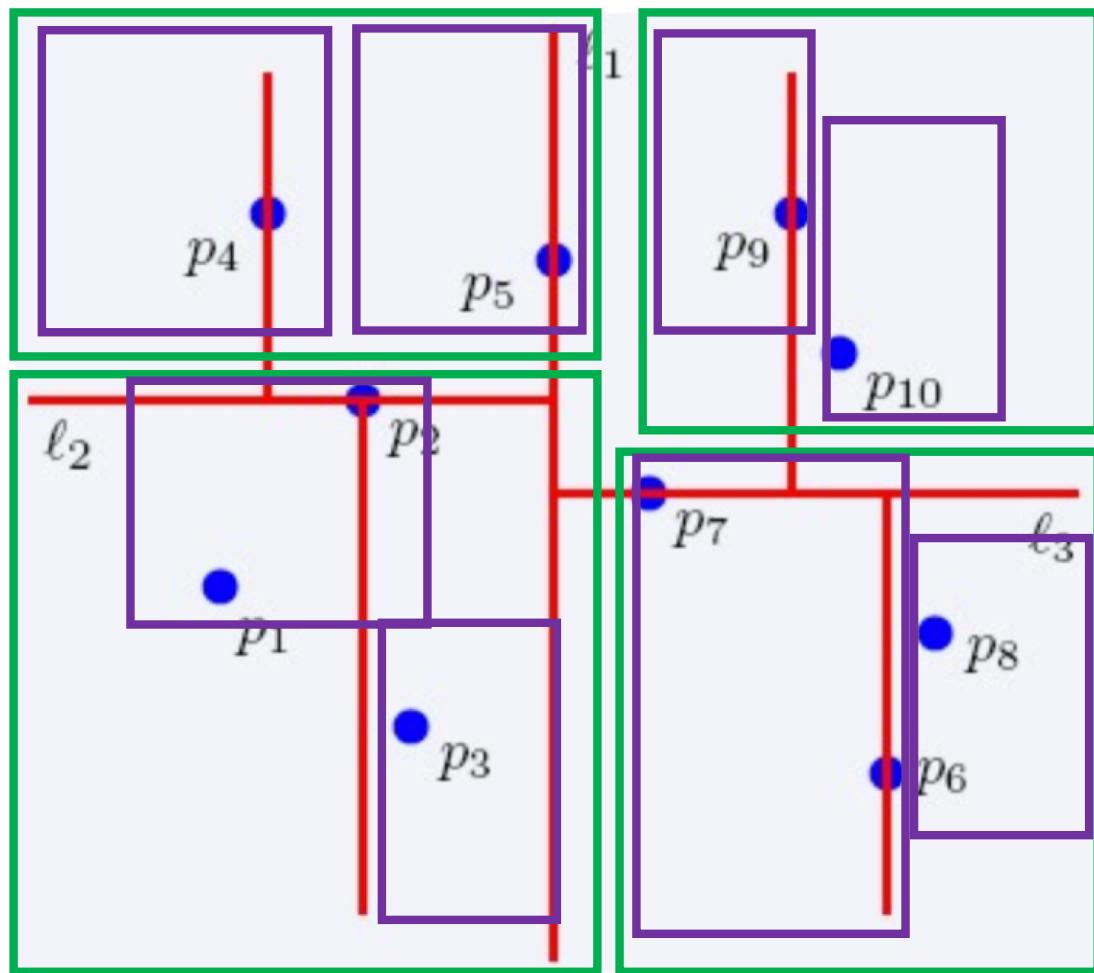
Medianas eixo x = p2, p4, p6 e p9



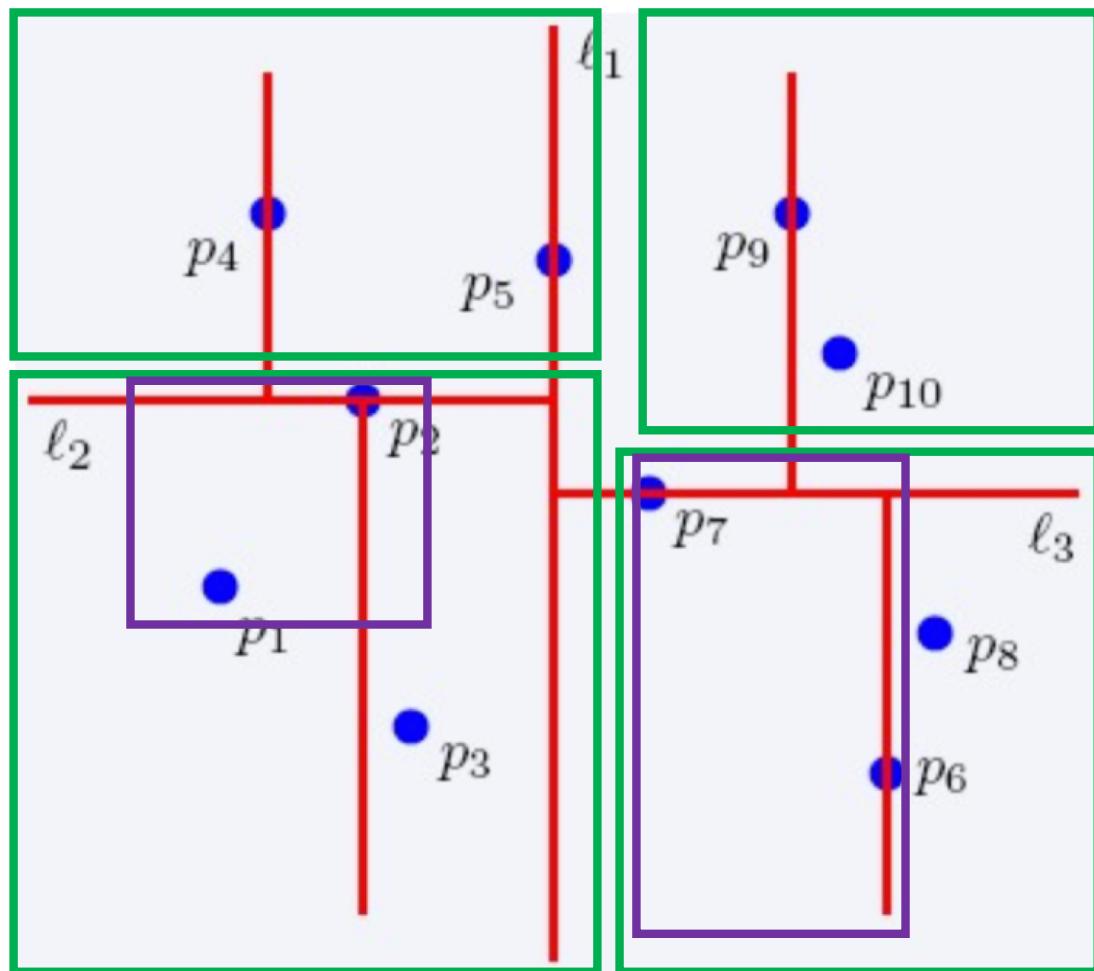
Medianas eixo x = p2, p4, p6 e p9



