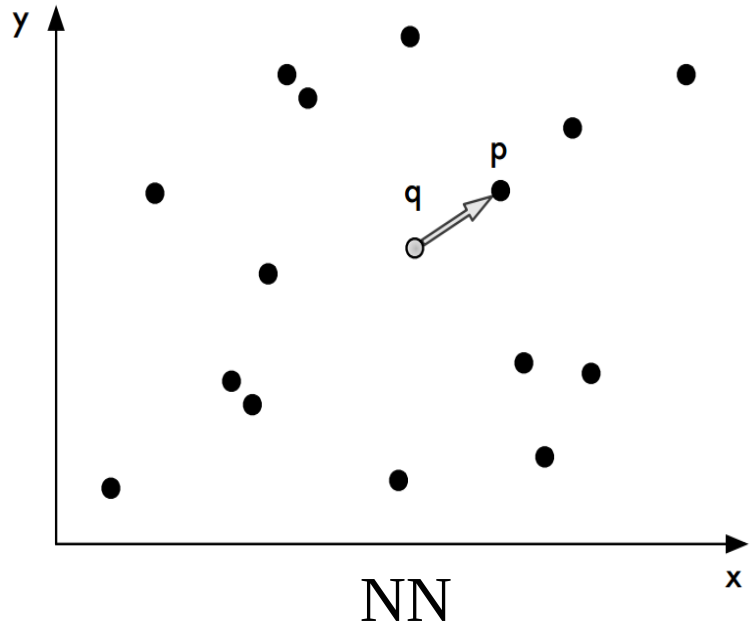
4.LSH

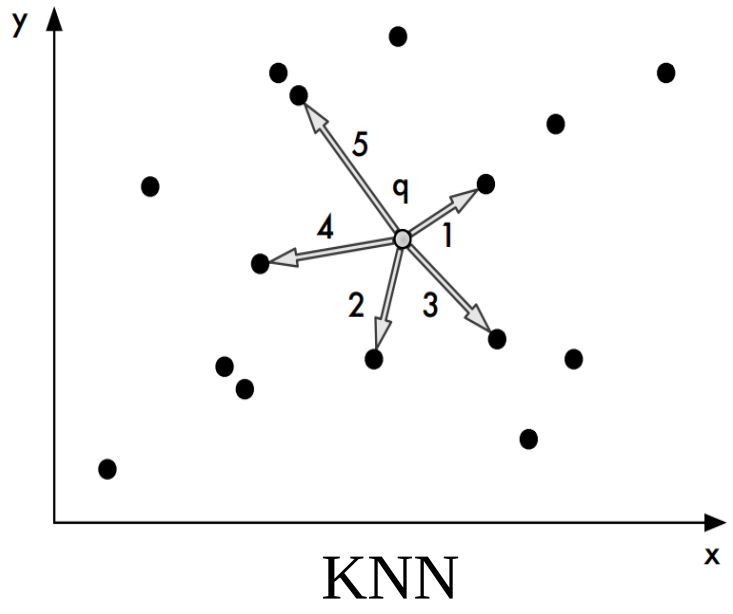
- Hashing的基本思想
- LSH预处理
- 检索算法流程

Locality-sensitive Hashing

- 1. Why use LSH?
- 用Nearest neighbor (NN) 或k-nearest neighbor (KNN)在数据库中检索和输入数据距离最近的1个或k个数据，一般情况下算法复杂度为O(n)（例如暴力搜索），优化情况下可达到O(log n)（例如二叉树搜索），其中n为数据库中的数据量。当数据库很大（即N 很大时），搜索速度很慢。



