软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(3):1006−1020 [doi: 10.13328/j.cnki.jos.007517] [CSTR: 32375.14.jos.007517]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

向量数据库的 K 近邻图高效更新方法*

王嘉翼, 徐士惠, 李国良

(清华大学 计算机科学与技术系, 北京 100084)

通信作者: 李国良, E-mail: liguoliang@tsinghua.edu.cn

摘　要: 在高维数据处理中, K 近邻图作为一种关键的数据结构, 广泛应用于聚类、图神经网络和推荐系统等领域. 然而, 随着预训练嵌入模型在非结构化数据建模与检索中的广泛使用, 嵌入模型的微调逐渐成为提升嵌入向量的语义表示能力的核心步骤. 嵌入微调通常会导致全部数据的向量表示发生系统性变化, 从而使原有 K 近邻图的邻接关系失效. 现有研究主要关注于如何为静态数据构建 K 近邻图, 缺乏对微调后的嵌入向量进行快速适应的研究. 为此, 提出一种面向嵌入模型微调场景的高效 K 近邻图更新方法 FastAdjust. 该方法基于嵌入模型微调为每条数据嵌入带来的影响较小的观察, 通过局部更新策略对原始 K 近邻图进行增量调整, 在确保最终 K 近邻图质量的同时, 显著提升更新效率. 具体而言, 首先, FastAdjust 利用基于乘积量化的聚类结构, 为每条数据高效且准确地定位可能成为邻居的数据子集, 缩小候选邻居搜索范围; 其次, 基于数据密度和嵌入变化幅度, FastAdjust 结合二者与数据 K 近邻变化程度的相关性, 为邻居关系变化程度不同的数据针对性地分配不同的更新资源, 从而提升整体更新效率. 真实数据集上的实验结果表明, FastAdjust 在嵌入模型微调的场景下能够快速调整 K 近邻图, 准确地适应数据嵌入的变化, 同时大幅减少计算开销, 具有良好的实用价值和扩展性.

关键词: K 近邻图; 近似最近邻搜索; 嵌入模型

中图法分类号: TP311

中文引用格式: 王嘉翼, 徐士惠, 李国良. 向量数据库的K近邻图高效更新方法. 软件学报, 2026, 37(3): 1006–1020. http://www.jos.org.cn/1000-9825/7517.htm

英文引用格式: Wang JY, Xu SH, Li GL. Efficient Updating Method for K-nearest Neighbor Graph in Vector Databases. Ruan Jian Xue Bao/Journal of Software, 2026, 37(3): 1006–1020 (in Chinese). http://www.jos.org.cn/1000-9825/7517.htm

Efficient Updating Method for K-nearest Neighbor Graph in Vector Databases

WANG Jia-Yi, XU Shi-Hui, LI Guo-Liang

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: In high-dimensional data processing, the K-nearest neighbor (KNN) graph is a critical data structure widely used in tasks such as clustering, graph neural networks, and recommendation systems. However, with the increasing use of pretrained embedding models in unstructured data modeling and retrieval, embedding model fine-tuning has become a key step in enhancing the semantic representation capability of embeddings. Such fine-tuning often leads to systematic changes in the vector representations of all data points, which invalidates the original neighborhood relationships in the KNN graph. Existing research primarily focuses on building KNN graphs for static data, lacking efficient solutions for adapting to updated embeddings after fine-tuning. To address this gap, this study proposes FastAdjust, an efficient KNN graph update method tailored for embedding model fine-tuning scenarios. Leveraging the observation that fine-tuning usually causes only minor changes to individual embeddings, incremental adjustments to the original KNN graph are performed

* 基金项目: 国家重点研发计划 (2023YFB4503600); 国家自然科学基金 (62525202, 62232009); 深圳市承接国家重大科技项目 (CJGJZD20230724093403007); 高速铁路与城轨交通系统技术国家工程研究中心实验室基础研究项目&中国国家铁路集团有限公司科技研究开发计划 (L2024W001)
本文由“向量数据库及 DB4LLM 技术”专题特约编辑高宏教授、李国良教授、张蓉教授推荐.
收稿时间: 2025-05-06; 修改时间: 2025-06-30; 采用时间: 2025-08-20; jos 在线出版时间: 2025-09-02
CNKI 网络首发时间: 2026-01-15

by FastAdjust through a local update strategy, significantly improving update efficiency while maintaining graph quality. Specifically, FastAdjust first employs a clustering structure based on product quantization to efficiently and accurately locate a subset of candidate neighbors for each data point, thus narrowing the search space. Secondly, based on data density and the magnitude of embedding variation, FastAdjust leverages their correlation with changes in the KNNs to adaptively allocate update resources according to the degree of neighbor relationship changes, thus improving overall update efficiency. Experimental results on real-world datasets demonstrate that FastAdjust efficiently and accurately adapts KNN graphs to embedding updates with significantly reduced computational cost, showing strong practical value and scalability.

Key words: K-nearest neighbor (KNN) graph; approximate nearest neighbor search; embedding model

随着深度学习和嵌入技术的发展, 将非结构化数据 (如文本、图像、音频) 转化为稠密向量的嵌入模型 (embedding model)[1]已成为构建高效语义表示与分析系统的核心手段. 在诸如文本聚类[2]、推荐系统[3]、图神经网络[4–6]建模等任务中, 基于嵌入向量构建的 K 近邻图 (K-nearest neighbor (KNN) graph)[7]是一种常用且有效的图结构, 可用于刻画数据点间的相似性关系. K 近邻图通过为每个数据点连接与其最相近的 K 个邻居构建边, 从而在向量空间中形成反映语义关系的图结构. 然而, 由于嵌入向量的维度较高, 准确构建 K 近邻图的时间成本较高. 因此, 通常采用近似方法来构建 K 近邻图, 以降低计算代价. 与之相似, 本文也聚焦于近似 K 近邻图的构建. 更准确的 K 近邻图是后续图学习、图搜索与推荐等应用的结构基础[8].

然而, 随着预训练嵌入模型在实际系统中的广泛部署, 如何应对模型微调 (fine-tuning) 带来的向量语义变化, 已经成为 K 近邻图构建与更新过程中亟待解决的关键问题. 在实际应用中, 如领域自适应、图神经网络辅助建图、交互式标签反馈以及推荐系统中的嵌入更新等场景中, 嵌入模型往往不能经过一次性训练后永久使用. 相反, 它们需要根据特定任务的分布特征、用户行为变化或新增的反馈数据进行持续优化和微调. 例如, 在领域自适应中, 模型需要将原始训练领域的嵌入能力迁移到目标领域; 在图建模中, 随着图结构的更新或边的变化, 节点嵌入也需随之调整; 而在交互式系统中, 用户反馈的标签信息能够引导嵌入空间更精确地拟合当前任务需求; 对于推荐系统, 则必须根据实时用户行为和点击日志动态更新嵌入, 以保持个性化推荐的有效性. 这些任务要求嵌入模型具备良好的可持续学习能力和对任务变化的适应能力.

在上述场景中, 嵌入模型被微调, 使得所有数据点的嵌入向量表示均发生变化, 导致原有 K 近邻图结构失效. 图 1 给出了嵌入模型微调前后 K 近邻图的变化示例, 在嵌入模型微调前后, 各个节点的嵌入向量发生变化, 导致向量之间的距离发生变化, 进而造成 K 近邻图中存储的各个节点前 K 近的邻居信息不再准确, 需要进行更新和调整.

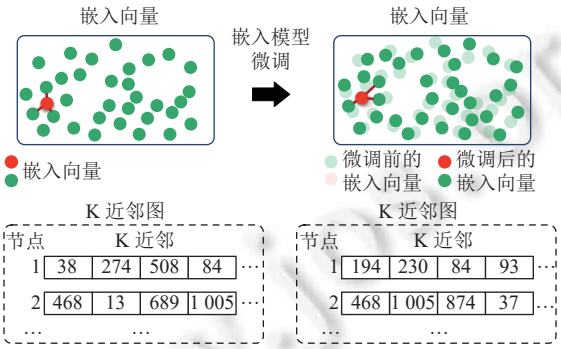

图 1 嵌入模型微调前后 K 近邻图的变化示例

现有的 K 近邻图构建方法虽然在静态场景下表现良好[9–11], 但缺乏对动态嵌入微调的支持. 现有的图增量更新算法大多针对图中少量节点的局部变动进行优化, 如优化数据插入或删除时的索引更新[12–18]. 然而, 这些更新方法难以有效应对模型微调导致的全局性、同步性的嵌入向量变化. 具体来说, 这种大规模变化的动态场景面临着以下挑战.

