

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

Aula IV: Encontrando objetos similares – parte II

# 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  $q$

# 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:

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

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  $g(p) = \langle h_1(p), \dots, h_k(p) \rangle$

Pré-processamento:

- 1) Para cada  $x$  no conjunto de dados mapeie  $x$  para os *buckets*  $g_1(x), \dots, g_L(x)$

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  $d(p, q) > r$ ?

# 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

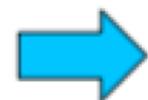
Permute aleatoriamente as linhas da matriz

$h(x)$ : 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

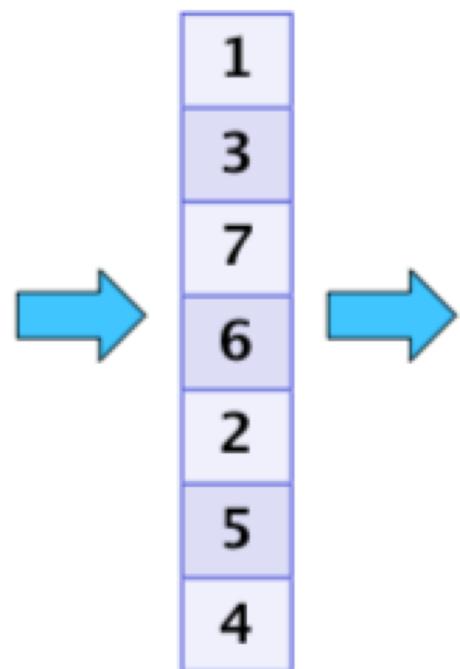
Escolha  $k$  (e.g., 100) permutações das linhas

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

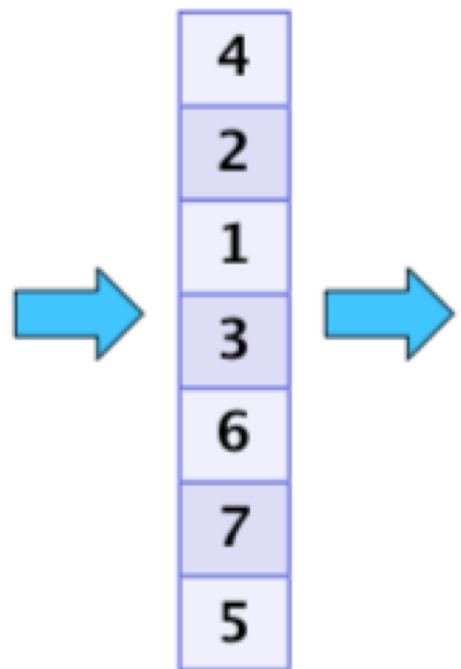


	x1	x2	x3	x4
1	1	0	1	0
3	0	1	0	1
7	1	0	1	0
6	1	0	1	0
2	1	0	0	1
5	0	1	0	1
4	0	1	0	1

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

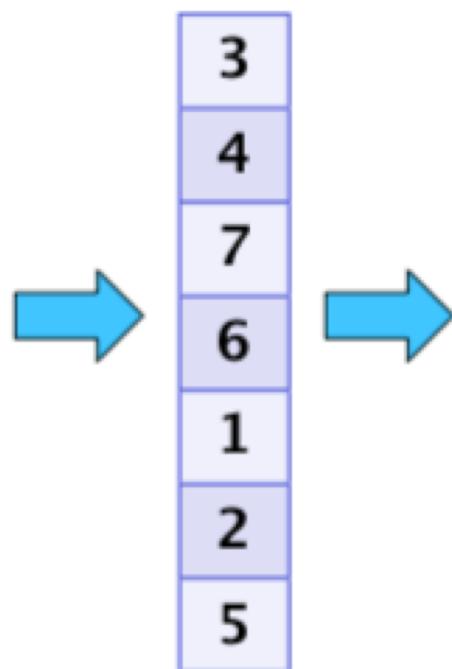


	x1	x2	x3	x4
4	0	1	0	1
2	1	0	0	1
1	1	0	1	0
3	0	1	0	1
6	1	0	1	0
7	1	0	1	0
5	0	1	0	1

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>