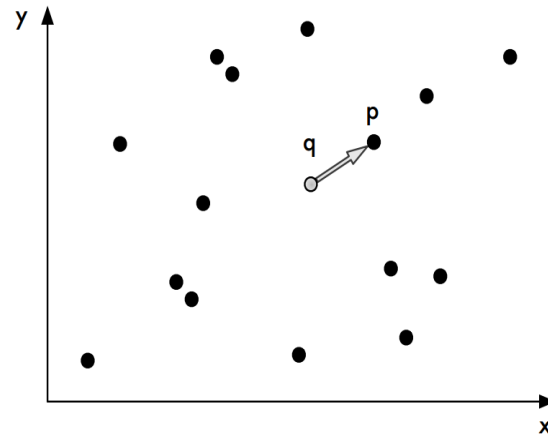


7. LSH

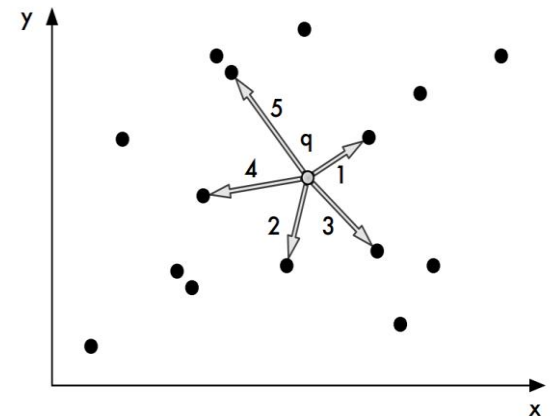
- 基于彩色直方图的图像特征提取
- LSH 预处理
- 检索算法流程

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。
- 而 LSH 算法可以很好的解决这一问题。在通常情况下，用户只需要得到被检索图片的一个近似即可，即在数据库中和输入数据距离最近的 1 个或 k 个数据。考虑到采用哈希可以将检索的开销降低至 $O(1)$ 左右，只需增大相似图片发生哈希冲突的概率，即可在检索时优先检索到这些图片获得令人满意的结果。基于此，Aristides Gionis 等提出了一种可用于高维度数据检索的 LSH 算法。
- [Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing.](#)



NN



KNN

基于彩色直方图提取图像特征

图像特征提取

- 数据（图像、视频、音频等）都可以表示成一个 d 维的整数向量 $\mathbf{p} = (p_1, p_2, \dots, p_d)$ 。
其中 p_i 是整数，满足 $0 \leq p_i \leq C$ ，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 \mathbf{p} 表示，构成方式如图所示。
其中 $H_i, i = 1, 2, 3, 4$ 是 3 维颜色直方图。其可以表现各种颜色在图像各部分的分布情况，进而反映图像的特征。

特征向量量化

- 对于得到的特征向量，每个分量满足 $0 \leq p_j \leq 1$ 。
将其按照以下规则量化，即将每个分量分别用 0, 1, 2 表示：

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

- 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。



Locality-Sensitive Hashing

Fact 1

- d 维整数向量 \mathbf{p} 可用 $d'=d*C$ 维的 Hamming 码表示: $v(\mathbf{p}) = \text{Unary}_C(p_1) \cdots \text{Unary}_C(p_d)$ 。
- 其中 $\text{Unary}_C(p_1)$ 表示 C 个二进制数, 前 p_1 个为 1, 后 $C-p_1$ 个为 0。
如当 $C = 10$: $\text{Unary}_C(5) = 1111100000$; $\text{Unary}_C(3) = 1110000000$ 。
- 如 $\mathbf{p} = (0, 1, 2, 1, 0, 2)$, 这里 $d = 6$, $C = 2$, 于是 $v(\mathbf{p}) = 00\ 10\ 11\ 10\ 00\ 11$ 。