(1) 高维空间搜索的复杂性. 模型微调后, 由于数据均发生改变, 高维嵌入空间中数据点的近邻关系可能发生

显著改变. 在广阔的搜索空间中难以准确、高效地定位潜在的新近邻集合.

(2) 更新影响的非均匀性. 嵌入向量的更新对不同数据点近邻关系的影响程度并不一致, 这使得为所有数据点均匀分配 K 近邻图更新资源的方式效率低下. 而这种不均匀性难以在计算出更新后的 K 近邻图前准确捕捉.

为了应对上述挑战, 本文提出了一种面向嵌入模型微调场景的高效 K 近邻图更新方法. 该方法的核心思想是避免全图重建, 聚焦关键区域的局部更新, 从而在保证 K 近邻图质量的前提下, 大幅提升更新效率. 为此, 我们设计了以下的关键更新策略.

(1) 基于乘积量化的候选近邻快速定位. 对于第 1 个挑战, 我们观察到, 尽管模型微调会导致全局嵌入向量的变化, 但对单个数据点而言, 其 K 近邻集合的变化通常具有一定的局部性——即新的近邻更有可能出现在微调前数据点的附近. 基于此, 本文首先在嵌入微调前对数据进行乘积量化处理. 在微调后, 我们基于量化结果可以快速估计数据点间的近似距离, 从而高效筛选出可能成为目标点新邻居的候选集合, 避免不必要的全局搜索, 显著减少搜索空间.

(2) 基于数据密度与数据变化幅度的动态更新资源分配. 对于第 2 个挑战, 尽管嵌入微调会对不同数据点的 K 近邻关系带来不同的影响, 并且这种影响无法直接计算, 但这种影响可以通过与其相关的关键指标进行预测. 为此, 本文提出基于数据密度和微调前后数据变化幅度, 估算 K 近邻变化程度的方法. 基于该预测结果, 本文为每个数据点针对性地分配不同的更新资源, 从而实现更新效率的提升.

通过上述策略, 本文所提出的方法在嵌入模型微调后, 能够在保证图结构准确性的同时, 有效降低 K 近邻图更新的时间与计算开销, 适用于大规模、动态演化的数据场景. 第 4 节在真实数据集上的实验结果验证了该方法在嵌入微调场景下的效率优势与结构准确性.

图 2 描述了嵌入模型微调的场景下, 不同 K 近邻图更新策略的比较. 第 1 种方式是直接忽略嵌入的变化, 继续使用微调前的嵌入数据建立的 K 近邻图进行查询, 如图 2(a) 所示. 这种不调整 K 近邻图的方法不需要对 K 近邻图中各点的邻居进行重新计算和调整, 因此不需要花费额外的时间, 效率高, 但由于嵌入发生了大面积变化, 因此产生的误差较大, 查询准确度低. 第 2 种方式是使用微调后的数据从头重新构建 K 近邻图, 并使用新的 K 近邻图进行查询, 从而达到较高的准确度, 如图 2(b) 所示. 尽管可以通过这种全量重建 K 近邻图的方法来解决此问题, 但重建过程计算成本高、资源开销大, 难以满足大规模在线系统中对实时性和效率的要求. 本文提出的面向嵌入模型微调场景的高效 K 近邻图更新方法如图 2(c) 所示. 该方法增量调整 K 近邻图, 基于微调后的数据对原 K 近邻图进行局部调整, 形成新的 K 近邻图, 并通过新 K 近邻图进行查询. 该方法通过快速定位候选数据与基于数据分布进行针对性更新资源分配, 实现了较高的准确度, 同时避免了重新构建 K 近邻图, 更新效率高. 整体上看, 我们的方法结合了前两种方法的优点, 取得了准确度和更新效率之间的平衡.

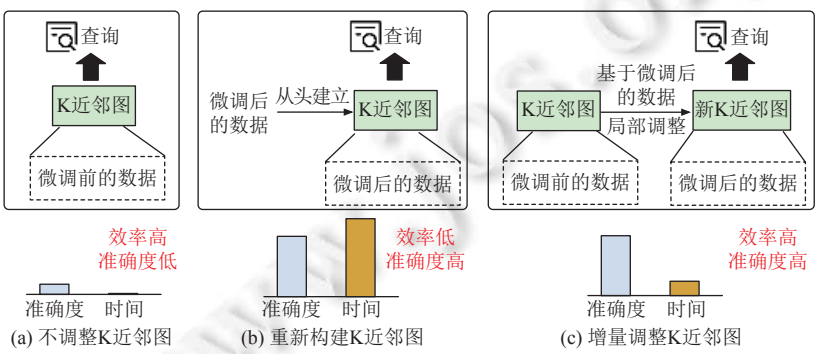

图 2 不同嵌入模型微调下 K 近邻图更新策略的比较

本文第 1 节介绍高效 K 近邻图构建和更新方法的相关工作和研究现状. 第 2 节介绍本文面临的嵌入模型微调的场景及其对 K 近邻图构建的影响. 第 3 节介绍本文提出的嵌入模型微调下的高效 K 近邻图更新算法. 第 4 节

通过在文本及图像的真实数据集上实验, 从 K 近邻图的准确度和更新的时间效率这两个方面全面验证本方法的有效性. 第 5 节总结全文, 并介绍未来工作.

1 相关工作

1.1 高效的 K 近邻图构建算法

K 近邻图是一种广泛应用于高维数据处理中的基础结构, 常用于聚类、图神经网络、图嵌入、推荐系统、图搜索等任务中[6]. 其基本思想是: 对于数据集中每一个样本点, 寻找与之距离最近的 K 个邻居, 并以此建立有向或无向图结构[7]. 在实际应用中, 由于数据量巨大、维度较高, 如何高效、准确地构建 K 近邻图是一个重要的研究问题[8].

目前已有多种用于构建 K 近邻图的算法被提出, 其中也包含用于近似最近邻查找的索引, 因为可以在近似最近邻索引建立后, 通过为每一条数据查询近似的 K 近邻来构建 K 近邻图. 这些方法主要可分为以下几类.

● 暴力搜索. 暴力搜索是最直观和精确的构建 K 近邻图的方法. 它通过计算所有点对之间的距离, 从中选取每个点距离最近的 K 个点作为其邻居, 并通常采用欧氏距离、余弦距离等度量方式. 该方法能获得最精确的 K 近邻图, 构图质量最优, 但时间复杂度高, 在大规模数据上难以应用, 且无法扩展到动态更新或流数据的场景.

● 基于树结构的方法. 其基本原理是通过构建空间划分树 (如 KD-Tree[9]) 对数据进行递归划分, 使得 KNN 查询时仅需遍历部分节点, 从而加快近邻搜索效率. Wang 等人[10]采用分治方法将点递归划分为子集, 形成随机划分树, 并在每个子集上构建精确邻域图, 通过邻域传播方法提高准确性. 这类方法在低维数据中性能优异, 能显著提升查询效率, 但在高维空间中空间划分效果变差, 导致这类方法退化为近似于线性扫描的过程. 同时, 该方法构建树结构的过程也需要较多时间和较大空间.

● 图增量构建方法. 这类方法先随机为每个点选择 K 个邻居来构建初始的邻接图, 再通过局部搜索或图遍历迭代优化邻居列表. Dong 等人[11]提出了 NN-Descent 方法, 该方法先随机初始化每个点的邻居集合, 然后利用“邻居的邻居也是可能的邻居”原则, 通过迭代地检查邻居和邻居之间的距离, 逐步优化邻接关系. Malkov 等人[12]提出了 HNSW 方法, 引入分层图结构, 从上层粗粒度搜索到下层精细搜索, 加速图的构建过程并提升查询准确性. 图增量构建方法在高维、大规模数据场景下表现出较好的效率与精度平衡, 且具有较好的可扩展性, 部分方法支持一定程度的动态更新, 适用于在线场景, 因此已成为主流方案. 然而, 这类方法的问题主要在于: 近似算法构建结果非最优, 可能影响后续任务精度; 算法的超参数 (如迭代轮数、邻接个数) 较多, 调节参数的过程复杂; 构图过程涉及随机性, 效果可能存在波动.

● 基于哈希和量化的方法. 这类方法借助局部敏感哈希 (locality sensitive hashing, LSH)、乘积量化 (product quantization, PQ)[19,20]等技术进行高效近似 K 近邻索引构建, 批量进行 K 近邻查询, 再合并结果构建 K 近邻图. Faiss (Facebook AI similarity search) 支持多种索引类型, 并结合 GPU 加速. 这类方法构图效率高, 适合处理千万级以上样本, 适用于大规模的工业级应用, 但构图结果的精度受限于索引结构和参数设置. 某些方法 (如量化) 可能对语义空间造成破坏.

