基于角距离的高维 DBSCAN 学习加速框架

Yifan Wang 佛罗里达大学 美国佛罗里达州盖恩斯维尔 wangyifan@ufl.edu

Daisy Zhe Wang 佛罗里达大学 美国佛罗里达州盖恩斯维尔 daisyw@ufl.edu

arXiv:2302.03136v1 [cs.IR] 6 Feb 2023

摘要

基于密度的聚类是数据科学中常用的工具。如今，许多数据科学研究都在使用高维神经嵌入。然而，传统的基于密度的聚类技术（如 DBSCAN）在高维数据上的性能会下降。本文提出 LAF，这是一种通用的学习型加速框架，可加速采用角距离度量的高维数据上的原始 DBSCAN 及其基于采样的变体。该框架由学习型基数估计器和后处理模块组成。基数估计器能够快速预测数据点是否为核心点，从而跳过不必要的范围查询；后处理模块则检测假阴性预测，并合并被错误分离的簇。评估结果表明，我们的 LAF 增强型 DBSCAN 方法在效率和质量方面均优于当前最先进的高效 DBSCAN 变体。

1 引言

如今，数据科学研究从深度神经模型生成的高维向量——神经嵌入——中获益良多。作为数据科学中广泛应用的技术，聚类已与嵌入相结合，例如， [22, 28] 利用聚类学习有效的段落嵌入， [19, 23] 利用聚类加速嵌入上的相似性搜索等。

作为一种具有代表性的聚类算法，具有噪声的基于密度的空间聚类（DBSCAN） [7]长期以来一直应用于低维空间数据（2D 或 3D）。然而，DBSCAN 通常效率较低，这是由计算密集型的范围查询造成的；而维数灾难使这一问题在高维数据中更加显著。具体而言，DBSCAN 将簇视为由低密度区域分隔的高密度区域。基于这一点，该算法反复将每个簇扩展至其相邻的高密度区域（其中的点称为核心点），直到簇完全被低密度区域包围（其中每个点要么是非核心点，要么是噪声）。对于每个点，DBSCAN 都必须执行大量范围搜索，以确定该点是否为核心点；这需要密集计算，限制了其在大规模高维数据分析中的应用。为提高 DBSCAN 的效率，已有研究提出了许多变体。例如，基于采样的 DBSCAN 变体 [5, 11, 12, 14, 21] 通过只在较小子集而非整个数据集上执行最繁重的计算来提高效率；另有一些研究通过加速 DBSCAN 中的范围查询 [15, 26], [1–4, 8, 9]来降低延迟，或在聚类过程中剪枝不必要的距离计算等。但其中许多方法是为低维到中维数据（大多低于 100 维）设计的。因此，它们仍不适用于具有数百至数千维的高维神经嵌入，例如 768 维的 BERT [6] 嵌入。

本文通过跳过对非核心点和噪声点的范围查询来解决这一问题，因为要确认一个点是否为非核心点或噪声点，只需知道其邻居数量。这可以通过基数估计来实现，即在执行查询之前估计查询结果数量的技术。在多维数据管理中，基数估计通常用于预测基于距离的范围查询所能找到的邻居数量，而无需实际执行查询；估计结果有助于优化范围搜索和相似性连接等关键操作。传统的范围查询基数估计依赖采样或核密度估计 [17]。近期研究将先进的机器学习应用于该问题，并提出了学习型基数估计 技术。这些技术通常基于回归模型，涵盖从非深度回归器（如 XGBoost）到深度回归器（如深度神经网络），其输入为查询点和距离阈值（即范围），输出为该范围内邻居数量的估计值。通过学习数据分布，学习型基数估计比传统方法能够做出更准确的预测。当前最先进的学习型基数估计方法采用了多种深度回归模型，例如卷积神经网络 [18],递归模型索引 [13, 24],深度格网络 [27], CardNet [24], SelNet [25], 等，这些模型在高维数据上表现出色。理论上和实践中，它们都能实现高效的预测。理论上，当模型结构固定时，其预测时间复杂度不随数据规模变化；实践中，图形处理器可以显著加速这些模型。由于神经模型具有泛化能力，训练时间并非问题：训练好的估计器可以用于任何分布相似的数据集。借助学习型基数估计器，可以在执行范围查询之前预测一个点是否为核心点，从而有效减少对非核心点和噪声点进行的不必要计算。

具体而言，我们的方法旨在用于基于角距离的聚类，例如余弦距离。本文不会探讨欧几里得距离等其他距离度量，原因有二：（1）角距离值得专门研究。在基于神经嵌入的应用中，余弦距离和欧几里得距离是衡量嵌入相似度的主要距离度量。因此，专注于余弦距离就足以惠及广泛的应用。（2）从理论上讲，我们的思路更适用于这种度量。角距离通常有界，例如，余弦距离的范围是 $0 \sim 2$，这使得基数估计器的训练比使用欧几里得距离更有效。具体而言，当训练集涵盖更多可能的输入值时，回归器通常能做出更好的预测

而对于取值范围无限的欧几里得距离（即 0 ~ +∞），训练集很难涵盖更多可能的输入值；对于有界的余弦距离，则更容易做到。例如，在评估中，我们使用 0.1 到 0.9 之间的余弦距离阈值构建训练集，这足以覆盖大多数情况。因此，其他距离不在本文讨论范围内。不过，我们的方法并未对距离度量施加严格限制，因此未来可以探索欧几里得距离。

