高维近似最近邻搜索：基于可靠且高效的距离比较操作

Jianyang Gao 南洋理工大学 新加坡
jianyang.gao@ntu.edu.sg

Cheng Long 南洋理工大学 新加坡 c.long@ntu.edu.sg

arXiv:2303.09855v1 [cs.DS] 17 Mar 2023

摘要

在高维欧几里得向量空间中进行近似 K 近邻（AKNN）搜索是一项基础且具有挑战性的问题。我们发现，在高维空间中，几乎所有 AKNN 算法的时间开销都由距离比较操作（DCO）的开销主导。每次操作都会扫描对象的所有维度，因此其运行时间随维数线性增长。为加快这一过程，我们提出了一种名为 ADSampling 的随机算法。对于大多数距离比较操作，该算法的运行时间随维数呈对数增长，且以高概率成功。此外，我们基于 ADSampling 开发了一种通用技术和两种针对特定算法的技术，可作为插件增强现有的 AKNN 算法。理论分析和实证研究均证实：（1）我们的技术几乎不会造成准确率损失；（2）它们能够持续提升效率。

ACM 引用格式：

Jianyang Gao 和 Cheng Long。2023年。高维近似最近邻搜索：采用可靠且高效的距离比较操作。收录于*2023*年国际数据管理会议（*SIGMOD '23*）论文集。ACM，纽约，美国，17页。https://doi.org/XXXXXXX.XXXXXXX

1 引言

高维欧几里得向量空间中的 K 近邻（KNN）搜索是一项基础问题，在信息检索 [41],数据挖掘 [13] 和推荐系统 [53]中有广泛应用。然而，由于维数灾难 [28],，精确 KNN 查询通常需要难以接受的响应时间。为实现更好的时间与准确率权衡，许多研究人员转向其宽松版本，即近似 K 近邻（AKNN）搜索 [15, 23, 26, 32, 46, 47]。

AKNN 已有许多算法，包括（1）基于图的方法 [20, 21, 29, 40, 45, 46]，（2）基于量化的方法 [4, 5, 23, 24, 26, 32]，3 [8, 12, 14, 47, 51] 4 [15,（）基于 22, 27, 28, 39, 44, 54, 55, 62] 树的方法和（）基于哈希的方法。特别是，我们关注内存型 AKNN 算法，这类算法假设所有原始数据向量和索引都可以存放在主存中 [8, 21, 32, 40, 45–47, 62]。这些算法都采用以下策略：首先生成一些K近邻候选项，然后从中找出 K 近邻。[1] 首先，它们生成 K 近邻候选项的方式有所不同。例如，基于图的方法会用图组织向量，并在图上进行基于启发式的搜索（例如贪心搜索）来生成候选项。其次，这些算法在从候选项中找出 K 近邻的方式上大体相同。具体而言，它们会维护一个 K 近邻集合 Q [2]（严格来说是一个最大堆），并针对一个新的候选对象，检查其与查询对象的距离是否不大于 Q 中的最大值。如果是，则将该候选对象及其距离作为键加入 Q；[3] 否则，丢弃该候选对象。我们将**检查一个对象与查询对象的距离是否不大于某个距离阈值，并在满足条件时返回其距离**这一计算称为距离比较操作（DCO）。给定对象**o** 和距离阈值 r，如果**o**与查询对象的距离至多为 r，则称**o**为正对象（相对于 r）；否则称为负对象。

所有现有 AKNN算法都采用以下方法，对一个对象和一个阈值执行 DCO。首先计算对象与查询对象之间的距离，然后将计算出的距离与阈值进行比较。我们将这种方法称为 FDScanning，因为该操作会扫描对象的全部维度。显然，FDScanning 的时间复杂度为 $O(D)$，其中 D 是对象的维度数。基于 FDScanning，几乎所有 AKNN算法的时间开销都主要由执行 DCO 所决定。我们以最流行的 AKNN算法之一 HNSW [46], 为例进行说明（其他算法的详细介绍将在第2.2节中提供）。设 N_s 为生成的 K近邻候选项数量。HNSW 的总时间开销为$O(N_sD+N_s\log N_s)$，其中 $O(N_sD)$ 是执行 DCO 的开销，而 $O(N_s\log N_s)$ 是其他计算的开销（详细分析见第2.2节）。在实践中，对于包含数百万个对象的大型数据集，D可能达到数百，而 $\log N_s$ 只有几十，因此执行 DCO 的开销占据了 HNSW 总开销的主要部分（我们将在第2.2节中通过实验证实这一结论）。例如，在包含 256 个维度的真实数据集 DEEP 上，DCO 占总运行时间的 77.2%。

我们有两个观察。首先，对于负对象上的 DCO，我们只需要确认这些对象的距离大于阈值距离，而无需返回其精确距离——请记住，负对象会被丢弃。因此，FDScanning 总会计算对象上 DCO 所涉及对象的距离，对负对象而言，这会进行不必要的计算。

允许免费为个人或课堂使用而制作本作品全部或部分内容的数字副本或纸质副本，但不得为营利或商业利益而复制或分发，且副本首页必须包含本声明和完整引文。对于本作品中归 ACM 以外其他方所有的组件，必须尊重其版权。注明出处后允许摘录。若要以其他方式复制、再版、发布到服务器或分发给他人，须事先获得具体许可和/或支付费用。请通过 permissions@acm.org 申请许可。
SIGMOD '23，*2023*年*6*月*18*日至*23*日，美国华盛顿州西雅图© 2023年计算机协会。ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/XXXXXXX.XXXXXXX

[1]基于图的方法会生成候选项，并从目前生成的候选项中迭代地找出 K 近邻 。[2] 在基于图的方法中，Q 的大小设为整数 $N_{ef} > K$，因为生成候选项需要第 N_{ef} 个近邻的距离。在其他 AKNN 方法中，Q 的大小设为 K。[3]当距离等于Q中的最大值时，可以选择不将其纳入。

其次，在 AKNN算法涉及的 DCO 中，通常大多数都针对负对象。一个迹象是，当对某个对象执行 DCO 时，距离阈值对应于某个较小的距离（例如，在许多 AKNN算法中，到目前为止观察到的第 K 小距离）。因此，该对象的距离很可能大于阈值，从而属于负对象。我们通过实证验证了这一说法（详情见第2.2节）。例如，对于具有代表性的算法 IVF， [32], 负对象的数量是正对象数量的 60 至 869 倍。综合来看，这些观察表明，对于回答 K近邻查询所涉及的大多数 DCO，FDScanning 都做了过多计算，因此仍有很大空间来提升 DCO [4] 的可靠性（近乎精确）和效率。

据我们所知，文献中尚无研究致力于实现比 FDScanning 效率更高且可靠的 DCO。一种直接的尝试是，在 DCO 中使用一些距离近似技术，例如乘积量化 [23, 24, 32] 和随机投影 [34]，以提高效率。然而，正如文献中普遍发现的 [38, 59] 以及我们实验中的实证结果所证实的那样，这些技术无法在实现显著时间开销节省的同时避免准确率损失，因此很难用于实现效率更高且可靠的 DCO。例如，根据 [59],，在包含一百万个128维向量的数据集 SIFT1M 上，不进行重排序时，没有一种量化算法能够达到超过60%的召回率。事实上，这些技术仅用于生成近邻候选项 [59],，而不用于（在 DCO 中）从生成的候选项中找出它们。

因此，本文提出一种名为 ADSampling 的新方法来实现这一目标。其核心是将对象投影到维度不同的空间中，并基于投影向量执行 DCO，以提高效率。与将所有对象都投影为维度相同的向量的传统随机投影技术 [15, 22, 34, 54],不同，ADSampling 在以下方面具有创新性。首先，在查询阶段，它可以灵活地将不同对象投影为维数不同的向量。其原理是，对于距离查询对象较远的负对象，将其投影到维数较少的空间中就足以实现可靠 DCO；而对于距离查询对象较近的负对象，则应将其投影到维数较多的空间中，以实现可靠 DCO。需要指出的是，对于正对象（即与查询对象的距离不超过阈值的对象），理想情况下其距离不应发生失真。ADSampling 通过两个步骤实现这种灵活性。(1) 在索引阶段（即查询到来之前），先通过随机正交变换 [11, 35] 对对象进行预处理。这一步只是随机旋转对象，不会使其距离失真。(2) 随后，在查询阶段处理不同对象的 DCO 时，对其变换后的向量采样不同数量的维度。我们验证了，这两个步骤生成的采样向量与通过随机投影得到的向量服从相同的分布，因此，基于采样向量得到的近似距离，如同基于随机投影得到的距离一样，能够很好地估计真实距离，同时实现前述维度灵活性。

其次，它会在查询阶段，根据对象上的 DCO 自适应地决定要为每个对象采样的维度数量，而不是在索引阶段预先设定一个固定数量（这需要掌握相关知识，且在实践中难以设定）。具体而言，它会逐步采样变换后的向量中的维度，直到能够根据采样后的向量，有把握地对该对象执行 DCO。利用采样后的向量，它通过计算对象的近似距离来判断是否已有足够证据支持可靠的 DCO，随后基于计算得到的近似距离进行假设检验。这之所以可行，是因为近似距离存在理论误差界（回顾前文所述：ADSampling 投影得到的向量与传统随机投影得到的向量等价）。

与 FDScanning 相比，ADSampling 能以更高的效率实现可靠的 DCO，原因如下。首先，对于每个负对象，我们证明 ADSampling 始终能够返回正确答案，并且其运行时间相对于 D 的期望为对数级（请注意，FDSanning 的运行时间相对于 D 为线性级）。其次，对于每个正对象，它都能以高概率成功，运行时间为 $O(D)$。第三，负对象的数量远多于正对象。本文的主要贡献总结如下。

（1）我们系统地回顾了现有的 AKNN算法，并指出了距离比较操作 （DCO）；该操作在 AKNN算法中无处不在。采用现有方法 FDScanning 时，对于几乎所有 AKNN算法，DCO 的开销都占据总体开销的主导地位；我们通过理论分析和实证研究验证了这一点。（第 2 节）（2）我们提出了一种新方法 ADSampling，可在高维欧几里得空间中以更高效率实现可靠的 DCO。具体而言，在大多数情况下（即针对负对象），ADSampling 的运行时间相对于 D 为对数级，并且始终返回正确答案。（第 3 节）（3）对于一般的 AKNN算法（记为 AKNN），我们用 ADSampling 替换其 DCO 所用的 FDScanning，从而得到一种新算法（记为 AKNN+）。我们证明，AKNN+ 算法以高概率保留对应 AKNN算法的结果，同时显著降低时间复杂度。（第 4 节）（4）我们进一步开发了两种针对特定 AKNN算法的技术，以提高 AKNN+ 算法的成本效益。例如，对于基于图的算法（以 HNSW+ 为代表），我们引入更多近似处理，得到一种新算法（称为 HNSW++）；对于其他算法（以 IVF+ 为代表），我们则通过缓存友好存储提升其成本效益。（第 5 节）（5）我们在真实数据集上进行了大量实验，验证了所提出的技术。例如，在准确率不变的情况下，ADSampling 可使 HNSW 的速度最高提升 2.65 倍，使 IVF 的速度最高提升 5.58 倍。此外，在准确率损失不超过 0.14% 的情况下，它可帮助 HNSW 最多节省 75.3% 的评估维度，并帮助 IVF 最多节省 89.2% 的评估维度。（第 6 节）

本文其余部分安排如下：第7节回顾相关工作，第8节总结全文。

[4]我们致力于实现可靠的 DCO，因为 DCO 用于从生成的候选项中寻找 K 近邻；如果其准确率受到影响，找到的 K 近邻的质量也会大幅下降。

2 距离比较操作 2.1 近邻查询与距离比较操作

设 O 为由 N 个对象组成的数据库，位于 D 维欧几里得空间 $\mathbb{R}^D$ 中，$\mathbf{q}$ 为一个查询对象。本文中，“对象”（分别对应“查询对象”）与“数据向量”（分别对应“查询向量”）可以互换使用。需要指出的是，对 O 的操作可以在查询对象$\mathbf{q}$ 到来之前进行（即索引阶段），而对查询对象$\mathbf{q}$ 的操作只能在其到来之后进行（即查询阶段）。对于对象$\mathbf{o}$，定义其与查询对象$\mathbf{q}$ 之差为$\mathbf{x}$，即$\mathbf{x} = \mathbf{o} - \mathbf{q}$。上下文明确时，我们用对应的向量$\mathbf{x}$ 指代对象$\mathbf{o}$ 。为避免歧义，“对象$\mathbf{o}$的距离”指其与查询向量$\mathbf{q}$之间的距离，记为 dis。**K 近邻（KNN）**查询旨在找出与查询对象$\mathbf{q}$距离最小的前 K 个对象。

本文研究距离比较操作（DCO），其定义如下。

定义 *2.1*（距离比较操作）。给定一个对象$\mathbf{o}$ 和一个距离阈值 r，**距离比较操作**（DCO）用于判断对象$\mathbf{o}$ 的距离是否不大于 r，若是，则返回 dis。特别地，如果 $dis \le r$，我们称对象$\mathbf{o}$ 为正对象；否则称其为负对象。

如第 1 节所述，DCO 在许多 AKNN算法中被广泛使用。这些算法通过计算对象$\mathbf{o}$ 的距离，并将其与 r进行比较，自然地对对象$\mathbf{o}$ 和距离 r 执行 DCO。我们将这种传统方法称为 FDScanning，因为它使用o 的所有维度来计算距离。显然，FDScanning 的时间复杂度为 $O(D)$。

