K-means Hashing: an Affinity-Preserving Quantization Method for Learning Binary Compact Codes

Kaiming He Fang Wen Jian Sun
Microsoft Research Asia

K-means 哈希：用于学习紧凑哈希码的保相似的量化方法

摘要

在计算机视觉领域，学习保存数据相似度的汉明距离方法成为风潮。散列函数需要量化向量空间，并产生相似性保持的二进制码。大多数现有哈希方法使用超平面（或核化超平面）进行量化和编码。在本文中，我们提出了一个利用了 k-means 量化的哈希方法。该方法执行 K-means 聚类，同时为量化单元建立二进制索引。单元之间的距离近似于它们之间的海明距离。我们进一步将算法推广到乘积空间，以学习较长的编码。实验证明我们的方法（KMH）优于最先进的各种散列编码方法。

1.Introduction

近似近邻（ANN）搜索被广泛用于图像/视频检索[27，11]，识别[28]，图像分类[23，2]，动作估计[25]，以及许多其他的计算机视觉问题。在 ANN 搜索中，较热的一个问题是使用紧凑码近似表示数据距离（或它们的顺序）。根据距离计算方法的不同，压缩编码大致分为两个流：基于海明方法，如局部敏感散列（LSH）[9,1，28]等[31，14，9，5，19，16，15]，和基于查找表的方法，比如矢量量化[7]和乘积量化[10，11，3]。

我们观察到这些紧凑的编码方法通常涉及两个阶段：(i)量化 – 特征空间被分为若干非重叠区域，每个区域具有唯一的索引；(ii)距离计算。现有的哈希方法使用超平面[1，13，29，5，19]或核化超平面[31，14，15]量化特征空间。一个超平面编码成一个比特位，其符号常常由一个散列函数来确定。两个样本之间的距离可以通过汉明距离近似表示(Fig1(a))：该计算在现在的 CPU 上是很快的，仅仅需要两个指令。

另一方面，基于查找表的方法[7，10，11，3]通过 k 均值[18]量化空间。K 均值比使用超平面的那些量化方法更具自适性，并且是在最小化量化误差时表现最优[7]。两个样本之间的距离由聚类中心之间的距离近似(Fig1(b))，它可以从预先计算的查找表中读取。乘积量化[7,10]是将 k 均值量化应用到更大位数的比特位的方法。

基于汉明的方法和基于查找的方法最近获得了越来越大的关注，并且不同的场景下各具优势。基于查找的方法，如[3，10]，比具有相同码长的哈希方法更准确。这要归功于自适应 k 均值量化和更灵活的距离查找。但是，查找法的距离计算比的汉明距离计算慢。海明方法也有优点，即距离计算是问题无关的：它们仅涉及编码阶段，但没有解码阶段。此属性是特别有利的，例如，在移动产品搜索[8]和内置的硬件系统。

在本文中，我们重点在通过计算汉明距离来学习紧凑二进制码。我们提出了一个新方案：我们通过 k-均值量化划分特征空间，用(Fig.1(c))区域中心之间的汉明距离近似样本距离。我们

希望汉明距离能保持 k 均值中心之间的欧几里德距离。一个朴素的解决办法是先利用 k 均值量化空间，然后分配距离保持的区域二进制码。但是这两个步骤只能得到次优解，这并不适用于实际问题。

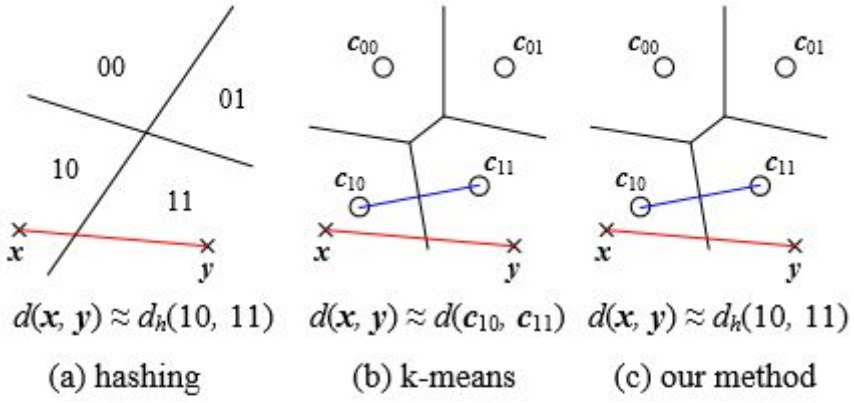


Figure 1: Compact encoding methods. A black line denotes a partition boundary, a circle denotes a k-means center, and a cross denotes a sample vector. Here d denotes Euclidean distance, and d_h denotes Hamming-based distance. The indices are denoted in binary forms.