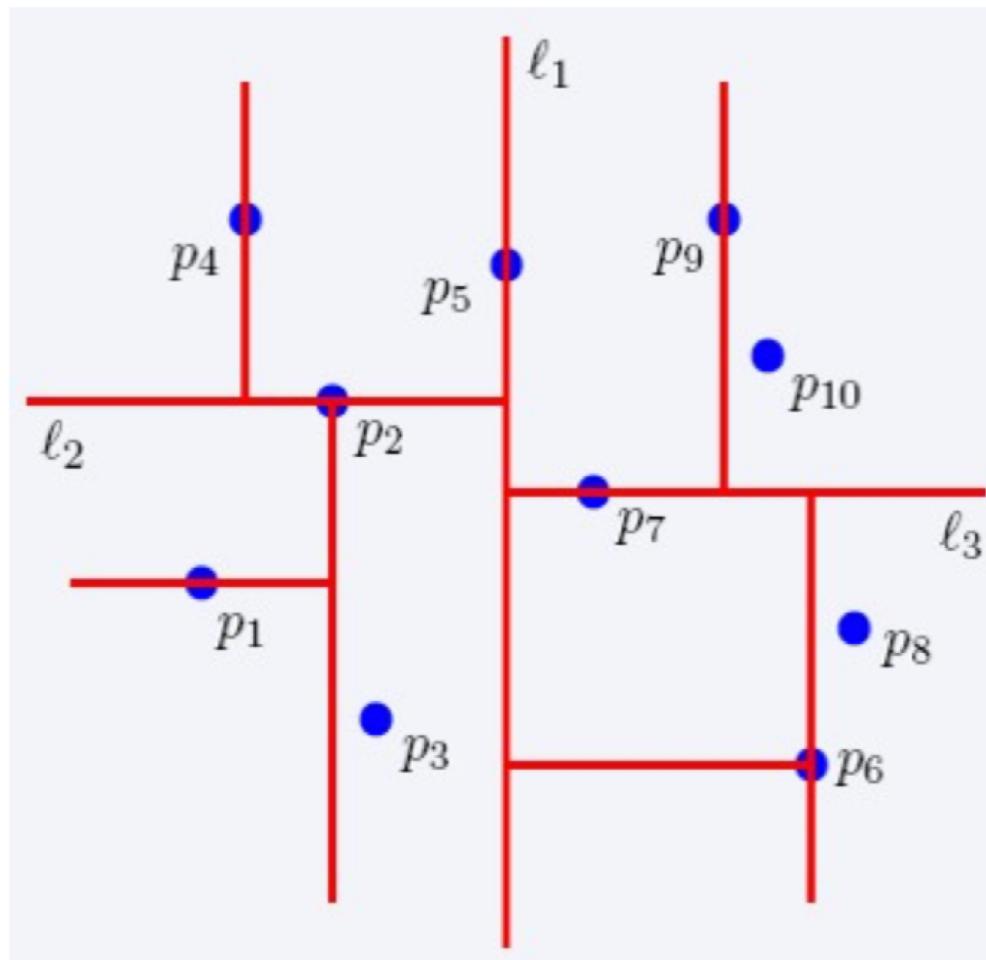
Medianas eixo x = p2, p4, p6 e p9



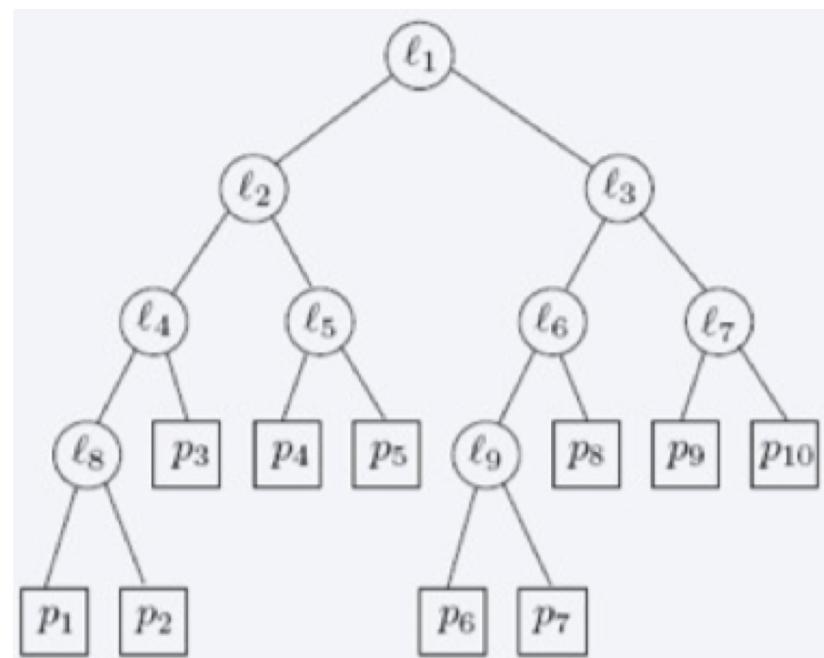
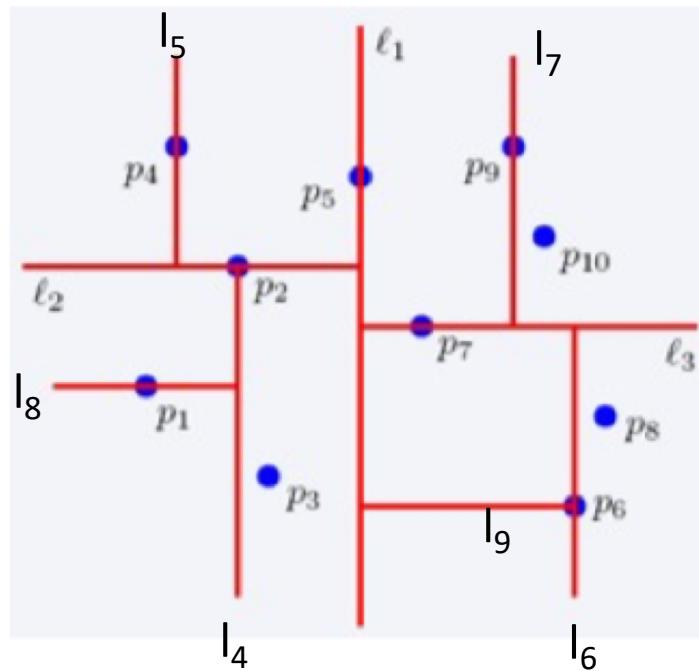
