Principal Component Hashing:An Accelerated Approximate Nearest Neighbor Search—summary

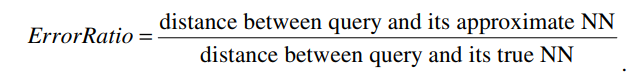
作者：李欣

**背景：**

NN搜索算法在许多方面都有应用，为了避免穷举进行搜索，很快搜索方案被提出，但是这些方法只是在低维数据分布上有效。当数据维度变得很大的时候，这些算法的复杂度和穷举法没有区别了。为了解决数据维度高的问题，很多近似NN算法被提出，主要以近似最近邻（ANN）和局部敏感哈希（LSH）为代表。

ANN首先通过二叉树搜索找到一个NN的candidate，然后进一步通过缩小搜索半径来检查其他可能性。在这个过程中减小了需要搜索的数目，但是同时也使得NN搜索的不准确性提高（潜在的candidate被排除在外）。

LSH是基于hash的近似NN搜索，对于错误率和计算复杂度有一个准确的定义：



LSH首先将搜索空间分成很多桶（buckets），这些桶中的所有向量都有相同的hash值。在进行NN搜索的时候，首先根据相同的规则计算出query的hash值得到其所属的桶。然后对桶中的其他向量进行搜索，此时桶中的向量是有限的，所以可以采用常规的搜索算法进行搜索，最终得到结果。但是也因为这个原因，并不能保证最终结果的准确性。

在基础的LSH以及其变体p-stable SLH中，hash函数并没有参考向量的分布情况，也因此会导致如下的问题：

1. 当query出现在向量分布的低密度区域时候，搜索可能会失败，这是因为没有一个桶含有与query有相同的hash值；
2. 当query出现在向量分布的高密度区域时候，查询的时间可能会大大增加，这是由于这些桶中含有大量的数据，进行下一步搜索的时候比较耗时。

本文也是针对以上两个问题，提出Principal Component Hashing（PCH），该方法在进行hash函数计算的时候利用了向量的分布信息。本方法有如下优势：

1. PCH将搜索空间分解成有限个含有相同期望数量的向量。这保证了对于不同的query向量，都可以有常数数量级的搜索时间，并且可以保证对于所有query向量都可以找到对应的NN候选。
2. PCH通过主成分上的有效距离计算从NN候选中找到NN向量。

**近似最近邻搜索：**

几乎所有基于NN搜索的研究，都是基于以下两点技术：

1. 减少距离计算的次数。距离计算次数的减少一般都是基于三角不等式[8,9,10]或者搜索空间分解[4,11,12]；
2. 距离计算的修剪（忽略不可能的计算）。当中途距离超过给定的暂定距离时，修剪停止距离计算。

这些研究领域已经相当成熟了，但是计算效率并没有显著提高，甚至会下降到穷举法同等数量级。

为了解决计算效率问题，近似最近邻算法被提出（以ANN和LSH为代表）。

**LSH：**

符号解释：x1,x2ϵX; D(x1,x2):x1,x2之间的距离; S⊂X: 存储向量集合; q: 查询向量；NN(q)⊆S：S中与q最近的向量，则LSH是找到一个近似最近邻向量NN’(q)使得：D(q,NN’(q))≤cD(q,NN(q))。其中c≥1（错误率）

Hash函数定义如下：

U是一个hash值集合，h(x):X−>U为hash函数，LSH需要满足如下条件：

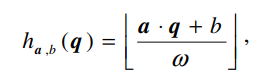
1. D(v,q)≤r1，则Pr(h(q)=h(v))≥p1
2. D(v,q)>r2，则Pr(h(q)=h(v))<p2

其中，p2≤p1且r2=c\*r1。

通过这些符合定义的hash函数，我们可以实现NN’(q)的查找，在L=nρ(c)时间内。其中ρ(c)=lnp1/lnp2，n为数据集中向量的数量。因此，很多工作都围绕ρ(c)展开，因为ρ(c)越小，性能越优。

**P-Stable LSH:**

该方法为LSH的一个实用性例子，通过欧式距离寻找近似NN向量。P-stable哈希函数定义如下：



其中：q为查询向量，a为一个向量，b,ω为常数。

该哈希函数将向量投影到向量**𝒶**上，并通过ω量化该轴。其中b可以看做是调节偏差，取值范围在[0, ω]，向量**𝒶**是从p-stable分布中选取的向量，比如：各向同性高斯（isotropic Gaussian），根据p-stable分布属性的不同，可证明该hash函数可以达到ρ(c)≤1/c[6].

