文章编号：2095-2163(2020)11-0070-04 中图分类号：TP399 文献标志码：A

基于邻居聚类的近似最近邻搜索

赵 增[1,2]，李明勇[1]，胡航飞[1]
（1 东华大学 计算机科学与技术学院，上海 201620；2 上海市计算机软件评测重点实验室，上海 200235）

摘 要：本文提出了一种新的基于图的方法，用于对高维特征向量的数据集进行近似最近邻搜索(ANNS)。大多数基于图的方法着重于提高图的构造质量，而本文的工作着重于图搜索的性能。基于近似 k 近邻(kNN)图来展示实验结果，并且存在许多用于构建近似 kNN 图的现有方法，例如 NN 下降、KGraph 或 Faiss。本文在图的构建阶段，首先初始化一个近似的 kNN 图，然后利用 K-means 聚类算法将邻居聚类；在查询阶段，使用贪婪搜索算法，遍历图并尝试贪婪地到达查询。为了提高查询性能，仅通过聚类信息比较其中一部分邻居，在实验中展示了如何降低查询成本和提高查询精度。

关键词：近似最近邻搜索；k 最近邻图；贪婪搜索算法

Approximate nearest neighbor search based on neighbor clustering

ZHAO Zeng[1,2], LI Mingyong[1], HU Hangfei[1]
(1 School of Computer Science and Technology, Donghua University, Shanghai 201620, China;
2 Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 200235, China)

【Abstract】 This article presents a new graph-based method for approximate nearest neighbor search (ANNS) on datasets of high dimensional feature vectors. Most graph-based methods focus on improving the quality of graph construction, but our work focuses on the performance of graph search. We report the results using approximate k-nearest neighbor (kNN) graph, and there are many existing methods for constructing approximate kNN graphs, such as NN-descent, KGraph or Faiss. In the construction stage of the graph, we first initialize an approximate kNN graph, and then filter the neighbors by the angle among the neighbors of each point on the graph. During the query stage, we use the greed-search algorithm, which walks over the graph and tries to reach the query greedily. To improve the query performance, we only compare a part of its neighbors by angle information. We show in the experiment how to achieve lower query time and query cost with high precision.

【Key words】 Nearest neighbor search; K-nearest neighbor graph; Greed-search algorithm

0 引 言

数十年来，最近邻居搜索(NNS)一直是一个热门话题，它在数据挖掘，机器学习和人工智能的许多应用中发挥着重要作用。当前，可用的数据集涵盖了广泛的应用程序和数据类型，包括图像、音频、视频、文本、合成和深度学习数据。SIFT、CIFAR 等图像数据集是将局部图像区域压缩到高维度空间中的单个点，这些外部点使用 64 到 512 个外部维度。

计算高维向量之间的欧几里得距离是 NNS 的基本要求。由于维数灾难，NNS 本质上很昂贵。具有 n 个数据点并在 n 维空间 R^d 中查询 q 的数据集 D，*NNS* 的目的是找到最接近 q 的点 $o^* \in D$。其中，o^* 称为 q 的最近邻居。定义如式(1)：

$$NN(O^*) = \arg\min_{p \in D}(p,q). \tag{1}$$

通常，最接近查询点 q 的 K 个点是从数据集中返回的，称为 K-最近邻居搜索(K-NNS)。查找 kNN 集的简单方法是计算查询 q 与数据集 D 中每个点之间的距离，并选择距离最小的点。当处理稀疏数据时，可以通过高级索引结构(例如，反向索引)有效地计算 NNS。但是，对于具有密集特征的数据，查找 NNS 的成本为 $O(n)$。当数据集很大时，耗时严重。对于高维 NNS，由于难以找到准确的结果，大多转向 NNS 的近似版本，即近似 k 最近邻搜索(K-ANNS)，在近二十年中已被广泛使用。