为此，我们提出保相似性的 k 均值哈希方法，其同时考虑量化和距离近似两个问题。该算法在 k 均值聚类阶段明确地保留了欧式距离和汉明距离之间的相似性。并容易推广到乘积空间学习位数更长的哈希。我们的方法（KMH），结合了 k-means 量化的自适应性和汉明距离计算的高速。

我们注意到一个被称为“k 均值局部敏感哈希”的方法[22]已经提出了[27]。我们指出，术语“哈希”在[22]指经典的桶模型，使得相似的数据将碰撞在同一个桶中。在本文中，我们遵循的术语和最近许多论文[1，31，14，29，19，15]意思相同，指的是用汉明距离计算的紧凑二进制编码。关于“哈希”的两种解释中可以在[24]中找到。

2.Affinity-Preserving K-means

2.1 basic model

我们的基本模型是使用 k 均值量化特征空间，并经由区域中心的汉明距离计算样本近似距离。

根据经典的矢量量化(VQ)[7]，我们将 d 维矢量 $x \in \Re^d$，映射成另一个矢量 $q(x) \in C = \{c_i \mid c_i \in \Re^d, 0 \le i \le k-1\}$。集合 C 称为码本[7]，$c_i$ 是一个码字，k 是码字的数目。如果比特位共有 b 位，那么最多有 $k = 2^b$ 个码字。VQ 将矢量分配给在码本与其最接近的码字。一般，码字是由 k-means 中心给出，因为它们使量化误差最小[7]。

给定两个向量 x 和 y，VQ 规定它们之间的距离是它们码字之间的距离：

$$d(\mathbf{x}, \mathbf{y}) \simeq d(q(\mathbf{x}), q(\mathbf{y})) = d(\mathbf{c}_{i(\mathbf{x})}, \mathbf{c}_{i(\mathbf{y})}), \qquad (1)$$

公式中，$d(x,y)=\|x-y\|$表示向量之间的欧式距离，$i(x)$表示 x 所在的区域。上述表示强调了距离计算仅取决于区域的划分：它可以从预计算的 k-k 查找表$d(\cdot,\cdot)$中读取。

为了用高速的汉明距离计算代替查找表，我们用汉明距离来替换查找表距离：

$$d(\mathbf{c}_{i(\mathbf{x})},\mathbf{c}_{i(\mathbf{y})}) \simeq d_h(i(\mathbf{x}),i(\mathbf{y})) \qquad (2)$$

其中，d_h表示两个区域 i 和 j 的汉明距离：

$$d_h(i,j) \triangleq s \cdot h^{\frac{1}{2}}(i,j). \qquad (3)$$

s是一个常参，h表示汉明距离 ，$h^{\frac{1}{2}}$是它的开方。开方将会在正交哈希法中发挥作用，并使公式能扩展到更长码位上去。s的作用：欧式距离d可以是任意范围的，而汉明距离h在[0，b]受到指定 b 比特的限制。我们将在 Sec 2.4 讨论如何确定 s。

总之，给定码本 C，我们通过$d_h(i(x),i(y))$近似表示 x 和 y 之间的的距离$d(x,y)$（Fig.1(c)）.从中可以明显看出，码字的索引会对相似距离产生影响。

2.2. A Naive Two-step Method

上述模型的朴素“两步走”解决方法如下：首先，通过 k-means 将空间量化成$k=2^b$个区域；然后为每个区域分配最优编码方案。我们为索引序列$I=\{i_0,i_1,\ldots,i_{k-1}\}$分配整数值为$\{0,1,\ldots,k-1\}$。给定码字序列$\{c_i\}$，我们认为能最小化汉明近似误差的码表是最佳索引方案：

$$\min_{\mathcal{I}} \sum_{a=0}^{k-1}\sum_{b=0}^{k-1} \left(d(\mathbf{c}_{i_a},\mathbf{c}_{i_b}) - d_h(i_a,i_b)\right)^2. \qquad (4)$$

该公式最小化向量$d(\cdot,\cdot)$和$d_h(\cdot,\cdot)$之间的差异。

穷尽所有方案去得到（4）的最优解，因为排列组合问题，该过程是极其复杂的：存在$(2^b)!$种组合方式。只有当 b≤3 位这个方案是可行的。当 b =4 它需要一天以上时间，如果 b>4，会花费更多时间。

更重要的是，即使使用穷举法，我们发现这两步法并不能很好地工作（证明见 Sec 4）。这是因为 k 均值量化将产生的任意范围的相关度矩阵$d(\cdot,\cdot)$。拟合这种矩阵将可能导致大的误差，因为海明距离只能在有限的范围产生几个离散值。