但是此方法不适应于高维空间，因为计算复杂度将会大大增加。

**Principal Component Hashing：**

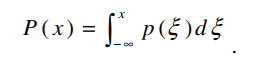
Three steps:

1. Hash value computation
2. NN candidates generation
3. Refinement of NN candidate to find proximate NN
4. **Hash value computation:**

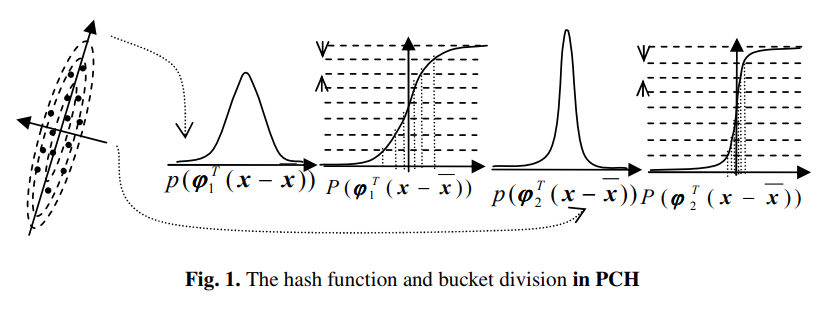
在p-stable LSH中，hash函数的参数与数据的分布无关，但是准确度和效率可能根据数据的分布而有差别。很容易出现前面背景里提到的两个问题。这就意味着ρ(c)并不能保证真实的性能，只是说明准确度与速度独立于数据的分布。

我们使用数据的分布来设计hash函数，在实际使用分布的主成分（principal component of the distribution）而不是**𝔞**，这是因为当向量被映射到主成分上时的标准差可以达到最大化。这就意味着，被映射的向量广泛地分布在主成分上。

当完成向量映射之后，需要将投影轴分到不同的桶中，最好的情况是所有桶中都含有相同数量的向量。此时如果知道在投影轴上数据分布的概率p(x)，则可以计算累计概率分布：

 （3）

显然，P(x)单调递增，且定义域为[-∞,+∞]，值域为[0,1].此时如果将值域切分为n+1个间隔[0,∆],[ ∆,2∆],……,[n∆,1]，则整个投影轴可以被分为n+1个没有交集的桶：(-∞,P-1(∆)],(P-1(∆),P-1(∆)],……,( P-1(∆),+∞),如下图所示：



这样不相交的划分，可以保证：

1. 每个query都会落到一个桶内
2. 每个桶包含相同的期望数量的向量。

这样非常适合进行近似NN搜索。

为了实现这一想法，文章做了一个假设：存储向量服从高斯分布  
这样，投影到主成分上的向量也服从高斯分布，这样就可以将Gaussian的p(x)用到被投影的向量上。

则（3）式可以被sigmoid函数近似：



这样的近似是为了设计一个快速hash函数，当使用第i个主成分时，hash函数为：



其中∆为间隔。

其中φi(i=1,…,M)是在给定数据集上执行PCA得到，由φi可以得到一系列独立的hash函数。这些hash函数对应于整个搜索空间的点阵分解。

使用上述hash函数，在轴i上的每一个桶都有一个hash值H，记作BiH，即：



其中S为搜索空间

1. **Generation of NN Candidates**

根据上述方法，计算出一个query的hash值hi(q)，应该寻找在中寻找candidates，这将大大减少NN候选集的大小，但是当query分布在桶的边界的时候，非常容易造成空候选集或者错误的搜索结果。

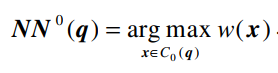
因此候选应该分布在那些至少有一个hash值为hi(q)的桶里，这就意味着初始的候选集C0(q)应该是Bihi(q)的合集。即：



但是，这样做的结果会导致候选集过大，本文又提出refinement of candidates来精简候选数据集。

1. **Refinement of NN Candidates**

当进行hashing的时候，可以计算每个存储向量x的命中频率，即散列值匹配的次数，用w(x)表示，根据这个值选择一个tentative NN向量NN0(q)。



则tentative距离𝓏=D(q, NN0(q))