接下来，我们回顾现有的 AKNN算法，并从理论和实证两方面验证 DCO 在这些算法中发挥的关键作用。

2.2 AKNN算法及其DCO

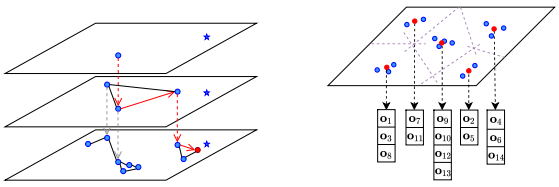


图 1：AKNN算法示意图。

2.2.1 基于图的方法。 基于图的方法是最先进的AKNN算法中的一类，在内存中AKNN查询 [20, 21, 29, 40, 45, 46] 的时间-准确率权衡方面表现突出。这类方法基于数据向量构建图，其中每个顶点对应一个数据向量。一种著名的基于图的方法是分层可导航小世界图（HNSW） [46]。它由若干层组成。第 0 层（基础层）包含所有数据向量，而第 $i+1$ 层仅随机保留第 i 层中的一部分向量。每层的规模随着层级升高而呈指数级递减。具体而言，顶层仅包含一个顶点。在每一层中，一个顶点与其若干个近似最近邻相连，而相邻层之间在各层中，只有当两个顶点表示同一个向量时，它们才会相连。图 1a 展示了 HNSW 图的示意图。

在查询阶段，首先在上层执行贪心搜索，以找到第 0 层（基础层）的一个良好入口。具体而言，搜索从顶层唯一的顶点开始。在每一层中，搜索都会迭代执行贪心搜索。每次迭代时，它会访问当前所在顶点的所有邻居，并前往距离最小的邻居。当所有邻居的距离都不小于当前所在顶点的距离时，搜索便会终止。随后，它转到下一层并重复这一过程，直至到达第 0 层。在第 0 层，它执行贪心束搜索 [60]（最佳优先搜索），这是大多数基于图的方法所采用的算法 [21, 31, 40, 45, 46]。具体而言，贪心束搜索维护两个集合：搜索集 S（按精确距离组织的最小堆）和结果集 R（按精确距离组织的最大堆）。搜索集 S 的大小不受限制，用于保存尚待搜索的候选项。结果集 R 的大小受 N_{ef}限制，用于保存迄今访问过的最近邻，其中数量为 N_{ef}，参数 N_{ef}用于控制时间-准确性权衡。开始时，将第 0 层的一个起始点同时插入 S 和 R。随后，搜索以迭代方式进行。每次迭代时，它从集合 S 中取出距离最小的对象，并枚举该对象的邻居。对于每个邻居，**先检查其与查询对象的距离是否不大于集合 R 中的最大距离；若是，则计算该距离**（即执行一次 DCO）。此外，如果该距离小于 R 中的最大距离，则：（1）将该对象同时放入集合 S 和集合 R（以计算得到的距离作为键）；（2）每当 R 中的对象数超过 N_{ef}时，从集合 R 中取出距离最大的对象，使 R 的大小保持不超过 N_{ef}。当 S 中的最小距离大于 R 中的最大距离时，搜索停止，并返回 R 中距离最小的 K 个对象。我们注意到，上层的贪心搜索对应于一个贪心束搜索过程

with $N_{ef} = 1$。

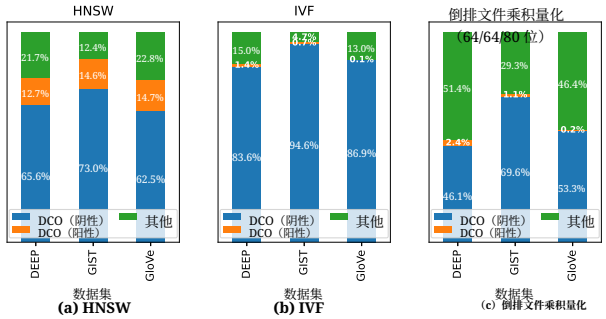


图2：AKNN 算法运行时间的分解。

DCO 与总时间开销的比较。我们回顾一下 HNSW 在采用 FDScanning 进行 DCO 时的时间复杂度。设 N_s 为 HNSW 访问的近邻对象候选项数。那么，DCO 的总开销为 $O(N_sD)$，更新集合 S 和 R 的开销为 $O(N_s \log N_s)$。因此，HNSW 的时间复杂度为 $O(N_sD + N_s \log N_s)$。在实践中，DCO 的总开销应是主要部分，因为在包含数百万个对象的大型数据集中，D 可能达到数百，而 $\log N_s$ 只有几十。我们也通过实证进行了验证。图2a展示了

在三个真实世界数据集上，以 $K = 100$ 为目标召回率时，HNSW 的时间消耗。结果表明，在维度从 256 到 960 不等的数据集上，DCO 占 HNSW 总运行时间的 77.2% 至 87.6%（如柱状图中蓝色和橙色部分所示）。

正对象与负对象。我们通过实验证实，对于 HNSW，负对象上的 DCO 数量显著多于正对象上的 DCO 数量。结果如图2a所示。请注意，图中负对象上的 DCO 成本与正对象上的 DCO 成本之比，反映了负对象数量与正对象数量之比，因为对负对象和正对象执行一次 DCO 的成本相同。根据实验结果，负对象的数量是正对象数量的 4.3 至 5.2 倍。

2.2.2 倒排文件索引。倒排文件索引是另一种常见的近似 K 近邻查询索引方法。根据 [38],，IVF 是近似 K 近邻查询的先进方法之一。事实上，根据第 6.2 节中的实验结果，它在某些数据集上的表现优于 HNSW。在索引阶段，该算法使用 K 均值算法对数据向量进行聚类，为每个簇建立一个桶，并将每个数据向量分配到对应的桶中。随后在查询阶段，对于给定的查询对象，该算法首先根据质心选出最近的 N_{probe} 个簇，将这些簇对应桶中的所有向量作为候选项取出，然后从取出的向量中找出近邻。此处，N_{probe} 是一个用于控制时间-准确性权衡的用户参数。在查找近邻时，一种常用方法是维护一个由大小为 K 的最大堆实现的 K 近邻集合 K。然后扫描所有候选项，对于每个候选项，**检查其距离是否不大于 K，若是，则计算其距离**（即执行一次 DCO）。如果 K 未满，此处的最大距离定义为 $+\infty$。如果该距离小于 K 中的最大距离，则将该候选项更新至 K 中（使用计算出的距离作为键）。最后返回 K 中的对象。图1b展示了 IVF 结构。

DCO 与整体时间开销的比较。我们回顾一下 IVF 在采用 FDScanning 执行 DCO 时的时间复杂度。设 N_s 为候选对象的数量。IVF 的总开销为 $O(N_sD + N_s \log K)$，其中第一项是 DCO 的开销，第二项是更新 K 的开销。可以看出，DCO 的开销占主导地位。我们像对 HNSW 所做的那样，通过实证进行验证。结果如图2b所示。根据结果，在维度从 256 到 960 不等的数据集上，DCO 耗时占 IVF 总运行时间的 85.0% 至 95.3%。

正对象与负对象的比较。我们通过实证验证，对于 IVF，负对象上的 DCO 数量显著多于正对象上的 DCO 数量。结果如图2b所示。根据结果，负对象的数量是正对象的 60 至 869 倍。

2.2.3 其他 *AKNN* 算法。在其他 AKNN算法中，包括基于树、基于哈希和基于量化的方法，DCO 也无处不在。基于树的方法 [8, 12, 14, 47, 51]通过树路由生成候选向量，并使用 DCO 找出近邻（与 IVF 类似）。基于哈希的方法 [15, 22, 27, 28, 44, 54, 55, 62] 通过哈希编码生成候选向量，并使用 DCO 找出近邻（与 IVF 类似）。基于量化的方法 [4, 5, 23, 24, 32, 57] 使用较短的量化编码生成候选项，并通过 DCO 进行重排序（以找出近邻，与 IVF 类似）。对于基于树和基于哈希的方法，DCO 的开销占主导地位，因为（1）一次树路由或哈希桶探测会生成多个候选项（需要执行多次 DCO），并且（2）树路由和哈希桶探测比 DCO 快得多（其时间复杂度为 $O(D)$）。对于基于乘积量化的方法 [5, 23, 24, 32],，DCO 会在其重排序阶段使用，但其开销并不占主导地位，因为主要开销在于计算量化编码。作为比较，我们展示了典型设置 [32, 57]下，倒排文件乘积量化（IVFPQ）这一基于量化的方法 [32], 在图2c中的时间分解结果。

3 ADSampling方法

回想一下，我们的目标是实现可靠的 DCO，同时比 FDScanning 具有更高的效率。为此，我们开发了一种名为 ADSampling 的新方法。其核心是将对象投影到维度更少的向量上，并基于投影向量执行 DCO，以提高效率。与传统且广泛采用的随机投影技术 [15, 22, 34, 54], 不同，ADSampling 不会将所有对象投影到维度相同的向量上，而是在以下方面有所创新。首先，在查询阶段，它可以灵活地将不同对象投影到维度数不同的向量上。我们将在第3.1节详细阐述这一思路的实现方式。其次，它会在查询阶段根据对对象执行的 DCO，自适应地决定每个对象需要采样的维度数，而不是在索引阶段预先设定一个固定的维度数（这需要大量先验知识，在实践中也难以设定）。我们将在第3.2节详细阐述这一思路的实现方式。此外，我们在第3.3节总结了 ADSampling，并证明对于负对象，其时间复杂度关于 D 呈对数级（显著优于 FDScanning 的时间复杂度 $O(D)$）。

3.1 对随机变换后的向量进行维度采样

为了提高 DCO 的效率，一个自然的想法是对对象进行随机投影 [34, 56]（即将对象（具体而言是其向量）与一个 $\mathbb{R}^{d\times D}$ 随机矩阵 P 相乘，其中 $d < D$ [5]），然后利用基于投影向量计算得到的近似距离来执行 DCO，即 $\sqrt{D/d}\|P\mathbf{x}\|$。众所周知，近似距离满足一个集中不等式，如下引理所示 [56]。

引理 3.1。对于给定对象 $\mathbf{x} \in \mathbb{R}^D$，随机投影 $P \in \mathbb{R}^{d\times D}$ 以 ϵ 乘法误差界保留其欧几里得范数的概率为

$$\mathbb{P}\left\{\left|\sqrt{\frac{D}{d}}\|P\mathbf{x}\| - \|\mathbf{x}\|\right| \leq \epsilon\|\mathbf{x}\|\right\} \geq 1 - 2e^{-c_0 d\epsilon^2} \tag{1}$$

[5]随机投影所用的随机矩阵有多种类型 [1, 15, 34, 36]。在本研究中，我们所说的随机投影是基于随机正交矩阵的随机投影，该矩阵可以通过对随机高斯矩阵进行正交化得到，其元素是相互独立的标准高斯随机变量 [11, 34, 35, 56]。

其中 c_0 是一个常数因子，且 $\epsilon \in (0, +\infty)$。

然而，对象一旦经过投影，相应的近似距离就会具有固定的分辨率。因此，在查询阶段，无法灵活地为不同对象采用不同的降维维度（相应地，近似距离也具有不同的分辨率）。

我们的目标是在查询阶段灵活地将不同对象投影为维度数不同的向量。为此，我们提出对对象进行随机变换（通过随机正交变换 [24, 34, 56]，从几何角度看即为随机旋转），然后灵活地对变换后的向量进行维度采样，以计算近似距离。形式化地说，给定一个对象$\mathbf{x}$，我们首先对$\mathbf{x}$应用一个随机正交矩阵 $P' \in \mathbb{R}^{D\times D}$，然后从中采样 d'行（为简便起见，取前 d'行）。结果记为（$P'\mathbf{x}$）$|_{[1,2,\ldots,d']}$。此方法有两个优点。首先，我们实现了灵活性，因为在查询阶段，可以针对不同的 d'灵活地从旋转后的向量中采样 d'个维度。其次，我们获得了有保证的误差界，因为对变换后的向量采样 d'个维度，等价于通过随机投影得到一个 d'维向量，下面将对此进行说明。

回想一下，对$\mathbf{x}$进行随机投影，就是将随机投影矩阵 $P \in \mathbb{R}^{d\times D}$作用于$\mathbf{x}$，结果记为 $P\mathbf{x}$。我们声称，$(P'\mathbf{x})|_{[1,2,\ldots,d']}$（我们提出的方法的结果）与 $P\mathbf{x}$（随机投影的结果）具有相同的分布。这基于矩阵乘法的一条基本性质：在矩阵乘法之前或之后进行行采样，结果相同：

$$(P'\mathbf{x})|_{[1,2,\ldots,d']} = P'|_{[1,2,\ldots,d']}\mathbf{x} \tag{2}$$

我们注意到，$P'_{[1,2,\ldots,d']}$对应于用于随机投影的随机矩阵 P，因为生成随机投影矩阵 P的一种常规方法是对随机正交矩阵 [11]进行行采样。因此，针对原始对象的随机投影集中不等式（如公式 (1) 所示）可应用于随机变换后的向量的维度采样，为我们接下来的讨论奠定了坚实基础。

我们将变换后的向量记为$\mathbf{y} := P'\mathbf{x}$。根据采样维度，我们可以计算$\mathbf{x}$的近似距离，记为 dis'，如下所示：

$$dis' := \sqrt{\frac{D}{d'}}\left\|\mathbf{y}|_{[1,2,\ldots,d']}\right\| \tag{3}$$

其中 d' 是采样维度的数量。值得注意的是，基于 d' 个采样维度计算近似距离的时间复杂度为 $O(d')$。此外，当采样了全部 D 个维度时，基于采样维度计算得到的距离 dis' 将等于真实距离 dis，这是因为随机正交变换会保持任意向量的范数不变（因为它只会旋转空间，而不会扭曲距离）。