1.2 K 近邻图更新方法

在 K 近邻图构建之后, 如果数据发生变化 (如新增点、删除点或已有点的特征发生更新), 完全重建 K 近邻图代价较高. 因此, 研究者提出了 K 近邻图的增量更新方法.

对于新增点的插入更新, 当一个新样本加入时, 无须重建整个图, 可以通过以下方式更新: 使用现有 K 近邻图 (或其索引结构) 对新点进行 K 近邻查询, 根据距离条件将该新点作为邻居加入这些近邻的邻接列表中, 并将其自身的邻接列表设置为查到的 K 个点. 在初始插入后, 可采用若干轮局部迭代进一步修正新点及其邻居的邻接关系, 提升图结构质量. 例如, Malkov 等人[12]提出的 HNSW 方法支持基于上述方法动态添加节点. 此外, 另一类常用的支持数据点的插入操作的方法[14–16]是动态建立一个辅助索引, 专门存储新增的数据点, 并阶段性地将辅助索引与全局的向量索引进行合并. Xu 等人[17]提出了基于倒排索引的索引原地更新策略, 用来避免维护辅助索引及定期更新全局索引的计算开销以及不均衡的计算负载. 然而这些方法只适用于插入数据的情况, 当全量数据的嵌入向量

因为嵌入模型的微调发生改变时, 仍然需要重新构建整个向量索引.

当某个节点的特征向量发生变化 (如嵌入更新), 其原本的近邻可能不再符合, 此时的解决方法为: 首先进行局部重建, 对该点重新进行 K 近邻搜索并更新其邻接列表, 然后对受影响的邻居进行修复; 接下来进行传播, 对更新点的邻居节点进行级联更新, 并设置传播深度上限以限制代价.

当节点删除时, 从其他节点的邻接表中移除该节点. 对于缺失的邻居, 可以通过局部重采样进行补充.

现有方法对于节点特征更新导致 K 近邻图更新的探索局限于单个或少部分节点的更新, 而本文侧重于解决嵌入模型经过微调等数据大规模变化时的情况, 在这种情况下, 数据点的改变不再只局限于一部分点, 而是所有点的嵌入都会发生变化. 我们将在第 2 节中详细介绍本文研究问题的应用场景.

2 嵌入模型微调的场景介绍

在处理非结构化数据 (如文本、图像或音频) 时, 常常难以直接利用其原始形式的数据建立明确的结构关系. 为了弥合这一差距, 近年来, 研究者广泛采用预训练模型将这类数据映射为稠密向量, 即嵌入表示 (embedding), 以捕捉其语义特征或内容相似性[1]. 通过将每个样本 (如文档、句子或实体) 编码为向量, 我们便可以进一步在向量空间中建模它们之间的邻近关系, 构建出反映语义结构的图.

在图结构构建任务中, 基于嵌入向量的 K 近邻图是一种常见且高效的表示方式. 该方法以每个节点的嵌入为基础, 通过计算其与其他节点之间的向量距离, 与最近的 K 个邻居节点构建连边, 从而形成表示语义相似关系的图结构. 这类图广泛应用于文本聚类、文档推荐、图神经网络建模等任务, 尤其适用于原始数据缺乏显式结构的场景.

然而, 当嵌入模型经过微调 (fine-tuning) 后, 原始的节点表示将发生显著变化, 从而影响节点之间的相对距离关系. 此时原有的 K 近邻图结构将不再准确, 必须进行更新以反映新的语义结构.

嵌入模型的微调通常出现在以下几类应用场景中.

(1) 文本语义理解任务中的领域自适应. 预训练的语言模型在通用语料上学习的嵌入可能无法准确捕捉特定领域中的语义细节. 例如, 在法律、医学或金融等专业领域, 通过引入领域标注数据对嵌入模型进行微调, 有助于提升语义相似度的判别能力. 在此过程中, 原有的 K 近邻图已无法准确反映领域语义, 必须基于新嵌入重新更新图结构.

(2) 图神经网络的辅助图构建. 在基于图神经网络 (GNN) 的任务中, 如节点分类、链接预测或社区检测, K 近邻图常作为结构先验引入模型中. 若嵌入通过对比学习或自监督方法持续优化, K 近邻图也需动态更新, 以保持结构与表示之间的一致性, 从而提升 GNN 模型性能.

(3) 交互式或迭代式标注流程. 在某些半监督学习或主动学习场景中, 嵌入模型在用户反馈的引导下逐步微调, 此时数据的向量表示将不断变化. 而由于 K 近邻图在数据可视化、聚类结果展示或样本筛选中扮演关键角色, 因此 K 近邻图需要随着嵌入更新而动态调整, 以确保交互效果的准确性和连贯性.

(4) 大规模文档检索与推荐系统中的嵌入演化. 在搜索与推荐系统中, 文档或用户的嵌入表示可能根据实时行为数据、偏好反馈进行周期性更新. 在这种检索系统中, K 近邻图可作为高效的近邻查找索引使用, 而嵌入的微调将直接影响召回质量, 因此必须高效更新 K 近邻图以保证系统性能.

综上所述, 嵌入模型微调不仅改变了向量表示, 同时也从根本上影响了基于相似度构建的图结构. 因此, 针对嵌入模型微调的场景设计一种高效的 K 近邻图更新机制, 以避免每次全量重建, 是直接影响诸多下游任务的关键问题.

3 嵌入模型微调下的高效 K 近邻图更新算法

3.1 框架概述

在嵌入模型驱动的搜索和推荐系统中, K 近邻图是一种常用的索引结构, 用于高效检索与查询点相似的数据.

然而, 当嵌入模型经过微调后, 数据点的嵌入向量会发生变化, 导致 K 近邻图中原有的邻居关系不再准确. 传统的解决方法通常依赖于全图重建来适应这些变化, 但这一过程计算量大、效率低, 难以满足大规模应用或更加实时的需求.

针对这一问题, 本文提出了一种高效的 K 近邻图更新方法 FastAdjust, 其系统框架如图 3 所示. FastAdjust 的核心思想是在嵌入微调之前, 对数据进行一系列预处理以得到对向量间关联更细致的信息, 使得在嵌入模型被微调后, 可以利用这些预处理得到的信息快速调整 K 近邻图, 从而有效适应嵌入向量的变化.

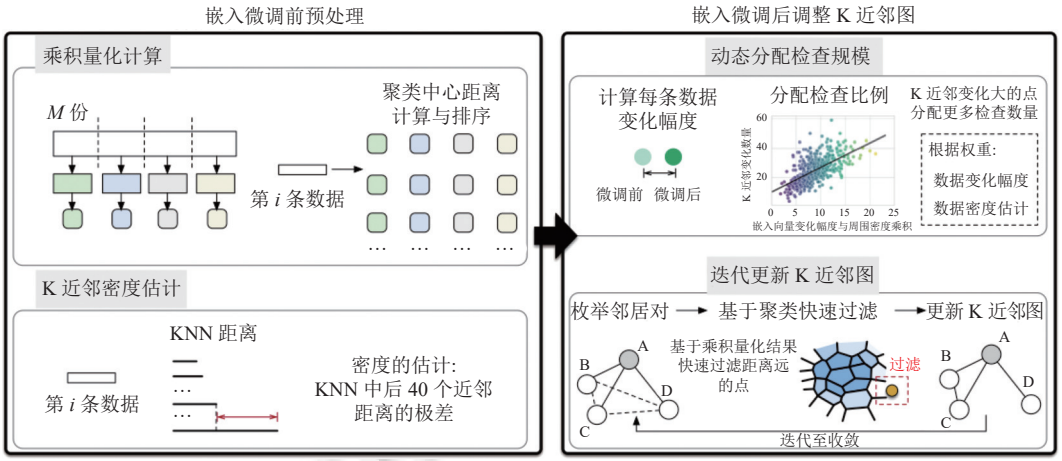

图 3 FastAdjust 系统框架图

在 FastAdjust 的预处理阶段, 其首先对数据进行基于乘积量化编码的处理, 借助这一信息, 在 K 近邻图更新阶段, 能够快速为每个数据点定位距离较近的数据子集. 此外, FastAdjust 还会利用微调前 K 近邻的分布情况来估算每个数据点周围数据的密度, 以辅助在 K 近邻图更新阶段更准确、快速地估计每个数据点 K 近邻关系的变化程度.