2.3. Affinity-Preserving K-means

上述讨论表明的两步方法只能得到次优解。（4）中的保相似性在 k 均值量化中并未被考虑。这促使我们考虑同时最小化量化误差和相似性错误。

传统的 k 均值算法最小化训练样本之间的平均量化误差E_{quan}：

$$E_{\text{quan}} = \frac{1}{n}\sum_{\mathbf{x}\in\mathcal{S}} \|\mathbf{x} - \mathbf{c}_{i(\mathbf{x})}\|^2, \qquad (5)$$

其中，S 是具有 n 个样本的训练集。传统 k 均值算法中使用 EM 算法最小化求解该误差：此处可用来分配区域编号 $i(x)$ 和更新码字 $\{\mathrm{c_i}\}$。

我们也想最小化（2）中的距离近似所产生的误差。考虑所有样本对的相似误差 E_{aff}：

$$E_{\text{aff}} = \frac{1}{n^2}\sum_{\mathbf{x}\in\mathcal{S}}\sum_{\mathbf{y}\in\mathcal{S}} \left(d(\mathbf{c}_{i(\mathbf{x})}, \mathbf{c}_{i(\mathbf{y})}) - d_h(i(\mathbf{x}), i(\mathbf{y}))\right)^2.$$

因为该公式中存在 n^2 个项，所以直接计算它是不可行的。不过，该公式可以写成下面的形式：

$$E_{\text{aff}} = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} w_{ij}\left(d(\mathbf{c}_i, \mathbf{c}_j) - d_h(i,j)\right)^2, \qquad (6)$$

$w_{ij} = n_i n_j / n^2$，n_i 和 n_j 是区域 i 和 j 中的样本个数。直观看来，E_{aff} 是相似矩阵 $d(\cdot,\cdot)$ 和 $d_h(\cdot,\cdot)$ 的带权差异值。

综合量化误差和相似误差，我们得到目标函数：

$$E = E_{\text{quan}} + \lambda E_{\text{aff}}, \qquad (7)$$

λ 是一个固定的权值（本文中，我们使用 10），我们采用交替的方式最小化这个目标函数：

分配阶段：固定 $\{\mathrm{c_i}\}$ 优化 $i(x)$。这步和传统 k-means 算法一致：每个样本 x 将会分配到码本 $\{\mathrm{c_i}\}$ 中与之最近的一个码字。

更新阶段：固定 $i(x)$ 优化 $\{\mathrm{c_i}\}$。与传统 k 均值算法不同，码字的更新取决于全局信息，包括（6）中的保相似矩阵 $d(c_i, c_j)$。所以我们一个个地更新码字 c_j，而保持其他码字 $\{\mathrm{c_i}\}_{\mathrm{i}\neq\mathrm{j}}$ 不变：

$$\mathbf{c}_j = \arg\min_{\mathbf{c}_j}\Big(\frac{1}{n}\sum_{\mathbf{x};i(\mathbf{x})=j} \|\mathbf{x} - \mathbf{c}_j\|^2 + 2\lambda\sum_{i;i\neq j} w_{ij}\left(d(\mathbf{c}_i, \mathbf{c}_j) - d_h(i,j)\right)^2\Big). \qquad (8)$$

这可以通过准牛顿法来解决[26]（在 matlab 中使用 fminunc 命令）。这一步中，对每个码字的更新仅进行一次。

初始化：以上迭代算法需要初始化参数$i(x)$，码本 C 和缩放参数 s（Eqn 3）。实际操作中，使用 PCA-Hashing 得到初始化参数。为了获得相应的 C 和 s，我们需要在已知哈希算法和我们的方法之间建立联系。在 Sec 2.4 中我们对此进行讨论。

我们将以上算法命名为保相似的 k-means。Algorithm1 给出了伪代码。该算法在 50~200 次迭代中收敛。

Algorithm 1 Affinity-Preserving K-means.

Input: a training set $\mathcal{S} = \{\mathbf{x}\}$, the bit number b
Output: an *ordered* set $\mathcal{C} = \{\mathbf{c}_i \mid i = 0, 1, ..., 2^b - 1\}$

1: Initialize $i(\mathbf{x})$, $\mathcal{C}$, s.
2: **repeat**
3: **Assignment**: for $\forall \mathbf{x} \in \mathcal{S}$ update $i(\mathbf{x})$ by x's nearest codeword's index.
4: **Update**: for $j = 0$ to $2^b - 1$, update $\mathbf{c}_j$ using (8).
5: **until** convergence

