



北京航空航天大学
BEIHANG UNIVERSITY



哈希比特选择

刘祥龙

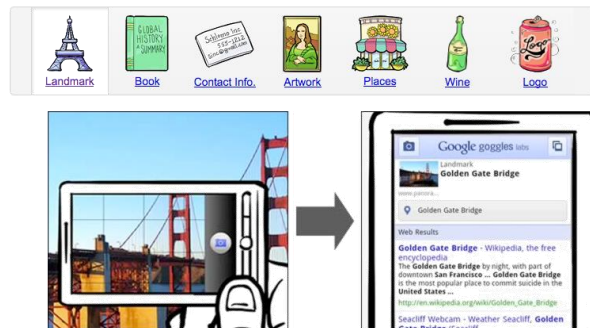
北京航空航天大学计算机学院

Joint work with **Dr. Junfeng He** and **Prof. Shih-Fu Chang**, Columbia Univeristity

背景



网页搜索



图片搜索

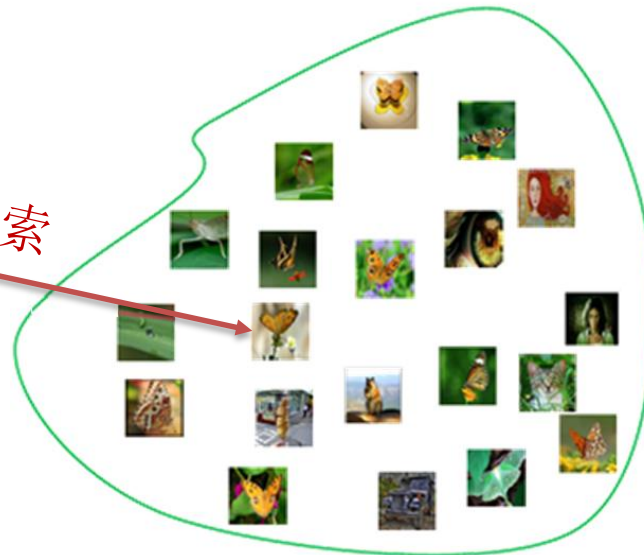


内容推荐



查询

最近邻搜索



数据库

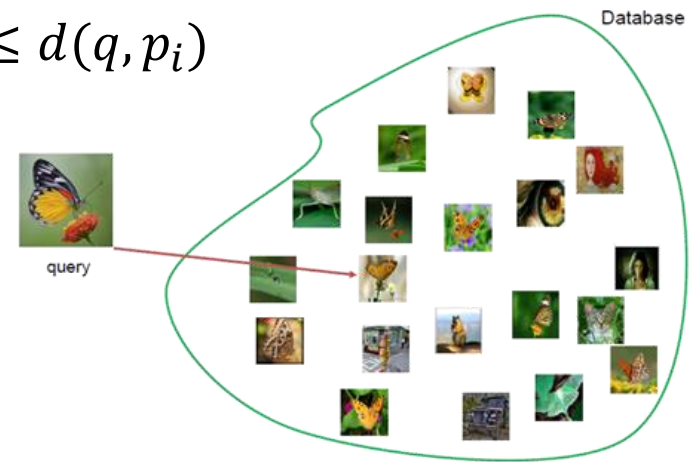
最近邻搜索

□ **定义：** 给定数据集 $P = \{p_i\}_{i=1\dots n}$ 及查询 q , q 的最近邻为：

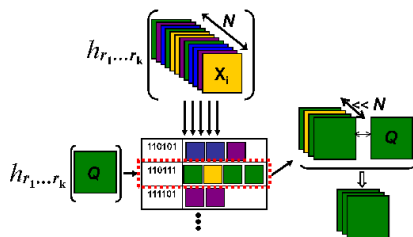
$$p^* \in P, \text{ 满足 } d(q, p^*) \leq d(q, p_i)$$

通常放松为近似最近邻

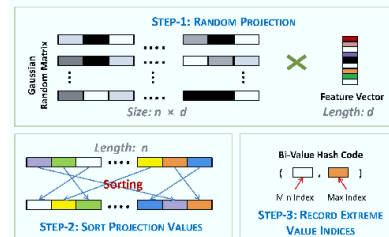
$$\hat{p}^* \in P, \text{ 满足 } d(q, \hat{p}^*) \leq (1 + \epsilon)d(q, p^*)$$



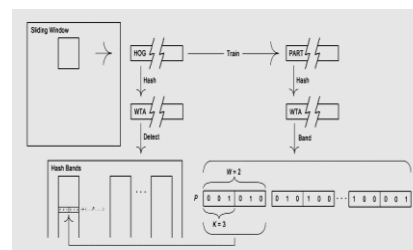
□ **应用：** 信息检索、大规模优化、物体检测、新闻推荐等



(Kulis et al., 2009)



(Mu et al., 2012) (Liu et al., 2012)



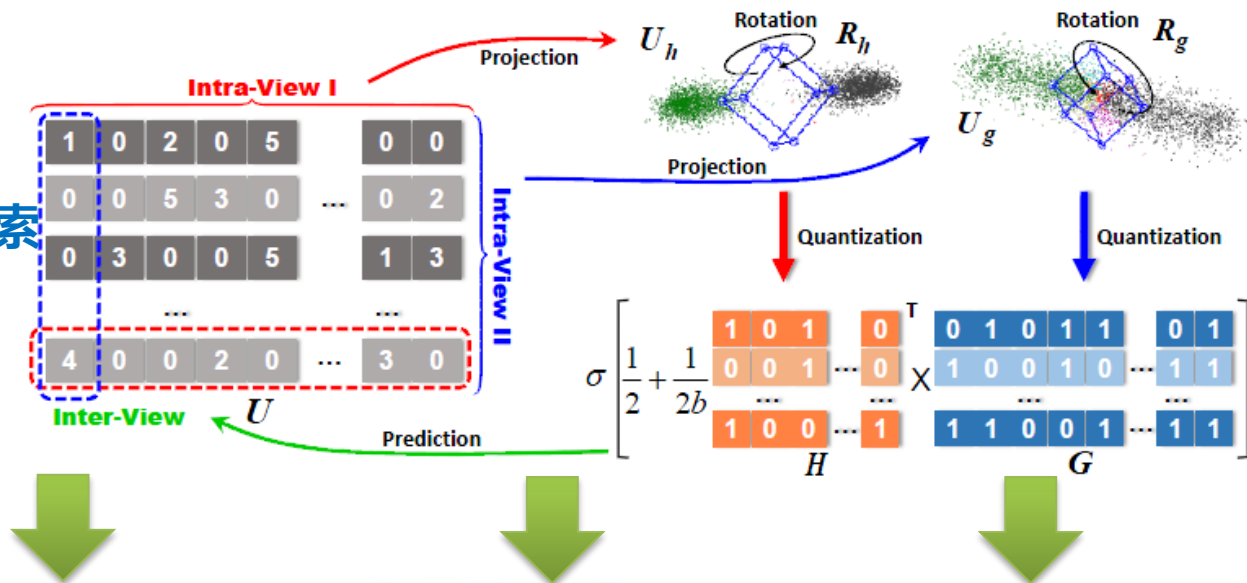
(Dean et al., 2013)