NN

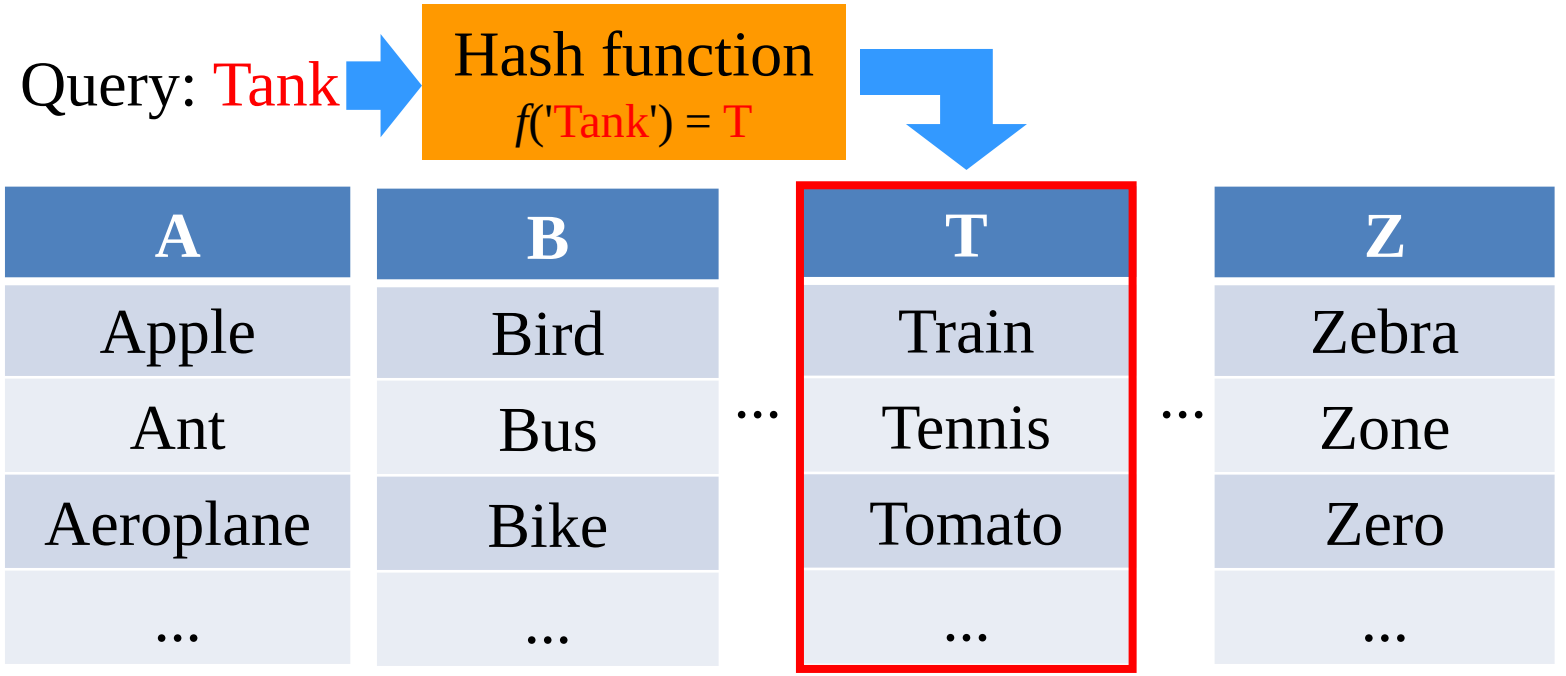


KNN

Hashing的基本思想

- Hashing的基本思想是按照某种规则（Hash函数）把数据库中的数据分类，对于输入数据，先按照该规则找到相对应的类别，然后在其中进行搜索。由于某类别中的数据量相比全体数据少得多，因此搜索速度大大加快。

- 一个查字典的类比：



数据的表示

- 数据(图像、视频、音频等)都表示成一个d维的整数向量

$$\boldsymbol{p}=(p_1,p_2,\cdots p_d)$$

- 其中p_i是整数，满足 $0\le p_i\le C$，这里C是整数的上限。
- 在本实验中，每幅图像用一个12维的颜色直方图$\boldsymbol{p}$表示，构成方式如右图所示。
- 其中 $H_i,i=1,2,3,4$ 是3维颜色直方图。
- **特征向量的量化**
- 上述得到的特征向量 $\boldsymbol{p}=(p_1,\cdots p_{12})$
- 每个分量满足 $0\le p_j\le 1$ 将其量化成
- 3个区间分别用0 1 2表示:

$$p_j=\begin{cases}0,\text{if } 0\le p_j<0.3\\1,\text{if } 0.3\le p_j<0.6\\2,\text{if } 0.6\le p_j\end{cases}$$

