学习度量索引复现报告

摘要

本课题对学习度量索引进行了改进和结果复现，学习度量索引是一种新的索引和搜索方法，这种方法使用机器学习模型来识别数据分布的特征，以便快速回答最近邻查询的问题，而不需要在数据之间进行精确的成对距离计算。在原方法的基础上，本课题对其中所采用的机器学习模型做出了一定的改进，通过原数据集和新数据集的实验证明了本课题提出的方法有一定的改进效果。

关键词：学习度量索引；相似性查询；机器学习；神经网络

1 引言

相似性查询是一种基于相似度度量的数据库查询方法，用于寻找与给定查询对象相似的数据项。在相似性查询中，通常会使用一些相似度度量方法（如余弦相似度、欧氏距离等）来衡量数据项之间的相似程度，从而找到与查询对象最相似的数据项。这种查询方法在信息检索、图像识别、推荐系统等领域都有广泛应用。

本课题复现的是提交给 2023SISAP 索引挑战的一篇文章[1]，该文章描述了学习度量索引（Learned Metric Index, LMI）的实验设置和性能，该索引使用相互关联的学习模型架构来回答相似性查询的问题。LMI 是一个由节点组成的层次树结构，该模型被训练用来对查询对象进行分类，模拟传统索引节点的行为。然而，查询并不是根据距离来确定对象的位置，而是通过执行一系列的预测来解决查询。这种与通常的索引创建和查询评估方法的不同导致了显著不同

的性能特征，通常在效率和有效性方面都优于传统的相似度搜索方法。

2 相关工作

2.1 M-tree

度量树（Metric tree, M-tree）的基本思路是逐层细分数据，并用球把数据包络起来，因此也称为包络球划分。M-Tree 是第一个面向外存并支持插入删除等动态操作的度量空间索引方法，其很多算法受到了一维索引的 B-树和多维索引的 R-树的启发[2]。其数据划分的方式为先选定若干个支撑点，然后把其它数据分配到以距其最近的支撑点为代表的子树。M-Tree 用一个球来表示一棵子树，即每棵子树都用一个中心（支撑点）和一个覆盖半径来表示，所有数据都被这个球覆盖。M-tree 扩展了传统向量空间之外的适用领域，在高维数据空间中表现良好，并且在文件增长的情况下也能很好地扩展。

2.2 M-index

度量索引（Metric Index, M-index）定义了一个从通用度量空间到数字域的通用映射模式[3]。该模式具有保持数据相似性的能力，即它将类似的度量空间对象映射到数字域中的接近数字。M-Index 的核心是一种通用的映射机制，它能够将数据实际存储在已建立的结构中，如 B+—树，甚至存储在分布式存储中。M-Index 的索引和搜索机制使用一组参考对象，并协同利用几乎所有已知的基于度量的数据分区、修剪和过滤原则。同时，M-Index 具有一组固定的参考对象，因此在插入数据对象期间的度量—函数评估的数量方面具有固定的构建成本。

2.3 LMI

学习度量索引是一种在度量数据中进行索引和搜索的新方法[4]。它基于一

个预先存在的树形索引结构，并为每个内部节点准备了一个机器学习模型。该方法通过对父节点分配的数据分段进行训练，以叶子节点构成数据对象的存储空间。相似性搜索问题被转化为一个分类问题，然后通过进行一系列的预测来解决查询。学习度量索引改变了索引构建和查询评估的标准范式，导致了非常不同的性能特征，并且在许多情况下，在效率和有效性方面都优于传统的相似性搜索方法。

3 本文方法

3.1 架构

学习度量索引的算法实现主要分为两个模块，第一个是划分模块，该阶段由聚类算法产生相似数据的类别来作为学习阶段的输入，然后使用神经网络来学习单个数据对象与其类别之间的关联，从而形成一个索引，该索引具有一个内部（根）节点和 n 个叶节点，对应于类别的数量。其中，K-Means 使用的是欧几里得距离，而数据集使用的是余弦距离。为解决这种不一致性，数据采用了 L2 归一化，其中认为负平方欧氏距离与余弦距离成正比。第二个是学习模块，该阶段涉及到解决一个有监督的多类分类问题，可用任意的一个分类器解决。原始的 LMI 使用了一个单一的全连接神经网络，即多层感知器（Multilayer Perception, MLP）。

3.2 搜索

LMI 的搜索主要由导航和桶级顺序搜索两部分组成，这两个部分的作用分别是减少搜索空间和计算剩余数据对象和原始空间中的查询之间的成对距离，其完整处理过程如图所示。首先，将 10000 个 768 维的查询对象通过主成分分析方法（Principal Component Analysis, PCA）降到 96 维，并作为导航阶段的卷

积神经网络（Convolutional Neural Network, CNN）的输入，得到一个概率分布矩阵，该矩阵捕获每个查询对象与每个类别的关系。其中，每个列被分解为单独的类别，在桶级顺序搜索中对来自给定类别的数据对象与相关查询之间的距离进行评估和排序，形成相互距离矩阵。然后，连续更新前 k 列，得到最终的输出。