3.2 结合假设检验的增量采样

剩下的一个问题是，如何确定需要采样多少个$\mathbf{y}$ 维度，才能有足够把握地得出关于 DCO 的结论（即，决定是否 $dis \le r$）。直观来说，采样维度越多，近似距离dis'就越准确，我们也就越能有把握地得出结论。另一方面，采样更多维度会增加计算近似距离的成本（因为该成本与采样维度的数量呈线性关系）。我们的目标是采样尽可能少的维度，同时确保足以有把握地得出结论。

具体而言，我们提出以增量 方式对$\mathbf{y}$ 的维度进行采样，即从少量维度开始。如果根据当前采样的维度无法得出有足够把握的结论，我们就继续采样更多维度，直到能够得出有足够把握的结论，或已采样全部维度。因此，问题就归结为：对于一定数量（例如 d'个）的采样维度，我们能否得出有足够把握的结论？用统计学术语来说，观测到的距离 dis'（基于采样维度计算得出）是真实距离 dis的一个估计量，其分布仅取决于真实值dis和采样维度的数量 d'。我们的任务是根据观测值 dis'，对真实值 dis得出结论（即判断 $dis \le r$是否成立）。这正是假设检验通常要解决的问题。受此启发，我们提出利用假设检验来解决该问题。具体而言，我们按如下方式进行假设检验。

(1) 我们定义原假设 H_0：$dis \le r$及其备择假设 H_1：$dis > r$。
(2) 我们使用 dis'作为 dis的估计量。引理 3.1 给出了dis'与 dis之间的关系（即，dis'与 dis之间的差值以 $\epsilon \cdot dis$为界，失败概率至多为 $2\exp(-c_0 d\epsilon^2)$）。(3) 我们将显著性水平 p设为 $2\exp(-c_0\epsilon_0^2)$，其中 ϵ_0 是一个需要通过经验调节的参数。由此，观测到的 dis'远大于 r（即，$dis' > (1+\epsilon_0/\sqrt{d})\cdot r$）这一事件的概率低于显著性水平 p（可根据引理 3.1，在$\epsilon = \epsilon_0/\sqrt{d}$且 H_0：$dis \le r$的条件下验证）。(4) 我们检查该事件是否发生（$dis' > (1+\epsilon_0/\sqrt{d})\cdot r$）。若发生，我们就可以拒绝$\neg H_0$，并以足够的置信度得出结论 H_1：$dis > r$；否则，我们无法得出这一结论。

假设检验的结果有三种情况。情况 1：我们拒绝该假设（即，我们得出结论 $dis > r$）并且 $d < D$。在这种情况下，时间开销（主要用于计算近似距离）为 $O(d)$，小于用 $O(D)$ 时间计算真实距离的开销。情况 2：我们无法拒绝该假设，并且 $d < D$。在这种情况下，我们会继续增量采样 $\mathbf{y}$ 的更多维度，并再次进行假设检验。情况 3：$d = D$。在这种情况下，我们已对 $\mathbf{y}$ 的所有维度进行采样，基于采样向量计算出的近似距离等于真实距离。因此，我们可以执行精确的 DCO。我们注意到，结合（可能是序贯的）假设检验进行增量维度采样的时间开销，严格小于 $O(D)$（当过程在情况 1 结束时），而等于 $O(D)$（当过程在情况 3 结束时）。

我们注意到，在相似性搜索的背景下，假设检验也曾用于确定局部敏感哈希所需的哈希值数量 [10, 52]。我们的技术与 [10, 52] 之间的差异

包括：(1) 我们的方法基于对变换后的向量维度进行采样的随机过程，而 [10, 52] 则基于对哈希函数进行采样，这导致两者的假设检验存在显著差异；(2) 我们的方法针对欧几里得距离函数，而[10, 52] 针对杰卡德和余弦相似度度量等相似度函数（将后者适配到欧几里得空间仍非易事）；(3) 我们的方法保证其效果不会差于计算精确距离的方法（在我们的情形下，即 FDScanning ），因为当采样完所有维度后，它就能得到精确距离；而 [10, 52] 没有这样的保证（当它们采样完所有哈希函数后仍无法给出确定结果时，就必须从头重新计算精确相似度）。

3.3 总结与理论分析

摘要。我们在算法 1 中总结了 ADSampling 的流程。该算法以变换后的数据向量 $\mathbf{o}'$、变换后的查询向量 $\mathbf{q}'$ 和距离阈值 r 为输入，并输出关于 $dis \leq r$的 DCO 结果：是则为 1（此时还会返回 dis ），否则为 0。需要指出的是，数据向量的变换在索引阶段进行，其开销可以由同一数据库上的所有后续查询分摊。查询向量的变换则在查询到来时于查询阶段进行，其开销可以由回答同一查询所涉及的所有 DCO 分摊。具体而言，算法使用变量 d 维护采样维度的数量，其中 $d = 0$ 初始为（第 1 行）。随后，当 $d < D$ 时，算法执行迭代过程（第 2 行）。每次迭代中，算法会逐步多采样一些维度，并相应地更新 d 和近似距离 dis'（第 3–4 行），然后基于近似距离 dis' 进行假设检验，并将 $dis \leq r$ 作为原假设（第 5 行）。最后，算法会按第 3.2 节所述的三种情况之一返回结果（第 6–11 行）。

算法1： ADSampling

输入： 一个变换后的数据向量 $\mathbf{o}'$、一个变换后的查询向量 $\mathbf{q}'$ 和一个距离阈值 r

输出： DCO 的结果（即，$dis \leq r$）：1 表示是，0 表示否；若结果为 1，还会返回精确距离

```
1  将采样维度的数量 d 初始化为 0
2  当 d < D 时执行
3      逐步再采样一些维度 y_i，并
         y_i = o'_i − q'_i
4      更新 d，并相应调整近似距离 dis'
5      对原假设进行假设检验
         H_0 as dis ≤ r 基于近似距离 dis'
6      如果 H_0 被拒绝且 d < D 那么                    // 情况 1
7          返回 0
8      否则如果 H_0 未被拒绝且 d < D 则                // 情况 2
9          继续
10     else                                          // 情况 3
11         若 1（且 dis'），如果 dis' ≤ r；否则为 0
```

失败概率分析。请注意，ADSampling 会在情况 1（假设被拒绝且 $d < D$）或情况 3（其中 $d = D$）。当算法在情况 3 下终止时，不会发生失败，因为此时近似距离 dis' 等于真实距离 dis，因此 DCO 结果是精确的。当算法在情况 1 下终止时，如果 $dis \leq r$成立，就会发生失败，因为此时算法会得出 $dis > r$的结论（即拒绝原假设）。下面分析失败概率。正如第 3.2 节所述，我们可以通过 ϵ_0控制失败概率。下面的引理给出了 ϵ_0 与采用 ADSampling 的 DCO 的失败概率之间的关系。

引理 3.2。对于 D 维空间中的 *DCO*，ADSampling 的失败概率为

$$\mathbb{P}\{failure\} = 0 \ if \ dis > r \tag{4}$$

$$\mathbb{P}\{failure\} \leq \exp\left(-c_0\epsilon_0^2 + \log D\right) \ if \ dis \leq r \tag{5}$$

证明。对于 $dis > r$ 的情况，正确性显而易见；对于另一种情况（$dis \leq r$），可如下验证。

$$\mathbb{P}\{failure\} = \mathbb{P}\left\{\exists d < D, dis' > (1 + \epsilon_0/\sqrt{d}) \cdot r\right\} \tag{6}$$

$$\leq \sum_{d=1}^{D-1} \mathbb{P}\left\{dis' > (1 + \epsilon_0/\sqrt{d}) \cdot dis\right\} \tag{7}$$

$$\leq \sum_{d=1}^{D-1} \exp\left(-c_0\epsilon_0^2\right) \leq \exp\left(-c_0\epsilon_0^2 + \log D\right) \tag{8}$$

其中，(6) 是因为当且仅当我们意外地拒绝某个 $d < D$ 的假设时，才会发生失败；(7) 应用了并集界以及 $dis \leq r$ 这一事实；(8) 则由引理 3.1 得出。□

ˆ时间复杂度分析。令 D 表示 ADSampling 采样的维度数。显然，ADSampling 的时间复杂度为 $O(\hat{D})$。由于该方法具有随机性，$\hat{D}$ 是一个随机变量。接下来，我们分析 $\hat{D}$ 的期望，记为 $\mathbb{E}[\hat{D}]$。首先，由于 $\hat{D}$ 始终至多为 D，因此我们知道 $\mathbb{E}[\hat{D}] \leq D$。此外，对于一个满足 $dis > r$ 的负对象上的 DCO，我们可以推导出 $\mathbb{E}[\hat{D}]$ 取决于 ϵ_0 和 $\alpha = (dis - r)/r$（我们称之为 dis 与 r 之间的距离差），如下所示（详细证明见附录 A ）。

引理 3.3. 当使用 ADSampling 对对象进行 *DCO*，且阈值为 r、$dis > r$ 时，令 $\alpha = (dis - r)/r$，则有

$$\mathbb{E}\left[\hat{D}\right] = O\left[\min\left(D, \alpha^{-2} \cdot \epsilon_0^2\right)\right] \tag{9}$$

上述结果与以下直觉十分吻合：(1) 当 dis 与 r 之间的距离差，即 $(dis - r)/r$，较大时，为了得出足够有置信度的结论，需要采样的维度更少；(2) 当 ϵ_0 较大时（即假设检验的显著性水平较小，这意味着对置信度的要求更高），需要采样的维度更多。

我们进一步推导 ADSampling 的时间-准确性权衡。

定理 3.4。当在对象和阈值 r 上使用ADSampling进行 *DCO*，且满足 $dis > r$时，令 $\alpha = (dis - r)/r$，我们有

$$\mathbb{E}\left[\hat{D}\right] = O\left[\min\left(D, \frac{1}{\alpha^2}\log\frac{D}{\delta}\right)\right] \tag{10}$$

以使其（正对象的）失败概率至多为 δ。

证明。令引理 3.2 中的失败概率等于 δ，得到相应的 ϵ_0。然后将 ϵ_0 代入引理 3.3，即得该定理。□

ADSampling 与 FDScanning 的比较。与 FDScanning 相比，ADSampling 以牺牲正对象的准确率为代价（定理 3.4），将负对象的复杂度从线性降低为相对于 D 呈对数级。我们强调，正对象的失败概率呈二**次指数级**衰减（引理 3.2），而负对象的时间复杂度则呈二**次**增长（引理 3.3），二者均与 ϵ_0有关。这表明，要实现近乎精确的 DCO，我们只需对少量维度进行采样。我们在第6.2.6节中通过实验验证了这些结果。结果表明，使用 ADSampling 作为插件时，一个精确的 K 近邻算法（即线性扫描）在 GIST 数据集上平均每个向量只需使用 55 个维度（原本为 960 个维度），即可达到 >99。9% 的召回率。

4 AKNN+：借助 ADSampling 作为插件组件改进 AKNN算法

回顾一下，AKNN算法（记作 AKNN）可以是现有众多算法中的任意一种 [15, 21, 32, 46, 47], ，其中涉及许多 DCO。在相关文献中，通常采用 FDScanning 执行 DCO，时间复杂度为 $O(D)$。鉴于 ADSampling 能够以更高的效率执行可靠 DCO，一个自然的想法是采用 ADSampling 执行 DCO，以改进 AKNN算法。具体而言，由于 ADSampling 基于经过随机变换的数据向量和查询向量，因此在任何查询到来之前，我们会对所有数据向量进行随机变换；查询到来时，则对查询向量进行随机变换。随后，我们基于变换后的数据向量和查询向量运行 AKNN算法。请回顾，变换数据向量的时间开销可以分摊到不同查询中，而变换查询向量的时间开销可以分摊到回答该查询所涉及的许多不同 DCO 中。在 AKNN算法的运行过程中，每当执行 DCO 时，我们就使用 ADSampling 方法。例如，对于 HNSW 等基于图的方法，在将新访问对象的距离与结果集 R 中的最大距离进行比较时，我们使用 ADSampling 方法。对于 IVF 等其他 AKNN算法，在从生成的候选项中选出最终的 K近邻时，我们使用 ADSampling 比较候选对象的距离与当前维护的 K 近邻集合 K 中的最大距离。

对于采用 ADSampling 处理 DCO 的 AKNN 算法 AKNN，我们称其为 AKNN+。例如，我们分别将采用 ADSampling 处理 DCO 的 HNSW 和 IVF 称为 HNSW+和 IVF+。

理论分析。回想一下，ADSampling 以牺牲正对象上 DCO 的准确率为代价，提高了负对象上 DCO 的效率。下面我们展示 AKNN+ 未能返回与 AKNN 相同结果的概率，与 AKNN+ 中涉及的负对象上的 DCO 时间复杂度之间的关系。基本上，为了保留 AKNN 返回的结果，只需确保所有 DCO 都产生正确结果，其数量至多为 N。于是根据并集界，AKNN+ 的失败概率可由各个单独 DCO 的失败概率之和上界。因此，将 ADSampling 的失败概率设为 $\delta = \delta\prime/N$，即可得到以下推论。

推论4.1. 令 $\delta\prime$ 表示 AKNN+ 未能返回与AKNN相同结果的概率。对距离差为 α 的负对象执行 *DCO* 的期望时间复杂度降低至

$$\mathbb{E}\left[\hat{D}\right] = O\left[\min\left(D, \frac{1}{\alpha^2}\log\frac{DN}{\delta\prime}\right)\right] \tag{11}$$

而剩余的时间开销（对正对象执行 *DCO* 及其他计算）保持不变。

此外，对于那些一次性生成所有候选对象的 AKNN+ 算法（例如 IVF），只需为近邻对象（K 个对象）而非所有对象（至多 N 个对象）生成正确的 DCO 结果，即足以返回与相应 AKNN 算法相同的结果。这是因为，一旦我们为真正的近邻对象执行 DCO 并得到正确结果，就能获得它们的精确距离（注意，它们全都是正对象），从而确保将它们作为最终答案返回。因此，我们得到以下推论。