我们提出**LAF**，一种通用的**学习加速**框架，基于角距离加速 DBSCAN 及其基于采样的变体。LAF 在每次范围查询前增加一个基数估计步骤，从而提高算法效率。如果预测某个点不是核心点或噪声点，就跳过对该点的范围查询，以减少计算。这种方法不仅适用于 DBSCAN，也适用于其基于采样的变体，因为处理采样子集时同样存在这类计算浪费。LAF 还包含一个后处理模块，通过检测并合并因漏检预测（即将核心点预测为非核心点或噪声点）而被错误拆分的簇，弥补聚类质量损失。据我们所知，我们是首批通过基数估计提高高维 DBSCAN 类聚类效率的研究者之一。我们已在 https://github.com/wyfunique/LAF-DBSCAN 开源代码。

我们的 LAF 增强型 DBSCAN 在评估中优于最先进的近似 DBSCAN 变体。具体而言，LAF 增强型 DBSCAN 的速度最高可达 DBSCAN 的 2.9 倍，比最先进的近似 DBSCAN 变体快 60% ~ 140%，同时在高维向量上保持较高的聚类质量；此外，LAF 也显著加速了所选的基于采样的 DBSCAN 变体（速度最高可达 6.7 倍），且质量损失极小，甚至没有质量损失。本文的主要贡献如下：

(1) 我们开发了 LAF，这是一种通用的学习型加速框架，可加速多种类 DBSCAN 聚类算法。(2) 我们利用该框架提出了一种新颖、高效的高维 DBSCAN 算法。(3) 我们在常用高维数据集上开展实验，展示了所提算法的高性能以及 LAF 的实用性。

2 方法

2.1 DBSCAN、LAF 增强型 DBSCAN

算法 1 以黑色文字展示了 DBSCAN 的伪代码（红色文字是 LAF 添加的，LAF 将在下文介绍）。DBSCAN 根据数据点在一定范围内的邻居数量，将数据点分类为核心点、非核心点和噪声点。给定距离函数 $d(\cdot, \cdot)$ 和距离阈值 ϵ，DBSCAN 会对每个数据点 P 执行范围查询，以查找其邻居 N $= \{Q | d(P, Q) < \epsilon\}$。如果邻居数量至少为 τ，则点 P 是核心点，当前簇会扩展到其邻居；否则，P 是非核心点或噪声点，当前簇不会从 P 扩展到其邻居。当当前簇无法继续扩展时，下一个簇将从其他某些未聚类核心点开始。该过程会重复进行，直到没有未分类的点（即算法 1 中的未定义）为止。非核心点与噪声点的区别在于其是否有核心点邻居。如果一个点本身是非核心点，但至少有一个核心点邻居，那么它就是非核心点，并且属于簇边界；而噪声点没有核心点邻居，不会被归入任何簇。为简化表述，在本文余下部分中，如果没有必要区分非核心点和噪声点，我们将二者统称为停止点 。

LAF 作为目标算法的插件运行：(1) 在每次范围查询之前放置基数估计器，只有当当前点被预测为核心点时，才执行范围查询。(2) 在聚类过程结束时插入后处理模块，用于检测将核心点错误地估计为非核心点的假阴性预测，并合并被此类误判停止点分隔的簇。这是为了弥补预测误差导致的有效性损失。基于 LAF，我们实现了一种高效的高维 DBSCAN，称为*LAF 增强型 DBSCAN*（又称 LAF-DBSCAN）。我们在算法 1 中给出其伪代码，并用红色文本突出显示 LAF 插入的代码行，其余代码行与原始 DBSCAN 相同。我们还开发了 *LAF-DBSCAN++*，这是经 LAF 增强的 DBSCAN++ [11] 版本，用于展示 LAF 加速 DBSCAN 变体的能力。详情将在第 3 节讨论。

基本上，LAF 插入了三个关键函数：CardEst、UpdatePar- tialNeighbors 和 PostProcessing，以及一个映射 E，用于记录所有被预测为停止点（称为预测停止点）的点及其部分邻居。这里我们使用“部分邻居”一词，是因为对于每个预测停止点，E 并不会记录其所有邻居，而只记录由 UpdatePartialNeighbors 生成的一个子集。CardEst 只需使用基数估计器来预测范围查询结果的数量，而另外两个函数都用于后处理。请注意，我们不会使用 CardEst 得到的 τ 的精确值来预测某个点是否为核心点。相反，τ 会乘以一个正因子 α，用于对预测基数进行阈值判定，如算法 1 第 6 行和第 22 行所示。这里使用 α 来调整假阳性率和假阴性率，使用户能够控制预测误差，并权衡速度与质量。具体而言，当 α 增大时，由于更多预测结果低于阈值，假阴性率会上升，从而带来更高的速度和更低的质量。当 α 减小时，假阳性率会上升，导致速度降低、质量提高。

如果某个点被预测为停止点（第 6–9 行、第 26–27 行），则将相应条目添加到 E 中；否则执行范围查询，并使用查询结果再次检查该点。同时，E 将通过 UpdatePartial Neighbors 使用查询结果进行更新。最后，后处理策略使用 E 更新聚类结果 C。

2.2 后处理策略

$\mathcal{E}$ 记录对每个预测停靠点而言，这是真实邻居的一个子集。UpdatePartialNeighbors 按照以下方式填充该集合（如算法2所示）：如果另一个点 P 将预测停靠点 P_n 判定为邻居，那么 P 也是 P_n 的邻居，并会被加入 $E(P_n)$。PostProcessing 函数（算法 3）会检测错误预测的停靠点，并合并被这些点分隔的簇。具体而言，如果 E 中的一个点至少有 τ 个部分邻居（第2行），则该点是假阴性。在这种情况下，PostProcessing 会随机选择其周围的一个簇作为