Medianas eixo x = p2, p4, p6 e p9



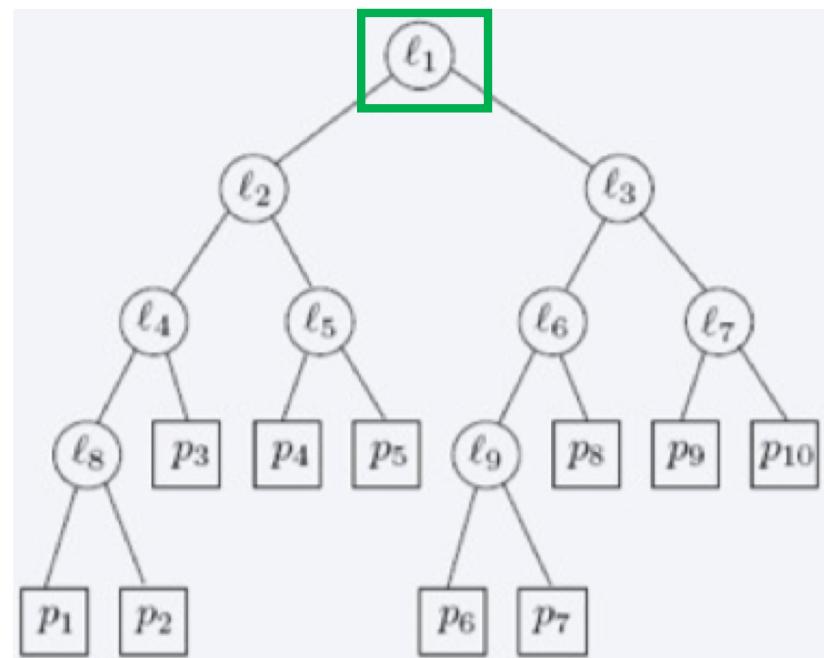
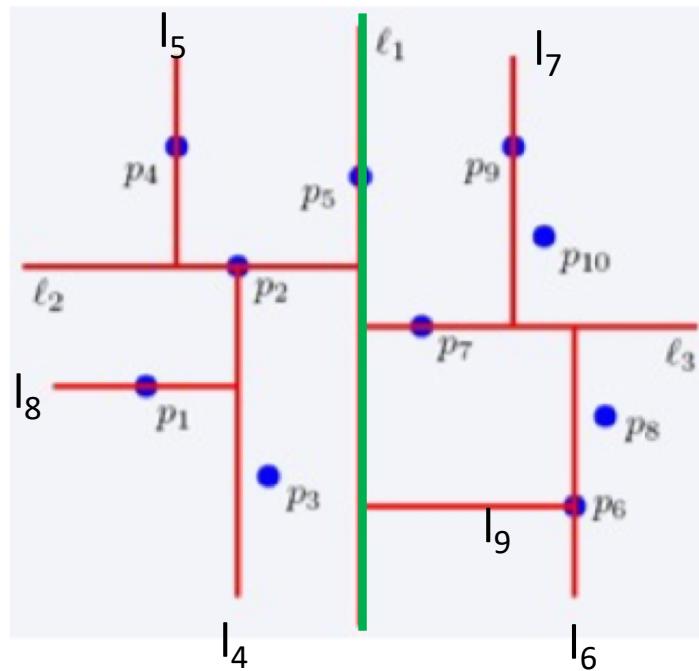
Medianas eixo y = p1 e p6