(Das et al., 2007)

MOOC平台下内容检索及推荐

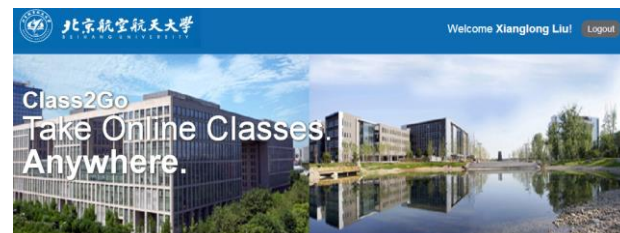
跨域最近邻搜索
用户-内容



课程内容检索



用户群组推荐



推荐课程

推荐教材



课程内容推荐

研究背景：最近邻搜索

□ 解决方案

— 线性扫描

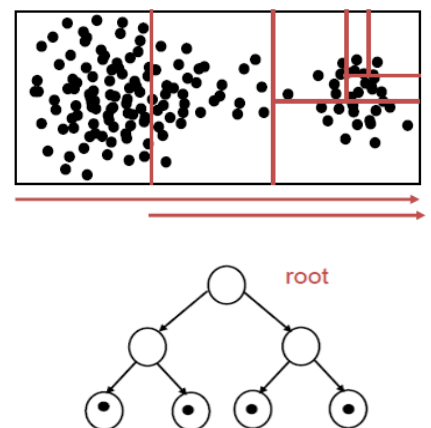
- 1B数据集，特征维度10K
- 存储40TB，线性扫描15小时

— 基于树结构：KD-tree, VP-tree, etc.

- 采用树结构存储
- 搜索时分而治之
- 处理高维特征时降为线性扫描

— 基于位置敏感哈希

- 采用哈希编码存储
- 搜索时计算编码相似度



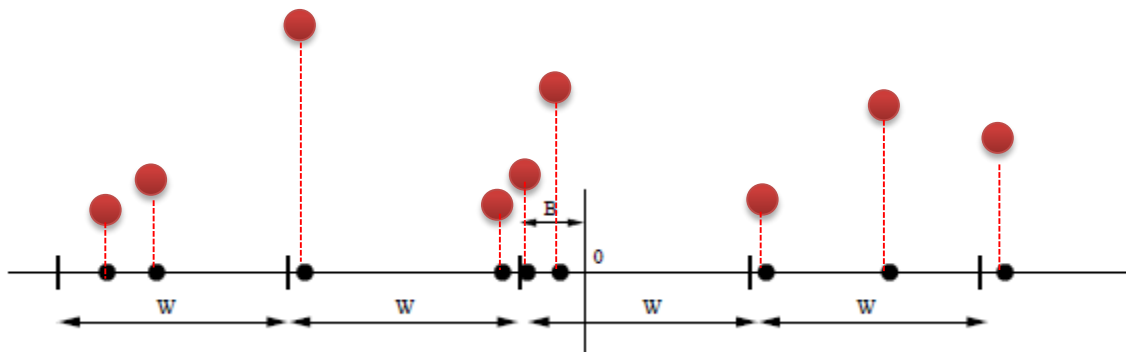
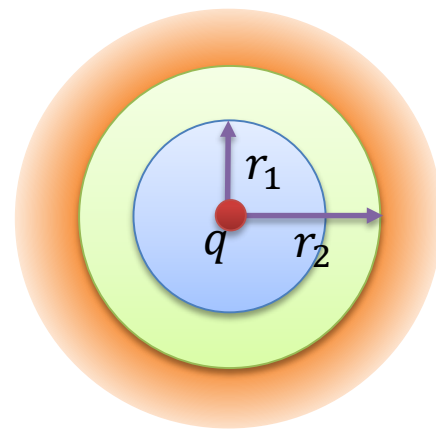
位置敏感哈希

□ **定义：** 给定距离度量 d ，哈希函数 $H = \{h : S \rightarrow U\}$ 称为

$(r_1; r_2; p_1; p_2)$ –敏感，对任意的 $q; p \in S$ 满足：

- 如果 $d(p; q) \leq r_1$ ，那么 $\Pr_H[h(q) = h(p)] \geq p_1$ ；
- 如果 $d(p; q) > r_2$ ，那么 $\Pr_H[h(q) = h(p)] \leq p_2$ 。

概率 $p_1 > p_2$ 并且距离阈值 $r_1 < r_2$ 。

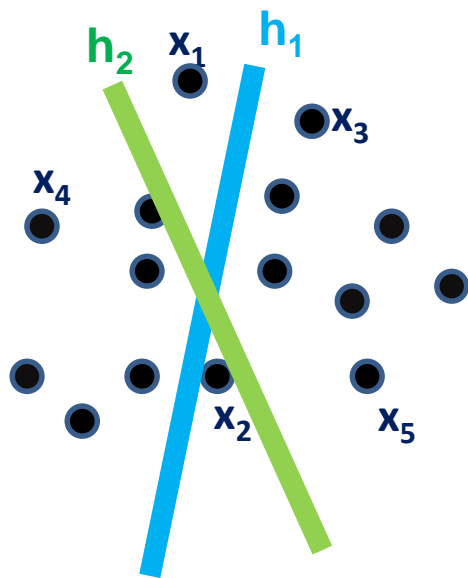


基于哈希的最近邻搜索

□ **原理**：相似数据的编码相似，保持了数据原始的相似性

□ **优势**：存储效率高、检索速度快

$$h(x) = \text{sgn}(w^T x + b)$$



X	x ₁	x ₂	x ₃	x ₄	x ₅
h ₁	0	1	1	0	1
h ₂	1	0	1	0	1
...
h _k
	010...	100...	111...	001...	110...

最近邻搜索方式

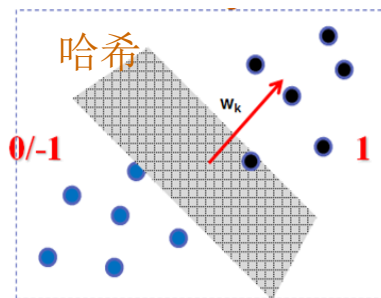
x_1	x_2	x_3	x_4	x_5	...	x_n
0010...	0111...	0110...	0101...	0100...	...	1101...

距离= 2 4 0 5 1 3

海明距离排序

命中结果

命中结果



查询数据

哈希表查询

命中结果

哈希表1

哈希桶	索引图像
0010...	
0110...	
⋮	⋮
1111...	