算法 1LAF 增强型 DBSCAN（LAF-DBSCAN）

输入： 数据集 D，距离函数 $d(\cdot,\cdot)$，距离阈值 ϵ，最小邻居数 τ，误差因子 α
输出： 从点到其簇 ID 的映射 C

```
簇 ID c：= 0
从预测停止点到部分邻居的映射 E：=∅
对每个 点 P in D 执行 C(P)：=未定义
对每个 点 P in D 执行
    如果 C(P) ≠ 未定义 则继续
    如果 基数估计(P) < ατ 则
        C(P) :=噪声
        如果 P 不在 E 则 E(P) := ∅
        继续
    邻居 N := 范围查询(D, d, P, ϵ)
    E :=UpdatePartialNeighbors(P, N, E)
    如果 |N| < τ 则
        C(P)：=噪声
        继续
    c：= c + 1
    C(P)：= c
    S：= N - {P}
    对于每个点 Q，执行
        若 C(Q) =噪声 则 C(Q) := c
        若 C(Q) ≠ 未定义 则继续
        C(Q) := c
        若 基数估计(Q) ≥ ατ 则
            N := 范围查询(D, d, Q, ϵ)
            E :=UpdatePartialNeighbors(Q, N, E)
            若 |N| ≥ τ 则 S := S ∪ N
        else
            若 Q 不在 E 则 E(Q) := ∅
C：= PostProcessing(C, E, τ)
返回C
```

算法2 UpdatePartialNeighbors

输入： 数据点 P，其邻居 N，映射 E
输出： 更新后的 E

```
对于每个 邻居 Pn ∈ N 执行
    如果 Pn ∈ E 则 E(Pn) := E(Pn) ∪ {P}
返回E
```

算法3 PostProcessing

输入： 从点到簇的映射 C，映射 E, τ
输出： 更新后的 C

```
对于每个点 P ∈ E 执行
    如果 |E(P)| ≥ τ 则
        从集合 E(P) 中随机选择一个非噪声邻居 P′
目标簇 ID c′ := C(P′)
将 E(P) 中的簇合并到目标簇中。
返回C
```

目标簇（第 3-4 行），并将其余被错误分开的簇合并到目标簇中（第 5 行）。

3 实验

3.1 实验设置

环境：一台 Lambda Quad 工作站，配备 28 个 3.30GHz Intel Core i9-9940X CPU、4 个 RTX 2080 Ti 图形处理器和 128 GB 随机存取存储器。
数据集：表 1 概述了用于评估的数据集，列出了它们的大小、数据维度、评估中使用的误差因子以及向量类型。下面将进一步介绍：

(1)《纽约时报》：包含 30 万个《纽约时报》新闻文章的词袋向量。我们从中随机采样 15 万个向量，对样本进行归一化，并通过高斯随机投影将其维度降至 256，这与*ANN- benchmark*[1]采用的方法相同。得到的数据集命名为*NYT-150k*。(2) Glove：在推文上预训练的 120 万个词嵌入（200 维）。我们从中采样 15 万个向量，并将采样得到的数据集命名为*Glove-150k*。(3) MS MARCO [16]：一个用于段落检索的基准数据集，包含 880 万个段落。我们采用与 [23]类似的方法处理该数据集，即使用与[23], 相同的深度模型为每个段落生成一个 768 维嵌入，将这些嵌入采样为若干子集，并将其命名为"MS-"加上规模（例如，"MS-100k"包含约 10 万个嵌入）。本文采样了 3 个数据集：MS-50k、MS-100k 和 MS-150k。

此外，我们对所有数据向量进行归一化，并按 8:2 的比例将每个数据集划分为训练集和测试集。对于每个数据集，我们首先在训练集上训练学习得到的基数估计器，然后在相应的测试集上评估所有方法；也就是说，报告的所有实验结果均是在这些测试集上收集的。

数据集	点数	Dim	α	Type
NYT-150k	150,000	256	1.15	词袋
Glove-150k	150,000	200	2.0	词嵌入
MS-150k	152,185	768	7.7	段落嵌入
MS-100k	107,400	768	2.0	段落嵌入
MS-50k	53,700	768	1.5	段落嵌入

表 1：评估数据集信息，包括点的数量（*#Points*）、数据维度（*Dim*）、LAF-DBSCAN 在各数据集上的误差因子 α以及向量类型（*Type*）。

指标： 如第 1 节所述，评估中使用的距离度量是余弦距离。对于仅支持欧几里得距离的一些基线方法，由于所有数据点都已归一化，我们使用公式 1 将余弦距离（d_{cos}）转换为欧几里得距离（d_{euc}），从而使我们的方法中的距离与基线方法中的距离等效。例如，根据该公式，当 $d_{cos} = 0.5$ 时，等效的 $d_{euc} = 1.0$，因此，如果我们在方法中设置距离阈值 $\epsilon = 0.5$，基线方法中的阈值将设为 1.0。

$$d_{euc}(\vec{u},\vec{v}) = \sqrt{2 d_{cos}(\vec{u},\vec{v})} \quad (\text{if } \|\vec{u}\| = \|\vec{v}\| = 1) \tag{1}$$

评估指标包括效率和有效性指标。对于效率，指标是聚类耗时（包括基数估计器的预测时间，不包括其训练时间）。对于有效性，指标包括：(1) 调整兰德指数（ARI）[10]；(2) 相对于真实值计算的调整互信息得分（AMI）[20], 。得分越高，聚类质量越好。这里我们使用原始 DBSCAN 的聚类结果作为真实值。

我们的方法：除了 LAF-DBSCAN 外，我们还使用 LAF 加速一种基于采样的 DBSCAN 变体 DBSCAN++ [11]。由此得到的方法称为 *LAF-DBSCAN++*，其目的是说明 LAF 不仅适用于 DBSCAN，也适用于其基于采样的变体（如第 1 节所述）。因此，它仅作为一种辅助方法，在评估中我们的主要方法仍然是 LAF-DBSCAN。在这两种方法中，基数估计器模型均为 RMI [13]，包含三个阶段，从顶层到底层分别包括 1、2、4 个全连接神经网络。每个神经网络有 4 个隐藏层，其宽度分别为 512、512、256 和