这个tentative距离𝓏是用来进行距离计算剪枝（Pruning）

1. **Tips for Improving the performance**
2. Bucket Overlapping

当query分布在桶间的边界的时候，NN(q)可能并不在桶中，这是由于桶间不存在交集的前提导致的。这里引入overlapped arrangement：



通过这个设定，成立，其中δ为一个范围。δ越大可以得到更加精确的搜索结果。

1. Cutoff the NN Candidates

注意到，当采用bucket overlapping时，候选集会增大。尽管采用有效的剪枝策略，但是大量的候选集还是会降低搜索速度。此时我们可以再次使用命中率w(x)来在保证精确度的同时减少候选集的大小，这是因为命中率越大成为NN(q)的可能性就越大。

在实际应用中，候选集C0(q)会根据w(x)进行降序排列，前b%会被抽取出来作为NN候选集C0’(q)。其中b便称为cutoff率。

1. **Extension of PCH to General Distribution（A-PCH）**

在完成桶分解之后（bucket decomposition）之后，可以将其构建成一个平衡二叉搜索树的结构。在搜索的过程中，进行二叉搜索寻找到query所落进的那个桶。之后的操作和普通的PCH相同。

**Experiments：**

做了两组对比实验：在A-PCH,ANN和LSH之间的对比实验；PCH和A-PCH之间的对比实验。实验结果都表明，PCH和A-PCH的效果都明显好于ANN以及标准p-stable LSH。同时，在数据不服从高斯分布的情况下，A-PCH比PCH的效果要好。

**Conclusion：**

本文介绍了两种基于PCA的hashing方法PCH以及它的变形A-PCH。两种Hashing方法都充分利用了存储数据的分布属性。

PCH将数据映射到主成分轴上，然后将每个轴分解成互不相交的含有相同数量数据的桶，这样设计可以保证1）不存在查询失败2）线性搜索时间。在搜索阶段，一个候选集通过hash函数被抽取出来，在这个过程中使用hash值的命中率来初始化NN候选集。通过这个initial guess以及NN候选集，我们可以很高效的挑选出近似NN，这个过程还可以通过剪枝来提高效率。

PCH假设存储数据服从高斯分布的，但是现实中往往并不是这样。进而本文又将PCH算法扩展到A-PCH，完全不依赖高斯分布的假设前提。

**Reference：**

1. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Transactions on Information

Theory IT-13(1), 21–27 (1967)

2. Zhang, Z.: Iterative Point Matching for Registration of Free-Form Curves and Surfaces.

Tech. Report INRIA, No 1658 (1992)

3. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun.

ACM 18(9), 509–517 (1975)

4. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm

for approximate nearest neighbor searching. Journal of the ACM 45, 891–923 (1998)

5. ANN: Library for Approximate Nearest Neighbor Searching,

http://www.cs.umd.edu/~mount/ANN/

6. Indyk, P., Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of

Dimensionality. In: Proceedings of the 30th ACM Symposium on Theory of Computing

(STOC 1998), pp. 604–613 (May 1998)

7. Datar, M., Indyk, P., Immorlica, N., Mirrokni, V.: Locality-Sensitive Hashing Scheme

Based on p-Stable Distributions. In: Proceedings of the 20th Annual Symposium on Computational

Geometry (SCG 2004) (June 2004)

8. Andoni, A., Indyk, P.: Near-Optimal Hashing Algorithms for Approximate Nearest

Neighbor in High Dimensions. In: Proc. of FOCS 2006, pp. 459–468 (2006)

9. Vidal, R.: An algorithm for finding nearest neighbor in (approximately) constant average

time. Pattern Recognition Letters 4, 145–158 (1986)

10. Mico, L., Oncina, J., Vidal, E.: A new version of the nearest-neighbor approximating and

eliminating search algorithm (AESA) with linear preprocessing time and memory requirements.

Pattern Recognition Letters 15, 9–17 (1994)

11. Brin, S.: Near neighbor search in large metric spaces. In: Proc. of 21st Conf. on very large

database (VLDB), Zurich, Switzerland, pp. 574–584 (1995)

12. Yianilos, P.Y.: Data structures and algorithms for nearest neighbor search in general metric

spaces. In: Proc. of the Fourth Annual ACM-SIAM Symp. on Discrete Algorithms,

Austin, TX, pp. 311–321 (1993)