近来，基于图的方法引起了人们的极大关注。例如 NSG[1]、HNSW[2]、EFANNA[3] 和 FANNG[4] 等方法。基于图的方法离线构造 kNN 图，可以将其视为高维空间中的大型网络图。使用基于图的方法所面临的挑战是精确 kNN 图的高构造复杂性，尤其是涉及大型数据集时，计算复杂性将成倍增加。许多研究人员转向建立近似的 kNN 图，但仍然很耗时。本文提出了一种新的基于图的搜索方法，该方法可以应用于各种基于图的搜索算法中。经实验验证，这种方法的搜索性能已经超过了最新的搜索算法，

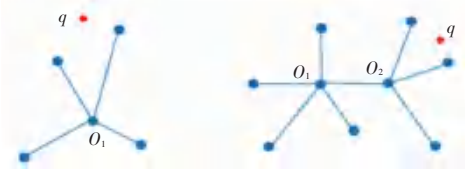
作者简介：赵 增(1996-)，男，硕士研究生，主要研究方向：近似最近邻搜索；李明勇(1979-)，男，博士研究生，主要研究方向：深度哈希；胡航飞(1995-)，男，硕士研究生，主要研究方向：近似最近邻搜索。

收稿日期：2020-10-08

在 Trevi 可以将查询成本缩短 40%以上，在 Audio 数据集上可以缩短 50%以上。

1　邻居选择对网络搜索质量的影响

在图网络上，每个点都拥有若干个邻居，例如图 1(a)中，O_1 的邻居拥有4个结点，点 q 是查询点。图 1(b) 中，O_1、O_2 互为邻居。因此若干个此类结点组合将构建成图网络。



(a)　　(b)

图 1　邻居和图网络

Fig. 1　Neighbor and Graph Network

在早期基于图的方法中，图上各个点使用精确的邻居点作为邻居集合，但由于构建精确 kNN 图的计算成本很高，因此许多研究人员转向构建近似的 kNN 图，即选择近似最近邻作为邻居集合。图上的每个结点都可能未连接到其实际邻居，而是连接到其近似邻居。此类方法尽管可以极大地提高索引构建速度，但可能会影响搜索精度。实际上，在 EFANNA 中的实验结果证明，低精度的近似 kNN 图仍然表现良好。这是因为 EFANNA 构造的近似 kNN 图的“错误”邻居实际上是更远的邻居。这些更远的邻居在搜索过程中扮演“高速公路”角色，这使搜索路径更快地到达查询点的邻域。

为了减少在图上搜索的时间，构造一个近似的 kNN 图通常需要降低图的出度。在有向图中，出度表示某个结点指向任意结点的边连接数量的总和，入度则表示任意结点指向某个结点的边连接数量的总和。通常，如果一个点具有较大的出度，那么它将成为 kNN 图的“交通枢纽”，这将增加搜索的复杂度。由此看来，从每个点的邻居候选集中选择最终邻居变得尤为重要。一些比较先进的算法使用有趣的边缘选择策略，例如 MRNG[1]、RNG[5]，并取得了引人的效果。

2　基于邻居聚类的搜索方法

由于构建精确 kNN 图的成本非常高，因此基于图的索引通常需要创建一个近似 kNN 图。在图上，每个数据点都连接到它的 k 个近似最近邻居。完成该算法需要二个阶段：构建图索引阶段和基于构建索引的查询阶段。

2.1　索引构建阶段

同 NSG 构建网络类似，使用 NN 下降的方法构建一个近似的 kNN 图，为图上的每一个结点计算邻居候选集，并设每个点的最大邻居上限是 R 个。计算数据集的近似中心(各个维度求和取均值)，对于图上的某一结点 e，从中心结点开始，使用贪婪搜索算法直到找到该结点 e。在搜索过程中，所有和点 e 发生欧氏距离计算的点，将被放入候选集合中。最后使用 MRNG 的边缘选择策略，将邻居集合筛选至 R 以下。为了查询阶段的快速搜索，将邻居集合进行 K-means聚类。如图 2 所示，点 p 拥有 7 个邻居，和点 p 具有相似角度的邻居将被聚为一类。使用每个邻居点和点 p 之间的余弦距离来聚类，余弦距离相似的点将被聚为一类。若指定聚类个数 K = 4，那么所有的邻居将被聚为 4 类，$C1$、$C2$、$C3$ 和 $C4$ 为聚类中心。这样在图上每个结点的邻居集合将被分为 4 类，这些聚类信息被保留并将在查询阶段使用。