推论4.2。 $\delta\prime$ 表示 AKNN+ 未能返回生成候选项的 K 近邻的概率。对于距离差为 α 的负对象，*DCO* 的期望时间复杂度降低至

$$\mathbb{E}\left[\hat{D}\right] = O\left[\min\left(D, \frac{1}{\alpha^2}\log\frac{DK}{\delta\prime}\right)\right] \tag{12}$$

其余时间开销（对正对象执行 *DCO* 以及其他计算）保持不变。

5 AKNN++：通过特定于算法的优化改进 AKNN+ 算法

5.1 HNSW++：迈向更高近似度

回顾一下，HNSW+ 维护一个结果集 R，其中包含一个大小为 N_{ef}、以距离为键的最大堆，且 $N_{ef} > K$。对于每个新生成的候选对象，它都会检查其距离是否不大于 R 中（某个对象的）最大距离；如果是，则将该对象插入集合 R。具体来说，它使用 ADSampling，以 R 中的最大距离作为阈值距离，对每个候选对象执行 DCO。我们发现集合 R 发挥着两个作用。首先，它维护迄今生成的候选项中距离最小的 K 近邻。这些 K 近邻将在算法结束时作为输出返回。其次，它维护迄今生成的候选项中 N_{ef}^{th}最大的距离。该距离在算法执行过程中用作 DCO 的阈值距离，其结果会影响候选项的生成。具体而言，如果 HNSW+ 生成的候选对象的距离不超过 R 中的 N_{ef}^{th}距离，则会将其添加到搜索集 S，以便进一步生成候选项。

这种双重角色设计源于在 HNSW+ 中，精确距离被用于同时实现这两种功能。如图 3a 所示，HNSW+ 始终使用精确距离（深绿色）维护 R 和 S，而 R 中最前面的 K 个对象就是 KNN 对象。对于第一种功能（维护 K近邻），使用精确距离是可取的，因为算法的输出是根据精确

距离定义的。然而，我们认为，对于第二种功能（维护 N_{ef}^{th} 最大距离），这样做可能不够经济，因为利用该距离生成候选项的过程是一种启发式方法（即贪心束搜索），而且使用近似距离也可能取得良好效果。

因此，我们提出将 R 的两种角色解耦，分别维护两个集合 R_1 和 R_2（如图3b所示）。集合 R_1 的大小为 K，依据精确距离（深绿色）构建。集合 R_2 的大小为 N_{ef}，依据距离构建，该距离可以是精确的，也可以是近似的。具体而言，对于每个新生成的候选对象，算法会检查其距离是否不大于集合 R_1中的最大距离；如果是，则将该候选对象插入集合 R_1。此外，这种 DCO 还会产生一个副产品，即 ADSampling 终止时观测到的距离 dis'（浅绿色）；该距离可能是精确的（如果已采样所有 D 维度），也可能是近似的（如果在 $d < D$ 时终止）。随后，它根据观测到的距离维护集合 R_2 和集合 S，方式与 HNSW+ 分别维护 R 和 S 类似。我们将基于这种角色解耦设计的算法称为 HNSW++。

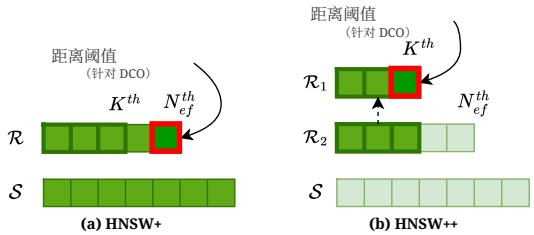


图3：HNSW+与HNSW++

理论分析。 我们注意到，与 HNSW+ 不同，后者以高概率返回与 HNSW 相同的结果，而 HNSW++ 并不旨在返回与 HNSW 相同的结果（尽管实践中它返回的结果几乎相同，这一点已在第6.2.2节中得到验证）。具体而言，HNSW++ 会生成一组候选项，这组候选项可能不同于 HNSW+ 或 HNSW 生成的候选项。在生成的候选项中，HNSW++ 保证以高概率返回其近邻，因为它仍通过 ADSampling 维护近邻，其保证与推论4.2中的保证相同。

HNSW++ 与 HNSW+。 与 HNSW+ 相比，HNSW++ 预计具有更好的时间-准确性权衡，原因如下。首先，考虑时间开销。在 ~~HN~~SW++ 中，对于每个 DCO，阈值距离是第 K^{th} 大的距离，小于 HNSW+ 中使用的距离（即第 N_{ef}^{th} 大的距离）。相应地，在 HNSW++ 中，α 值（定义为 $(dis - r)/r$）大于 HNSW+ 中的值。因此，根据第3.3节中 ADSampling 的时间复杂度分析，此 DCO 的时间开销会小于 HNSW+ 中的时间开销。其次，考虑效果。尽管 HNSW++ 和 ~~HNSW+~~ 使用不同的距离来生成候选项，但我们预计它们生成的候选项质量相近，因为：(1) 两种算法使用的距离应该相近（在某些情况下甚至相同）；(2) 用于生成候选项的方法，即贪心束搜索，具有启发式性质，没有充分迹象表明它更偏向于精确距离而非近似距离。

备注。我们指出，HNSW++ 技术也可以用于其他基于图的方法 [21, 31, 40, 45, 46]。这是因为这些算法在查询阶段也会基于集合 R 进行贪心束搜索。

5.2 IVF++：迈向缓存友好

在原始 IVF 算法中，同一簇中的向量按顺序存储。计算这些向量的距离时，算法会依次扫描这些向量的所有维度，因而具有很强的引用局部性，所以缓存友好。图4a展示了相应的数据布局（箭头所示）以及所需的数据（彩色背景所示）。虽然 IVF+ 扫描的维度少于 IVF，但采用相同的数据布局时，它并不具备缓存友好性。具体而言，当 IVF+ 结束对一个数据向量的维度采样时，后续维度很可能已经从主存加载到缓存中，尽管这些维度并不需要。图4b展示了相应的数据布局和所需的数据。

我们提出重新组织候选项的数据布局，并相应调整候选项输入 ADSampling 的维度顺序，从而实现更具缓存友性的数据访问。回想一下，对于每个候选对象，ADSampling 一定会先采样该候选对象的若干个维度（例如 d_1个），然后根据假设检验的结果逐步采样更多维度。也就是说，每个候选对象的前 d_1 个维度必定会被访问。基于这一点，我们将所有候选项的前 d_1 个维度按顺序存储在数组 A_1 中，并将所有候选项剩余的 $D - d_1$ 个维度按顺序存储在另一个数组 A_2中。我们指出，重新组织数据布局的过程可以在索引阶段完成。在查询阶段，对候选项使用 ADSampling 进行 DCO 时，我们按以下顺序访问候选项的维度：第一个候选对象的前 d_1 个维度，第二个候选对象的前 d_1个维度，.....，最后一个候选对象的前 d_1 个维度；第一个候选对象的 $D - d_1$ 个维度，.....，最后一个候选对象的 $D - d_1$ 个维度。图4c展示了相应的数据布局和所需的数据。我们将由此得到的算法称为 IVF++。IVF++ 和 IVF+ 产生的结果完全相同，但前者的缓存友好性更好，因为它利用了所有候选项前 d_1 个维度的引用局部性。

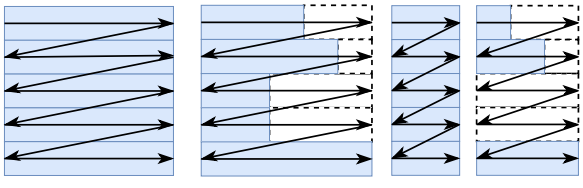


图4：数据布局和所需数据

理论分析。 由于 IVF++ 和 IVF+ 仅数据布局不同，因此它们具有相同的理论保证（推论4.2）。

备注。 我们指出，用于提高 IVF+ 缓存友好性的技术，也可用于改进其他一些 AKNN+ 算法，包括基于树的方法 [8, 14, 47, 51], 、基于量化的方法 [5, 32] 以及基于哈希的方法 [15, 16, 22]。这是因为所有这些算法都会批量生成候选项，然后对候选项重新排序，以找出 K 近邻。

6 实验

6.1 实验设置

数据集。 我们使用了六个规模和维度各不相同的公开数据集6，其详细信息见表1。这些数据集已被广泛用于对 AKNN 算法进行基准测试 [38, 40, 43]。我们指出，这些公开数据集同时提供数据和查询向量。

表1：数据集统计信息

数据集	Size	D	查询集大小	数据类型
Msong	992,272	420	200	音频
DEEP	1,000,000	256	1,000	图像
Word2Vec	1,000,000	300	1,000	Text
GIST	1,000,000	960	1,000	图像
GloVe	2,196,017	300	1,000	Text
Tiny5M	5,000,000	384	1,000	图像

算法。 对于可靠 DCO，我们将提出的方法 ADSampling 与传统的 FDScanning 和 PDScanning（部分维度扫描）进行比较，下面将对此作出说明。PDScanning 会逐步扫描原始向量的各个维度；当基于已扫描的部分 d维度计算出的距离（即 $\sqrt{\sum_{i=1}^{d} x_i^2}$）大于距离阈值 r 时，便终止扫描。需要说明的是，PDScanning 从零个维度开始扫描，而不是预先设定维度数量，因为：（1）维度数量难以确定；（2）鉴于维度是逐步扫描的，从某个特定数量的维度开始扫描与从零个维度开始扫描的性能非常相近。还需指出，PDScanning 是一种用于 DCO 的精确算法，其最坏情况下的时间复杂度为 $O(D)$。我们将采用 PDScanning 处理 DCO 的 AKNN算法称为 AKNN*，而将采用进一步优化数据布局的版本称为 AKNN**（仅适用于 IVF）。我们不将乘积量化、随机投影等距离近似方法纳入比较，因为如第 1 节所述并在第 6.2.4 节进一步验证的那样，这些方法难以实现可靠的 DCO。对于 AKNN算法，我们主要关注 HNSW [46] 和 IVF [32]，以展示 DCO 的应用背景，因为基准测试 [3, 40]表明，它们分别代表了两种最先进的 AKNN算法。需要说明的是，这些方法在业界得到广泛采用（包括 Faiss [33], 、Milvus [58] 和 PASE [61]）。为使比较更加全面，我们还纳入了表现最佳的基于树的方法之一 Annoy [2]（基准测试见 [3, 40]）以及一种基于哈希的方法 PMLSH [62]。需要指出的是，与 HNSW 和 IVF 相比，这些方法在时间-准确率权衡方面的表现并不理想。由于篇幅有限，相关结果列于附录 B。

性能指标。 我们使用两个指标来衡量准确性：(1) 召回率 [3, 32, 40, 46], , 即成功检索到的真实 K 近邻数量与 K 的比值；(2) 平均距离比 [22, 27, 48, 49, 54], , 即检索到的 K 对象相对于真实 K 近邻的距离比的平均值（等于距离相对误差的平均值加一）。我们采用每秒查询数（QPS），即每秒处理的查询数量，来衡量效率。请注意，查询时间按端到端方式测量（即包含对查询向量进行随机变换的时间）。我们在第 6.2.3 节分解时间开销。我们还测量算法评估的维度总数。对于 AKNN算法，这表示候选项的维度总数（因为每个候选对象的所有维度都用于计算其距离）。对于 AKNN+ (AKNN*) 和 AKNN++ (AKNN**) 算法，这表示候选项采样（扫描）的维度总数（因为对每个候选对象，仅使用采样（扫描）的维度来近似计算其距离）。所有提及的指标均在整个查询集上取平均值。

实现。 AKNN 算法的实现分为两个阶段。在索引阶段，我们首先使用 NumPy 库生成一个随机正交变换矩阵，将其存储起来，并将该变换应用于所有数据向量。然后，将变换后的向量（对于 AKNN、AKNN* 和 AKNN**，则为原始向量）输入现有的 AKNN 算法。具体而言，对于 HNSW、HNSW+、HNSW++ 和 HNSW*（请注意，它们具有相同的图结构），我们的实现基于 hnswlib [46], ；对于 IVF、IVF+、IVF++、IVF* 和 IVF**（请注意，它们具有相同的簇结构），我们基于 Faiss 库 [33]实现 K-means 聚类。在查询阶段，所有算法均使用 C++实现。对于新的查询对象，在运行 AKNN+ 和 AKNN++算法时，我们首先使用 Eigen 库 [25] 对查询向量进行变换，以实现快速矩阵乘法（AKNN、AKNN* 和 AKNN** 不涉及变换）。然后，将该向量输入 AKNN、AKNN+、AKNN++、AKNN* 和 AKNN** 算法。按照 [40, 60], , 我们禁用所有特定于硬件的优化，包括 SIMD、内存预取和多线程（包括 Eigen 库中的相关优化），以便专注于比较算法本身。

参数设置。对于 HNSW，有两个预设参数用于控制图的构建，分别是 M，用于控制连接邻居的数量，以及 $efConstruction$，用于控制近似最近邻的质量。我们遵循其原始工作 [46] 中的参数设置，其中参数设为 $M = 16$ 和$efConstruction = 500$。对于 IVF，根据 Faiss 库 7 的建议，簇的数量应约为数据库基数的平方根。由于我们关注的是百万级数据集，因此将其设为 4,096 ϵ_0 1 5, 。对于 1 8, 2 1, 2 4, 2 7 3 0 6 2 5 ADSampling，我们改变其取值为 . . . . . 和 ., 并在第 . . . 节研究其影响。根据结果，我们采用 $\epsilon_0 = 2.1$ 作为默认设置。请注意，在 ADSampling 中，它会逐步对数据向量的部分维度进行采样，并在迭代过程中执行假设检验。为避免频繁进行假设检验带来的开销，我们在每次迭代中采样Δ_d 个维度。默认情况下，我们设定 $\Delta_d = 32$。其参数研究也见第6.2.5节。