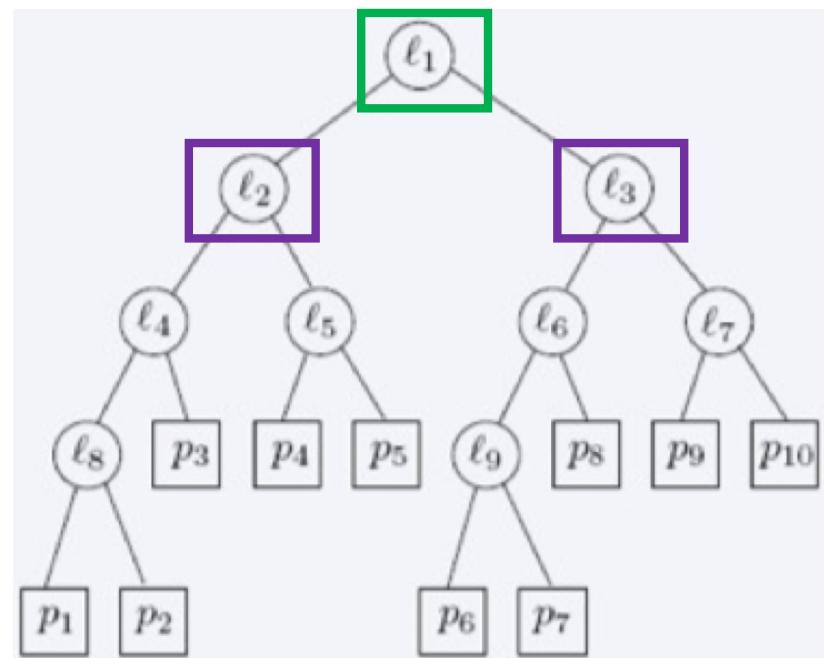
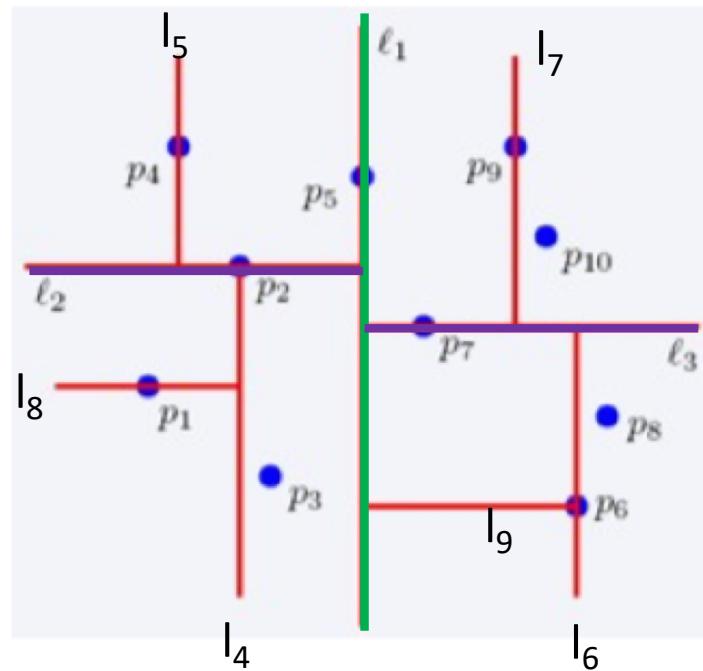
Resultado