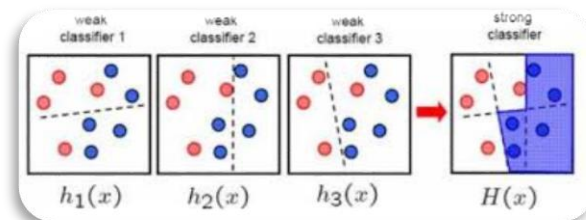
哈希表L

命中结果

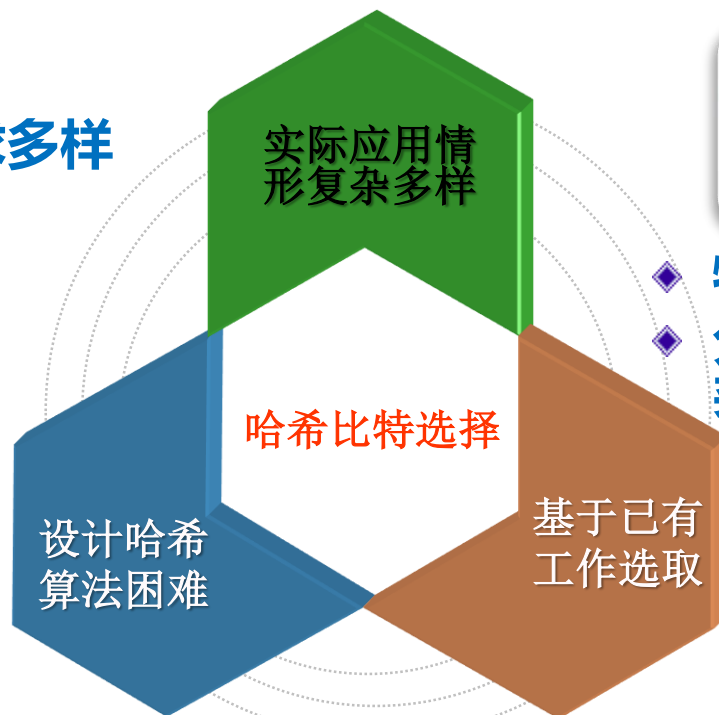
哈希桶	索引图像
0010...	
0110...	
⋮	⋮
1111...	

问题

- ◆数据类型复杂
- ◆应用场景及需求多样



特征选择：选取有效特征
分类器组合：组合已有分类器



- ◆针对应用需要定制或设计新的哈希算法



多特征哈希

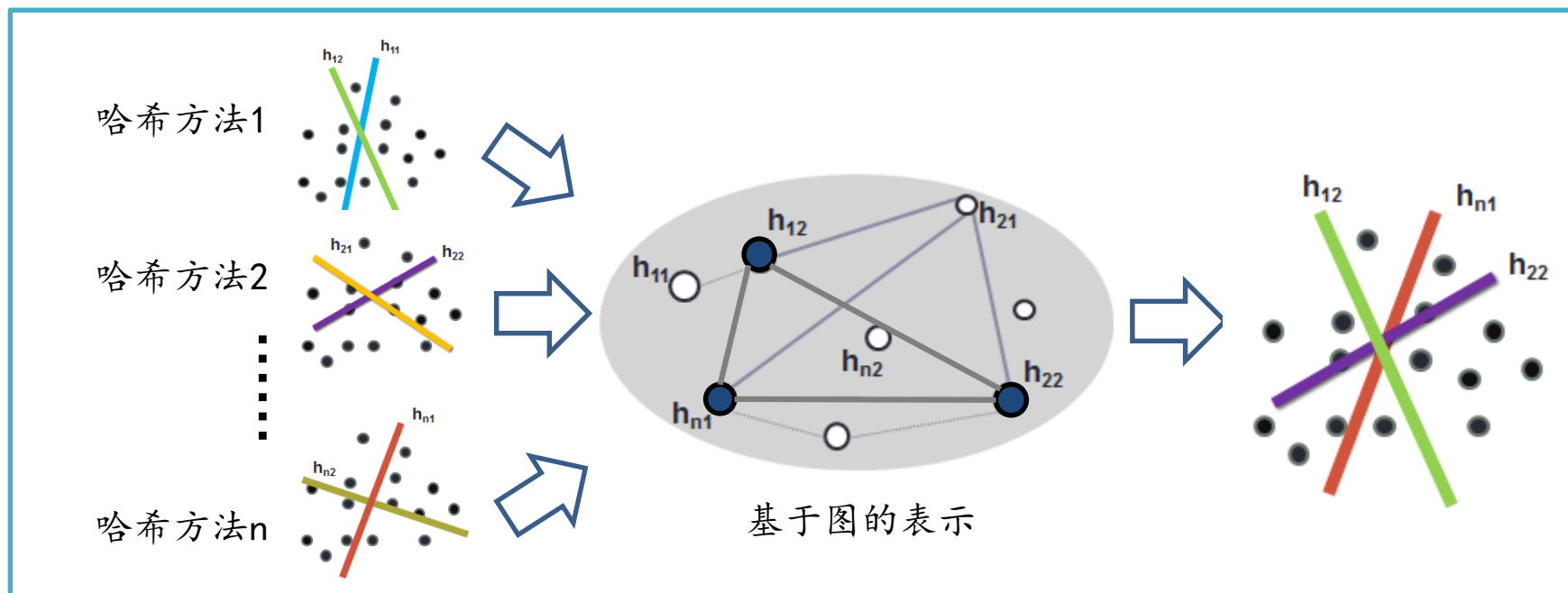


蝴蝶
树叶
鲜花
多标签哈希

哈希比特选择


□思路：建立哈希函数池，依据应用需求选择最优的函数子集

- 可使用不同哈希方法、参数、特征等来生成候选函数
- 依据应用需求，可定制不同的选择标准



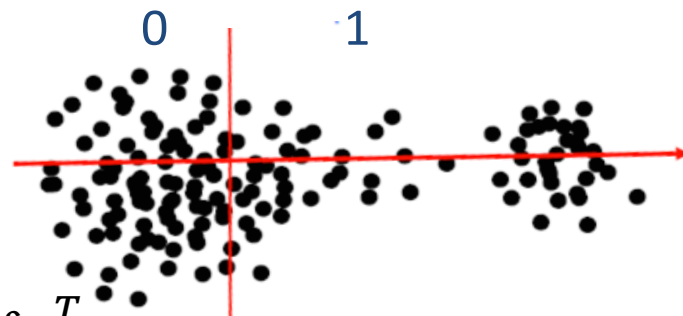
哈希比特选择

□ 问题描述

- **哈希函数池**: B 个哈希函数 $H = \{h_1, h_2, \dots, h_B\}$ ($h_i: R^d \rightarrow \{-1, 1\}$) 索引为 $V = \{1, 2, \dots, B\}$
- **数据集**: $X = [x_1, x_2, \dots, x_N] \in R^{d \times N}$
- **哈希比特**: $Y_i = h_i(X) = [h_i(x_1), h_i(x_2), \dots, h_i(x_N)] \in \{-1, 1\}^N$
 y_i 的 N 次抽样
随机二元向量 $y: y_i = h_i()$
- **目标**: 选择对应用有效的 l 个哈希函数