2.4. Relation to Existing Methods

如果对常量进行调整，我们的方法将成为经典的向量量化或哈希算法。实际上，我们的方法权衡了这两种方法。

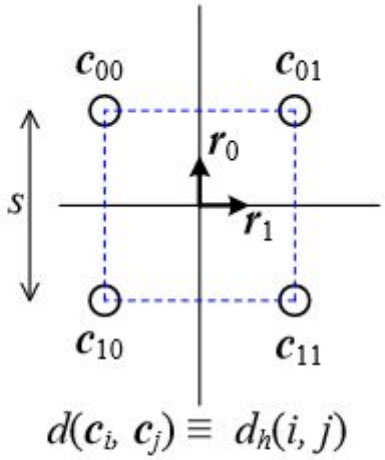


Figure 2: Relation between vector quantization and orthogonal hashing. The vertexes of a hyper-cube are used as codewords. In this 2-d example the hyper-cube is a square. The partition boundaries of the cells are orthogonal hyperplanes (lines in 2-d).

Vector Quantization [7].

如果允许使用预先计算的查找表$d(\cdot,\cdot)$，我们可以去掉（7）中的相似性误差E_{aff}，通过设置λ= 0。其结果是，（8）仅仅通过样本均值就能解决更新的问题。此时，我们的方法就

是经典 K-means 算法。

Orthogonal Hashing and Iterative Quantization [5].

如果我们设定 λ=∞（7），那么 $d(\cdot,\cdot)$ 和 $d_h(\cdot,\cdot)$ 必须相同，最小化（7）相当于迭代量化（ITQ）[5]。

为简单起见，我们假设数据是 b 维的（PCA[5]）。如果这两个查找表 $d(\cdot,\cdot)$ 和 $d_h(\cdot,\cdot)$ 是相同的，$k=2^b$ 个码字等价于一个 b 维超立方体的顶点。码本由下式给出：

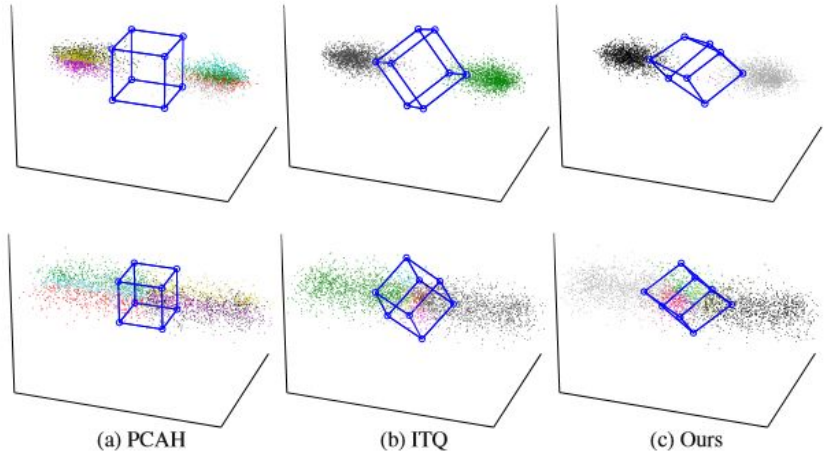
$$C_{\text{cube}} = \{\mathbf{c} \mid \mathbf{c} \in \mathbb{R}^b, \mathbf{c}\cdot\mathbf{r}_t = \pm\frac{1}{2}s, \forall t = 0, 1, ..., b-1\}, \quad (9)$$

其中，s 是这个超立方体的边长，向量 $\{r_t\}$ 是 b 维正交基（Fig.2）。很容易地看到，所得到的区域的分区边界是正交超平面（图 2 中）。任意两个码字之间的欧几里德距离等于到汉明距离：$d^2(\mathbf{c}_i, \mathbf{c}_j) = s^2 \cdot h(i,j) = d_h^2(i,j)$。在图 2 中很容易看到这个事实。 上述等价在[5]中有所提及。

假设数据以零为中心。（9）中的唯一自由旋转矩阵 R，其列是 $\{r_t\}$。那么，最小化目标函数（7）（λ=∞）相当于最小化量化误差 w.r.t.旋转矩阵 R。这和 ITQ[5]的成本函数是完全一样的。

这种关系提供了在等式（3）中确定 s 的一种方式。我们通过 C_{cube} 初始化码本，使用 PCA 的基为 $\{r_t\}$。将（9）代入（5），容易证明 E_{quan} 是 s 上的二次函数。我们通过最小化 E_{quan} 初始化 s。我们初始化后固定 s。从理论上讲，我们可以在每次迭代后更新 s，但我们在实践中观察边缘效应。

2.5. A Geometric View



	PCAH	ITQ	Ours
E_{quan} ($\times 10^3$)	28.7	8.6	***8.2***
E_{dist} ($\times 10^3$)	29.4	16.2	***16.1***

	PCAH	ITQ	Ours
E_{quan} ($\times 10^3$)	7.72	4.51	***4.31***
E_{dist} ($\times 10^3$)	8.11	7.35	***7.14***

Figure 3: A geometric view of Hamming distance: (a) PCAH, (b) ITQ, (c) ours. A circle denotes a codeword, and a line linking two circles means their Hamming distance is 1. A dot denotes a data point, with different colors indicating different encoded indices. Two synthetic datasets are shown here. Each database consists of 3 randomly selected principal components from the SIFT1M datasets [10]. This figure is best viewed in color version.