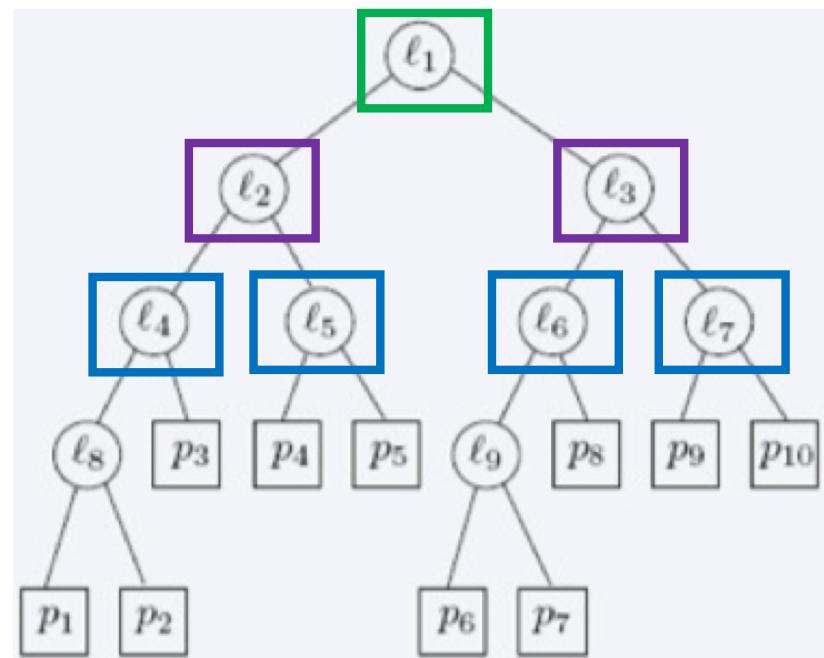
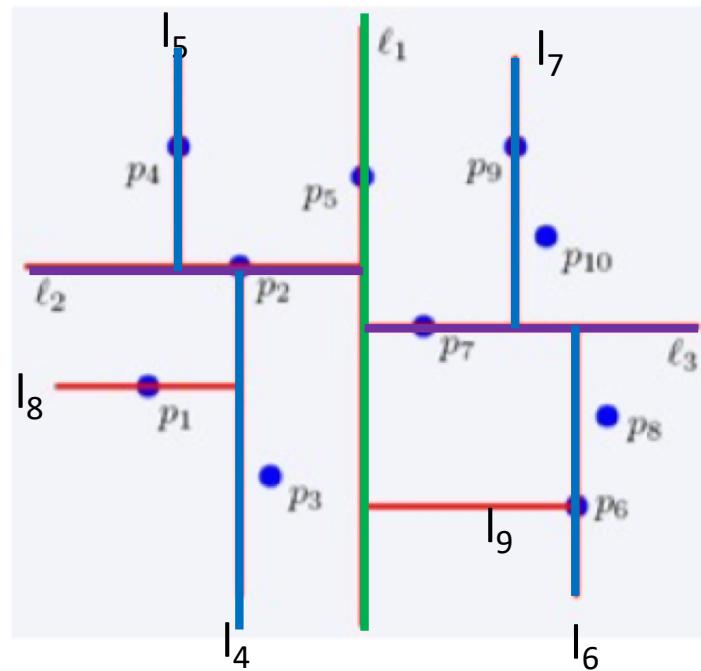
Resultado



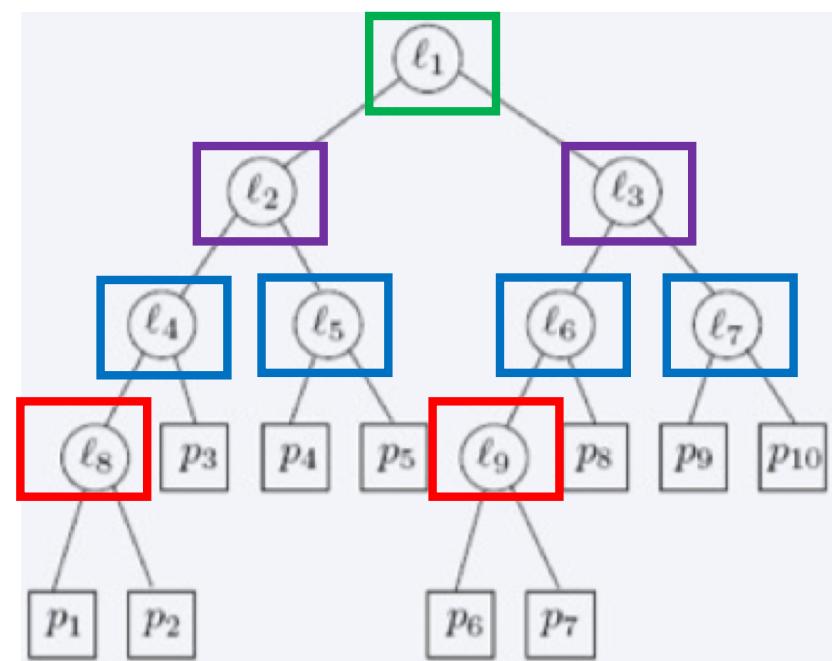
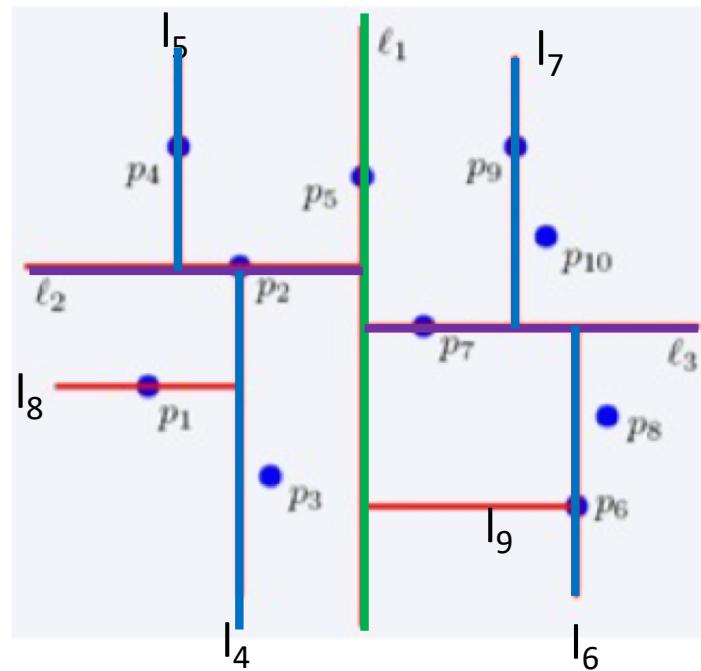
Resultado



Resultado



Resultado



Complexidade

Muito similar ao procedimento de construção de uma árvore de busca binária convencional!