所有 C++ 源代码均在 Ubuntu 20.04 LTS 下使用 g++ 9.4.0 编译，并采用 -O3 优化。Python 源代码（用于索引阶段）在 Python 3.8 上运行。所有实验均在配备 AMD Threadripper PRO 3955WX 3.9GHz 16 核/32 线程处理器和 64GB 内存的机器上进行。代码和数据集可在 https://github.com/gaoj0017/ADSampling 获取。

[6]请注意，我们的技术几乎不会额外占用空间（唯一的额外空间开销来自 $D \times D$ 随机正交矩阵，与规模为 N、维度为 D的庞大数据库相比，这点开销可以忽略不计）。因此，它们不会影响 AKNN 算法的可扩展性。为验证这些技术在加速 AKNN 算法方面的有效性，我们将重点研究百万级数据集。

[7]https://github.com/facebookresearch/faiss/wiki/Guidelines-to-choose-an-index

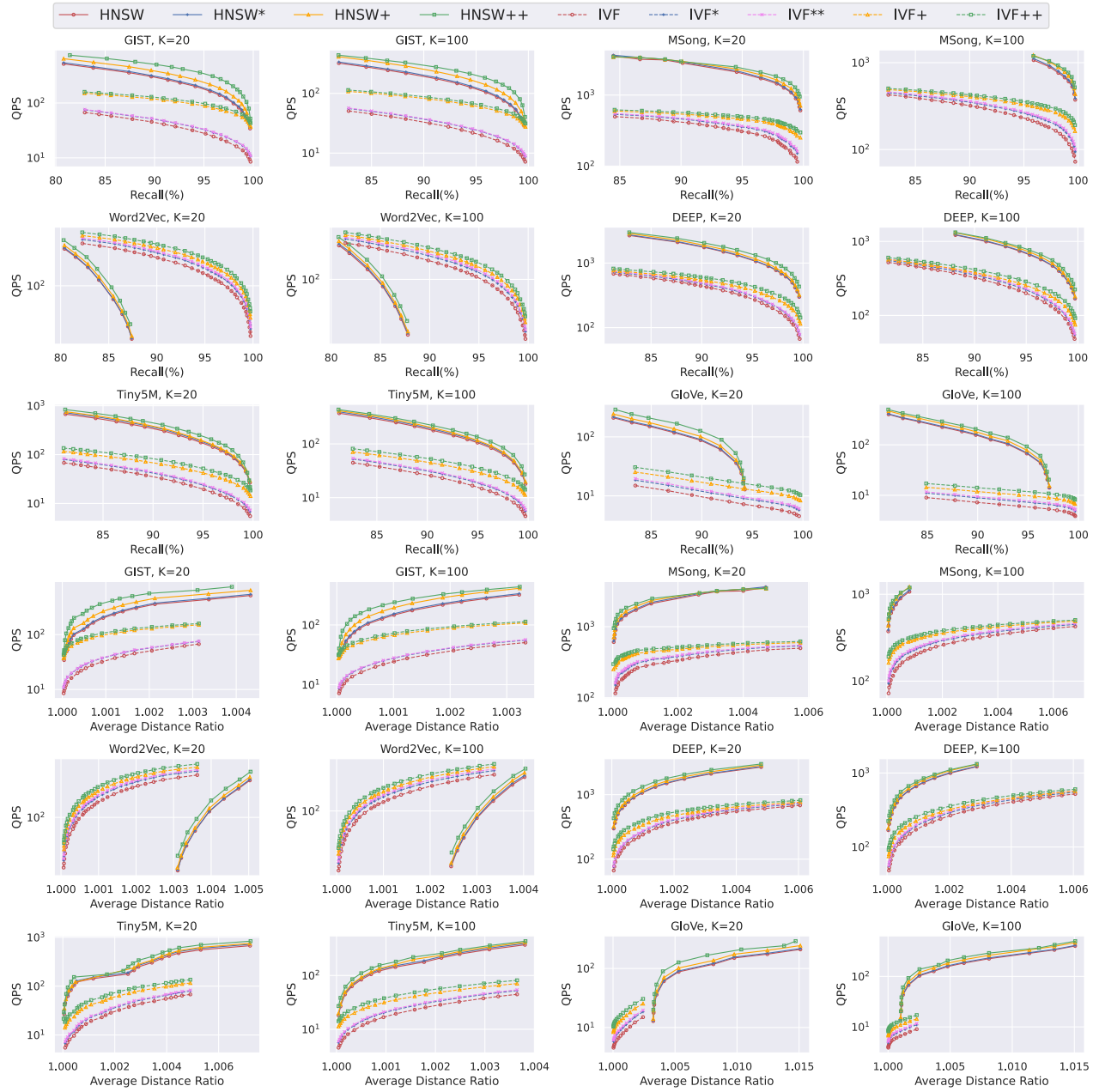


图5：时间-准确率权衡（HNSW 和 IVF）。

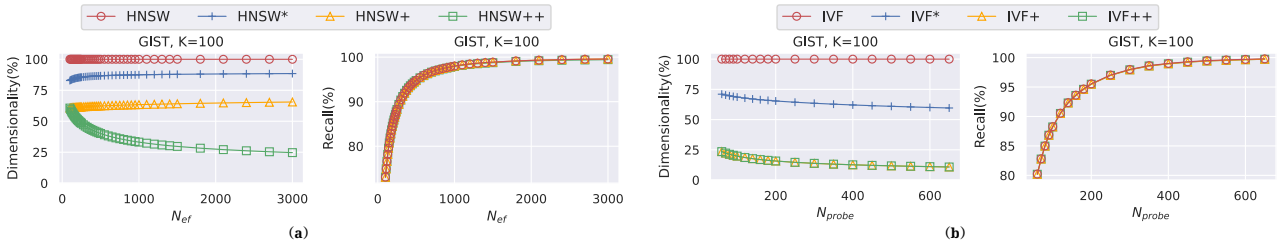


图6：评估维度和准确率。

6.2 实验结果

6.2.1 总体结果（时间-准确性权衡）。我们在图5中绘制了QPS-召回率曲线（上方各图，越靠右上越好）和 QPS-比率曲线（下方各图，越靠左上越好），分别调整 HNSW/HNSW*/HNSW+/HNSW++ 的 N_{ef} 以及 IVF/IVF*/IVF**/IVF+/IVF++ 的 N_{probe}。出于实际需求，我们只关注召回率至少为 80% 的区域。总体而言，从图5的结果中可以清楚地观察到：(1) AKNN+算法（以橙色曲线表示）优于基础 AKNN 算法（以红色曲线表示）；(2) AKNN++算法（以绿色曲线表示）进一步优于 AKNN+ 算法；(3) 基线方法 HNSW*（以蓝色曲线表示）在所有测试数据集上对 HNSW 的提升都很有限；(4) 基线方法 IVF*（以蓝色曲线表示）和 IVF**（以紫色曲线表示）始终明显逊于 IVF+ （而 IVF++ 的优势更为显著）。

此外，我们还有以下观察结果。(1) 与 HNSW 相比，我们的技术对 IVF 带来的提升更大（即使 IVF 的表现优于 HNSW，例如在 Word2Vec 上）。我们认为，这是因为 HNSW 中除 DCO 外的其他计算比 IVF 中的计算开销更大（如图2所示）。(2) 总体而言，我们的技术在高精度区域带来的提升大于在低精度区域带来的提升（例如，GIST 上 95% 与 85% 的比较）。这是因为 AKNN算法追求更高准确率时，不可避免地会生成更多距离差较大的低质量候选项 α，而对此类候选项，只需更少维度即可实现可靠的 DCO。(3) 数据布局优化对 IVF+ （即 IVF++ 与 IVF+）带来的提升大于对 IVF*（即 IVF** 与 IVF*）带来的提升。这是因为 ADSampling 具有对数复杂度，而基线方法 PDScanning 具有线性复杂度。具体而言，使用 ADSampling 时，前 Δ_d 个维度足以完成许多 DCO，因此可以避免访问图4c中的第二个数组 A_2。而使用 PDScanning 进行 DCO 时，仍会频繁访问第二个数组，因为它需要超过 Δ_d 个维度。

6.2.2 评估维度与召回率的结果。接下来，我们研究 GIST、K、HNSW 的评估维度数量和召回率
在相同的搜索参数设置下（HNSW 为 N_{ef}，IVF 为 N_{probe}），AKNN/AKNN+/AKNN++/AKNN*（AKNN** 的曲线与 AKNN* 完全相同，因此省略）的结果。为便于比较，评估维度数量以其相对于 AKNN 的比例（百分比）表示。结果如图6所示。

总体结果。从图6中可以清楚地看出，AKNN+ 和 AKNN++ 评估的维度远少于 AKNN，同时召回率几乎相同。具体而言，在 GIST 上，对于所有测试的N_{ef}取值，与 HNSW 相比，HNSW+ 和 HNSW++ 的准确率损失不超过 0.14%；与 IVF 相比，IVF+ 和 IVF++ 的准确率损失不超过 0.1%。与此同时，HNSW++ 可减少总维度的 39.4% 至 75.3%，HNSW+ 可减少 34.5% 至 39.4%，IVF+/IVF++可减少 76.5% 至 89.2%。基线方法 HNSW* 可减少 10.9% 至 15.7%，这解释了其对 HNSW 的改进幅度较小。IVF*/IVF** 可减少 28.9% 至 40.4%。

HNSW+ 与 HNSW++相比。我们进一步比较 HNSW+ 和 HNSW++。根据图6a，我们得到以下观察结果。(1) HNSW++评估的维度更少，这在很大程度上解释了

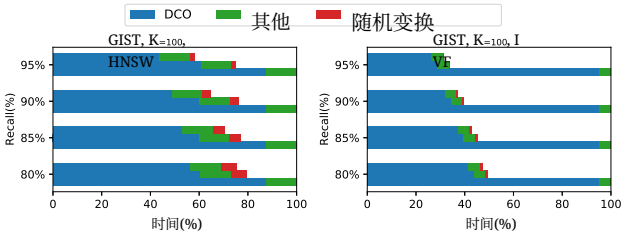


图7：时间开销分解（在特定召回率下，从上到下的三条横条分别表示 AKNN++、AKNN+ 和 AKNN 算法。时间开销以原始 AKNN 算法的开销为基准进行归一化。）

HNSW++ 的运行速度快于 HNSW+这一结果。(2) HNSW++ 达到几乎相同的召回率，与 HNSW+相当，这在经验上表明，使用近似距离进行图路由，与使用精确距离具有几乎相同的效果。(3) HNSW+ 评估的维度（其占 HNSW 评估维度的百分比）随 N_{ef}增加而增加，而 HNSW++ 评估的维度则随 N_{ef}增加而减少。这是因为 HNSW+以 N_{ef}^{th}近邻的距离为阈值执行 DCO；该距离随 N_{ef} 增加而增大，因此，较大的 N_{ef}会带来较小的距离差 α，这意味着要进行可靠的 DCO，就需要评估更多维度。对于 HNSW++，如 6.2.1 节所述，在追求高召回率时，AKNN算法不可避免地会生成许多 α较大的低质量候选项，因此 DCO 所需的维度更少。

IVF+ 与 IVF++。IVF+ 和 IVF++ 仅数据布局不同，因此它们的准确率和评估维度完全相同。

6.2.3 时间成本分解结果。请注意，应用 ADSampling 需要额外开销，用于对数据向量和查询向量进行随机变换。具体而言，变换数据向量的开销发生在索引阶段，可由同一数据库上的后续所有查询分摊。查询向量的变换则在查询阶段、收到查询时执行，其开销可由回答同一查询所涉及的所有 DCO 分摊。为简便起见，我们将此步骤（即 Johnson-Lindenstrauss变换 [19, 34]）实现为矩阵乘法操作，其耗时为 $O(D^2)$。请注意，使用先进算法可以更快地执行此步骤 [19],，例如，使用 Kac随机游走时耗时为 $O(D\log D)$。我们展示了数据集 GIST 上的时间成本分解结果。该数据集的维数最高，因此随机变换的开销也最大。我们在图7中分解了时间开销。请注意，对于 HNSW+/HNSW++，随机变换的开销最多占原始 HNSW 总开销的 6.18%。随着准确率提高，该比例会降低（例如，召回率为 95% 时，该比例降至 2.02%）。对于 IVF+，该比例不超过 1.11%。

6.2.4 评估距离近似技术对可靠 ***DCO*** 的可行性结果。接下来，我们研究两种距离近似方法对可靠 DCO 的可行性，包括随机投影和乘积量化 [32]（典型设置为每个分区包含256 个质心 [32, 59]）。我们将 ADSampling、PDScanning 和 FDScanning 的结果纳入比较。为了测试一种方法所能达到的最佳召回率，