如上所讨论的，任何正交散列方法（例如，ITQ[5]和 PCAH[29，5]）可以被视为使用旋转超立方体的顶点作为码字的矢量量化方法。Fig.3(a)和(b)示出了几何当 b =3 位时的几何视图。

我们的方法允许“拉伸”的超立方体同时旋转它，如 Fig.3(c)。拉伸失真由 E_{aff} 控制。虽然

ITQ 中有最小量化误差 E_{quan}，我们的方法由于拉伸得到了更小的 E_{quan}，如图 3.我们评估经验平均距离误差 E_{dist}：

$$E_{\text{dist}} = \frac{1}{n^2} \sum_{\mathbf{x} \in \mathcal{S}} \sum_{\mathbf{y} \in \mathcal{S}} \left(d(\mathbf{x}, \mathbf{y}) - d_h(i(\mathbf{x}), i(\mathbf{y})) \right)^2 . \quad (10)$$

图 3 中的表显示我们的方法有更小的 E_{dist}。

注意在图 3 中表中数据受系数 s 的影响。所以对每个数据集，对每个方法我们使用相同的 s，s 的计算见 Sec 2.4。（PCA & ITQ）

图 3 中的每个数据集由 3 个在 SIFT1M 数据集[10]随机选择的主成分构成。有趣的是，我们注意到，在第一个数据集（图 3 顶部，含有 SIFT 的最大主成分）大致有两个簇。即便这些方法中给出 3 位至多 $2^3 = 8$ 个簇，无论 ITQ 和我们的方法都把数据划分成大致两个簇（彩色图 3 中的上方）。这两个集群的码字是在超立方体的对角线上，汉明距离最大（=3）。虽然使用 3 位编码两个集群貌似是低效的，但如果考虑到距离这是值得的。相反，PCAH 将数据分成 8 个均衡的群集，欧式距离是难以通过汉明距离被保留。在图表 3 中（顶部）示出它具有最大 E_{dist}。

3.Generalization to a Product Space

像传统 k 均值，如果比特数 b 很大，因为该算法需要计算和存储 2^b 码字，保相似的 k 均值是不实际的。乘积量化（PQ）方法[10]通过在乘积空间的子空间分别训练 k-means 解决了这个问题。我们表明，我们的算法可以自然地推广到乘积空间。

3.1. From Product Quantization to Hamming Distance Approximation

[10]中的 PQ 方法将空间 $\Re^D$ 分解成子空间的笛卡尔乘积。具体来说，一个矢量 $x \in \Re^D$ 被表示为 M 个子矢量的串联：$x = [\hat{x}^1, \ldots, \hat{x}^m, \ldots, \hat{x}^M]$，其中在 $\hat{x}^m$ 上标 m 表示第 m 子向量。一个有 $k = 2^b$ 中的子码字的子码本 $\hat{C}^m$ 将在第 m 子空间独立训练。在乘积空间 $\Re^D$ 上，任何码字 c 是从 M 个子码本的串联 M 个子码字得出。以这种方式，$\Re^D$ 上有 $K = k^M = 2^{Mb}$ 个不同码字，由 B = Mb 比特位索引。该算法仅需要计算和存储 $\cdot M \cdot 2^b$ 个子码字。

与 VQ 相同（1），PQ 通过码字之间的距离近似两个向量之间的距离：

$$\begin{aligned} d(\mathbf{x}, \mathbf{y}) &\simeq d(q(\mathbf{x}), q(\mathbf{y})) \\ &= \sqrt{\sum_{m=1}^{M} \left(d(q^m(\hat{\mathbf{x}}^m), q^m(\hat{\mathbf{y}}^m)) \right)^2}, \quad (11) \end{aligned}$$

其中，q^m 表示在第 m 子空间的量化器。在 PQ 中，（11）的距离是通过 M 个独立的 k-by-k 查找表来计算。

与 Sec 2.1 的 Eqn（2）相同，我们通过汉明距离近似查找表距离（11）：

$$d(q(\mathbf{x}), q(\mathbf{y})) \simeq \sqrt{\sum_{m=1}^{M} \left(d_h(\hat{i}^m(\hat{\mathbf{x}}^m), \hat{i}^m(\hat{\mathbf{y}}^m)) \right)^2}, \quad (12)$$