[1]https://github.com/erikbern/ann-benchmarks

128。该估计器曾在 [24]，中作为一个强基线使用，我们直接借用了其中的代码。对于每个训练集，基数估计器训练 200 个轮次，批大小为 512。

尽管还有其他学习型基数估计器，例如 CardNet [24] 和 SelNet [25]，，但本文不探讨哪种估计器最适用于我们的方法，因为这超出了本文的范围。具体而言，本文旨在揭示这种新思路在加速 DBSCAN 方面的潜力；在我们的评估中，RMI 已经表现出足够好的效果，能够证明这一潜力。**基线方法：** 基线方法如下：

(1) DBSCAN：原始的 DBSCAN。其聚类结果用作其他方法的真实值。

(2) DBSCAN++ [2] [11]：一种近似 DBSCAN 变体，通过对数据集进行采样，并将最繁重的计算限制在样本内来加速 DBSCAN。具体而言，DBSCAN++ 会对数据点的一个子集进行采样，并在整个数据集的范围内检测其中的核心点。随后，簇首先围绕子集内的这些核心点生长，最后将子集外所有未分类的数据点直接分配给距离最近的核心点。我们的 LAF-DBSCAN++ 方法建立在 DBSCAN++之上。

(3) KNN-BLOCK DBSCAN [3] [3]：一种近似 DBSCAN 变体，通过 K 近邻查询剪除不必要的距离计算，从而提高效率。我们在表格和图中将其简称为"KNN-BLOCK"。

(4) BLOCK-DBSCAN [4] [2]：一种与 KNN-BLOCK DBSCAN 类似的方法，但借助基于覆盖树的范围查询实现。

(5) ρ-近似 DBSCAN [5] [8, 9]：一种近似 DBSCAN 变体，通过采用近似因子 $\rho\,(\rho > 0)$ 放宽密度判据来加速 DBSCAN。

我们的方法和基线方法主要都使用 C++ 实现。**参数：** 除权衡评估外，所有实验中的关键参数均设置如下；权衡评估中的参数设置将在第 3.4 节单独介绍。(1) 不同实验中的距离阈值 ϵ 和邻居阈值 τ 均动态设置，具体设置将明确说明。(2) 对于 LAF-DBSCAN，误差因子 α 针对不同数据集采用临时设定，详见表 1。对于 LAF-DBSCAN++，其 α 固定为 1.0。(3) 对于 DBSCAN++，采样比例 p 会根据预测核心点的比例自动设置。具体而言，我们首先通过基数估计器得到被预测为核心点的点所占比例（记为 R_c），然后 $p = \delta + R_c$，其中 δ 是用户设定的偏移量，范围为 0.1 到 0.3。在我们的评估中，最终的 p 通常在 $0.2 \sim 0.6$ 范围内。并且，LAF-DBSCAN++ 的 p 与 DBSCAN++ 保持一致。(4) 对于 KNN-BLOCK DBSCAN，我们控制用于 K近邻搜索的 k均值树的两个参数：分支因子（设为 10）和待检查叶节点的比例（设为 0.6）。(5) 对于 BLOCK-DBSCAN，我们控制覆盖树的基数（设为 2），以及计算内部核心块之间最小距离时的最大迭代次数（即 [2]，中的 RNT 设为 10）。(6) 对于 ρ-近似 DBSCAN，我们设置 $\rho = 1.0$。

[2]代码可在 https://github.com/jenniferjang/dbscanpp 获取 [3]代码可在 https://github.com/XFastDataLab/KNN-BLOCK-DBSCAN 获取 [4]代码可在 https://github.com/XFastDataLab/BLOCK-DBSCAN 获取 [5]代码和二进制文件可在 https://sites.google.com/view/approxdbscan 获取；我们使用 2.0 版本

3.2 具有代表性的 (ϵ, τ) 与合适的 α 选择

我们根据噪声比例选择噪声点所占的比例。合适的 (ϵ, τ) 应满足：(1) 噪声比例较低至中等；(2) 簇数足够多，因为噪声过多或大多数点被归入极少数簇时，聚类便失去了意义。因此，我们通过网格搜索选择 (ϵ, τ) 的值，使得：(1) 大多数数据集的噪声比例小于 0.6；(2) 大多数数据集的簇数大于 20。表 2 展示了部分统计数据，每个单元格都包含相应情况下的一组*(噪声比例, 簇数)*。满足条件的单元格已突出显示。例如，(0.55, 5) 和 (0.6, 5) 都是合适的 (ϵ, τ)，因为它们中的任意一个都能使 3 个数据集中至少 2 个满足条件；而应避免使用 (0.5, 5) 和 (0.7, 5)。最后，我们选取三个 (ϵ, τ) 值，并在全文中报告：(0.5, 3)、(0.55, 5) 和 (0.6, 5)。