比特选择标准

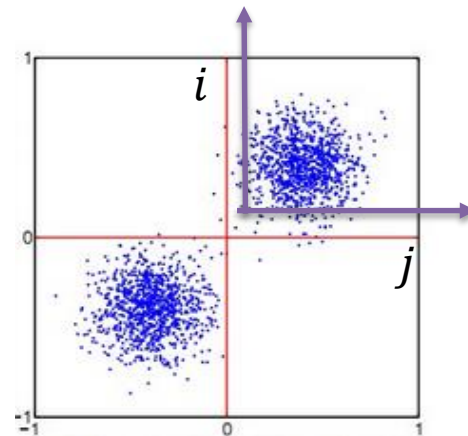
□ 个体质量：最近邻保持能力



第*i*个哈希函数的质量 $\pi_i = e^{-\gamma y_i \mathcal{L} y_i^T}$

□ 整体质量：相互独立不冗余

第*i*个和*j*个哈希函数的独立性 $a_{ij} = e^{-\lambda \text{MI}_{ij}}$



第*i*个和*j*个哈希函数的互信息 $\text{MI}_{ij} = \sum_{y_i, y_j} p(y_i, y_j) \log \frac{p(y_i, y_j)}{p(y_i)p(y_j)}$

比特选择算法

□选择方法：选择既能保持数据相似关系又相互独立的一组哈希函数

从 B 个哈希函数中，寻找期望的大小为 l 的子集：

$$\begin{aligned} \max_z \quad & \frac{1}{2} z^T \hat{A} z \\ \text{s. t.} \quad & z \in \{0, 1\}^B, |z|_0 = l \end{aligned}$$

z^* ：标识选中的哈希比特

\hat{A} ：融合相似度保持能力 π 和独立性 A ，表征内聚性

$\frac{1}{2} z^T \hat{A} z$ ：度量了选中哈希比特之间的内聚性

比特选择算法

□ 近似求解：放松离散约束

从 B 个哈希函数中，寻找期望的大小为 l 的子集：

$$\begin{aligned} & \max_z \frac{1}{2} z^T \hat{A} z \\ \text{s.t. } & z \in R^B, z \geq 0, |z|_1 = 1 \end{aligned}$$

二次规划

z^* ：候选哈希比特的重要性（得分）

近似解：得分最大的 l 个哈希比特，最大化选中哈希函数的内聚性

\hat{A} ：融合相似度保持能力 π 和独立性 A

非负性

对称性

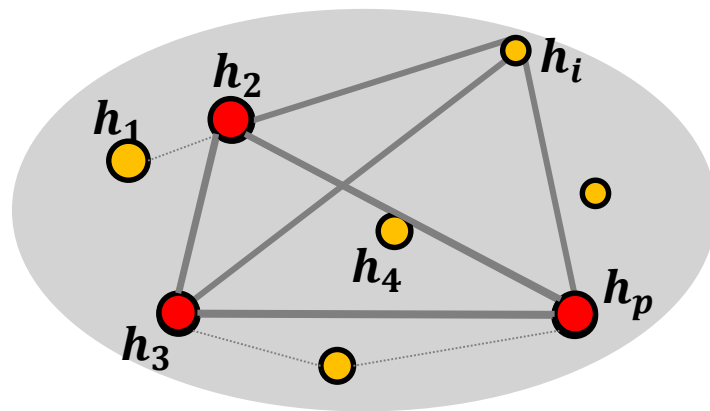
单调性

$$\hat{A} = \Pi A \Pi, \Pi = \text{diag}(\pi)$$

理论分析

候选哈希比特图 $G = (V, E, A, \pi)$

- $V = \{h_1, \dots, h_p\}$: 节点集合, 对应候选哈希比特
- E : 边集合, 对应候选哈希比特相互关联
- $\pi = [\pi_1, \dots, \pi_p]^T$: 节点权重, 对应哈希比特最近邻保持能力
- $A = (a_{ij})$: 边权重, 对应哈希比特之间的独立性

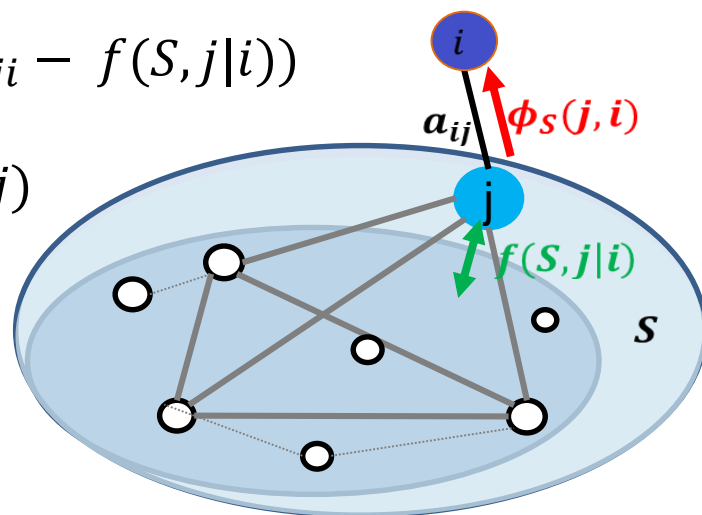


比特选择可视为节点和边加权图 G 上稠密子集发现：子图应该包含较大权重的节点和边，即高度的内部同质性

理论分析

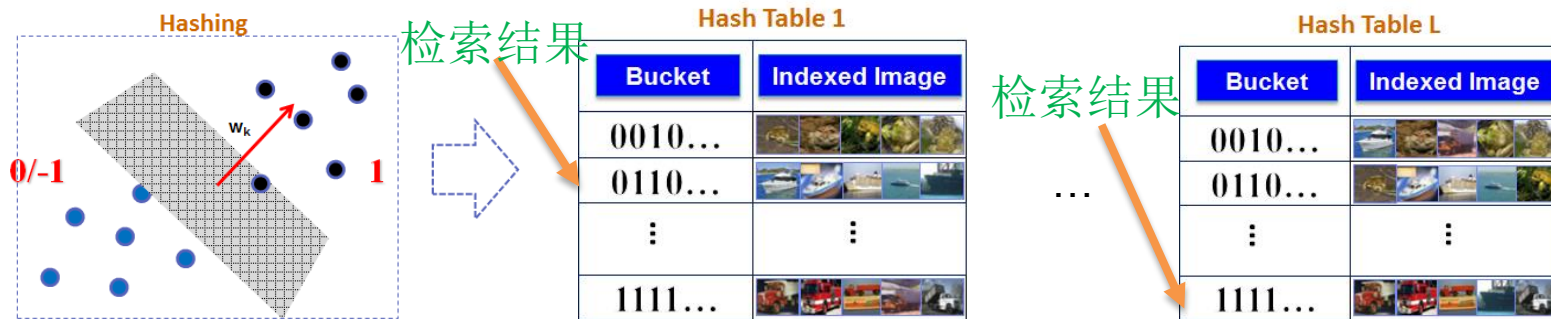
□ 正则主导集合

- 外部连接 (Markov 跳变) $\varphi_S(j, i) = \frac{\pi_j}{\pi_i} (a_{ji} - f(S, j|i))$
- 诱导权重 $w_S(i) = \sum_{j \in S \setminus \{i\}} \varphi_{S \setminus \{i\}}(j, i) w_{S \setminus \{i\}}(j)$
- 正则主导集合 $\{i | i \in S, w_S(i) > 0\}$
- 对应的哈希函数称为**主导哈希函数**