这里的符号 $\hat{i}^m$ 指示该索引来自于第 m 个子码本 $\hat{C}^m$。把方程（2）代到（11），我们有：

$$d(q(\mathbf{x}), q(\mathbf{y})) \simeq \sqrt{\sum_{m=1}^{M} s^2 \cdot h(\hat{i}^m(\hat{\mathbf{x}}^m), \hat{i}^m(\hat{\mathbf{y}}^m))}$$
$$= s \cdot \sqrt{h(i(\mathbf{x}), i(\mathbf{y}))}. \qquad (13)$$

当 $i(x)$ 是 M 个子区域的串联时，等号成立。

此方程意味着我们可以在每个子空间独立地应用算法 1，并使用 M 个子区域的串联作为最终索引 $i(x)$。在这种情况下，汉明距离仍然近似于欧式距离。如果该距离排序是唯一需要关注的，系数 s 和（13）中的平方根能被忽略，因为它们是单调的。

注意，等式（13）需要 s 使所有子空间保持不变。我们可以通过在全空间使用 PCAH 初始 s（这非常容易处理）。但在实践中，我们当 s 在每个子空间独立初始化时表现更好。

3.2. Decomposing the Product Space

为了将全空间 $\Re^D$ 分解成 M 个子空间的乘积，我们使用以下简单的准则。

与[31，29，3,5]采用相同的标准，比特位独立，我们期望在我们的方法中子空间是独立的。为此，我们通过 PCA 投影（不降维）预处理数据。根据 PQ 方法的另一个常见的标准[11，12]，我们对每个子空间的方差进行平衡。我们定义方差为子空间特征值的乘积。我们采用一个简单的贪心算法来实现平衡：我们根据特征值从大到小的顺序对主成分排序，依次将它们赋值给具有最小方差的 M 个桶中的一个。在同一个桶中的主成分构成一个子空间。

这种分解方法被称为特征值分配，它的理论基础在[4]中。

3.3. Algorithm Summary

有了上面的子空间分解，我们分别将保相似的 k 均值应用到每个子空间。此步骤产生 M 个子码本，每个子码本具有 2^b 个 $\frac{D}{M}$ 维码字。

为了编码任何向量，我们先将其划分成 M 子向量。每个子向量将在子码本里找到最近的子码字。这 M 个分区连接成为 B= M·b 位的二进制码。编码一个查询项的复杂性是 $O(D^2 + 2^b D) = O(D^2 + 2^{\frac{b}{M}} D)$，其中 D^2 来自于 PCA 投影。在实践中，我们使用 b≤8，编码成本可以忽略。

我们公布我们算法的 Matlab 的代码，以便于理解细节。

4. Experimental Results

我们在两个公开数据集上评估 ANN 搜索算法的性能。第一个数据集是 SIFT1M[10]，含有 1 百万个 128-D SIFT 特征[17]和 10000 独立查询项。第二个数据集是 GIST1M，含 1 百万 384-D GIST 特征[21]。我们随机从 80M 数据集[28]中抽取 1 万个查询样本。我们将内阁查询项的 K-欧式距离内的邻居认为是基础事实。实验中，我们设置 K =10。稍后我们将对 K 进行讨论。

我们采用海明排序的搜索策略，该策略在许多哈希方法中普遍采用[1，31，29，19，5，6]。海明排序是根据数据与查询项的汉明距离进行排序。头 N 个样本将被检索。海明排序是一个穷举的线性搜索方法，但在实践中非常快速：它需要大约 1.5ms 去扫描 1 百万个 64 位的哈希码（单核 C++实现，IntelCorei7 2.93GHzCPU and8GB RAM）。汉明排序可以通过最近的一个方法[20]进一步加快。我们评估头 N 个海明近邻上的召回率。定义为检索结果中真实的近邻占所有真实邻居的比重。

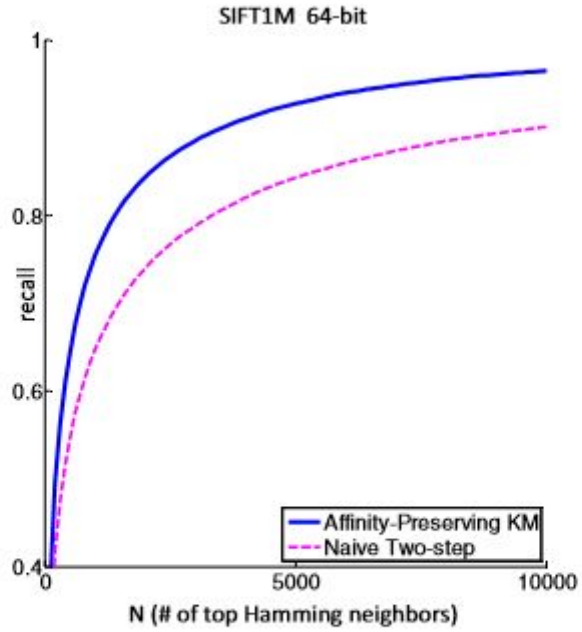


Figure 4: Comparison with the naive two-step method in SIFT1M with 64 bits.