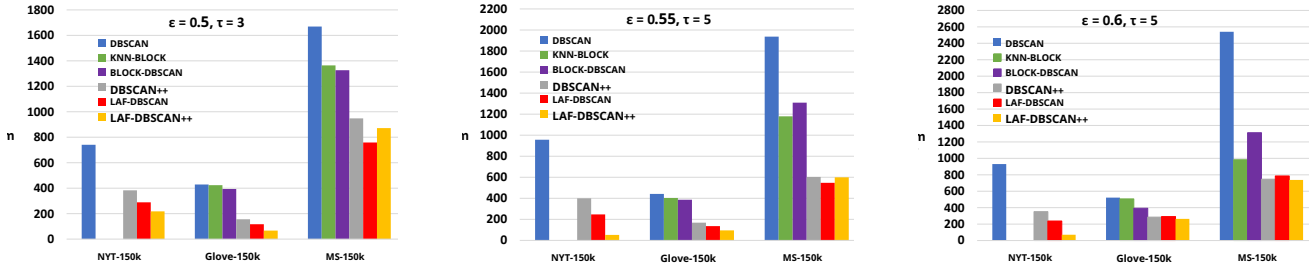
$(\epsilon,\ \tau)$	MS-50k	MS-100k	MS-150k
(0.5, 3)	(0.63, 654)	**(0.53, 1071)**	**(0.47, 1225)**
(0.5, 5)	(0.83, 174)	(0.72, 348)	(0.64, 380)
(0.55, 5)	(0.65, 183)	**(0.48, 223)**	**(0.39, 175)**
(0.6, 5)	**(0.38, 92)**	**(0.21, 70)**	**(0.15, 47)**
(0.7, 5)	(0.005, 1)	(0.0007, 1)	(0.0004, 1)

表2：关于噪声比例和簇数的部分统计数据。这些数据是在每个数据集上使用不同的 ϵ 和 τ 运行 DBSCAN 得到的。表中数据集名称下方的每个单元格都包含一对数值 （噪声比例、簇总数）**，合适的数值对以粗体标出。**

本节还讨论了误差因子 α 的适当设置。基本上，没有可量化的方法可以预测最佳的 α，因为这取决于数据集。本文采用的方法是网格搜索，我们的观察结果可以帮助指导用户：当

	$(\epsilon,\ \tau)$	方法	NYT-150k	Glove-150k	MS-150k
ARI	(0.5,3)	KNN-BLOCK	-	0.8597	**0.6004**
		BLOCK-DBSCAN	-	**0.8825**	0.4953
		DBSCAN++	**0.7933**	0.8129	0.4218
		LAF-DBSCAN	0.7731	0.8660	0.4134
		LAF-DBSCAN++	0.7321	0.7746	0.4113
	(0.55,5)	KNN-BLOCK	-	0.6942	0.1862
		BLOCK-DBSCAN	-	0.8508	0.2283
		DBSCAN++	1.0	0.7869	0.1321
		LAF-DBSCAN	1.0	**0.8520**	**0.2309**
		LAF-DBSCAN++	1.0	0.7444	0.1138
	(0.6,5)	KNN-BLOCK	-	0.2665	-0.0444
		BLOCK-DBSCAN	-	0.6399	0.0046
		DBSCAN++	1.0	0.7801	**0.3687**
		LAF-DBSCAN	1.0	**0.8797**	0.2643
		LAF-DBSCAN++	1.0	0.7653	0.3519
AMI	(0.5,3)	KNN-BLOCK	-	0.4994	**0.4254**
		BLOCK-DBSCAN	-	0.6613	0.3945
		DBSCAN++	0.6872	0.6369	0.3965
		LAF-DBSCAN	**0.7050**	**0.7558**	0.4196
		LAF-DBSCAN++	0.6245	0.5947	0.3879
	(0.55,5)	KNN-BLOCK	-	0.3391	0.1738
		BLOCK-DBSCAN	-	0.6364	0.2626
		DBSCAN++	1.0	0.6578	0.2288
		LAF-DBSCAN	1.0	**0.7554**	**0.3017**
		LAF-DBSCAN++	1.0	0.6068	0.2210
	(0.6,5)	KNN-BLOCK	-	0.1427	0.0390
		BLOCK-DBSCAN	-	0.4988	0.1259
		DBSCAN++	1.0	0.7061	**0.2836**
		LAF-DBSCAN	1.0	**0.8167**	0.2763
		LAF-DBSCAN++	1.0	0.6822	0.2750

表3：三种最大数据集上近似方法的聚类质量（AMI 和 ARI 分数）



(a) $\epsilon = 0.5, \tau = 3$。 **(b)** $\epsilon = 0.55, \tau = 5$。 **(c)** $\epsilon = 0.6, \tau = 5$。

图1：三个最大数据集上所有方法的聚类时间

(ϵ, τ)	MS-50k	MS-100k	MS-150k
(0.5, 3)	864.7秒/206.6秒	3499.8秒/882.7秒	6931.1秒/1669.5秒
(0.55, 5)	854.7秒/180.3秒	3367.0秒/827.6秒	6595.1秒/1936.6秒
(0.6, 5)	753.3秒/219.9秒	2817.9秒/1041.1秒	5385.9秒/2539.9秒

表 4：不同数据集规模下 ρ-近似 DBSCAN 与 DB- SCAN 的聚类时间。每个单元格显示一对时间"t_1/t_2"，其中 t_1 是 ρ-近似 DBSCAN 的时间，t_2 是在相同（ϵ，τ）和数据集条件下 DBSCAN 的时间。

向量类型固定时（例如，稠密神经嵌入），α应随着数据集规模增大或数据维度升高而增大。表 1 中 Glove 和三个 MS 数据集的结果可以印证这一点。原因可能是训练集中的偏差。例如，根据表 2，随着数据规模增大，噪声比例降低，这意味着核心点所占比例增加。训练中的这种偏差使基数估计器更倾向于预测较大的值。因此，α也应相应增大。