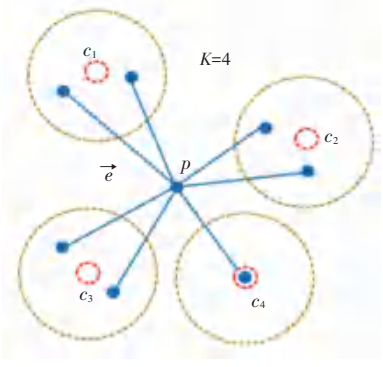


图 2　邻居集合的聚类过程

Fig. 2　Neighbor set clustering process

2.2　查询阶段

本文对基于传统的贪婪搜索算法进行改进，使用随机方法初始化候选集。由于所有结点的邻居集合都在索引阶段进行聚类，因此可以指定在搜索过程中要检查的聚类数 k'。在图 3 中，聚类 K 的数目为 3，点 1 和点 2 在两个不同的聚类中，点 3 和点 4 在同一聚类中。当迭代起始点为 p 时，计算点 p 的 3 个聚类中心和查询点 q 之间的角度(用余弦相似度代替)。如果指定 k' = 2(k' < = K)，并且 a_1、a_2、a_3 的角度分别为 30°、100°、120°，则只需要检查点 2、3、4。其原因是，点 1 所在的聚类中心和查询点 q 之间的角度太大，则不必计算。反之，如果检查太多的聚类，那么必然会增加计算成本。如果 k' = K，算法就需要检查所有邻居集合中的所有点，那么将失去构造包含聚类信息图的意义。如果检查的聚类太少，即使可以降低计算成本，也很难实现高精度。使用此方法一直迭代检查整个图网络，最终查询路径会在查询点的邻域附近收缩，迭代次数和查询轮次的

个数有关。图 3 中橘黄色曲线代表查询点 q 的最近邻邻域，邻域内有极有可能包含点 q 的真实最近邻。合理的检查聚类个数将降低成本并实现高精度。通过调整参数可以很容易获得要检查的最佳聚类个数。

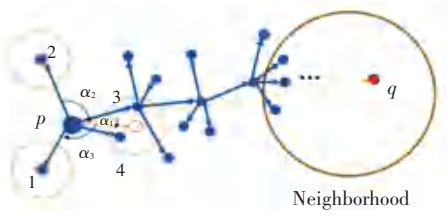


图 3 基于角度的贪婪搜索算法

Fig. 3 Greedy-search algorithm based on angle

3 实验结果分析

通过实验，将对公共数据集进行详细分析，以证明本文方法的有效性。

实验使用 Audio 和 Trevi 两个数据集。Audio 是音频数据集，Trevi 是图像数据集。Audio 拥有 192 维度的特征向量 53 387 个，Trevi 拥有 4 096 维度的特征向量 99 900 个。在 Audio 数据集的实验中，统一使用独立于数据集之外的 200 个 192 维的特征向量作为查询。Trevi 数据集同样使用 200 个4 096维的特征向量作为查询。程序代码以 C ++编写，并由带有“O3”选项的 g++5.4 编译。所有数据集上的实验都是在配备 i5-8300H CPU 和 16GB 内存的计算机上进行的。