也可以用别的量化方法，目的是使0 1 2的分布尽可能平均

- 于是最终得到的特征向量的每个元素满足$p_j\in\{0,1,2\}$

LSH预处理

d维整数向量 $\boldsymbol{p}$可用$d'=d*C$维的Hamming码表示:

$$v(\boldsymbol{p}) = \text{Unary}_C(p_1)\cdots\text{Unary}_C(p_d)$$

其中 $\text{Unary}_C(p_1)$表示C个二进制数，前p1个为1，后C-p1个为0。如当C=10:

$$\text{Unary}_C(5) = 1111100000 \qquad \text{Unary}_C(3) = 1110000000$$

如$\boldsymbol{p}$=(0,1,2,1,0,2)，这里d=6,C=2,于是

$$v(\mathbf{p}) = 00\,10\,11\,10\,00\,11$$

选取集合 $\{1, 2, \ldots, d'\}$ 的L个子集$\{I_i\}_{i=1}^{L}$，定义v($\boldsymbol{p}$)在集合

$$I_i = \{i_1, i_2, \cdots i_m\}: 1 \le i_1 < i_2 < \cdots < i_m \le d'$$

上的投影为 $g_i(\boldsymbol{p}) = p_{i1}p_{i2}\cdots p_{im}$，其中$p_{ij}$为v($\boldsymbol{p}$)的第$i_j$个元素。对于上述$\boldsymbol{p}$，它在{1,3,7,8}上的投影为(0,1,1,0)

哈希函数计算

- 不必显式的将d维空间中的点$\boldsymbol{p}$映射到d'维Hamming空间向量v($\boldsymbol{p}$)。
- I|i表示I中范围在(i-1)*C+1~i*C中的坐标：

$$I = \{1, 3, 7, 8\}, I|1 = \{1\}, I|2 = \{3\},$$
$$I|3 = \phi, I|4 = \{7, 8\}, I|5 = I|6 = \phi$$

- v($\boldsymbol{p}$)在I上的投影即是v($\boldsymbol{p}$)在I|i(i=1,2,…,d)上的投影串联，v($\boldsymbol{p}$) 在I|i上的投影是一串1紧跟一串0的形式，需要求出1的个数：