Obtém-se uma árvore binária:

- Tamanho:
- Profundidade:
- Tempo para construção:

2 minutos

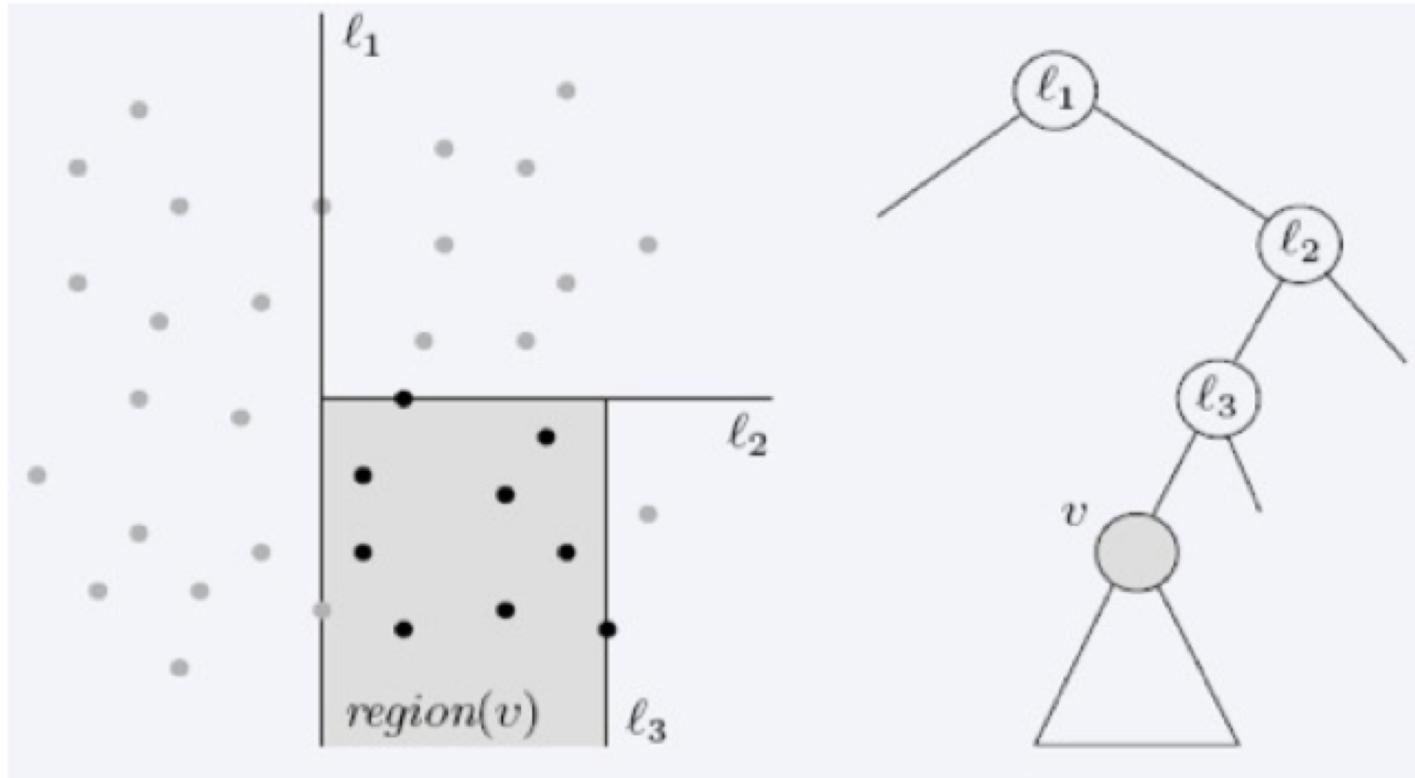
Complexidade

Muito similar ao procedimento de construção de uma árvore de busca binária convencional!

Obtém-se uma árvore binária:

- Tamanho: $O(n)$
- Profundidade: $O(\log n)$
- Tempo para construção: $O(n \log n)$

Região do nó v



$Region(v)$: sub-árvore com raiz sendo v define a região de v (Retângulo)

Busca em uma kd-tree

Útil para fazer busca intervalar, i.e., especificando um retângulo R

Objetivo, encontrar todos os pontos dentro de R

Busca intervalar

Search (v, R)

Se **v** é folha:

retorne o ponto em **v**, se estiver em **R**

Senão:

Se **region(v)** está contida em **R**:

reporte todos pontos da sub-árvore de **v**

Senão:

Se **region(left(v))** tem intercessão com **R**:

Search(left(v), R)

Se **region(right(v))** tem intercessão com **R**:

Search(right(v), R)

Pergunta

Como utilizar esse método para todos os pontos dentro de uma distância r (ou inferior) de um certo ponto q ?

3 minutos...