定理: 如果 z^* 是采用 $\hat{A} = \Pi A \Pi$ ($\Pi = \text{diag}(\pi)$) 的二次规划问题的严格局部解, 那么它的支持集 $\sigma = \sigma(z)$ 是图 $G = (V, E, A, \pi)$ 的正则主导集合, 假设 $w_{\sigma \cup \{i\}}(i) > 0$ 对所有 $i < \sigma$ 成立。

多哈希表构造



目标：构造 L 个互补哈希表 $\{T_l; l = 1, \dots, L\}$, 每一个表包含来自 H 的 K 个哈希函数 $T_l = \{h_{l_1}, \dots, h_{l_K}; \{l_1, \dots, l_K\} \in V\}$.

传统方法：顺序选取或随机选取

通常只针对一种哈希算法及特征

忽略了哈希表之间的关系

基于哈希比特选择的方法

可支持多种哈希算法、特征等

考虑多哈希表间的互补性

哈希函数选择标准

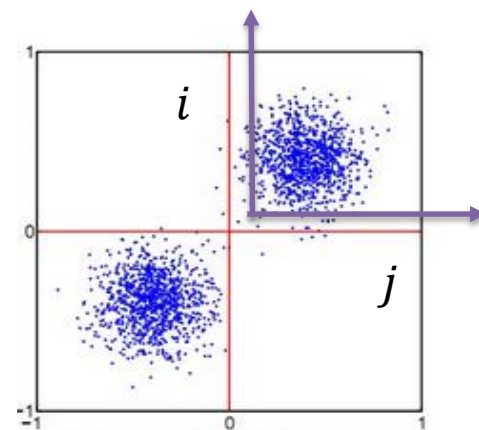
□ 个体质量：最近邻保持能力，同构/异构近邻预测准确率

第 i 个哈希函数的质量 $\pi_i = e^{\gamma y_i S y_i^T}$ $S_{ij} = \begin{cases} 1, & \text{if } (i, j) \in \mathcal{M} \\ -1, & \text{if } (i, j) \in \mathcal{C} \\ 0, & \text{otherwise} \end{cases}$

□ 整体质量：相互独立不冗余

第 i 个和 j 个哈希函数的独立性 $a_{ij} = e^{-\lambda \text{MI}_{ij}}$

第 i 个和 j 个哈希函数的互信息 $\text{MI}_{(ij)} = \sum_{y_i, y_j} p(y_i, y_j) \log \frac{p(y_i, y_j)}{p(y_i)p(y_j)}$



直接构造方法

顺序从候选的哈希函数池 H 中选择主导哈希函数构成当前哈希表

Algorithm 1 Hash Table Construction using Dominant Hash Functions (DHF).

- 1: **Input:** data \mathbf{X} , hash function set \mathcal{H} , similarity \mathbf{S} ;
 - 2: **Initialize:** $\mathbf{Y}_i \leftarrow h_i(\mathbf{X}), i = 1, \dots, B$;
 - 3: Compute weights π and \mathbf{A} by Eq. (3) and Eq. (4);
 - 4: **for** $l = 1$ to L **do**
 - 5: Optimize \mathbf{z}^* of problem (5) with respect to \mathcal{H} ;
 - 6: Set $\sigma \leftarrow \text{topK}(\mathbf{z}^*)$ and $\mathcal{V} \leftarrow \{1, \dots, |\mathcal{H}|\} \setminus \sigma$;
 - 7: Set $\mathcal{T}_l \leftarrow \{h_i : h_{i \in \sigma} \in \mathcal{H}\}$;
 - 8: Update $\mathcal{H} \leftarrow \mathcal{H} \setminus \mathcal{T}_l$ and $\mathbf{A} \leftarrow \mathbf{A}_{\mathcal{V}}$;
 - 9: **end for**
 - 10: **Output:** hash tables $\mathcal{T}_l, l = 1, \dots, L$.
-

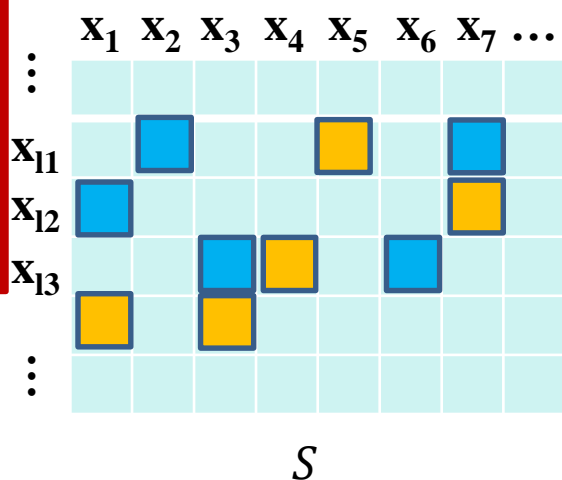
互补哈希表构造思路

□ 互补哈希表：最近邻能够在其中至少一个哈希表中找到

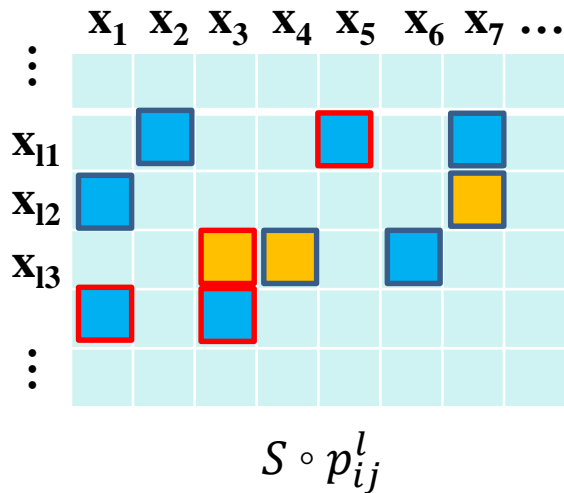
□ Boosting方法：通过调整最近邻对的权重，使当前构造的哈希表纠正前面哈希表的预测错误