我们使用交叉验证，从{2,4,8}中选择参数 b（每个子空间位数）。子空间数 M 由 $\frac{B}{b}$ 给出。分块处理时，当 B=64，我们的方法在 SIFT1M 上（b=4）的训练时间约 3 分钟，在 GIST1M（b=8）为 20 分钟，使用的是未优化的 Matlab 代码。使用流数据时，查询编码时间（<0.01ms）能与其他基于超平面的哈希方法相媲美，与汉明排序时间（1.5ms/一百万的数据）相比是可忽略的。

Comparisons with the naive two-step method

Sec 2.2 中的朴素两步策略也可以应用到乘积空间。我们已经实现了此两步法（b=4），它需要一天以上的时间来训练。 图 4 比较了这种方法和我们的方法（SIFT1M，B=64，M=16）。

图 4 示出的朴素方法是低劣的，即便它能穷尽两个相似矩阵。这意味着，这个方法只能获得次优解。当将海明保相似矩阵拟合到任意矩阵时，相似误差可能很大。

Comparisons with state-of-the-artshashing methods

我们将所提出的 K-means 哈希（KMH）方法与其它优秀的无监督哈希方法 - 局部敏感哈希（LSH）[1]，谱哈希（SH）[31]，主成分分析哈希（PCAH）[29，5]，和迭代量化（ITQ）[5]进行了比较。我们比较了一个半监督哈希——最小损失哈希（MLH）[19]，为此，我们使用 10,000 个样本去生成伪标签。这些方法都是公开可用的，并且我们使用了它们的默认设置。

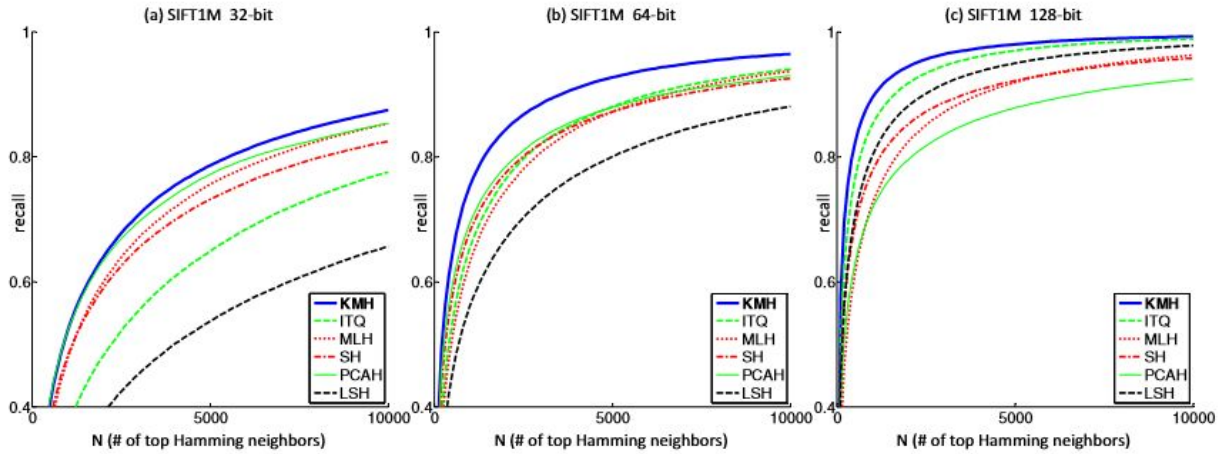


Figure 5: ANN search performance of six hashing methods on SIFT1M using 32, 64, and 128-bit codes. In this figure, K=10 Euclidean nearest neighbors are considered as the ground truth. Our method uses b=2 in the 32-bit case, and b=4 in the 64/128-bit cases.