在嵌入模型微调后, FastAdjust 进一步利用“邻居的邻居也很可能是邻居”这一观察, 不断对 K 近邻图中的节点采样邻居对, 通过代价较低的过滤或是实际距离计算, 判断邻居对能否成为彼此的邻居 (在下文中, 将这一过程称作检查). 通过迭代的方式, FastAdjust 能够在嵌入微调后的向量空间上逐步更新微调前构建的 K 近邻图. 与传统的 NN-descent 方法不同, FastAdjust 充分考虑了嵌入微调带来的细微变化特性, 从而实现更加高效的更新. 具体来说, 下文将详细介绍 FastAdjust 使用的两个关键子策略: 候选数据定位与动态更新资源分配. 这两个子策略分别通过避免不必要的距离计算和识别不同数据点 K 近邻变化幅度, 并以此分配更新资源, 实现优化 K 近邻图调整效率的目标.

(1) 基于乘积量化的候选数据定位: 微调前后数据的嵌入向量变化有限, 因此微调前距离很远的嵌入向量在微调后仍然难以形成近邻关系. 基于这一特点, FastAdjust 提出了一种基于乘积量化的候选数据定位方法, 利用预处理阶段计算的信息, 快速定位距离较近的数据点, 并以低代价高效过滤较远的点.

(2) 基于数据密度的动态更新资源分配: 此外, FastAdjust 还依据预处理阶段计算的密度信息, 结合数据微调后的变化程度, 为每个数据点分配不同的更新资源 (检查次数), 从而实现更高效、更精准的 K 近邻图更新.

通过这种高效的 K 近邻图调整策略, FastAdjust 能够避免全图重建的巨大开销, 在嵌入微调后迅速调整图结构, 从而有效减少嵌入模型微调对 K 近邻图结构造成的影响.

3.2 基于乘积量化的候选数据定位

在模型微调的场景中, 一个显著的特点是微调对每个数据点的影响一般是有限的, 也就是数据嵌入的变化幅度较小. 因此, 微调后数据点的嵌入向量变化及其邻居变化通常是局部性的: 微调后某个数据点的新邻居往往是其微调前距离较近的点. 基于这一观察, 为了避免全局搜索带来的高计算开销, 我们可以在调整 K 近邻图时, 重点关

注微调前与目标数据点相近的数据, 从而避免遍历所有数据点.

为此, 本文提出了一种基于乘积量化的候选数据点定位方法, 通过快速缩小搜索范围来提升效率. 直观来说, 确定哪些数据点可能成为某一数据点的“新邻居”, 可以通过聚类进行优化. 比如首先将数据点进行聚类, 然后检查每个数据点所属簇内的数据点. 由于微调后的嵌入变化通常较小, 因此数据点的新邻居很可能集中在其所在簇内的邻近区域.

然而, 直接对数据进行聚类会引入误差, 主要原因是大多数数据点并不位于聚类中心, 这可能导致一些不在同一簇中但与其距离较小的点被忽略, 进而影响最终的 K 近邻图质量. 为了解决这一问题, 另一种思路是为每个数据点维护一个候选邻居集合, 包含距离其较近的数据点. 这个思路可以看作是为每个数据点维护一个更大的近邻列表. 但当近邻列表的大小增加时, 计算和存储开销也会显著增大.

为了有效解决这一问题, 本文提出了一种基于乘积量化[19,20]的方法, 用于限定每个数据点的候选邻居集合, 从而避免无谓的全局搜索. 如图 3 所示, 乘积量化通过将每个数据点的向量分割为多个子向量, 并计算这些子向量与预先计算的聚类中心的距离来实现数据的聚类和划分. 具体来说, 假设 D 维向量被分为 M 个子向量, 每个子向量的维度为 $\frac{D}{M}$. 每个划分后的子向量会被聚类到 c 个聚类中心中, 并通过与其距离最近的聚类中心的下标进行编码. 由此, 每个数据点都会被转换为一个长度为 M 的 PQ 编码, 表示每个子向量对应的聚类中心编号. 以图 3 中的例子来说, 原始的向量被划分为 M=4 个子向量, 在图中表示为不同的颜色, 每个子向量独立地由其最近的聚类中心表示, 因此原始的向量被子向量分别对应的聚类中心编码为 4 个整数下标.

此外, PQ 中各个子向量的聚类中心之间的距离会在预处理阶段被计算并存储. 由此, 对于两条使用 PQ 编码过的数据, 其近似距离可以通过查聚类中心之间的距离表, 以 $O(M)$ 的复杂度计算. 与精确计算的 $O(D)$ 复杂度相比, 这种方法能够加速 $\frac{D}{M}$ 倍.

● 快速判断是否需要计算实际距离. 为了提高计算效率, 在为目标点判断采样数据时基于建立的乘积量化结果进行过滤, 从而用较低的代价避免与距离较远的数据点计算距离. 作为辅助, 在预处理阶段, 本文为每条数据计算并存储其每个子向量与其前 λ 近的 PQ 聚类中心编号, 其中, λ 是一个超参数 (例如, 取 λ=0.5c 表示取前 50% 近的聚类中心, λ 的大小与微调程度相关, 微调程度越大, λ 也取较大). 然后, 在 K 近邻图的调整阶段, 本文通过算法 1 判断是否需要进一步计算采样点与目标点的实际距离.

算法 1. 判断采样数据是否需要计算与目标点的实际距离.

输入: 目标点及其 PQ 编码, 采样数据及其 PQ 编码, 检查阈值 θ;
输出: 是否需要计算实际距离.

1. 检查采样数据的 PQ 编码是否每一个都是离目标点 PQ 编码前 λ 近的
2. **If** 均为前 λ 近 **then**
3. **Return** True
4. **else**
5. $dist \leftarrow$ 使用目标点与采样点的 PQ 编码计算近似距离
6. $dist' \leftarrow$ 目标点到当前第 K 近邻的距离
7. **If** $dist - dist' > \theta$ **then**
8. **Return** False
9. **else**
10. **Return** 随机数 $(0,1) > \frac{dist - dist'}{\theta}$
11. **end**
12. **end**

相较于每次计算实际距离, 该算法通过代价低的判断, 快速过滤掉和目标点距离较远的点. 其过滤流程又分为两个层级, 首先, 对于每一个 PQ 编码都是离目标点编码前 λ 近的点, 需要计算实际距离, 因为其与目标点的距离大概率比较小. 对于不符合此条件的点, 算法通过乘积量化的近似距离计算, 快速估计其与目标点的距离. 如果近似距离较大 (与目标点的第 K 近距离的差距大于 θ), 则将其过滤掉, 不计算其与目标点的实际距离; 否则按照估计的近似距离, 以一定的概率进行实际距离计算. 这一概率随估计距离的增大而减小, 保证了优先检查离目标点更近的数据.

● 示例说明. 如图 4 所示, 采用本文提出的方法后, 具有不同 PQ 编码的数据会按照不同的比例被检查 (计算实际距离). 例如, 图中深蓝色区域离目标点较近, 由于其 PQ 编码对应的每一个聚类中心都离目标点前 λ 近, 因此所有该区域的采样点都会与目标点计算实际距离. 对于浅蓝色区域, 其中的数据按照与目标点的距离, 以不同的概率被检查, 而距离目标点较远的白色区域则被直接排除. 图 4 的例子表明本文提出的方法能够高效又准确地识别出与目标点距离较近的区域, 从而准确地缩小候选范围, 有效避免计算较远数据点的实际距离, 显著提高 K 近邻图的更新效率.

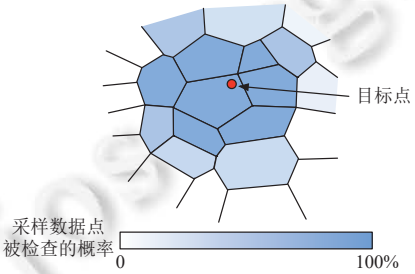

图 4 嵌入空间不同部分采样点的检查概率

3.3 基于数据密度的动态更新资源分配

经过微调后, 不同数据点的邻居关系变化程度有所不同. 因此, 为了提高 K 近邻图的更新效率, 不应该均匀分配对每个点检查候选点的次数, 而应根据邻居关系变化的程度进行有针对性的分配. 这种方法能够在较短时间内显著提升 K 近邻图的质量. 然而, 邻居关系变化程度是一个依赖于微调后的 K 近邻图、只有在图更新后才能准确计算的变量. 为了解决这一问题, 我们需要探索与邻居关系变化幅度相关性较高的其他变量, 通过这些变量来替代邻居关系的直接计算, 从而指导节点间检查次数的分配.