为了衡量不同算法的 ANNS 性能，使用召回率和成本作为评估准确性的标准。平均召回率和平均成本则是多个查询点的结果求均值得到。给定一个查询点，所有算法均应返回 k 个点。需要比较这 k 个点中有多少个在真正的 k 个最近邻居中。假设给定查询返回的 k 个点的集合为 R'，而查询的真实 k 个最近邻居集合为 R，则召回率定义如式(2)：

$$recall(R') = \frac{|R' \cap R|}{|R|} \times 100\%. \qquad (2)$$

另一个绩效评估指标是成本。在查询阶段，将计算与查询点进行欧几里德距离计算的点。假设数字为 C，数据集中的点总数为 N，则将成本定义为式(3)：

$$cost(C) = \frac{C}{N}. \qquad (3)$$

将 HNSW 和 NSG 两种最新的图算法来作为比较，以此来验证实验的高效性。HNSW 基于可导航小世界(NSW)[6]提出的分层图结构，是 NSW 的改进版本，并且在性能上有很大的提高。HNSW 具有多个实现版本，例如 Faiss、Hnswlib。实验中使用性能更好的 Hnswlib 进行比较。NSG 是基于 kNN 图的方法，其中该图上每个点的邻居集都通过 MRNG 方法进行裁剪。在查询阶段，每个查询点都从相同的导航结点开始搜索。NSG 可以很好地近似单调的搜索路径。此外，NSG 在淘宝(阿里巴巴集团)的电子商务搜索场景中显示出卓越的性能，并已以十亿个结点的规模集成到其搜索引擎中。

通过在搜索阶段增加候选集列表的长度来增加召回率，可以得出一些有趣的结论。

实验中，统一比较在高精度下(99%以上)，对比 3 种方法的查询成本。通过验证，相同召回率下，本文的方法需要更少的查询成本。图 4 是 Audio 和 Trevi 两个数据集上的召回率(Recall)和成本(cost)的对应曲线。为了方便比较，cost 直接使用查询点的访问个数，200 个查询点取平均作为性能评估指标。图中展示的是 top20 的结果(求前 20 近邻)。在 Audio 数据集的 Recall 达到 99.75%以上时，本文方法所需要的 cost 低于其它二种方法。同理，在 Trevi 数据集的 Recall 达到 98%以上时，同样得到类似的效果。图 5 展示的是，在其它参数不变的情况下，3 种方法求 top50 的结果。从实验结果可以看出，本文的方法依然展示出优越的性能。通过 Trevi 数据集可以看出，高维度的数据集在本文方法上依然有效。

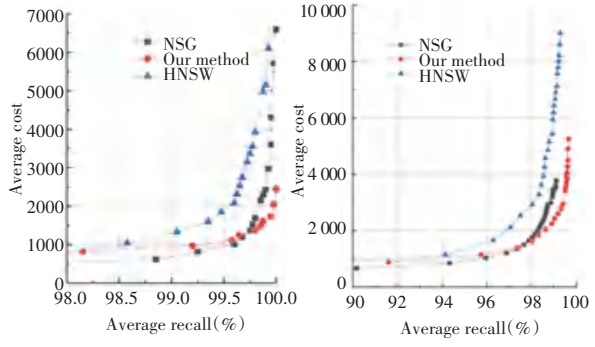


图 4 3 种算法的召回率-成本曲线(Top20)

Fig. 4 The recall-cost curves of three algorithms

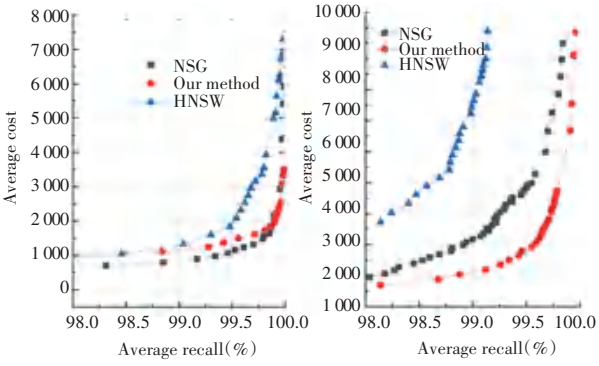


图 5 3 种算法的召回率-成本曲线(Top50)

Fig. 5 The recall-cost curves of three algorithms

(下转第 78 页)