■ > 0 ■ < 0

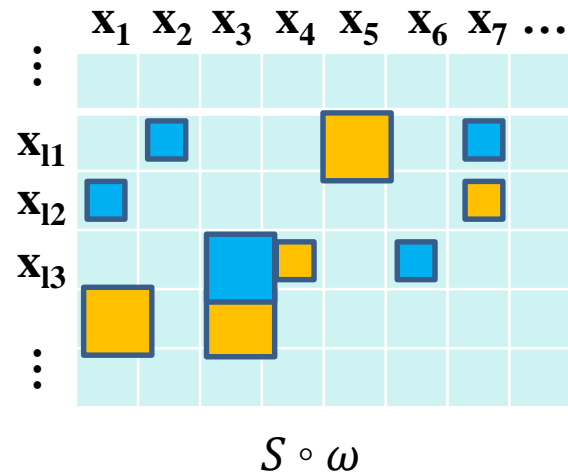
近邻相似矩阵



预测及错误



更新近邻相似矩阵



互补构造方法

采用Boosting方式：顺序从候选的哈希函数池 H 中选择能够正确预测前面错分的最近邻关系的主导哈希函数构成当前哈希表

(1) 预测最近邻关系: 当前 l 个哈希表在近邻对 x_i 和 x_j :

$$\underline{\mathbf{p}_{ij} = \mathbf{d}_{ij}^l - \mu,}$$

$$\mathbf{d}_{ij}^l = \min_{i=1, \dots, l} \sum_{h \in \mathcal{T}_i} \|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|^2$$

(2) 更新最近邻相似度: 错分的最近邻对的权重将被放大, 相反正确的权重将缩小

$$\underline{\mathbf{S}_{ij} = \mathbf{S}_{ij} \omega_{ij}}$$

$$\omega_{ij} = \begin{cases} \exp(-\alpha \mathbf{p}_{ij}), & \text{if } (i, j) \in \mathcal{M} \\ \exp(\alpha \mathbf{p}_{ij}), & \text{if } (i, j) \in \mathcal{C} \\ 0, & \text{otherwise} \end{cases}$$

实验对比方法-哈希比特选择

□ 比特选择的对比算法

– 朴素方法

- 随机方法 (Random)
- 贪婪策略 (Greedy)：仅考虑单个比特质量
- 主导集合方法 (DomSet) (Pavan et al., 2007)：边加权图的稠密子图发现，仅考虑比特之间的独立性

– 可配置哈希方法 (LRH) (Mu et al., 2011)

- 最大化所选比特的近邻结构保持能力

□ 比特生成的哈希算法

- 线性哈希：LSH、PCAH、PCAR、ITQ、RMMH
- 非线性哈希：SPH (Heo et al., 2012)
- 多特征哈希：MFH (Song et al., 2011)
- 两比特哈希：HH (Liu et al., 2011), DBH (Kong et al., 2012)

实验数据集

□ 语义最近邻

- 多类别数据：CIFAR-10（60K）

384维GIST + 300维SIFT BoW

- 多标签数据：NUS-WIDE（270K）

128维小波纹理特征 + 225维颜色矩特征

□ 空间距离最近邻

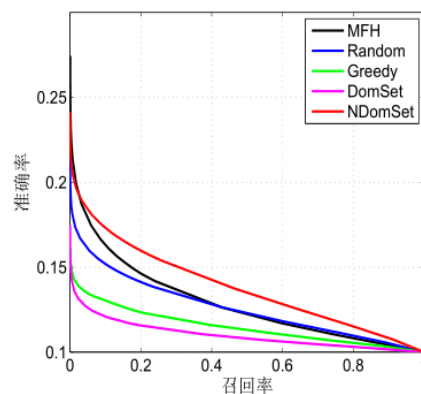
- GIST-1M（1M）：960维GIST特征
- SIFT-1M（1M）：128维SIFT特征
- 欧式距离排序的前5%作为最近邻

实验结果-基本哈希算法

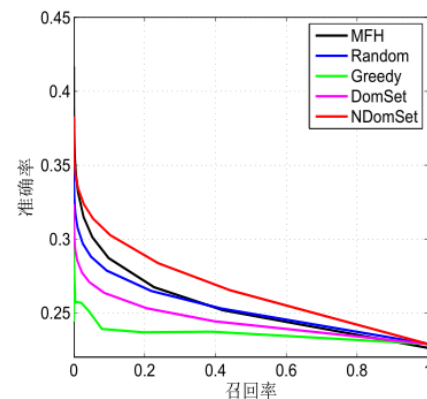
单特征

500 候选比特		32 比特	64 比特	128 比特
LSH	LRH	3.71	6.93	11.27
	RANDOM	3.83±0.13	6.88±0.24	11.15±0.33
	GREEDY	3.19±0.22	5.18±0.17	8.84±0.15
	DOMSET	1.94±0.06	4.13±0.13	8.31±0.17
	NDOMSET	5.14±0.11	8.22±0.20	12.07±0.19
PCAR	LRH	3.89±0.20	7.02±0.24	11.53±0.23
	RANDOM	4.24±0.38	6.98±0.43	11.81±0.20
	GREEDY	3.41±0.13	5.59±0.25	9.47±0.17
	DOMSET	1.97±0.15	4.21±0.24	8.77±0.17
	NDOMSET	5.48±0.30	8.93±0.33	12.44±0.24
ITQ	LRH	4.24	7.47	11.27
	RANDOM	4.38±0.20	7.47±0.20	11.15±0.33
	GREEDY	3.74±0.14	7.47±0.20	11.15±0.33
	DOMSET	4.21±0.23	7.47±0.20	11.15±0.33
	NDOMSET	5.20±0.24	8.30±0.1	12.07±0.21
SPH	LRH	5.61	10.09	16.24
	RANDOM	6.05±0.33	9.65±0.66	15.67±0.62
	GREEDY	4.39±0.28	9.87±0.50	16.64±0.42
	DOMSET	3.23±0.14	6.25±0.34	11.26±0.37
	NDOMSET	7.13±0.24	11.71±0.39	17.07±0.38
RMMH	LRH	5.61	10.65	17.05
	RANDOM	5.81±0.26	10.75±0.25	16.53±0.43
	GREEDY	5.74±0.48	9.56±0.43	15.65±0.59
	DOMSET	3.78±0.22	7.72±0.38	13.97±0.49
	NDOMSET	7.06±0.26	11.78±0.24	17.56±0.41

多特征



(a) CIFAR-10 准确率-召回率@ 32 比特



(b) NUS-WIDE 准确率-召回率@ 32 比特

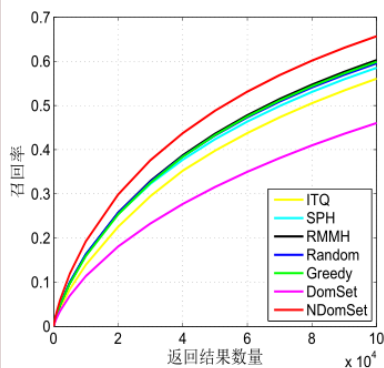
500 候选比特	CIFAR-10		NUS-WIDE	
	32 比特	64 比特	32 比特	64 比特
MFH	13.44±0.09	12.74±0.04	25.28±0.37	25.91±0.37
RANDOM	13.30±0.74	14.32±0.42	25.69±0.51	26.34±0.56
GREEDY	12.17±0.42	12.92±0.41	23.91±0.37	24.16±0.36
DOMSET	11.19±0.06	11.97±0.06	24.66±0.49	25.66±0.51
NDOMSET	14.80±0.34	15.64±0.35	27.17±0.23	27.74±0.16