直观来看, 嵌入向量本身在微调前后的变化幅度和数据点周围的数据密度是两个影响邻居关系变化幅度的关键因素. 本身嵌入向量变化较大的点, 其邻居关系通常会发生较大的变化. 而在数据密集的区域, 数据点间的邻居关系对嵌入向量的微小变化更加敏感. 因此, 即使嵌入向量发生较小的变化, 也可能对邻居关系产生显著影响. 例如, 图 5 中微调后嵌入向量变化较大的数据点, 它们的邻居关系通常发生了显著变化. 而在低密度区域, 即使嵌入向量发生较大变化, 邻居关系却几乎未发生变化. 例如, 图中蓝色点的周围数据稀疏, 与蓝色点之间距离的差异较大, 因此即便其嵌入向量发生了较大的变化, 邻居关系变化也较小 (如其 3 个近邻在微调前后并无变化). 相反, 在高密度区域 (如图中的红色点), 由于其周围数据较为密集, 与红色点之间距离的差异较小, 即便嵌入向量变化较小, 也可能导致邻居关系的显著变化 (如红色点的 3 个近邻中有 2 个发生了变化).

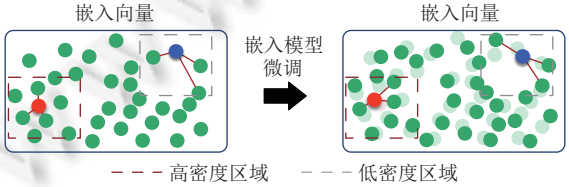

图 5 不同密度区域 K 近邻关系变化情况的示例

基于上述观察, 本文提出了一种数据密度驱动的动态更新资源分配策略. 该策略通过计算每个数据点的变化幅度和周围数据的密度, 估算其邻居关系的变化程度, 并据此为每个数据点针对性地分配检查次数. 这种策略确保了邻居关系变化较大的数据点获得更多检查次数, 从而提高这些区域的 K 近邻图的准确性. 而对于邻居变化较小的区域, 这种方法则可以减少不必要的检查次数.

为了实现这一策略, 首先需要为数据密度提供一种量化的评估方法. 对于每个数据点来说, 与其最近的邻居不再是 K 近邻的变化不容易发生, 因为要改变这种邻居关系, 需要相对较大的嵌入向量变化. 相反, 那些相对较远的邻居更容易发生不再是 K 近邻的变化. 并且, 如果这些较远的 K 近邻点与数据点的距离比较接近, 则说明它们在空间距离上比较密集, 小幅度的变化可能显著影响它们的顺序, 甚至使原本非 K 近邻的点成为 K 近邻. 基于上述分析, 我们选取 K 近邻中靠后的若干个近邻 (例如, 对于 100 近邻图选择后 40 个近邻), 并使用它们距离极差的倒数来衡量数据密度. 若这些近邻的极差较大, 说明该点周围邻居的距离差异较大, 数据密度较小, 量化指标较小 (例如图 5 中的蓝色点周围). 另一方面, 数据点微调前后的变化幅度可以直接通过微调前后嵌入向量的距离来衡量.

在图 6 中, 我们绘制了微调前后嵌入向量变化幅度与周围密度的乘积以及 K 近邻变化数量的散点图. 结果表明, 该乘积与 K 近邻变化数量之间有较强的正相关性. 这表明该值较大时, K 近邻变化的幅度更有可能较大.

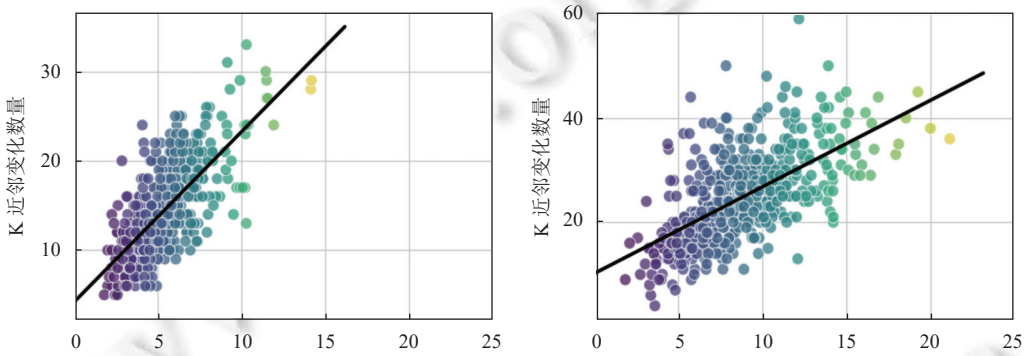

图 6 嵌入向量变化幅度与数据周围密度的乘积与 K 近邻变化数量的相关性

基于这一观察, 我们将该乘积作为权重, 为不同数据点分配不同的检查次数. 通过这种方式, 算法能够根据邻居关系的变化幅度, 动态调整各数据点的检查次数: 对于邻居关系变化较大的区域, 分配更多计算资源, 从而进行更精细的邻居搜索和更新; 在邻居关系变化较小的区域, 则采用较低频的更新策略, 从而减少不必要的计算开销. 通过这一机制, 有限的计算资源能够得到更合理的分配, 使得 K 近邻图能够在保证质量的同时实现高效更新, 从而更准确地反映微调后嵌入向量的嵌入空间结构.

4 实验与分析

4.1 实验设置

为了验证本文方法的有效性, 我们在如下 3 个广泛使用的真实数据集上进行实验.

- StackExchange (Stack) 数据集 (https://huggingface.co/datasets/teven/stackexchange)
- Wikipedia (Wiki) 数据集 (https://huggingface.co/datasets/wikimedia/wikipedia)
- DigiFace 数据集 (https://microsoft.github.io/DigiFace1M/)

其中, StackExchange 和 Wikipedia 是文本数据集. 除此以外, 为了验证本文方法在其他模态数据上的适用性, 我们还在广泛使用的图像数据集 DigiFace[21]上进行了实验, 针对图像模态的数据进行评估.

数据集的详细参数如表 1 所示.

表 1 实验数据集

属性	Stack	Wiki	DigiFace
数据类型	文本	文本	图像
数据规模	1 000 000	1 000 000	144 000
平均词数	296	715	N/A
嵌入维度	768	768	768
微调前100近邻平均距离	0.941	0.927	6.35
微调后100近邻平均距离	0.985	0.976	8.74

文本语义表示方面, 本文采用 Sentence Transformer 中的 all-mpnet-base-v2 模型提取文本的语义嵌入, 每个嵌入向量维度为 768. 我们使用 STS (semantic textual similarity) 数据集对该模型进行微调. STS 数据集由大量句子对组成, 每对句子被标注了一个从 0 到 5 的相似度得分, 反映两个句子在语义上的接近程度. 该数据集覆盖了同义句、翻译句和含义相近或相异的表达形式, 非常适合作为文本嵌入模型的微调数据. 对嵌入模型使用 STS 数据集进行微调, 能够提升模型对语义相似度的判断能力. 这对于语义搜索、同义句检测和其他文本理解任务的应用具有重要意义.

图像的嵌入向量方面, 本文采用 ViT (vision Transformer) 模型[22]计算图像的嵌入, 采用在 ImageNet-21K 上进行过预训练的 vit-base-patch16-224-in21k 作为微调前的模型. DigiFace 数据集提供了不同图片中人物身份的标记, 我们使用数据集 DigiFace 中未被用于嵌入向量计算的部分, 使用对比学习对预训练的嵌入模型进行微调.

在微调过程中, 我们采用余弦相似度损失函数, 设置学习率为 10^{-5}, 默认以 16 的批大小 (batch size) 对嵌入模型训练 1 个轮次.

对于乘积量化, 我们设定划分数 $M=4$, 即将每个 768 维嵌入向量分为 4 个子向量, 每个子向量聚类数 $c=256$, 并在预处理阶段对每个子向量选取前 40%, 即 $\lambda=0.4c=102$ 个距离最接近的聚类中心. 检查阈值默认 θ 取 0.2.

为了全面评估本文方法的效果, 我们将本文方法与以下 3 种基线进行对比.

- Stale: 直接使用微调前的嵌入向量构建的 K 近邻图, 未做任何调整.
- NN-descent: 在微调后的嵌入向量上, 从头执行 NN-descent 算法构建 K 近邻图.
- NN-descent-init: 以微调前的嵌入向量上建立的 K 近邻图作为初始图, 在微调后的嵌入向量上继续执行 NN-descent 算法对 K 近邻图进行优化.