3.3 效率与有效性评估

我们首先在三个最大的数据集 NYT-150k、Glove-150k 和 MS-150k 上评估每种方法的效率和有效性。表 3 通过 ARI 和 AMI 分数（越高越好）报告了所有近似方法的聚类质量。由于 DBSCAN 作为真实值，因此未列入表中。图1展示了这些方法的聚类时间。由于 KNN-BLOCK DBSCAN 和 BLOCK-DBSCAN 存在未知错误，无法在 NYT-150k 上运行，因此它们在 NYT-150k 上的结果未出现在表 3 和图1中。需要注意的是，由于 ρ- 近似 DBSCAN 在高维数据上的效率极低，我们没有在表 3 、图1或后续任何实验中纳入该方法。具体而言，[8], 越大，ρ 会使 ρ-近似 DBSCAN 更高效，而 ρ 在 [8]中的取值范围为 0.001 至 0.1。然而，尽管我们在评估中将 ρ 增大到了 1.0，该方法仍表现出较低的效率，甚至比朴素 DBSCAN 更慢，如表 4 所示。这意味着它深受维数灾难的影响，不应应用于高维空间。[2] 进一步解释了 ρ-近似 DBSCAN 存在这一问题的原因。

观察到：（1）在大多数情况下，LAF-DBSCAN 和 LAF-DBSCAN++ 的效率最高。例如，LAF-DBSCAN 的速度最高可达 DBSCAN 的 2.9 倍，同时比 DBSCAN++快 1.6 倍、比 KNN-BLOCK DBSCAN 快 2.2 倍、比 BLOCK-DBSCAN 快 2.4 倍。（2）LAF-DBSCAN 在大多数情况下都能获得最高的聚类质量；在 NYT-150k 这三个方法得分相同时，LAF-DBSCAN 所需时间仅为 DBSCAN++的 60% ～ 70，如图1所示。（3）LAF-DBSCAN++ 的聚类质量通常略低于 DBSCAN++，但效率提升显著（即速度最高可达 DBSCAN++的 6.7 倍），因此其速度—质量权衡能力优于 DBSCAN++，更具实用性。（4）受维度灾难影响，所有方法在 MS-150k 上的表现都不如在其他数据集上。具体而言，维度越高，分布通常越复杂。对于 LAF 来说，这类分布更难拟合，因而会产生更多漏检预测（FN）。例如，当 $\epsilon = 0.5$，$\tau = 3$时，NYT/Glove/MS-150k 中的 FN 数量分别为 5687/2010/7425，这与表3中的结果呈负相关。更复杂的分布也会降低采样的代表性和近邻搜索的有效性，从而降低基线方法的聚类质量。

3.4 速度-质量权衡评估

我们使用 MS-150k 和 Glove-150k，并采用设置 $\epsilon = 0.5$，$\tau = 3$，以展示第3.3节中讨论的除 ρ-近似 DBSCAN 之外所有近似方法的速度—质量权衡能力。我们通过调整采样比例 p 来改变 DBSCAN++ 和 LAF-DBSCAN++ 的性能；具体做法是将偏移量 δ（见第3.1节）在 0.1 ～ 0.9 范围内变化，而对于 LAF-DBSCAN，误差因子 α 在 1.1 到 15.0 之间变化（在 LAF-DBSCAN++ 中固定为 1.0）。对于 KNN-BLOCK DBSCAN，我们将分支因子在 3 ～ 20 范围内变化，并将叶节点比例设为 0.001 到 0.3。对于 BLOCK-DBSCAN，我们将覆盖树基数从 1.1 调整到 5，同时将最大迭代次数固定为 10。

如图2和图3所示，在 Glove-150k 和 MS-150k 上，LAF-DBSCAN 和 LAF-DBSCAN++在高质量区域（例如 AMI ＞ 0.4 的区域）都具备最佳的速度—质量权衡能力。鉴于现实世界通常要求相对较高的聚类质量，我们得出结论：在实际应用中，LAF-DBSCAN 和 LAF-DBSCAN++ 比所有基线方法都具备更好的权衡能力。此外，在两个数据集上，LAF-DBSCAN++ 相比 DBSCAN++呈现出更好的权衡，这意味着 LAF 在质量损失相对极小的情况下显著减少了 DBSCAN++ 的聚类时间，进一步证明了 LAF 对广泛 DBSCAN变体的优势和实用性。

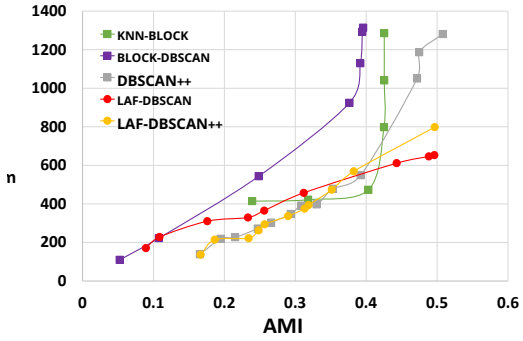


图2：数据集 MS-150k 上近似方法的速度-质量权衡曲线

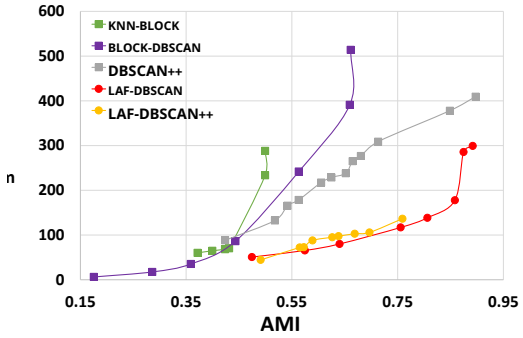


图3：近似方法在数据集Glove-150k上的速度-质量权衡曲线

	方法	MS-50k	MS-100k	MS-150k
ARI	KNN-BLOCK	0.7577	0.3828	0.1862
	BLOCK-DBSCAN	**0.7710**	0.4632	0.2283
	DBSCAN++	0.7238	0.4690	0.1321
	LAF-DBSCAN	0.7581	**0.5524**	**0.2309**
	LAF-DBSCAN++	0.6455	0.3077	0.1138
AMI	KNN-BLOCK	0.5708	0.2736	0.1738
	BLOCK-DBSCAN	0.6134	0.3518	0.2626
	DBSCAN++	0.6264	0.4494	0.2288
	LAF-DBSCAN	**0.7043**	**0.5034**	**0.3017**
	LAF-DBSCAN++	0.5328	0.3197	0.2210