Fact 2

- 选取集合 $\{1, 2, \dots, d'\}$ 的 L 个子集 $\{I_i\}_{i=1}^L$, 定义 $v(\mathbf{p})$ 在集合 $I_i = \{i_1, i_2, \dots, i_m\}: 1 \leq i_1 < i_2 < \dots < i_m \leq d'$ 上的投影为 $g_i(\mathbf{p}) = p_{i_1}p_{i_2} \cdots p_{i_m}$, 其中 p_{i_j} 为 $v(\mathbf{p})$ 的第 i_j 个元素。
对于上述 \mathbf{p} , 它在 $\{1, 3, 7, 8\}$ 上的投影为 $(0, 1, 1, 0)$ 。

Locality-Sensitive Hashing - 计算方法

- 值得注意的是，一般不必显式的将 d 维空间中的点 \mathbf{p} 映射到 d' 维 Hamming 空间向量 $v(\mathbf{p})$ 。
- 令 $\{I|i\}$ 表示 I 中范围在 $(i-1)*C+1 \sim i*C$ 中的坐标，则 $v(\mathbf{p})$ 在 I 上的投影即是 $v(\mathbf{p})$ 在 $\{I|i\} (i = 1, 2, \dots, d)$ 上的投影串联。 $v(\mathbf{p})$ 在 $\{I|i\}$ 上的投影是一串 1 紧跟一串 0 的形式，要求出 1 的个数，并进行串联。
- 如 $I = \{1, 3, 7, 8\}$ ，则 $\{I/1\} = \{1\}$ ， $\{I/2\} = \{3\}$ ， $\{I/3\} = \emptyset$ ， $\{I/4\} = \{7, 8\}$ ， $\{I/5\} = \emptyset$ ， $\{I/6\} = \emptyset$ 。
 $\{I/1\}$ 中小于等于 $p_1 = 0$ 的个数为 0，投影：0；
 $\{I/2\}$ - 2 中小于等于 $p_2 = 1$ 的个数为 1，投影：1；
 $\{I/4\}$ - $3 * 2$ 中小于等于 $p_4 = 1$ 的个数为 1，投影：10；
串联得到：(0, 1, 1, 0)。

LSH 预处理

- 首先, 生成 l 个 Hash Table。最简单的生成 l 个 Hash Table 的方式即为生成 l 个 Hash 函数, 即 $g(\mathbf{p})$ 。
- 对于容量为 N 的数据集 $P = \{\mathbf{p}_i\}_{i=1}^N$, $g(\mathbf{p})$ 可能的输出有 n 个, 其中 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。由于采用了 LSH, 具有类似的特征的数据发生冲突的概率更大。
- 在遇到冲突时, 论文选择不插入对应的哈希表。这是由于当两个点距离很近时, 其冲突的概率非常高, 即会在多个哈希表中均发生冲突。放弃在某个哈希表中插入不会对最终结果产生显著影响, 比较邻近的点仍然会被找到。
- 具体的算法见伪代码。

Algorithm Preprocessing

Input A set of points P ,

l (number of hash tables),

Output Hash tables $\mathcal{T}_i, i = 1, \dots, l$

Foreach $i = 1, \dots, l$

Initialize hash table \mathcal{T}_i by generating
a random hash function $g_i(\cdot)$