评估指标方面, 我们使用召回率 (*Recall*@100, 即前 100 个真实邻居中被正确检索出的比例) 衡量不同方法得出的 K 近邻图的准确性, 数值越高表示 K 近邻图越准确. 我们使用欧氏距离衡量向量间的距离 (由于本文使用的嵌入模型中, 嵌入向量会被标准化为单位长度, 使用欧氏距离与 Cosine 距离等价). 为了评估建立出的 K 近邻图结构上与真实的 K 近邻图的差异, 我们还计算了各个节点度数的均方根误差 (root mean square error, RMSE) 进行评估. 此外, 为评估不同方法的执行效率, 我们记录每种方法在微调后调整 K 近邻图所耗费的时间, 时间消耗越短表示方法越高效.

所有实验均在一台 Ubuntu 服务器上完成, 服务器的配置参数为 256 GB 内存、4 张 NVIDIA RTX 3090 显卡, 以及 Intel(R) Xeon(R) Gold 6242R CPU @ 3.10 GHz 处理器.

4.2 K 近邻图准确度比较

本文对比分析了在嵌入向量微调后, K 近邻图执行更新 3 min 时, 不同方法所生成的 K 近邻图的召回率, 结果如图 7 所示. 实验结果表明, FastAdjust 在各个数据集上均显著优于未对 K 近邻图进行调整的 Stale 以及采用 NN-descent 进行增量更新的方法. 以 Stack 数据集为例, FastAdjust 的召回率达到了 96.4%, 而 Stale、NN-descent 和 NN-descent-init 的召回率分别仅为 84%、85% 和 89%.

Stale 方法的召回率较低, 主要原因在于其仍使用微调前的嵌入向量构建的 K 近邻图. 由于微调后向量的空间分布已发生改变, 原有的邻居关系失去了准确性, 导致检索性能下降. NN-descent 方法虽然重新构建了 K 近邻图,

但其忽略了微调前后向量变化幅度有限的事实, 从头计算所有邻居关系, 需要花费大量时间, 难以在有限时间内完成有效更新, 因而在短时间内获得的 K 近邻图质量仍然较差, 召回率远低于 FastAdjust. 另一方面, 尽管 NN-descent-init 方法尝试以旧图作为初始化加速更新过程, 但其在图结构更新过程中, 仍然没有考虑微调前后向量的关联, 产生大量冗余的计算与判断, 导致整体效率较低, 在有限的适应时间内准确度较低.

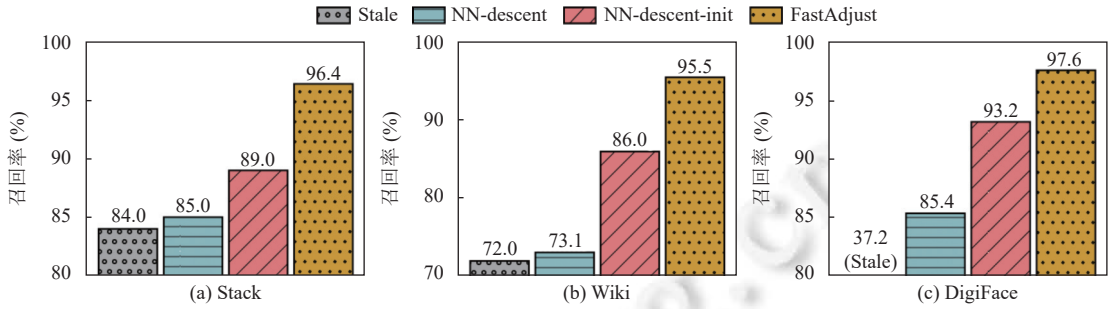

图 7 不同方法更新 K 近邻图, 3 min 后召回率的比较

相较而言, FastAdjust 充分利用了微调前的 K 近邻图结构以及微调前后向量变化幅度有限这一关键特性. 它通过聚类引导的候选邻居定位策略和基于局部密度的动态更新资源分配机制, 显著提高了 K 近邻图更新的效率和质量. 在相同的时间限制下, FastAdjust 能够更快地为每个样本重新识别更准确的近邻, 从而在较短的更新时间内显著提升召回率.

4.3 时间效率比较

本文对比分析了不同方法在 K 近邻图调整过程中, 召回率随更新时间变化的表现, 结果如图 8 所示. 从图中可以看出, FastAdjust 能够在极短的时间内显著修复因嵌入向量微调而准确度下降的 K 近邻图, 快速提高其召回率. 例如, 在 Stack 数据集上, FastAdjust 在 2 min 内即达到了 95% 的召回率, 而排名第 2 的 NN-descent-init 达到 95% 的召回率则耗时 13.4 min, 前者在效率上实现了 6.7 倍的提升. 整体来看, 与其他方法相比, FastAdjust 在达到高召回率所需时间上具有显著优势, 体现了其在效率上的卓越性能.

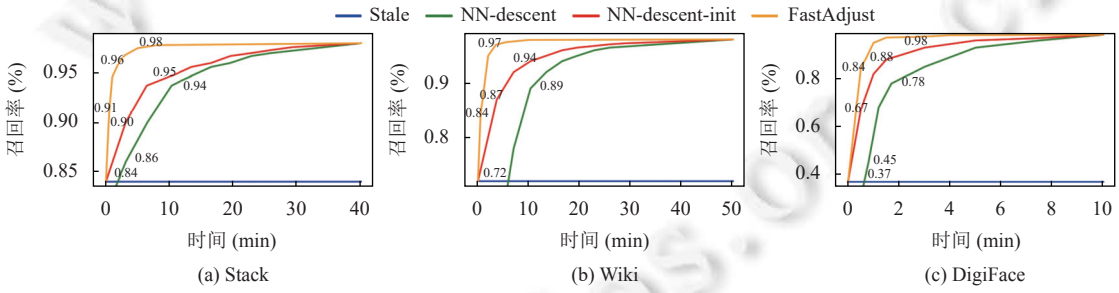

图 8 不同方法 K 近邻图召回率随更新时间的变化

这种效率提升归功于 FastAdjust 所采用的一系列高效更新策略: 一方面, 其基于乘积量化的数据定位方法能够准确识别需要更新的区域, 显著减少了冗余计算; 另一方面, 其根据数据密度动态分配检查次数的机制, 有效提升了资源利用效率, 进而提升更新效率. 这些策略共同保障了 FastAdjust 在效率和精度之间实现了良好的平衡.

4.4 图结构准确度的评估

K 近邻图本身的结构信息反映了不同数据点之间的相关关系, 例如不同数据点的入度即衡量了其在全局中与其他数据的密切程度. 一个质量更高的 K 近邻图不仅应该在近邻关系的召回率上取得较高的结果, 也应该尽可能与真实的 K 近邻图保留相似的图结构. 为了评估建立出的 K 近邻图结构上与真实的 K 近邻图的差异, 我们还计算了各个节点入度与真实 K 近邻图的入度的均方根误差 (RMSE), 从图的结构层面进行评估.

实验结果如图 9 所示. 从实验结果可以看出, FastAdjust 的 RMSE 显著低于其他基线方法. 由此, 本文从召回率以外的另一个角度, 证明了 FastAdjust 能够快速调整 K 近邻图的结构, 在很短的调整时间内取得与真实 K 近邻图相似的图结构, 使其适应嵌入模型微调对 K 近邻图产生的结构性变化.

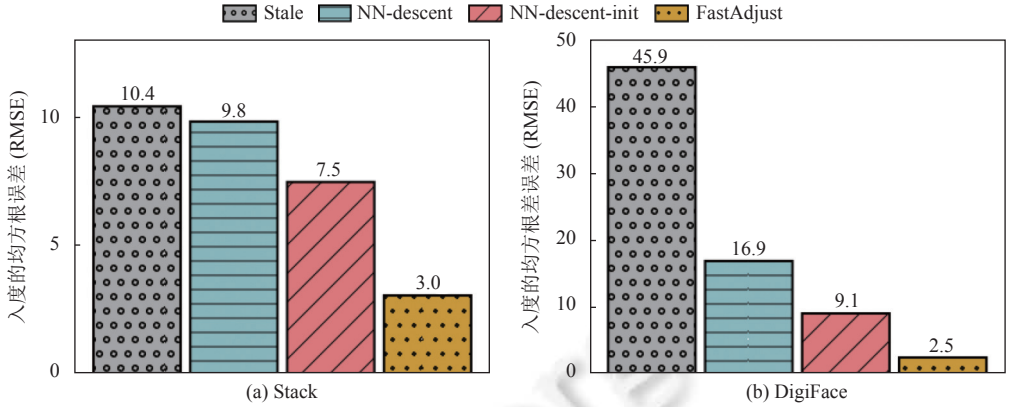

图 9 不同方法更新 K 近邻图, 3 min 后节点入度均方根误差的比较

4.5 不同参数与 K 近邻变化个数的相关性

为了验证第 3.3 节中所提出的基于数据密度的动态更新资源分配的合理性, 本节评估了不同变量与数据 K 近邻微调前后变化个数的相关系数, 结果如表 2 所示.