实验结果-多哈希算法混合

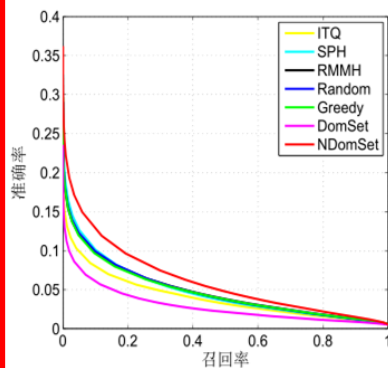
基本哈希算法

多比特哈希算法

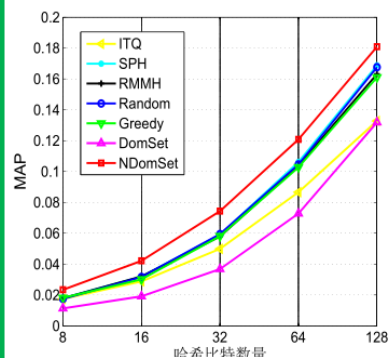
召回率



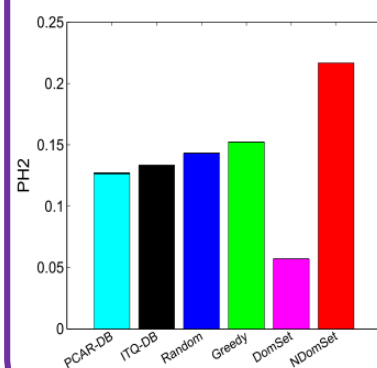
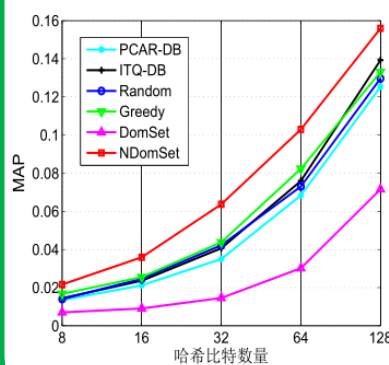
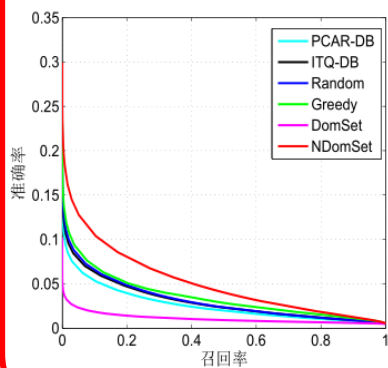
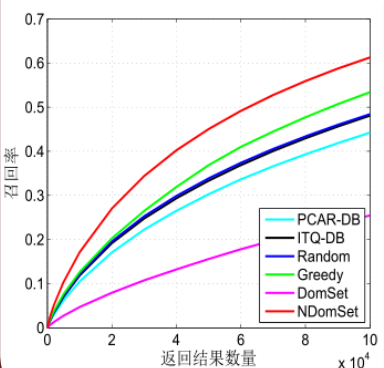
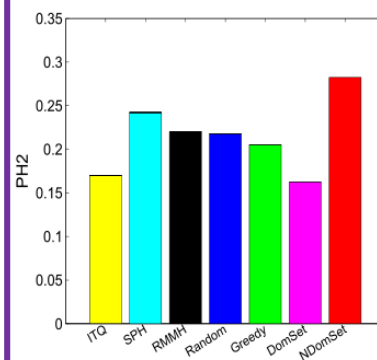
准确率-召回率



MAP



PH2



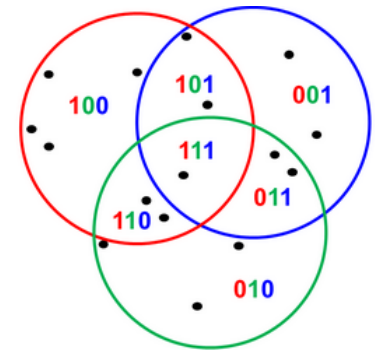
实验对比方法-多哈希表构造

□ 哈希表构造的对比算法

- 随机选取方法 (RAND)
- 直接构造方法 (DHF)

□ 比特生成的哈希算法

- 线性哈希: LSH、PCAH、PCAR、ITQ、RMMH
- 非线性哈希: SPH (Heo et al., 2012)



实验结果-基本哈希算法

哈希表查询

SIFT-1M 上多个 LSH 哈希表的 PH2 性能 (%)

# 哈希表→	1	4	8	12	16
RAND	21.91	18.29	16.20	14.42	13.15
DHF	26.29	28.87	26.88	23.24	16.22
RDHF	26.29	29.25	27.60	23.80	16.44

不同基本哈希算法上多表构建方法的 PH2 性能 (%)

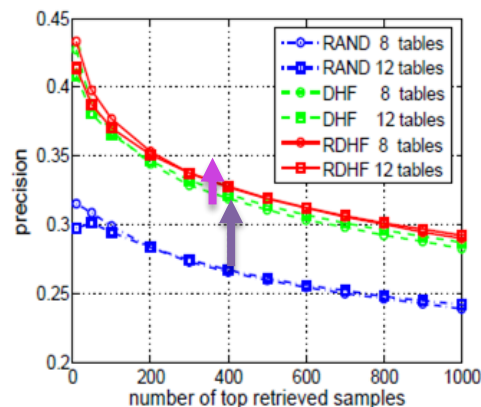
	哈希算法 →	LSH			KLSH			RMMH		
	# 表 →	8	12	16	8	12	16	8	12	16
SIFT-1M	RAND	16.20	14.42	13.15	19.20	17.46	16.09	26.02	24.87	23.73
	DFH	26.88	23.24	16.22	29.97	25.34	18.86	31.87	30.62	28.63
	RDFH	27.60	23.80	16.44	29.40	26.00	19.45	31.23	30.95	29.16
GIST-1M	RAND	8.80	8.51	8.24	11.83	11.20	10.73	11.58	10.96	10.58
	DFH	9.21	9.30	8.72	14.37	13.63	11.88	13.28	12.68	11.59
	RDFH	9.68	9.67	8.99	14.27	14.00	12.31	13.45	13.03	12.04