Complexidade

$O(n^{1/2} + P)$

P é o número de pontos a ser reportado

Fácil de generalizar para caso d -dimensional

Construção: $O(n \log n)$

Espaço: $O(n)$

Busca: $O(n^{1 - 1/d} + P)$

Construção no caso d-dimensional

Similar ao caso **2d**

Na raiz nós dividimos os pontos em dois conjuntos de acordo com eixo x_1

Nos filhos da raiz, a partição é realizada de acordo com o eixo x_2

...

Na profundidade $d + 1$, volta-se ao eixo x_1

A recursão é interrompida quando apenas um ponto restar no conjunto

Dúvida

E se **d** for muito grande?

Hash de similaridade

Ideia: Construir uma função hash h que dado um par de pontos p, q :

Se $d(p, q) \leq r$, então $P[h(p) = h(q)]$ é alta

Se $d(p, q) \geq cr$, então $P[h(p) = h(q)]$ é baixa

Obviamente, depende do domínio dos pontos, função de distância d . Para cada caso há uma h apropriada

Como utilizar?

Distância de Hamming

Para o caso de vetores binários de dimensão d , conta o número de posições que dois vetores p e q diferem

Defina:

$h(p) = p_i$, i.e., o i -ésimo bit de p

Se i é selecionado aleatoriamente, então:

$P[h(p) = h(q)] =$

2 minutos

Distância de Hamming

Para o caso de vetores binários de dimensão d , conta o número de posições que dois vetores p e q diferem

Defina:

$h(p) = p_i$, i.e., o i -ésimo bit de p

Se i é selecionado aleatoriamente, então:

$$P[h(p) = h(q)] = 1 - d(p, q) / d$$

Como utilizar para conseguir o ponto mais próximo?

Podemos apenas computar h uma vez (para uma posição aleatória)?

Como será a taxa de sucesso se fizermos isso?

Como utilizar para conseguir o ponto mais próximo?

Defina $\mathbf{g}(\mathbf{p}) = \langle h_1(\mathbf{p}), \dots, h_k(\mathbf{p}) \rangle$

Pré-processamento:

- 1) Selecione $\mathbf{g}_1(\mathbf{p}), \dots, \mathbf{g}_L(\mathbf{p})$
- 2) Para cada \mathbf{x} no conjunto de dados mapeie \mathbf{x} para os *buckets* $\mathbf{g}_1(\mathbf{x}), \dots, \mathbf{g}_L(\mathbf{x})$

Pode ser feito em tempo $O(nkL)$

Como utilizar para conseguir o ponto mais próximo?

Consulta de um ponto q

- 1) Compute $g_1(q), \dots, g_L(q)$ e obtenha os pontos mapeados em cada um desses *buckets*
- 2) Para cada x obtido no passo anterior verifique se $d(x, q) \leq r$. Se sim, retorne x

Pode ser feito em tempo **$O(dL)$**

Funciona?

Qual a probabilidade de sucesso?

Qual a probabilidade de retornar p , se $d(p, q) > r$?

- vem na lista =)

Comparando conjuntos

Suponha que você tenha uma quantidade grande de documentos, cada um deles representado como um conjunto.

Como encontrar documentos similares?

Na aula passada, nós vimos a distância de Jaccard

Comparar todos os possíveis pares não é viável

Até mesmo comparar dois documentos pode ser caro

Melhorando

Associe a cada objeto uma assinatura (de tamanho bem menor)

Compare assinaturas ao invés de objetos

Encontre os pares de assinaturas similares e então verifique se os objetos associados também são similares

Similaridade de Jaccard

Considere dois objetos, **x** e **y**

Onde representamos conjuntos através de vetores binários

	x	y
a	1	1
b	1	0
c	0	1
d	0	0

Similaridade é 1/3

Ideia

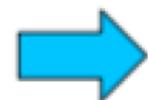
Permute aleatoriamente as linhas da matriz

$h(x)$: primeira linha (da matriz permutada) em que x tem valor 1

Veja que aqui estamos considerando objetos como colunas

Exemplo

	x1	x2	x3	x4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	1	0	1	0
7	1	0	1	0



1
3
7
6
2
5
4



	x1	x2	x3	x4
1	1	0	1	0
3	0	1	0	1
7	1	0	1	0
6	1	0	1	0
2	1	0	0	1
5	0	1	0	1
4	0	1	0	1

1	2	1	2
---	---	---	---

Fato Interessante

$P[h(x) = h(y)]$ é igual a
similaridade de Jaccard entre x e y

5 minutos

Mais interessante ainda

A distância de Jaccard é uma métrica!

5 minutos

Algoritmo

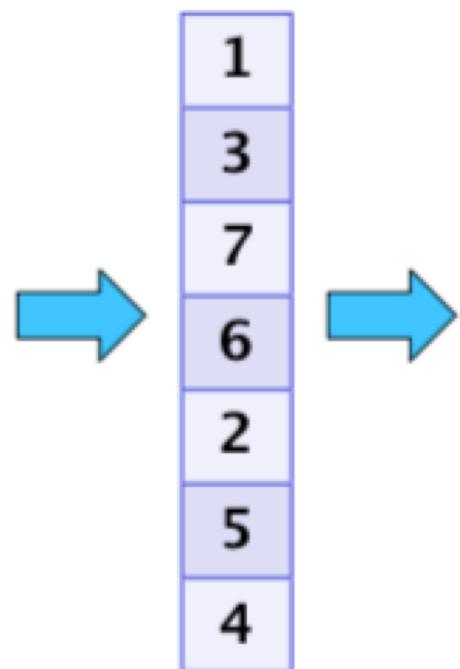
Escolha k (e.g., 100) permutações das linhas

Para cada objeto x , crie um vetor $\text{sig}(x)$, onde $\text{sig}(x)[i]$ é a primeira linha da permutação i que tem um 1
(i.e., repita $h(x)$ k vezes de forma independente)

Pode-se então comparar os vetores de assinatura ao invés dos conjuntos originais!