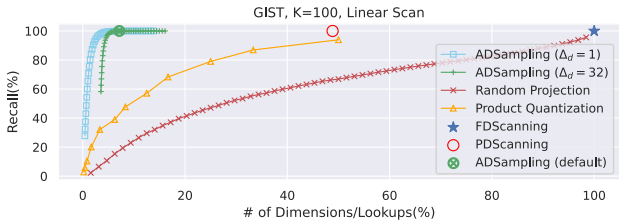


图 8：可靠 DCO 的可行性（召回率）。

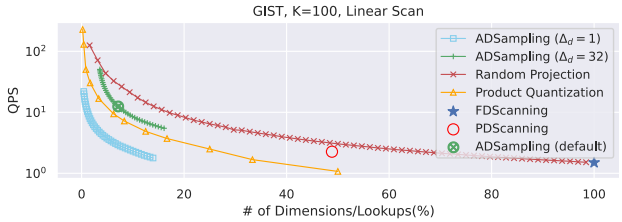


图 9：可靠 DCO 的可行性（每秒查询数）。

我们采用精确的 K 近邻算法（即线性扫描）进行这项实验。具体而言，对于随机投影和乘积量化，我们扫描所有数据对象，并返回近似距离最小的 K 个对象。对于 ADSampling 和 PDScanning，与 IVF 一样，我们维护一个 K 近邻集合，并依次对每个对象执行 DCO。我们在图 8 中绘制召回率与维度数/查找次数 [8] 的曲线。对于随机投影，我们改变投影向量的维数，并观察到：(1) 它仅减少 1.04% 的维度，却造成了 3.71% 的准确率损失；(2) 当维数减半时，其召回率不超过 70%。对于乘积量化，我们改变其量化码大小（即分区数），并观察到，即使在最佳情况下（即每两个维度用一个编码表示），它仍会造成 6.2% 的准确率损失。因此，乘积量化和随机投影都无法实现可靠的 DCO，且效率显著优于 FDScanning。

对于 ADSampling，我们测试了两种设置 $\Delta_d = 1$ 和 $\Delta_d = 32$。前者代表我们的方法所能达到的最佳召回率与维数权衡，后者则代表一种较实用的设置，其假设检验频率较低（即我们的默认设置）。我们通过将 ϵ_0 从 0.0 变动到 4.0 来绘制相应曲线。我们观察到，对于 $\Delta_d = 1$，它仅采样总维数的 6.61%，即可达到 >99.9% 的召回率；对于 $\Delta_d = 32$，它采样总维数的 7.11%，即可达到 > 99.9% 的召回率。因此，与 FDScanning 相比，ADSampling 实现了更优的召回率与维数权衡。

在图 9 中，我们绘制了每秒查询数与维度/查找次数的曲线。我们有以下观察结果。(1) ADSampling（采用默认设置），在图中标记为绿色圆圈内的绿色十字，其每秒查询数显著高于 FDScanning 和 PDScanning。这是因为它仅利用了总维度的 7.11%，同时召回率超过 99。9%。(2) 在维数相同时，随机投影的效率优于 ADSampling。这是

[8]对于乘积量化，它指的是量化码大小，评估每个编码时都需要查找内存中的表（即随机访问内存）。对于其他方法，它指的是维度数，评估每个维度时都需要进行一些算术计算。请注意，两者不能直接比较，因为在现代 CPU 上，前者的速度远慢于后者。

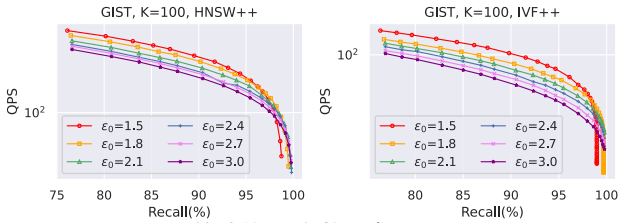


图10：AKNN++ 算法的 ϵ_0 参数研究。

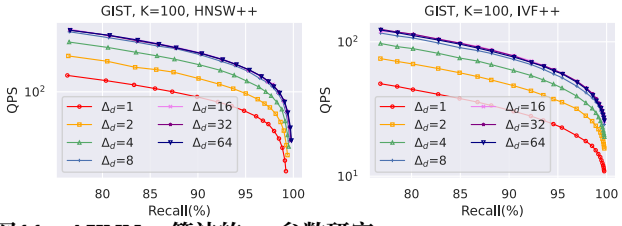


图11：AKNN++ 算法的 Δ_d 参数研究。

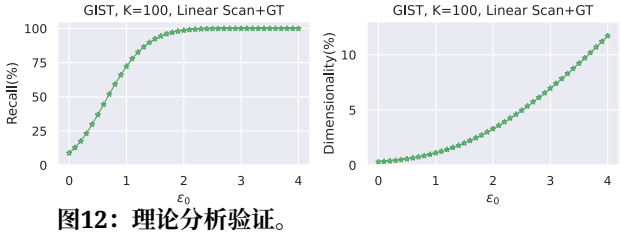


图12：理论分析验证。

因为随机投影的维数固定，并且可以将投影向量按顺序组织在数组中，从而实现更好的缓存友好性。不过，我们注意到，当随机投影的每秒查询数与 ADSampling（默认设置）相同时，其召回率不超过 40%。

6.2.5 参数研究结果。 参数 ϵ_0 是 ADSampling 算法的一个关键参数，因为它直接控制准确率与效率之间的权衡（请注意，较大的 ϵ_0 意味着假设检验的显著性水平较低，进而意味着假设检验结果更准确）。图10绘制了不同 ϵ_0下 HNSW++ （左图）和 IVF++ （右图）的每秒查询数-召回率曲线。总体而言，从图中可以看出，随着 ϵ_0增大，每秒查询数-召回率曲线会向右下方移动。这是因为较大的 ϵ_0 能提高准确率，但会降低效率。我们观察到，当 $\epsilon_0 = 2.1$时，准确率损失很小；继续增大 ϵ_0 则会降低效率。因此，为了在不损失太多准确率的情况下提高效率，我们建议将 ϵ_0 设为约 2。1。HNSW+ 和 IVF+ 的结果相似，因篇幅所限不再赘述。

图 11 绘制了采用不同 Δ_d 时 HNSW++ 和 IVF++ 的 QPS-召回率曲线。我们观察到，过于频繁地进行假设检验（例如，当 $\Delta_d = 1$ 时）会损害性能。值得注意的是，较小的 Δ_d 意味着在收集到足够信息后可以立即终止采样，但假设检验所需的算术运算会更多。我们的实证研究表明，当 $\Delta_d = 16$，32，64 时，可以实现最佳权衡。

6.2.6 理论结果验证实验。 我们进一步通过实验证明引理 3.2 和 3.3。作为一项验证研究，实验设置有所不同。为了消除 AKNN 算法造成的准确率损失，我们采用线性扫描开展验证研究

，因为线性扫描本身就是一种精确的 K 近邻算法。请注意，在 KNN查询处理中，前面的 DCO 的结果会影响后续 DCO 的距离阈值，这会给验证研究带来一定偏差。为消除这种偏差，我们提供一个固定的距离阈值（真实值中第K近邻的精确距离），使各个 DCO 彼此独立。为测试实际所需的维数，我们设置 $\Delta_d = 1$。在此设置下，召回率表示正对象中成功完成的 DCO 所占比例（请注意，负对象对应的 DCO 永远不会失败），因此可在实验中反映采用 ADSampling 的单个 DCO 的成功概率。图12左侧面板显示，失败概率确实按照二次指数趋势衰减，并在约 $\epsilon_0 = 2$ 时达到接近 100% 的准确率。右侧面板显示，评估维度的数量按照二次趋势增长；当 ϵ_0 较小时，增长较为缓慢（当 $\epsilon_0 = 2$ 时，评估维度总数约为普通FDScanning的 3%）。

7 相关工作

近似K近邻搜索。现有的 AKNN 算法可分为四类：（1）基于图的方法 [20,21, 29, 40, 45, 46]，（2）基于量化的方法 [4, 5, 23, 24, 26, 32]，（3）基于树的方法 [8, 12, 14, 47, 51]，（4）基于哈希的方法。具体而言，基于图的方法在内存中AKNN查询方面表现出色。内存受限时，基于量化的方法效果显著。基于哈希的方法则提供严格的理论保证。有关详情，请参阅近期的教程 [15, 22, 27, 28, 以及 39, 44, 54, 55, 62]（ ）[17, 50]，综述和基准测试 [3, 9, 40, 49, 60]。此外，还有大量研究采用机器学习（ML）来加速 AKNN [6, 16, 18, 38]。 [6, 18] 将强化学习应用于图路由，以取代贪心束搜索。 [38] 通过学习实现搜索的提前终止。 [16] 则利用机器学习构建索引结构。需要注意的是，上述方法均将机器学习用于候选生成。这些基于机器学习的方法与我们的技术相互独立，而我们的技术可以帮助它们在生成的候选项中找到 K 近邻。

AKNN 随机投影。 如前文所述并经验证，在对 K 近邻候选项进行重排序的阶段，随机投影几乎无法用于获得可靠的 DCO；但在生成 K 近邻候选项的阶段，随机投影已广泛应用于局部敏感哈希 [15, 22, 27, 28, 44, 54, 55]和随机划分/投影树 [14, 51]。我们的研究与这些研究的不同之处在于：(1) 我们可以灵活地将不同对象投影为维度各异的向量，而这些研究将所有对象投影为维度相同的向量；(2) 我们通过假设检验，根据对象的 DCO 自动设置其投影向量的维度数，而这些研究需要人工设置维度数；(3) 在从生成的候选项中找出近邻的阶段，我们使用投影向量（用于 DCO），而这些研究在生成候选项的阶段使用投影向量。因此，这些研究与我们的研究是正交的。

AKNN 的维度采样。 我们注意到，一种面向 MAB（多臂老虎机）的方法 [37] 也采用维度采样，并声称具有对数复杂度。我们的研究与[37] 在以下方面有所不同。在问题方面， [37] 针对 AKNN 问题本身，旨在找到一个超集，该超集包含 K近邻。将 [37]用于 DCO（本文的研究重点）并非易事。在理论方面， [37]的对数复杂度依赖于对数据的一些强假设（这些假设在实践中可能并不成立），而我们的方法不依赖任何假设。在技术方面，（1） [37] 直接对原始向量进行采样，而我们先对向量进行随机变换，再对变换后的向量进行采样。我们的方法具有这样的优势：近似距离的误差界基于随机投影的集中不等式，不像 [37] 那样依赖任何假设；（2） [37] 使用一些上下界来确定采样维度的数量，而我们采用序贯假设检验。我们的方法没有假阳性，而 [37] 既有假阳性，也有假阴性。总之， [37] 与我们的工作仅共享维度采样这一高层次思路，在问题、理论和技术等诸多方面均有不同。

8 结论与讨论

我们发现，距离比较操作是几乎所有 AKNN 算法的主要时间开销，并展示了提升其效率的机会。我们提出了一种用于 DCO 的新随机算法，在大多数情况下，该算法的运行时间相对于 D 呈对数级，并且以高概率成功。基于此，我们进一步为 AKNN 算法开发了一种通用技术和两种针对特定算法的技术。实验结果表明，增强后的 AKNN 算法始终优于原始算法。我们还为所有技术提供了严格的理论分析。

我们希望重点介绍这些技术的以下扩展和应用。(1) 通过简单变换，我们的技术可以轻松扩展到另外两种广泛采用的相似度度量，即余弦相似度和内积。具体而言，对给定数据和查询向量进行基于余弦相似度的相似性搜索，等价于对归一化后的数据和查询向量进行欧几里得最近邻搜索，因此可以应用 ADSampling。判断内积 $\mathbf{o}$,$\mathbf{q}$ 是否 $\geq r$，可以化简为判断 DCO： $\|\mathbf{o}/\|\mathbf{o}\| - \mathbf{q}/\|\mathbf{q}\|\| \leq \sqrt{2 - 2r/(\|\mathbf{o}\| \cdot \|\mathbf{q}\|)}$ ~~是否成立，其中距离阈值等于 $\sqrt{2 - 2r/(\|\mathbf{o}\| \cdot \|\mathbf{q}\|)}$ [9]~~。(2) ~~DCO 在聚类 [42] 和异常值检测~~ [7]等许多其他高维数据管理与分析任务中也十分常见。我们的技术有望在保持准确率的同时降低 DCO 的开销，从而加速这些任务的现有方法。

9 致谢

本研究由新加坡教育部通过其学术研究基金（第二级资助项目 MOE-T2EP20220- 0011 和 MOE-T2EP20221-0013 ）资助。本文所表达的任何观点、研究结果、结论或建议均属于作者，不代表新加坡教育部的立场。我们感谢 Yi Li（SPMS，南洋理工大学）解答了许多有关高维概率的问题，也感谢匿名审稿人提供建设性的反馈和宝贵建议。

[9]可验证如下：$\mathbf{o}, \mathbf{q} \geq r \Longleftrightarrow \mathbf{o}/\|\mathbf{o}\|, \mathbf{q}/\|\mathbf{q}\| \geq r/(\|\mathbf{o}\| \cdot \|\mathbf{q}\|) \Longleftrightarrow \|\mathbf{o}/\|\mathbf{o}\|\|^2 - 2\,\mathbf{o}/\|\mathbf{o}\|, \mathbf{q}/\|\mathbf{q}\| + \|\mathbf{q}/\|\mathbf{q}\|\|^2 \leq 2 - 2r/(\|\mathbf{o}\| \cdot \|\mathbf{q}\|) \Longleftrightarrow \|\mathbf{o}/\|\mathbf{o}\| - \mathbf{q}/\|\mathbf{q}\|\|^2 \leq 2 - 2r/(\|\mathbf{o}\| \cdot \|\mathbf{q}\|)$.