比特选择

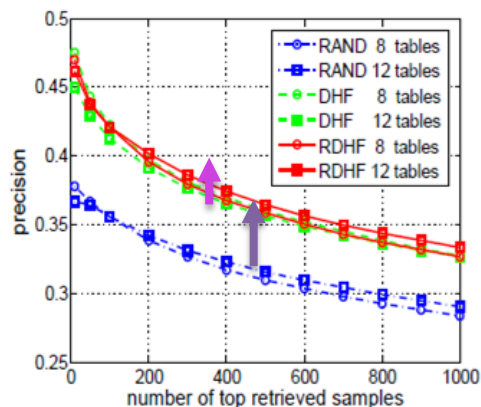
哈希表互补

实验结果-基本哈希算法

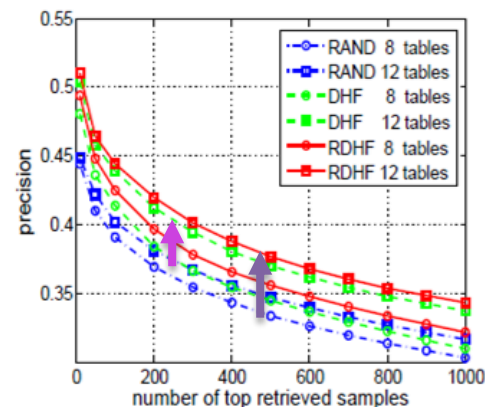
海明距离排序



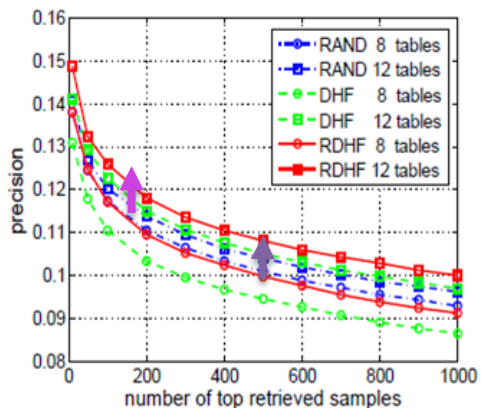
(a) Precision of LSH on SIFT-1M



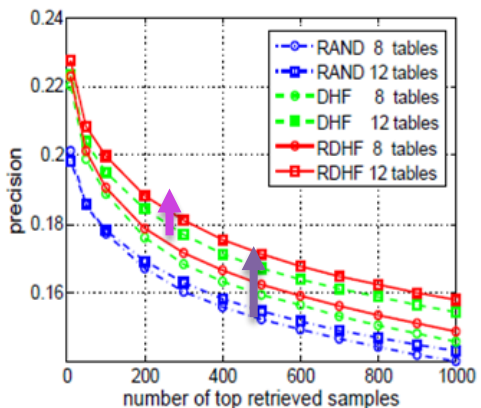
(b) Precision of KLSH on SIFT-1M



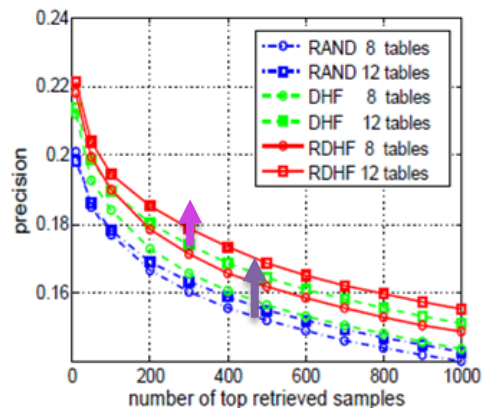
(c) Precision of RMMH on SIFT-1M



(d) Precision of LSH on GIST-1M



(e) Precision of KLSH on GIST-1M

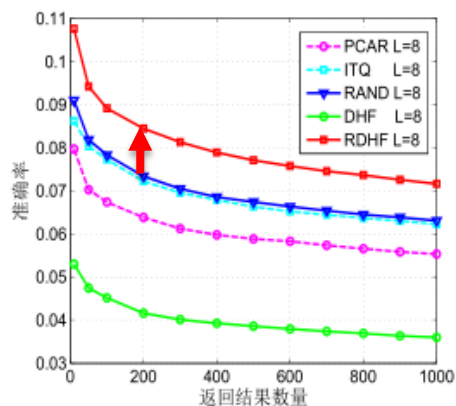


(f) Precision of RMMH on GIST-1M

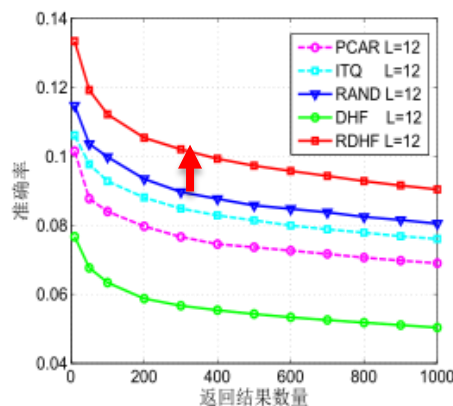
实验结果-多哈希算法混合

GIST-1M 上基于多种哈希算法的多表构造方法的 PH2 性能 (%)

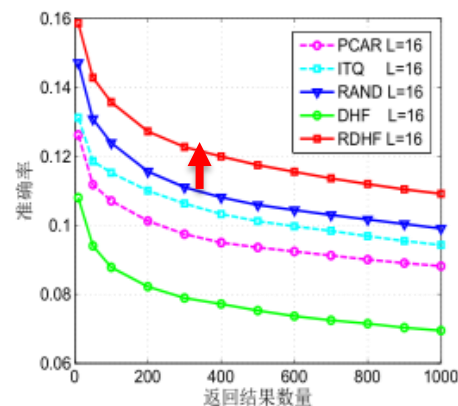
	准确率			召回率		
	8 表	12 表	16 表	8 表	12 表	16 表
PCAR-DB	8.42	8.41	8.31	4.90	5.79	6.74
ITQDB	8.93	8.72	8.60	5.24	6.50	7.57
RAND	9.34	9.26	9.10	5.82	7.01	8.19
DHF	7.36	7.13	6.98	1.11	2.12	3.37
RDHF	11.32	10.82	10.44	6.89	9.40	10.88



(a) 准确率 @ 8 哈希表



(b) 准确率 @ 12 哈希表



(c) 准确率 @ 16 哈希表

基于多种哈希算法的多表构造方法的海明距离排序的性能对比

结论

- ✓ 提出了通用哈希解决方案：哈希比特选择，可以使用不同的哈希算法、参数配置、特征等，支持各种应用场景
- ✓ 给出了哈希比特选择标准、快速求解方法以及基于正则主导集合的理论解释
- ✓ 基于比特选择提出了通用的互补哈希表构造方法

1. Xianglong Liu, Junfeng He, Bo Lang, Shih-Fu Chang. Hash Bit Selection: a Unified Solution for Selection Problems in Hashing. IEEE CVPR, 2013.
2. Xianglong Liu, Junfeng He, Bo Lang. Reciprocal Hash Tables for Nearest Neighbor Search. AAAI, 2013.



谢谢！

请批评指正