Exemplo, $i = 1$

	x1	x2	x3	x4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	1	0	1	0
7	1	0	1	0

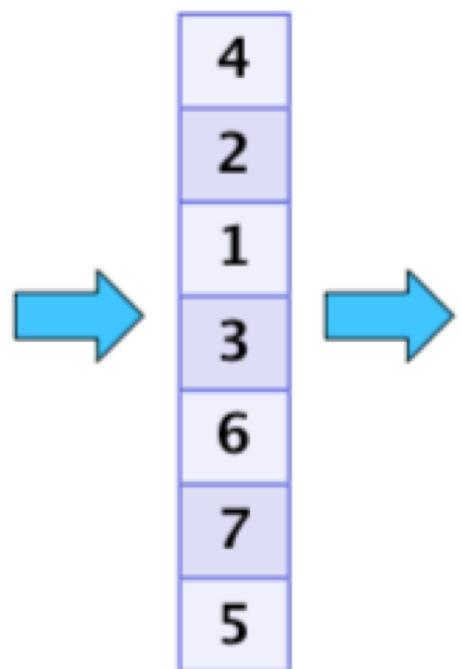


	x1	x2	x3	x4
1	1	0	1	0
3	0	1	0	1
7	1	0	1	0
6	1	0	1	0
2	1	0	0	1
5	0	1	0	1
4	0	1	0	1

1 2 1 2

Exemplo, $i = 2$

	x1	x2	x3	x4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	1	0	1	0
7	1	0	1	0

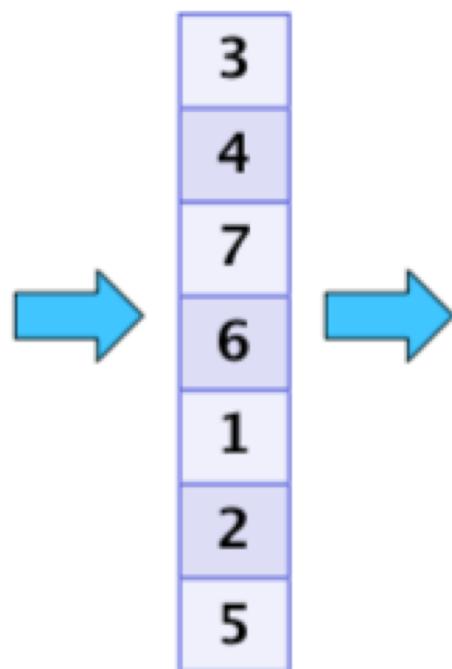


	x1	x2	x3	x4
4	0	1	0	1
2	1	0	0	1
1	1	0	1	0
3	0	1	0	1
6	1	0	1	0
7	1	0	1	0
5	0	1	0	1

2	1	3	1
---	---	---	---

Exemplo, $i = 3$

	x1	x2	x3	x4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	1	0	1	0
7	1	0	1	0



	x1	x2	x3	x4
3	0	1	0	1
4	0	1	0	1
7	1	0	1	0
6	1	0	1	0
1	1	0	1	0
2	1	0	0	1
5	0	1	0	1

3	1	3	1
---	---	---	---

Exemplo, comparando assinaturas

	x1	x2	x3	x4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	1	0	1	0
7	1	0	1	0



x1	x2	x3	x4
1	2	1	2
2	1	3	1
3	1	3	1

	actua	signs
(x1,x2)	0	0
(x1,x3)	0.75	2/3
(x1,x4)	1/7	0
(x2,x3)	0	0
(x2,x4)	0.75	1
(x3,x4)	0	0

Esta é uma solução viável?

Quantas permutações de n elementos existem?

Quantos bits são necessários para representar tal número?

É fácil gerar uma permutação de n elementos (n grande) de forma totalmente aleatória?

Na prática

Substitui-se permutações por funções hash h_1, \dots, h_k
com coeficientes aleatórios

Na prática

$s_i(x) = \text{mínimo de } \{h_i(j), \text{ para todo elemento } j \text{ de } x\}$

	x1	x2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

	x1	x2
1	0	1
2	2	0

$$h(r) = r + 1 \bmod 5$$

$$g(r) = 2r + 1 \bmod 5$$

Como usar essa abordagem para encontrar vizinhos mais próximos?

1 minuto

Como usar essa abordagem para encontrar vizinhos mais próximos?

5 minutos

Verifique quantas vezes dois conjuntos colidem

Várias outras variantes

Várias otimizações

SimHash para distância do cosseno

MinHash generalizada para distância generalizada de Jaccard

Leitura recomendada

<http://www.mmdu.org>

Capítulo 3

Aula baseada em slides de:

<http://www.mmds.org>

<https://www.cs.bu.edu/~evimaria/cs565-13.html>