表5：不同规模数据集上所有近似方法的聚类质量（$\epsilon = 0.55, \tau = 5$）

3.5 可扩展性评估

为评估可扩展性，我们在三个不同规模的 MS 数据集上运行了所有方法，并报告了 $\epsilon = 0.55$ 和 $\tau = 5$的结果，因为其他（ϵ， τ）的结果相似。质量分数列于表5中，效率结果展示在图4中；为便于查看，我们在图中用聚类时间数值标注了 MS-150k 的各个点。这些结果证明，基于以下观察，我们采用 LAF 增强的方法具有很高的有效性和可扩展性：(1) 与效率和有效性评估中的情况类似，在这里 LAF-DBSCAN 在大多数情况下仍能达到最佳质量，并且在最大数据集上具有最高速度（其时间最短，为547.2秒）。(2) 随着数据规模增大，LAF-DBSCAN 的聚类时间增长最慢，表明其可扩展性高于基线方法。(3) 在大多数情况下，LAF-DBSCAN++ 的质量接近 DBSCAN++，而在其他情况下

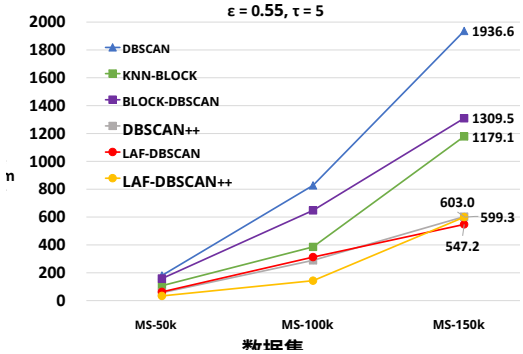


图4：不同规模数据集上所有方法的聚类时间（$\epsilon = 0.55, \tau = 5$）

(ϵ, τ)	数据集	MC/TC	MP/TPC	ASMC
(0.5, 3)	NYT-150k	63/92	209/19358	3.32
(0.55, 5)	Glove-150k	39/81	250/7879	6.41
(0.55, 5)	MS-150k	159/176	1107/18384	6.96

表 6：LAF-DBSCAN 完全漏检簇的统计数据。*MC*，完全漏检簇的数量；*TC*，真实簇的总数；*MP*，漏检数据点的数量；*TPC*，属于真实簇的数据点总数，即非噪声点；*ASMC*，完全漏检簇的平均大小。

随着数据规模的增加，它们迅速接近，表明 LAF-DBSCAN++比 DBSCAN++具有更高的可扩展性。

3.6 漏检簇分析

除了第2节讨论的错误拆分簇误差外，如果某个簇的所有核心点都被错误地预测为非核心点或噪声点，该簇也可能被完全漏检。幸运的是，这种簇完全漏检的错误对聚类质量的影响可以忽略不计，因为它通常发生在非常小的簇中。我们根据表3选取 LAF-DBSCAN 在每个数据集上聚类质量最低的情形（即，在 NYT-150k 上为 $(\epsilon,\tau) = (0.5,3)$，在 Glove-150k 和 MS-150k 上为 (0.55,5)），并在表6中报告完全漏检簇的相关信息。尽管在最坏情况下，LAF-DBSCAN 完全漏检的簇超过 50%，但它仍能找到主要簇，因为漏检簇合计仅包含 ~6% 的非噪声点。漏检簇的平均大小（ASMC）也非常小（即平均仅包含 3 ~ 7 个点），不会对整体聚类质量造成实质性影响。因此，本文不再进一步讨论这种错误。

4 结论与未来工作

为提升角距离下高维 DBSCAN 类聚类的效率和可扩展性，我们提出了 LAF，这是一种通用的学习型加速框架，它利用学习型基数估计技术减少聚类过程中不必要的范围查询，并通过检测漏检数据点、合并被错误拆分的簇来弥补质量损失。评估结果表明，与最先进的高效 DBSCAN 方法相比，LAF增强方法在保持较高质量的同时，效率显著更高，速度—质量权衡也更优。

其性能也优于基线方法。本研究的主要局限在于适用的距离度量范围有限。但由于距离度量不受硬性约束，我们的方法很容易适配其他距离，这将作为未来的研究方向。未来工作还将研究所用基数估计器的影响，并深入探讨合适的 α 等问题。