表 2 不同变量与 K 近邻变化个数的相关系数

变量	Stack	Wiki
位移距离	0.411	0.391
1/(前100近邻的平均距离)	−0.365	−0.367
1/(前100近邻距离的极差)	0.223	0.203
1/(前100近邻中后40近邻距离的极差)	0.564	0.575
(位移距离)/(前100近邻的平均距离)	−0.071	−0.182
(位移距离)/(前100近邻距离的极差)	0.332	0.278
(位移距离)/(前100近邻中后40近邻距离的极差)	**0.649**	**0.636**

从表 2 中相关系数的结果可以看出, 不同变量对 K 近邻变化个数的影响存在一定的规律, 这些关系为我们提供了对 K 近邻结构变化的深入理解. 微调前后数据位移距离 (即数据变化幅度) 与 K 近邻变化个数之间的相关系数为正 (例如, Stack 和 Wiki 上的相关系数分别为 0.411 和 0.391), 说明当位移距离增大时, K 近邻的变化个数也倾向于增加. 这表明, 数据微调后较大的位移可能导致 K 近邻结构发生较大的调整. 前 100 近邻中后 40 近邻距离的极差同样与 K 近邻的变化个数显示出较强的正相关 (如 Stack 和 Wiki 分别为 0.564 和 0.575), 说明前 100 近邻中后 40 近邻的距离差异对 K 近邻变化个数有较大影响. 位移距离/前 100 近邻中后 40 近邻距离的极差是与 K 近邻变化个数相关系数最高的变量 (Stack 和 Wiki 分别为 0.649 和 0.636). 这表明这一变量与 K 近邻的变化有较强的正相关性. 当这个比值较大时, K 近邻的变化个数一般也显著增加, 因此, 通过基于该变量的取值为每条数据动态分配更新资源, 也就是分配不同的检查次数, 可以提高对计算资源的利用效率, 为 K 近邻变化较大的数据分配更多资源来调整其邻居关系, 从而加速整个 K 近邻图的调整.

通过分析不同变量与 K 近邻变化个数的相关系数, 我们发现, 数据点的位移距离和近邻距离的差异程度对微调前后 K 近邻变化的程度有重要影响. 这些结果支持了第 3.3 节提出的基于数据密度的动态分配更新资源的合理性. 未来的研究可以进一步探讨这些变量在实际应用中的具体影响, 以优化 K 近邻图更新算法在不同数据分布下的表现.

4.6 消融实验

为了验证 FastAdjust 的各个部分的有效性, 本节通过比较不同的优化方法组合来评估各个优化的有效性. 为此, 我们将 FastAdjust 与以下方法进行比较.

● FA-noLoc. 基于 FastAdjust 方法, 但不使用第 3.2 节提出的基于乘积量化的候选数据定位, 也就是对于每一对枚举到的点均计算实际距离.

● FA-noAlloc. 基于 FastAdjust 方法但不使用第 3.3 节提出的为不同节点分配不同检查次数的优化.

● NN-descent-init. 在 FastAdjust 不使用上述两种优化的时候会退化为 NN-descent-init, 也就是以微调前的嵌入向量上建立的 K 近邻图作为初始图, 在微调后的嵌入向量上继续执行 NN-descent 算法对 K 近邻图进行更新.

图 10 展示了通过使用不同的优化方法, 在 Stack 和 Wiki 数据集上更新 K 近邻图, 3 min 时的召回率. 结果显示, FA-noLoc 和 FA-noAlloc 相较于 FastAdjust 均有一定程度的性能下降, 但仍然优于 NN-descent-init. 其中 FA-noLoc 相较于 FA-noAlloc 的召回率下降幅度相对更大, 这是因为在更新 K 近邻图时, FA-noAlloc 能够在早期阶段就较为准确地判别出不同数据微调前后 K 近邻关系变化幅度, 从而能够在早期的 K 近邻图更新中更多地利用 K 近邻关系变化较小的节点, 基于这些节点较为准确的近邻信息优化 K 近邻变化较大的节点, 从而能够快速提升 K 近邻图的质量.

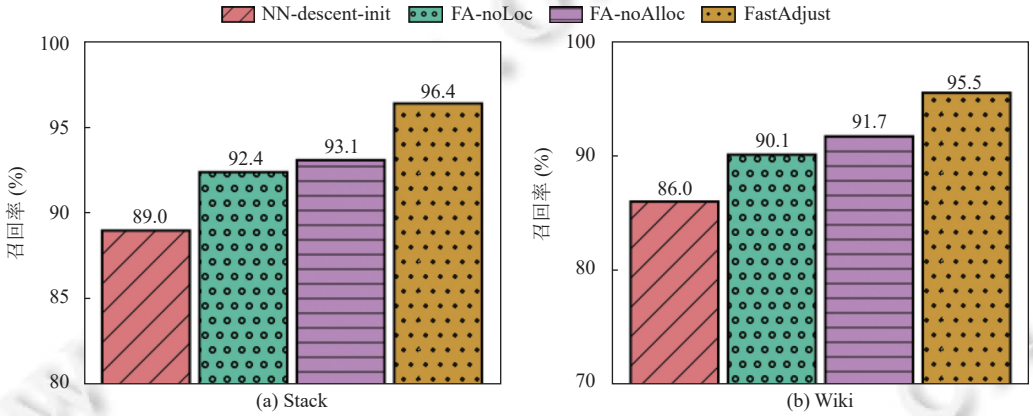

图 10 去除不同优化后的 FastAdjust 方法更新 K 近邻图, 3 min 时的召回率

4.7 阈值的误差分析

检查阈值 θ 是第 3.2 节中基于乘积量化的候选数据定位方法中的一个超参数. 它被用于根据乘积量化计算出的近似距离, 判断是否需要对实际距离进行精确计算. 较小的 θ 会减少实际距离计算的次数, 从而提高效率, 但也可能由于近似误差遗漏一些近邻点; 与之相反, 较大的 θ 会保留较多的候选点进行精确计算, 虽然计算开销更大, 但能降低遗漏近邻的风险.

本文在 DigiFace 数据集上比较了在采用不同阈值 θ 时, 减少的距离计算的比例以及判断错误发生的比例, 结果如表 3 所示.

表 3 在 DigiFace 数据集上, 不同阈值 θ 下减少实际距离计算的比例及判断错误的概率

阈值 θ	减少实际距离计算比例 (%)	错误率 (%)
0.1	76	0.10
0.2	76	0.10
0.5	76	0.10
1.0	76	0.10
2.0	74	0.09
5.0	57	0.05

从表 3 的结果可以看出, 阈值 θ 有较为灵活的取值空间, 在一个较大的取值空间中, 均可以在较低的错误率下, 较大幅度地减少实际距离计算的数量. 在实际使用时, 由于这一参数的取值与嵌入向量数据本身的分布有关, 可以先在一部分采样数据上对该超参数进行选择, 然后再为余下的数据使用所选的取值.

4.8 不同微调幅度的影响

图 11 展示了在 Stack 数据集上, 不同微调幅度下, 各方法在对 K 近邻图调整 3 min 后的召回率表现. 微调幅度通过对嵌入模型设置不同的微调轮次 (*epoch*) 实现.

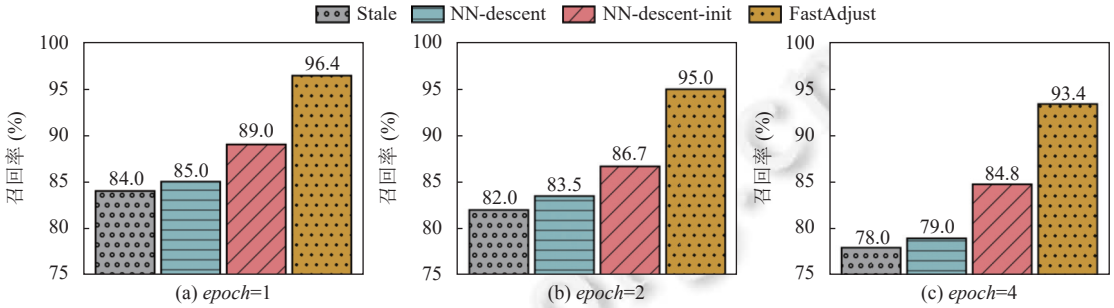

图 11 不同方法 K 近邻图召回率随微调幅度的变化

实验结果表明, 在微调幅度较小时, 本文提出的方法能够迅速适应嵌入变化对 K 近邻图结构的影响, 在极短时间内将近邻关系恢复至与重新构建 K 近邻图相似的准确度, 验证了本文方法的高效性与有效性.