$$|\{I|i\} - C * (i-1) \leq x_i|$$

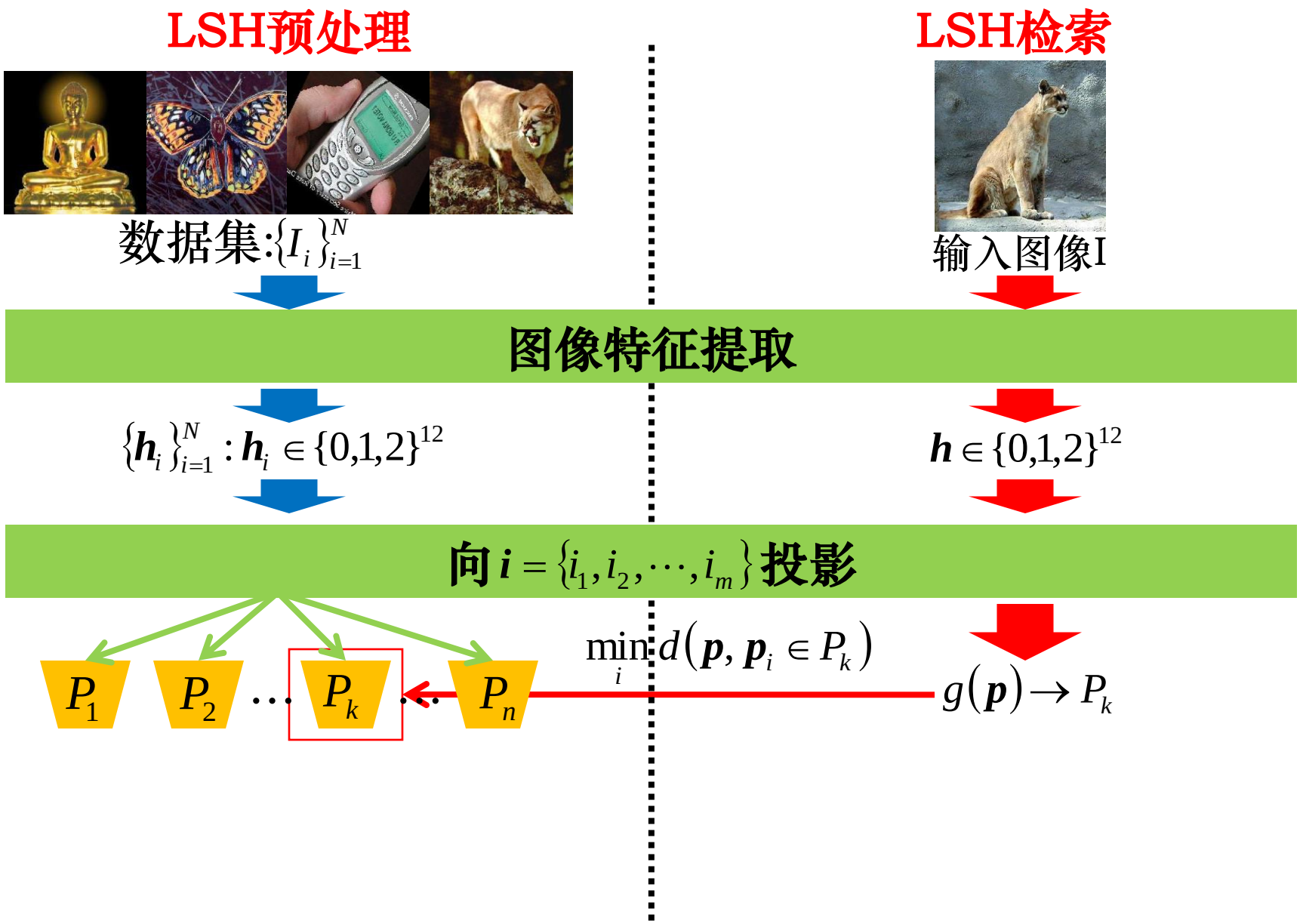
- 比如 $\{I|1\}$ 中小于等于 $x_1 = 0$ 的个数为0，投影：0；
- $\{I|2\} - 2$ 中小于等于 $x_2 = 1$ 的个数为1，投影：1；
- $\{I|4\} - 3 * 2$ 中小于等于 $x_4 = 1$ 的个数为1，投影：10；
- 串联得到：(0,1,1,0)

LSH检索

g($\boldsymbol{p}$)被称作Hash函数，对于容量为N的数据集 $P=\{\boldsymbol{p}_i\}_{i=1}^{N}$，g($\boldsymbol{p}$)可能的输出有n个，n远小于N，这样就将原先的N个数据分成了n个类别，其中每个类别中的数据具有相同的Hash值，不同类别的数据具有不同的Hash值。

对于待检索的输入$\boldsymbol{p}$，先计算g($\boldsymbol{p}$)，找到其对应的类别，然后在该类别的数据集中进行搜索，速度能够大大加快。

检索算法流程


LSH预处理
LSH检索
数据集:$\{I_i\}_{i=1}^N$
输入图像I
图像特征提取
$\{\boldsymbol{h}_i\}_{i=1}^N : \boldsymbol{h}_i \in \{0,1,2\}^{12}$
$\boldsymbol{h} \in \{0,1,2\}^{12}$
向$\boldsymbol{i} = \{i_1, i_2, \cdots, i_m\}$投影
P_1
P_2
P_k
P_n
$\min_i d(\boldsymbol{p}, \boldsymbol{p}_i \in P_k)$
$g(\boldsymbol{p}) \to P_k$

练习

- 利用LSH算法在图片数据库中搜索与目标图片最相似的图片。自行设计投影集合，尝试不同投影集合的搜索的效果。对比NN与LSH搜索的执行时间、搜索结果。

拓展思考

- 本练习中使用了颜色直方图特征信息，检索效果符合你的预期吗？检索出的图像与输入图像的相似性体现在哪里？
- 能否设计其他的特征？