参考文献

[1] Yewang Chen、Shengyu Tang、Nizar Bouguila、Cheng Wang、Jixiang Du 和 HaiLin Li。2018。通过剪枝减少不必要的距离计算，实现适用于高维数据的快速 DBSCAN 聚类算法。《模式识别》 83 （ 2018 ），375–387。[2] Yewang Chen 、Lida Zhou、Nizar Bouguila、Cheng Wang、Yi Chen 和 Jixiang Du。2021。BLOCK-DBSCAN：面向大规模数据的快速聚类。《模式识别》109（2021），107624。https://doi.org/10.1016/j.patcog.2020.107624[3] Yewang Chen、Lida Zhou、Songwen Pei、Zhiwen Yu、Yi Chen、Xin Liu、Jixiang Du 和 Naixue Xiong。2019。KNN-BLOCK DBSCAN：面向大规模数据的快速聚类。《IEEE 系统、人类与控制论汇刊：系统》51，6（2019），3939–3953。[4] Difei Cheng、Ruihang Xu 和 Bo Zhang。2021。基于密度的聚类方法的快速密度估计。arXiv 预印本 arXiv:2109.11383（2021）。[5] Igor de Moura Ventorim、Diego Luchi、Alexandre L. Rodrigues 和 Flávio Miguel Varejão。2021。BIRCHSCAN：一种将 DBSCAN 应用于大型数据集的抽样方法。《Expert Syst. Appl.》184（2021），115518。[6] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2018。BERT：用于语言理解的深度双向变换器预训练。https://doi.org/10 48550 1810 04805 。/ARXIV。. Martin Ester、Hans-Peter Kriegel、Jörg Sander、Xiaowei Xu 等。1996。一种用于在含噪声的大型空间数据库中发现簇的基于密度的算法。载于 KDD，第 96 卷。226–231。[8] Junhao Gan 和 Yufei Tao。2015。重新审视 DBSCAN：错误主张、不可修复性与近似。载于《2015 年 ACM SIGMOD 数据管理国际会议论文集》（澳大利亚维多利亚州墨尔本）（SIGMOD '15）。美国计算机协会，纽约州纽约，美国，519–530。https://doi.org/10 1145 2723372 2737792 。/。Junhao Gan 和 Yufei Tao。欧几里得 DBSCAN 的困难性与近似。ACM Trans. Database Syst. 42，3，文章 14（2017 年 7 月），45 页。https://doi.org/10 1145 3083897 。/ Lawrence Hubert 和 Phipps Arabie。1985。比较划分。《分类学杂志》2，1（1985），193–218。[11] Jennifer Jang 和 Heinrich Jiang。2018。DBSCAN++：迈向快速且可扩展的密度聚类。https://doi.org/10 48550 1810 13105 。/ARXIV。. Heinrich Jiang、Jennifer Jang 和 Jakub Lacki。通过子采样相似性查询加速 DBSCAN。《神经信息处理系统进展》33（2020），22407–22419。[13] Tim Kraska、Alex Beutel、Ed H Chi、Jeffrey Dean 和 Neoklis Polyzotis。2018。学习型索引结构的价值。载于《2018 年数据管理国际会议论文集》。489–504。[14] Diego Luchi、Alexandre Loureiros Rodrigues 和 Flávio Miguel Varejão。2019。将 DBSCAN 应用于大型数据集的抽样方法。《模式识别快报》117（2019），90–96。[15] Yinghua Lv、Tinghuai Ma、Meili Tang、Jie Cao、Yuan Tian、Abdullah Al-Dhelaan 和 Mznah Al-Rodhaan。2016。一种适用于复杂结构数据集的高效、可扩展的基于密度的聚类算法。《神经计算》171（2016），9–22。[16] Tri Nguyen、Mir Rosenberg、Xia Song、Jianfeng Gao、Saurabh Tiwary、Rangan Majumder 和 Li Deng。2016。MS MARCO：人工生成的机器阅读理解数据集。载于 CoCo@ NIPS。[17] Jianbin Qin、Wei Wang、Chuan Xiao、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）。美国计算机协会，纽约州纽约，美国，4062 4063 10 1145 3447548 3470811 –。https://doi.org/。/。[18] Ji Sun、Guoliang Li 和 Nan Tang。学习型相似性查询基数估计。载于《2021 年数据管理国际会议论文集》。1745–1757。[19] Yao Tian、Tingyun Yan、Xi Zhao、Kai Huang 和 Xiaofang Zhou。2022。度量空间中精确相似性搜索的学习型索引。https://doi.org/10.48550/ ARXIV.2204.10028[20] Nguyen Xuan Vinh、Julien Epps 和 James Bailey。2010。用于聚类比较的信息论度量：变体、性质、归一化及偶然性校正。《机器学习研究杂志》11（2010），2837–2854。[21] P Viswanath 和 V Suresh Babu。2009。Rough-DBSCAN：一种面向大型数据集的快速混合式基于密度的聚类方法。《模式识别快报》30，16（2009），1477–1488。[22] Xiao Wang、Craig Macdonald、Nicola Tonellotto 和 Iadh Ounis。2021。用于多表示稠密检索的伪相关反馈。载于 *ACM SIGIR*信息检索理论国际会议论文集。297–306。[23] Yifan Wang、Haodi Ma 和 Daisy Zhe Wang。2022。LIDER：一种面向大规模稠密段落检索的高效高维学习型索引。https: //doi.org/10.48550/ARXIV.2205.00970[24] Yaoshu Wang、Chuan Xiao、Jianbin Qin、Xin Cao、Yifang Sun、Wei Wang 和 Makoto Onizuka。2020。相似度选择的单调基数估计：一种深度学习方法。载于*ACM SIGMOD* 数据管理国际会议论文集。1197–1212。[25] Yaoshu Wang、Chuan Xiao、Jianbin Qin、Rui Mao、Makoto Onizuka、Wei Wang、Rui Zhang 和 Yoshiharu Ishikawa。2021。面向高维数据的一致且灵活的选择率估计。载于*2021* 年数据管理国际会议论文集。2319–2327。[26] Shaoyuan Weng、Jin Gou 和 Zongwen Fan。2021。 *h*-DBSCAN：一种简单快速的大数据 DBSCAN 算法。载于亚洲机器学习会议。PMLR，81–96。[27] Seungil You、David Ding、Kevin Canini、Jan Pfeifer 和 Maya Gupta。2017。深度格网络与部分单调函数。《神经信息处理系统进展》30（2017）。[28] Jingtao Zhan、Jiaxin Mao、Yiqun Liu、Jiafeng Guo、Min Zhang 和 Shaoping Ma。2022。通过约束聚类学习离散表示，实现高效有效的稠密检索。载于第十五届 *ACM*网络搜索与数据挖掘国际会议论文集（线上活动，亚利桑那州，美国）*(WSDM '22)*。美国计算机协会，纽约州纽约，美国，1328–1336。https://doi.org/10.1145/3488560.3498443