Foreach $i = 1, \dots, l$

Foreach $j = 1, \dots, n$

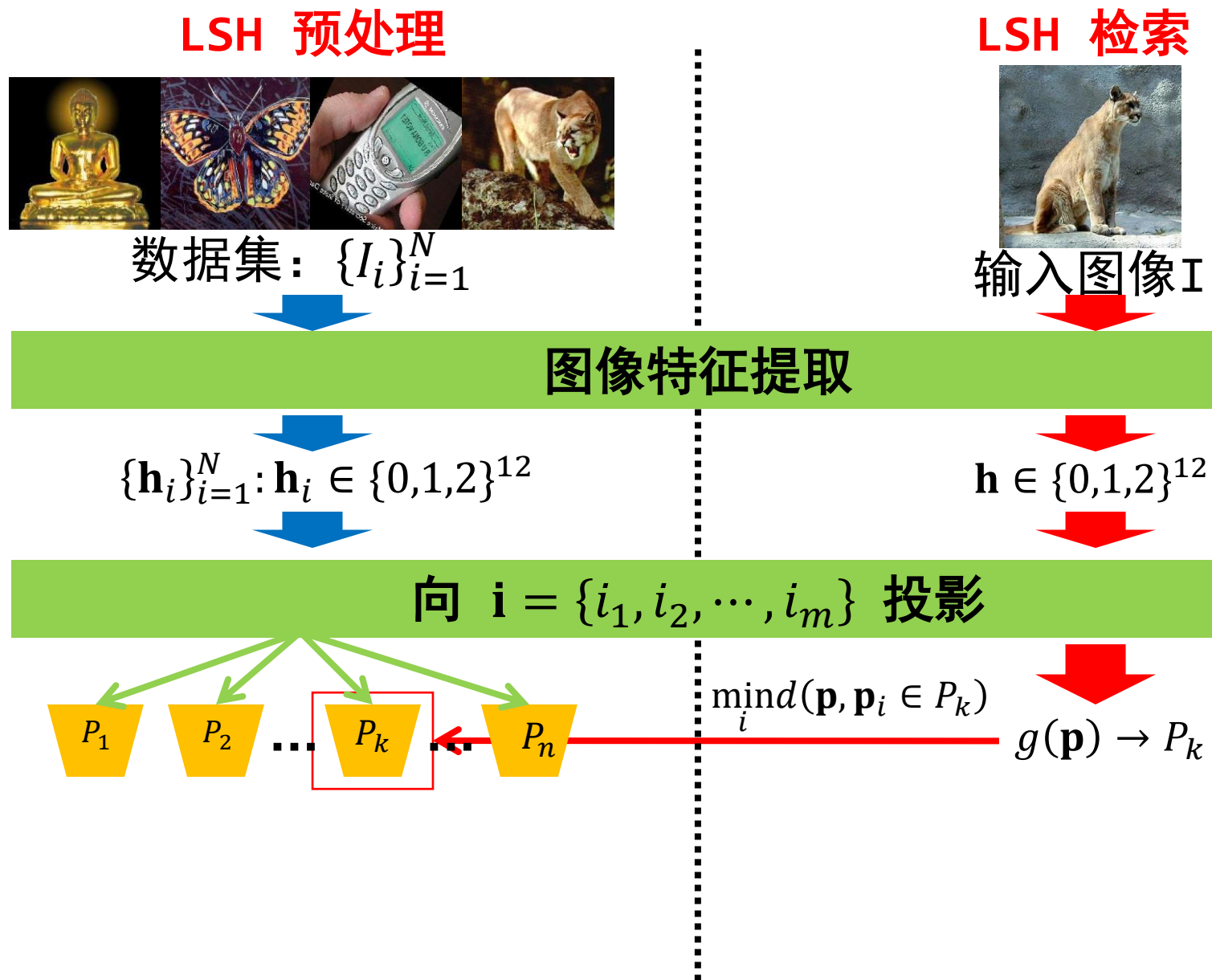
Store point p_j on bucket $g_i(p_j)$ of hash table \mathcal{T}_i

LSH 检索

- 对于目标特征向量，分别调用 l 个哈希函数，从 l 个哈希表中找到与之冲突的向量。并从中找出距离最近的 k 个点（本实验中令 $k = 1$ ）。其中，距离最近的点的标签即为特征向量的类别。
- 根据论文作者的测试，当特征函数为颜色直方图时，使用的距离函数可以为 l_1 -norm，也可以为 l_2 -norm。其中 l_1 -norm 的检索效果略优于 l_2 -norm，但差别不大。
- 具体的算法见伪代码。

```
Algorithm Approximate Nearest Neighbor Query  
Input A query point  $q$ ,  
        $K$  (number of appr. nearest neighbors)  
Access To hash tables  $\mathcal{T}_i, i = 1, \dots, l$   
       generated by the preprocessing algorithm  
Output  $K$  (or less) appr. nearest neighbors  
 $S \leftarrow \emptyset$   
Foreach  $i = 1, \dots, l$   
     $S \leftarrow S \cup \{\text{points found in } g_i(q) \text{ bucket of table } \mathcal{T}_i\}$   
Return the  $K$  nearest neighbors of  $q$  found in set  $S$   
/* Can be found by main memory linear search */
```

LSH 流程示意



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。

拓展与思考

- 本实验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？