参考文献

[1] Nir Ailon 和 Bernard Chazelle。2009。《快速 Johnson–Lindenstrauss 变换与近似最近邻》。SIAM J. Comput. 39, 1（2009），302–322。https://doi.org/10 1137 060673096 10 1137 060673096 。/ arXiv:

[2] https://doi.org/ 。/ Annoy。 。Annoy。https://github.com/spotify/annoy。[3] Martin Aumüller、Erik Bernhardsson 和 Alexander Faithfull。2020。《ANN-Benchmarks：近似最近邻算法的基准测试工具》。Inf. Syst. 87, 2020 13 10 1016 2019 02 006 C（1月），

[4] 页码。https://doi.org/ 。/j.is. 。 。Artem Babenko 和 Victor Lempitsky。 。《用于极限向量压缩的加性量化》。载于《2014 IEEE 计算机视觉与模式识别会议》。931 938 10 1109 2014 124 – 。https://doi.org/ 。/

[5] CVPR. 。Artem Babenko 和 Victor Lempitsky。 。《倒排多索引》。IEEE模式分析与机器智能汇刊 37, 6（2015），1247–1260。https://doi.org/10 1109 2014 2361319 。/TPAMI. 。Dmitry Baranchuk、

[6] Dmitry Persiyanov、Anton Sinitsin 和 Artem Babenko。2019。《学习在相似性图中进行路由》。载于《第36届国际机器学习会议论文集》（《机器学习研究论文集》，第97卷），Kamalika Chaudhuri 和 Ruslan Salakhutdinov（编）。PMLR，475–484。https://proceedings.mlr.press/v97/baranchuk19a.html [7] Stephen D. Bay 和 Mark Schwabacher。2003。《利用随机化和简单的剪枝规则，以近线性时间挖掘基于距离的离群点》。载于《第九届 ACM SIGKDD 知识发现与数据挖掘国际会议论文集》（华盛顿特区）（KDD '03）。计算机协会，纽约，美国，29 38 10 1145 956750 956758 – 。

[8] https://doi.org/ 。/ 。Alina Beygelzimer、Sham Kakade 和 John Langford。 。《用于最近邻搜索的覆盖树》。载于《第23届国际机器学习会议论文集》。97–104。[9] Leonid Boytsov 和 Bilegsaikhan Naidan。2013。《构建高效实用的非度量空间库》。载于《相似性搜索与应用》，Nieves Brisaboa、Oscar Pedreira 和 Pavel Zezula（编）。Springer Berlin Heidelberg，柏林，海德堡，280–293。[10] Aniket Chakrabarti 和 Srinivasan Parthasarathy。2015。《用于自适应局部敏感哈希的序贯假设检验》。载于《第24届国际万维网会议论文集》（意大利佛罗伦萨）（WWW '15）。国际万维网会议指导委员会，日内瓦共和国和州，CHE，162 172 10 1145 2736277 2741665 – 。https://doi.org/ 。/

[11] 。Krzysztof M Choromanski 、 Mark Rowland 和 Adrian Weller。2017。《结构化随机正交嵌入的惊人效力》。神经信息处理系统进展 30（2017）。[12] Paolo Ciaccia、Marco Patella 和 Pavel Zezula。1997。《M-树：度量空间相似性搜索的高效访问方法》。载于《第23届超大型数据库国际会议论文集》（VLDB '97）。Morgan Kaufmann 出版社，旧金山，美国，426–435。[13] T. Cover 和 P. Hart。1967。《最近邻模式分类》。IEEE Transactions on Information Theory 13, 1（1967），21–27。https://doi.org/10.1109/TIT.1967.1053964[14] Sanjoy Dasgupta 和 Yoav Freund。2008。《随机投影树与低维流形》。载于《第四十届 ACM 计算理论研讨会论文集》。537–546。[15] Mayur Datar、Nicole Immorlica、Piotr Indyk 和 Vahab S Mirrokni。2004。《基于 p-稳定分布的局部敏感哈希方案》。载于《第二十届计算几何年度研讨会论文集》。253–262。[16] Yihe Dong、Piotr Indyk、Ilya Razenshteyn 和 Tal Wagner。2020。《学习用于最近邻搜索的空间划分》。载于《国际学习表征会议》。https://openreview.net/forum?id=rkenmREFDr [17] Karima Echihabi、Kostas Zoumpatianos 和 Themis Palpanas。2021。《高维向量相似性搜索的新趋势：人工智能驱动、渐进式和分布式》。VLDB基金会会议录 14, 12（2021年7月），3198–3201。https://doi.org/10.14778/3476311. 3476407[18] Chao Feng、Defu Lian、Xiting Wang、Zheng Liu、Xing Xie 和 Enhong Chen。2022。《在邻近图上进行强化路由以实现高效推荐》。ACM Trans. Inf. Syst.（2022年1月）。https://doi.org/10.1145/3512767 已接收。[19] Casper Benjamin Freksen。2021。《Johnson-Lindenstrauss 变换导论》。CoRR abs/2103.00564（2021）。arXiv:2103.00564 https://arxiv.org/abs/ 2103.00564[20] Cong Fu、Changxu Wang 和 Deng Cai。2021。《利用卫星系统图进行高维相似性搜索：效率、可扩展性与未索引查询兼容性》。IEEE模式分析与机器智能汇刊（2021）。[21] Cong Fu、Chao Xiang、Changxu Wang 和 Deng Cai。2019。《利用导航式扩展图进行快速近似最近邻搜索》。VLDB基金会会议录。12, 5 2019 461 474 10 14778 3303753 3303754 （1月），–。

[22] https://doi.org/ 。/ 。Junhao Gan、Jianlin Feng、Qiong Fang 和 Wilfred Ng。 。《基于动态碰撞计数的局部敏感哈希方案》。载于《2012 ACM SIGMOD数据管理国际会议论文集》（美国亚利桑那州斯科茨代尔）（SIGMOD '12）。计算机协会，纽约，美国，541–552。https://doi.org/10.1145/2213836.2213898

[23] Tiezheng Ge、何恺明、柯其发和孙剑。2013。用于近似最近邻搜索的优化乘积量化。载于 IEEE 计算机视觉与模式识别会议论文集。2946–2953。[24] 龚云超、Svetlana Lazebnik、Albert Gordo 和 Florent Perronnin。2013。迭代量化：一种用于大规模图像检索的二进制编码学习的普罗克鲁斯特方法。IEEE模式分析与机器智能汇刊 35, 12 2013 2916 2929 10 1109 2012 193 ( )，–。

[25] https://doi.org/ 。/TPAMI. 。Gaël Guennebaud、Benoît Jacob 等。 。Eigen v 。http://eigen.tuxfamily.org。[26] 郭睿琦、Philip Sun、Erik Lindgren、Quan Geng、David Simcha、Felix Chern 和 Sanjiv Kumar。2020。利用各向异性向量量化加速大规模推理。载于国际机器学习会议。PMLR，3887–3896。[27] 黄强、Jianlin Feng、张一凯、Qiong Fang 和 Wilfred Ng。2015。用于近似最近邻搜索的查询感知局部敏感哈希。VLDB基金会论文集 9，1（2015），1–12。[28] Piotr Indyk 和 Rajeev Motwani。1998。近似最近邻：迈向消除维数灾难。载于第三十届 ACM 计算理论年度研讨会论文集。604–613。[29] Masajiro Iwasaki。2016。用于邻近搜索的剪枝双向K近邻图。载于《相似性搜索与应用》，Laurent Amsaleg、Michael E. Houle 和 Erich Schubert（编）。施普林格国际出版公司，查姆，20–33。[30] Vishesh Jain、Natesh S. Pillai、Ashwin Sah、Mehtaab Sawhney 和 Aaron Smith。2022。利用 Kac随机游走实现快速且内存最优的降维。应用概率年刊 32，5（2022），4038–4064。https://doi.org/10.1214/22-AAP1784[31] Suhas Jayaram Subramanya、Fnu Devvrit、Harsha Vardhan Simhadri、Ravishankar Krishnawamy 和 Rohan Kadekodi。2019。DiskANN：在单节点上实现快速、精确的十亿点最近邻搜索。载于《神经信息处理系统进展》，H. Wallach、H. Larochelle、A. Beygelzimer、F. d'Alché-Buc、E. Fox 和 R. Garnett（编），第32卷。Curran Associates, Inc.

h

t

t

p

s://

p

roceedings.neurips.cc/paper/2019/file/09853c7fb1d3f8ee67a61b6bf4a7f8e6-Paper.pdf

[32] Hervé Jégou、Matthijs Douze 和 Cordelia Schmid。2010。用于最近邻搜索的乘积量化。IEEE模式分析与机器智能汇刊 33，1（2010），117–128。[33] Jeff Johnson、Matthijs Douze 和 Hervé Jégou。2019。利用图形处理器进行十亿级相似性搜索。IEEE大数据汇刊 7，3（2019），535–547。[34] William B Johnson 和 Joram Lindenstrauss。1984。将利普希茨映射扩展到希尔伯特空间 26。当代数学 26（1984），28。[35] Hervé Jégou、Matthijs Douze、Cordelia Schmid 和 Patrick Pérez。2010。将局部描述符聚合为紧凑图像表示。载于2010年IEEE计算机学会计算机视觉与模式识别会议。3304–3311 10 1109 2010 5540039 。https://doi.org/ 。/CVPR. 。Daniel

[36] M. Kane 和 Jelani Nelson。 。更稀疏的 Johnson-Lindenstrauss 变换。ACM期刊 61, 1, 4 2014 23 10 1145 2559902 文章（1月），

[37] 页。https://doi.org/ 。/ Daniel LeJeune、Reinhard Heckel 和 Richard Baraniuk。 。近似 *k*近邻计算的自适应估计。载于第二十二届国际人工智能与统计会议论文集（机器学习研究论文集，第89卷），Kamalika Chaudhuri 和杉山将（编）。PMLR，3099–3107。https://proceedings.mlr.press/v89/lejeune19a.html [38] Conglong Li、Minjia Zhang、David G. Andersen 和 Yuxiong He。2020。通过学习式自适应提前终止改进近似最近邻搜索。载于2020年ACM SIGMOD数据管理国际会议论文集（美国俄勒冈州波特兰）（SIGMOD '20）。计算机协会，纽约，美国，2539–2554。https://doi.org/10.1145/3318464.3380600[39] 李金峰、严晓、张健、徐安、James Cheng、刘杰、Kelvin K. W. Ng 和 Ti-chung Cheng。2018。用于学习哈希的通用高效查询方法。载于2018年国际数据管理会议论文集（美国德克萨斯州休斯敦）（SIGMOD '18）。计算机协会，纽约，美国，1333–1347。https://doi.org/10.1145/3183713.3183750[40] 李文、Ying Zhang、Yifang Sun、Wei Wang、李明杰、张文杰和林学民。2019。高维数据上的近似最近邻搜索——实验、分析与改进。IEEE知识与数据工程汇刊 32，8（2019），1475–1488。[41] 刘颖、张登胜、陆国军和马维英。2007。具有高层语义的基于内容的图像检索综述。模式识别 40，1（2007）262 282 10 1016 2006 04（），–。https://doi.org/ 。/j.patcog. 。

[42] 1982 S. Lloyd。 。PCM中的最小二乘量化。IEEE信息论汇刊 28, 2 1982 129 137 10 1109 1982 1056489 ( )，–。

[43] https://doi.org/ 。/TIT. 。陆可敬、工藤峰一、Chuan Xiao 和石川义治。 。HVS：基于沃罗诺伊图、用于解决近似最近邻搜索问题的分层图结构。VLDB基金会会议录 15，2（2021年10月），246–258。https://doi.org/10 14778 3489496 3489506 . / 。陆可敬、

[44] 王宏亚、Wei Wang 和工藤峰一。2020。VHP：通过虚拟超球面划分实现近似最近邻搜索。VLDB基金会论文集 13，9（2020），1443–1455。

[45] Yury Malkov、Alexander Ponomarenko、Andrey Logvinov 和 Vladimir Krylov。2014。基于可导航小世界图的近似最近邻算法。*Information Systems* 45（2014），61–68。https://doi.org/10.1016/j.is.2013. 10.006[46] Yu A. Malkov 和 D. A. Yashunin。2020。利用分层可导航小世界图实现高效且稳健的近似最近邻搜索。*IEEE*模式分析与机器智能汇刊 42，4（2020），824–836 10 1109 2018 2889473 。https://doi.org/ 。/TPAMI. 。[47] Marius Muja 和 David G Lowe。 。2014 。面向高维数据的可扩展最近邻算法。*IEEE*模式分析与机器智能汇刊 36，11（2014），2227–2240。[48] Marco Patella 和 Paolo Ciaccia。2008。近似相似性搜索的多重面向。载于第一届相似性搜索与应用国际研讨会（*sisap 2008* 10 21 10 1109 2008 18 ）。–。[49] https://doi.org/ 。/SISAP. 。Marco Patella 和 Paolo Ciaccia。 。2009 。近似相似性搜索：一个多面向问题。离散算法杂志 7，1（2009年3月），36–48。https://doi.org/10 1016 2008 09 014 . /j.jda. 。Jianbin Qin、Wei Wang、Chuan Xiao、[50] Ying Zhang 和 Yaoshu Wang。2021。面向数据科学的高维相似性查询处理。载于第*27*届*ACM SIGKDD*知识发现与数据挖掘会议论文集 （线上活动，新加坡）*(KDD '21)*。计算机协会，纽约，美国，4062 4063 10 1145 3447548 3470811 –。https://doi.org/ 。/ 。[51] Parikshit Ram 和 Kaushik Sinha。 。2019 。重新审视用于最近邻搜索的kd树。载于第*25*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集。1378–1388。[52] Venu Satuluri 和 Srinivasan Parthasarathy。2011。用于快速相似性搜索的贝叶斯局部敏感哈希。*arXiv*预印本 *arXiv:1110.1328*（2011）。[53] J. Ben Schafer、Dan Frankowski、Jon Herlocker 和 Shilad Sen。2007。协同过滤推荐系统。Springer Berlin Heidelberg，柏林，海德堡，291 324 10 1007 978 3 540 72079 9_9 –。https://doi.org/ 。/- - - - Yifang Sun、Wei Wang、Jianbin Qin、Ying Zhang 和林学民。2014。SRS：使用微型索引在高维欧几里得空间中解决c-近似最近邻查询。*VLDB*基金会论文集 （2014）。[55] Yufei Tao、Ke Yi、Cheng Sheng 和 Panos Kalnis。2010。在高维空间中高效、准确地搜索最近邻和最近点对。*ACM*数据库系统汇刊（*TODS*）35，3（2010），1–46。[56] Roman Vershynin。2018。高维概率：数据科学中的导论与应用。剑桥大学出版社。https://doi.org/10. 1017/9781108231596[57] Jun Wang、Wei Liu、Sanjiv Kumar 和 Shih-Fu Chang。2016。大数据索引的学习哈希——综述。*IEEE*会议论文集 104，1（2016），34–57。https://doi.org/10 1109 2015 2487976 . /JPROC. 。Jianguo Wang、Xiaomeng Yi、[58] Rentong Guo、Hai Jin、Peng Xu、Shengjun Li、Xi-angyu Wang、Xiangzhou Guo、Chengming Li、Xiaohai Xu、Kun Yu、Yuxing Yuan、Yinghao Zou、Jiquan Long、Yudong Cai、Zhenxiang Li、Zhifeng Zhang、Yihua Mo、Jun Gu、Ruiyi Jiang、Yi Wei 和 Charles Xie。2021。Milvus：专用向量数据管理系统。载于*2021*年国际数据管理会议论文集 （线上活动，中国）*(SIGMOD '21)*。计算机协会，纽约，美国，2614–2627。https://doi.org/10 1145 3448016 3457550 . / 。[59] Jingdong Wang、Ting Zhang、jingkuan song、Nicu Sebe 和 Heng Tao Shen。2018。学习哈希综述。*IEEE*模式分析与机器智能汇刊 40, 4 2018 769 790 10 1109 2017 2699960 ( ), –。https://doi.org/ 。/TPAMI. 。[60] Mengzhao Wang、Xiaoliang Xu、Qiang Yue 和 Yuxiang Wang。 。2021 。基于图的近似最近邻搜索：全面综述与实验比较。*VLDB*基金会会议录 14，11（2021年7月），1964–1978。https://doi.org/10 14778 3476249 3476255 . / 。Wen Yang、Tao Li、Gai Fang 和 [61] Hong Wei。2020。PASE：PostgreSQL超高维近似最近邻搜索扩展。载于*2020*年*ACM SIGMOD*数据管理国际会议论文集 （美国俄勒冈州波特兰）*(SIGMOD '20)*。计算机协会，纽约，美国，2241–2253。https://doi.org/10.1145/3318464.3386131[62] Bolong Zheng、Zhao Xi、Lianggui Weng、Nguyen Quoc Viet Hung、Hang Liu 和 Christian S Jensen。2020。PM-LSH：用于高维近似最近邻搜索的快速、精确 LSH 框架。*VLDB*基金会论文集 13，5（2020），643–655。