随着微调轮次的增加, 嵌入向量的变化幅度也随之增大, 导致每个数据点的近邻关系发生更显著的变化. 如图 11 中 Stale 方法的结果所示, 基于微调前的数据构建的 K 近邻图在此情形下的召回率明显下降. 对于 NN-descent 方法而言, 面对更剧烈的 K 近邻关系变化, 所需的迭代轮数与距离计算次数也显著增加, K 近邻图的调整过程因此变得更为耗时.

相比之下, FastAdjust 在面对较大的微调幅度时, 依然能够在较短时间内完成 K 近邻图的有效更新, 以适应嵌入向量的变化. 这是因为即使在向量变化幅度更大的情况下, 该方法仍能较为精准地定位可能产生新邻居的数据区域, 并能较为准确地根据不同数据的预期邻居变化幅度, 针对性地分配适当的更新资源, 从而显著提升更新效率.

5 总结与未来工作

基于本研究所提出的高效 K 近邻图更新方法 FastAdjust, 本文总结了在嵌入模型微调后的 K 近邻图调整过程中取得的成果, 并展示了其显著的效率优势和准确性. 通过局部增量调整策略以及针对性地分配更新资源, FastAdjust 能够有效地适应嵌入向量的变化, 避免了全图重建所带来的高昂计算开销, 提升了更新速度及查询性能.

然而 FastAdjust 仍存在一定的局限性, 其目前仅支持嵌入模型微调前后对 K 近邻图这一向量索引的更新, 并且其无法直接适应嵌入模型的大幅度更新或是更换嵌入模型的场景. 这些局限性使得目前 FastAdjust 支持的应用场景依然有限. 未来的工作将进一步优化该方法的适用范围, 例如为其他向量索引提供支持, 从而支持更一般的场景; 并探索在微调幅度较大的情况下的处理策略, 例如如何应对当微调幅度巨大, 或是直接更换为其他嵌入模型时, 数据的邻居关系发生了完全变化, 变得难以发现局部的邻居更新关系的情况. 此外, 结合更加自适应的资源分配策略, 例如为本文提出的各种优化方法自动化地确定超参数, 也将是未来要探索的方向.

References

[1] Reimers N, Gurevych I. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: ACL, 2019. 3982–3992. [doi: 10.18653/V1/D19-1410]

[2] van der Maaten L, Geoffrey H. Visualizing data using t-SNE. Journal of Machine Learning Research, 2008, 9(86): 2579–2605.

[3] Zhang JH, Zhu YQ, Liu Q, Wu S, Wang SH, Wang L. Mining latent structures for multimedia recommendation. In: Proc. of the 29th ACM Int'l Conf. on Multimedia. ACM, 2021. 3872–3880. [doi: 10.1145/3474085.3475259]

[4] Franceschi L, Niepert M, Pontil M, He X. Learning discrete structures for graph neural networks. In: Proc. of the 36th Int'l Conf. on Machine Learning. 2019. 1972–1982.

[5] Deng CH, Li XY, Feng Z, Zhang ZR. GARNET: Reduced-rank topology learning for robust and scalable graph neural networks. In: Proc. of the 1st Learning on Graphs Conf. 2022. 198: 3:1–3:23.

[6] Pan JJ, Wang JG, Li GL. Survey of vector database management systems. The VLDB Journal, 2024, 33(5): 1591–1615. [doi: 10.1007/S00778-024-00864-x]

[7] Eppstein D, Paterson MS, Yao FF. On nearest-neighbor graphs. Discrete & Computational Geometry, 1997, 17(3): 263–282. [doi: 10.1007/PL00009293]

[8] Pan JJ, Wang JG, Li GL. Vector database management techniques and systems. In: Proc. of the Companion of the 2024 Int'l Conf. on Management of Data. Santiago: ACM, 2024. 597–604. [doi: 10.1145/3626246.3654691]

[9] Silpa-Anan C, Hartley R. Optimised KD-trees for fast image descriptor matching. In: Proc. of the 2018 IEEE Conf. on Computer Vision and Pattern Recognition. Anchorage: IEEE, 2008. 1–8. [doi: 10.1109/CVPR.2008.4587638]

[10] Wang J, Wang JD, Zeng G, Tu ZW, Gan R, Li SP. Scalable k-NN graph construction for visual descriptors. In: Proc. of the 2012 IEEE Conf. on Computer Vision and Pattern Recognition. Providence: IEEE, 2012. 1106–1113. [doi: 10.1109/CVPR.2012.6247790]

[11] Dong W, Moses C, Li K. Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proc. of the 20th Int'l Conf. on World Wide Web. Hyderabad: ACM, 2011. 577–586. [doi: 10.1145/1963405.1963487]

[12] Malkov YA, Yashunin DA. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2020, 42(4): 824–836. [doi: 10.1109/TPAMI.2018.2889473]

[13] Sun J, Li GL, Pan J, Wang J, Xie YQ, Liu RC, Nie W. GaussDB-vector: A large-scale persistent real-time vector database for LLM applications. Proc. of the VLDB Endowment, 2025, 18(12): 4951–4963. [doi: 10.14778/3750601.3750619]

[14] Krishnaswamy R, Manohar MD, Simhadri HV. The DiskANN library: Graph-based indices for fast, fresh and filtered vector search. IEEE Data Engineering Bulletin, 2024, 48(3): 20–42.

[15] Sundaram N, Turmukhametova A, Satish N, Mostak T, Indyk P, Madden S, Dubey P. Streaming similarity search over one billion Tweets using parallel locality-sensitive hashing. Proc. of the VLDB Endowment, 2013, 6(14): 1930–1941. [doi: 10.14778/2556549.2556574]

[16] Wang JG, Yi XM, Guo RT, Jin H, Xu P, Li SJ, Wang XY, Guo XZ, Li CM, Xu XH, Yu K, Yuan YX, Zou YH, Long JQ, Cai YD, Li ZX, Zhang ZF, Mo YH, Gu J, Jiang RY, Wei Y, Xie C. Milvus: A purpose-built vector data management system. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 2614–2627. [doi: 10.1145/3448016.3457550]

[17] Xu YM, Liang HY, Li J, Xu ST, Chen Q, Zhang QX, Li C, Yang ZY, Yang F, Yang YQ, Cheng P, Yang M. SPFresh: Incremental in-place update for billion-scale vector search. In: Proc. of the 29th Symp. on Operating Systems Principles. Koblenz: ACM, 2023. 545–561. [doi: 10.1145/3600006.3613166]

[18] Gollapudi S, Karia N, Sivashankar V, Krishnaswamy R, Begwani N, Raz S, Lin YY, Zhang Y, Mahapatro N, Srinivasan P, Singh A, Simhadri HV. Filtered-DiskANN: Graph algorithms for approximate nearest neighbor search with filters. In: Proc. of the ACM Web Conf. 2023. Austin: ACM, 2023. 3406–3416. [doi: 10.1145/3543507.3583552]

[19] Jégou H, Douze M, Schmid C. Product quantization for nearest neighbor search. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2011, 33(1): 117–128. [doi: 10.1109/TPAMI.2010.57]

[20] Ge TZ, He KM, Ke QF, Sun J. Optimized product quantization for approximate nearest neighbor search. In: Proc. of the 2023 IEEE Conf. on Computer Vision and Pattern Recognition. Portland: IEEE, 2013. 2946–2953. [doi: 10.1109/CVPR.2013.379]

[21] Bae G, de La Gorce M, Baltrušaitis T, Hewitt C, Chen D, Valentin J, Cipolla R, Shen JJ. Digiface-1M: 1 million digital face images for face recognition. In: Proc. of the 2023 IEEE/CVF Winter Conf. on Applications of Computer Vision. Waikoloa: IEEE, 2023. 3515–3524. [doi: 10.1109/WACV56688.2023.00352]

[22] Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai XH, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N. An image is worth 16×16 words: Transformers for image recognition at scale. In: Proc. of the 9th Int'l Conf. on Learning Representations. 2021.

作者简介

王嘉翼, 博士生, CCF 学生会员, 主要研究领域为人工智能和数据管理的交叉技术.

徐士惠, 博士生, 主要研究领域为基于人工智能的数据库优化技术.

李国良, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为大数据, 数据库, 数据科学.