图 5 和图 6 示出在两个数据集上的比较结果。我们测试了 B =32，64，和 128 位。我们的方法始终优于所有位号的所有竞争对手。我们也注意到，除了我们的方法，就没有方法优于其余的竞争对手。ITQ 最有竞争力，通常使用 64 和 128 位。这意味着降低了量化误差是合理的目标之一。 PCAH 在 32 位 SIFT1M 上执行得非常好，但在其他情况下较差。

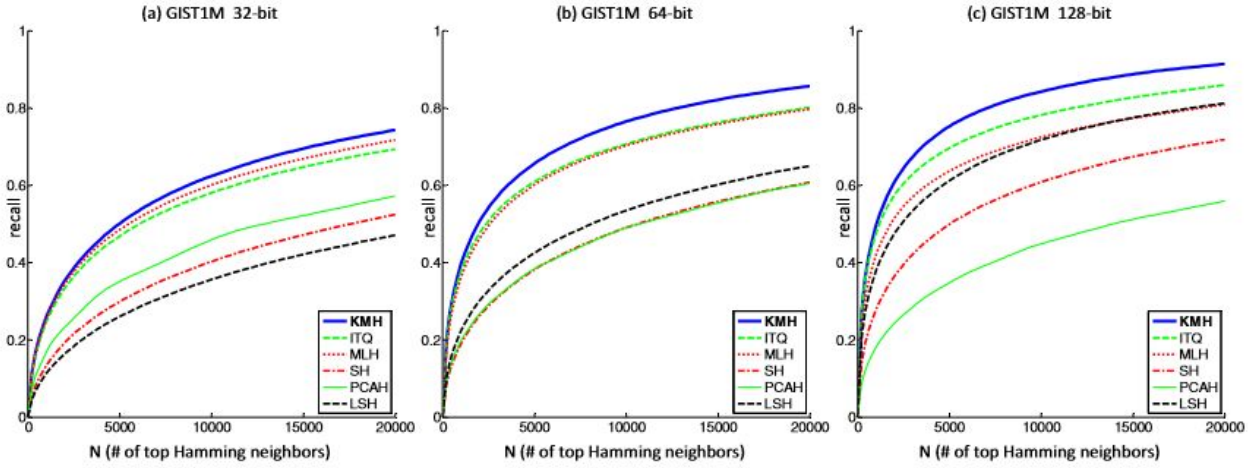


Figure 6: ANN search performance of six hashing methods on GIST1M using 32, 64, and 128-bit codes. In this figure, K=10 Euclidean nearest neighbors are considered as the ground truth. Our method uses b=8 in all cases.

Performance under various K

最近的一篇文章[30]已经注意到，评估哈希方法时，决定基础事实的阈值（如在我们的实验中设置 K 近邻）可能具有特别影响。在图 7，我们评估了不同的 K 值（1 到 1000）。由于篇幅有限，我们只显示 ITQ 和 MLH 的结果，因为我们发现 K 较大时它们更胜一筹。在图 7 中，我们看到，在一系列的 K 下，我们的方法一直执行得更好。这表明我们的方法的具有稳健性。

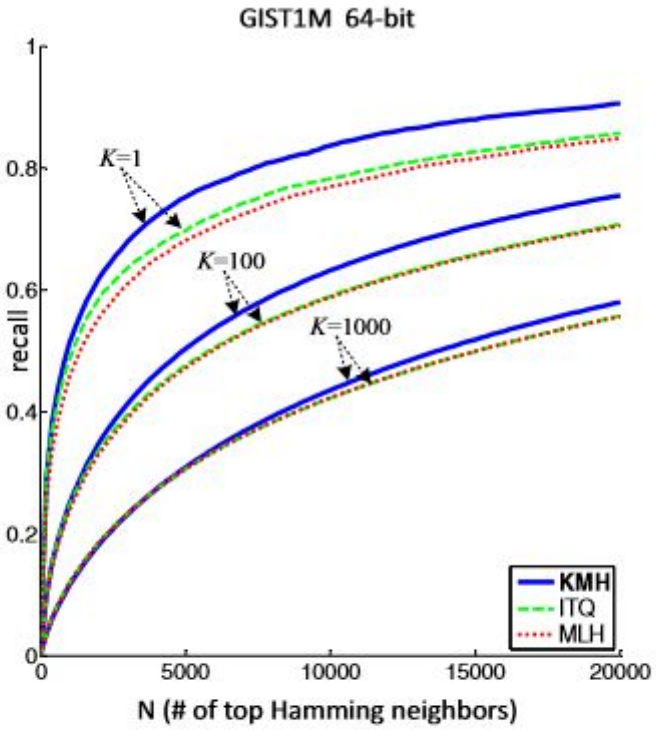


Figure 7: Comparisons in GIST1M with 64 bits under different metric of ground truth nearest neighbors (K=1, 100, 1000). The result of K=10 is in Fig. 6.

5. Discussion and Conclusion

我们提出了基于 k 均值的紧凑二进制编码方法。与使用超平面量化的大多数哈希方法相比，我们的方法具有 K 均值的适应性。我们的保相似的 K-means 算法允许以码字之间的距离代替欧式距离，同时不需要使用查找表。因此，我们的方法还享有汉明距离计算的优点。实验已经表明，我们的方法优于许多哈希方法。