附录
理论分析

在本节中，我们将详细证明引理 3.3。我们假设每次额外采样一个维度的y 。设 $\gamma(d) = (1+\epsilon_0/\sqrt{d})$。我们有

$$\mathbb{E}\left[\hat{D}\right] = \sum_{d=1}^{D} d \cdot \mathbb{P}\left\{\hat{D}=d\right\} = \sum_{d=1}^{D} \mathbb{P}\left\{\hat{D}\geq d\right\} \tag{13}$$

$$= \sum_{d=1}^{D} \mathbb{P}\left\{\forall p < d, \sqrt{\frac{D}{p}}\|\mathbf{y}|_{1,2,\ldots,p}\| \leq \gamma(p)\cdot r\right\} \tag{14}$$

$$= \sum_{d=1}^{D} \mathbb{P}\left\{\forall p < d, \sqrt{\frac{D}{p}}\|\mathbf{y}|_{1,2,\ldots,p}\| \leq \gamma(p)\cdot \frac{\|\mathbf{y}\|}{1+\alpha}\right\} \tag{15}$$

$$\leq 1 + \sum_{d=1}^{D-1} \mathbb{P}\left\{\sqrt{\frac{D}{d}}\|\mathbf{y}|_{1,2,\ldots,d}\| \leq \gamma(d)\cdot \frac{\|\mathbf{y}\|}{1+\alpha}\right\} \tag{16}$$

ˆ 其中，(14) 是因为 $D \geq d$ 表示所有先前的假设检验均无法拒绝该假设这一事件，而 (16) 将与所有检验（即 $\forall p < d$）对应的事件放宽为与最后一次检验（即 $p = d-1$）对应的事件。

我们记 $\tilde{d} := \epsilon_0^2/\alpha^2$， $d_0 := \text{ceil}(\tilde{d})$，并将$d \leq d_0$ 的概率放宽为 1：

$$\mathbb{E}\left[\hat{D}\right] \leq 1 + d_0 + \sum_{d=d_0+1}^{D} \mathbb{P}\left\{\sqrt{\frac{D}{d}}\|\mathbf{y}|_{1,2,\ldots,d}\| \leq \frac{\gamma(d)}{1+\alpha}\|\mathbf{y}\|\right\} \tag{17}$$

让我们聚焦于（17）的最后一项，并据此推导如下：

$$\sum_{d=d_0+1}^{D} \mathbb{P}\left\{\sqrt{\frac{D}{d}}\|\mathbf{y}|_{1,2,\ldots,d}\| \leq \frac{\gamma(d)}{1+\alpha}\|\mathbf{y}\|\right\} \tag{18}$$

$$= \sum_{d=d_0+1}^{D} \mathbb{P}\left\{\sqrt{\frac{D}{d}}\|\mathbf{y}|_{1,2,\ldots,d}\| \leq \left[1-\left(1-\frac{\gamma(d)}{1+\alpha}\right)\right]\|\mathbf{y}\|\right\} \tag{19}$$

$$\leq \sum_{d=d_0+1}^{D} \exp\left[-c_0 d\left(1-\frac{\gamma(d)}{1+\alpha}\right)^2\right] \tag{20}$$

$$= \sum_{d=d_0+1}^{D} \exp\left[-\frac{c_0\alpha^2}{(1+\alpha)^2}\left(\sqrt{d}-\sqrt{\tilde{d}}\right)^2\right] \tag{21}$$

$$\leq \int_{d_0}^{D} \exp\left[-\frac{c_0\alpha^2}{(1+\alpha)^2}\left(\sqrt{x}-\sqrt{\tilde{d}}\right)^2\right] \mathrm{d}x \tag{22}$$

$$= \int_{d_0}^{D} \exp\left[-\frac{c_0\epsilon_0^2}{(1+\alpha)^2}\left(\sqrt{\frac{x}{\tilde{d}}}-1\right)^2\right] \mathrm{d}x \tag{23}$$

$$= \tilde{d}\int_{d_0/\tilde{d}}^{D/\tilde{d}} \exp\left[-\frac{c_0\epsilon_0^2}{(1+\alpha)^2}\left(\sqrt{u}-1\right)^2\right] \mathrm{d}u \tag{24}$$

$$\leq \tilde{d}\int_{1}^{+\infty} \exp\left[-\frac{c_0\epsilon_0^2}{(1+\alpha)^2}\left(\sqrt{u}-1\right)^2\right] \mathrm{d}u \tag{25}$$

其中，(19) 将其改写为符合引理 3.1 格式的形式，(20) 应用引理 3.1，(21) 代入 $\gamma(d)$，(22) 将 (21) 放宽为积分，(24) 代入 $u = x/\tilde{d}$，(25) 则将积分放宽为 $[1,+\infty)$。

接下来，我们首先分析 $\alpha \leq \epsilon_0$ 的情况，如下所示。

$$(25) \leq \tilde{d}\int_{1}^{+\infty} \exp\left[-\frac{c_0\epsilon_0^2}{(1+\epsilon_0)^2}\left(\sqrt{u}-1\right)^2\right] \mathrm{d}u \tag{26}$$

$$\leq \tilde{d}\int_{1}^{+\infty} \exp\left[-\frac{c_0}{4}\left(\sqrt{u}-1\right)^2\right] \mathrm{d}u \tag{27}$$

其中，（26）是因为 $\alpha \leq \epsilon_0$；当为获得合理的准确率而设置 $\epsilon_0 \geq 1$ 时，可得（27）。注意，积分收敛，因此有一个常数上界。对于 $\alpha \leq \epsilon_0$的情况，我们有

$$\mathbb{E}\left[\hat{D}\right] = 1 + d_0 + O(\tilde{d}) = O(\tilde{d}) = O\left(\alpha^{-2}\cdot\epsilon_0^2\right) \tag{28}$$

对于 $\alpha > \epsilon_0$的情况，其期望维数必定不大于 $\alpha = \epsilon_0$ 的情况，因为它的距离差 α 更大。因此，其期望维数的上界为 $O(1)$。

B 基于树和基于哈希的方法的结果

对于 Annoy，按照 [40], ，我们将树的数量设为 $N_{tree} = 50$。在索引阶段，我们将原始数据向量输入 Annoy 的索引算法（注意，Annoy、Annoy+ 和 Annoy* 具有相同的索引结构）。随后在查询阶段，对于 Annoy/Annoy*，我们将索引和原始数据向量加载到主存中，将原始查询向量输入 Annoy 的查询算法以生成候选项，并使用 FDScanning/PDScanning 对候选项重新排序。对于 Annoy+，我们将索引和变换后的数据向量加载到主存中，将原始查询向量输入 Annoy 的查询算法以生成候选项，并使用 ADSampling 对候选项重新排序。对于 PMLSH，按照[62], ，我们将随机投影的维数设为 15，并将 PM-树内部节点和叶节点的大小设为 16。与 Annoy 类似，在索引阶段，我们基于原始数据向量构建索引。在查询阶段，我们将原始查询向量输入 PMLSH 的搜索算法以生成候选项，并分别基于原始向量、原始向量和变换后的向量，使用 FDScanning、PDScanning 和 ADSampling 对候选项重新排序。对于这两种方法，我们通过改变访问的候选项数量来控制时间-准确率权衡。本实验不采用数据布局优化，因为该优化不适用于索引集成（例如 Annoy 的树集成）。我们绘制了所比较算法的 QPS-召回率曲线和 QPS-平均距离比曲线，如图 13所示。结果表明，AKNN+ 始终显著优于 AKNN* 和 AKNN算法。

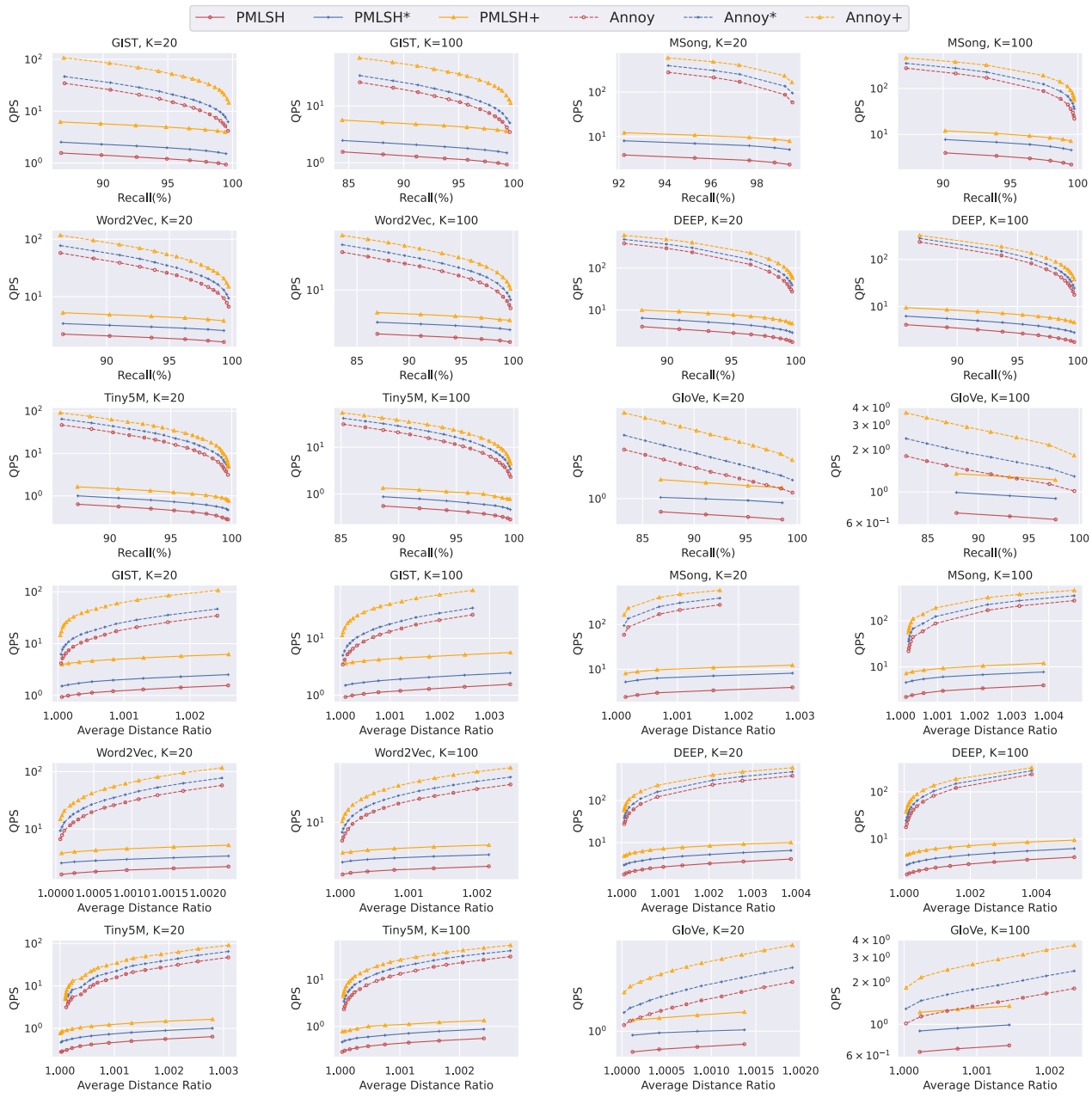


图13：时间-准确率权衡（PMLSH 和 Annoy）。