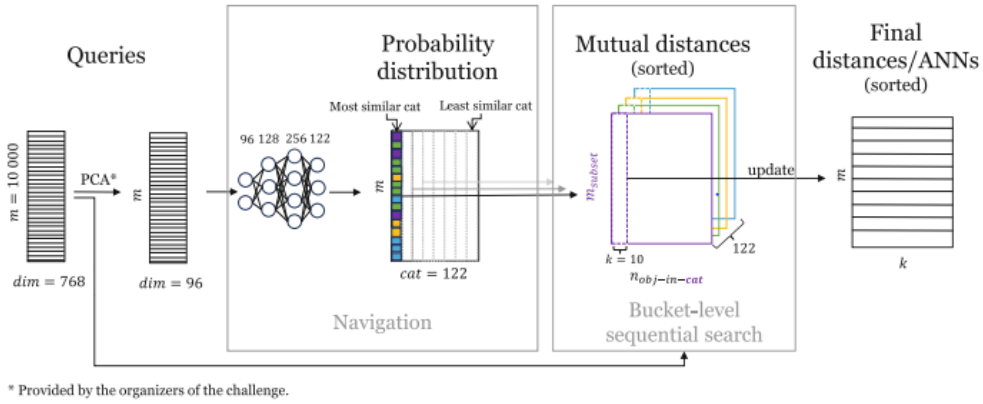

图 1 LMI 搜索完整处理过程

4 复现细节

4.1 与已有开源代码对比

本课题对 LMI 原方法进行了改进，在导航阶段的神经网络选择中添加了卷积神经网络，其主要代码改动如下图所示。其中，红框内为在源代码基础上添加的部分：

```
    if model_type == 'MLP-9':
        self.layers = torch.nn.Sequential(
            torch.nn.Linear(input_dim, 8),
            torch.nn.ReLU(),
            torch.nn.Linear(input_dim, 16),
            torch.nn.ReLU(),
            torch.nn.Linear(16, output_dim)
        )
    # CNN model
    if model_type == 'CNN':
        self.layers = torch.nn.Sequential(
            torch.nn.Linear(input_dim, 256),
            torch.nn.ReLU(),
            torch.nn.Linear(256, 128),
            torch.nn.ReLU(),
            torch.nn.Linear(128, output_dim)
        )
    self.n_output_neurons = output_dim
```

图 2 代码主要添加片段 1

```
class Model(nn.Module):
    """ The model class representing the index. """

    def __init__(self, input_dim=768, output_dim=1000, model_type=None, num_classes=None):  # noqa: C901
        super().__init__()
        self.input_shape = self.input_shape
        self.num_classes = num_classes
        self.model = self.build_model()

        def build_model(self):
            model = models.Sequential([
                layers.Conv2D(32, (3, 3), activation='relu', input_shape=self.input_shape),
                layers.MaxPooling2D((2, 2)),
                layers.Conv2D(64, (3, 3), activation='relu'),
                layers.MaxPooling2D((2, 2)),
                layers.Conv2D(64, (3, 3), activation='relu'),
                layers.Flatten(),
                layers.Dense(64, activation='relu'),
                layers.Dense(self.num_classes, activation='softmax')
            ])
            return model

        def compile_model(self):
            self.model.compile(optimizer='adam',
                               loss='sparse_categorical_crossentropy',
                               metrics=['accuracy'])

        def train(self, train_images, train_labels, epochs):
            self.model.fit(train_images, train_labels, epochs=epochs)

        def evaluate(self, test_images, test_labels):
            return self.model.evaluate(test_images, test_labels)
```

图 3 代码主要添加片段 2

4.2 实验环境

本课题采用 Python 编写，Python 版本：3.11；操作系统：Windows 10；处理器：Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz .00 GHz；GPU：Intel(R) UHD

Graphics 630。

4.3 创新点

在导航阶段，原始的 LMI 采用 MLP 得到概率分布矩阵，本课题将其替换成了 CNN，其相比 MLP 具有如下优势：

（1）对图像和空间数据的处理更有效：CNN 可以通过卷积层和池化层识别图像中的特征，并且可以保留空间结构信息，更适合处理图像和空间数据。

（2）参数共享和权值共享：CNN 可以通过参数共享减少模型参数的数量，从而降低模型复杂度，减少过拟合的可能性。

（3）局部感知能力：CNN 可以有效地识别图像中局部特征，而不受全局变化的影响，这使得它在处理图像等数据时更加有效。

（4）对平移和旋转不变性更强：CNN 通过卷积操作可以获取到图像中的局部特征，这使得模型对于图像的平移和旋转不变性更强。

5 实验结果分析

本课题分别在 300K 和 100K 的数据集上进行了实验，实验结果分别如图 4 和图 5 所示。由于搜索时间主要依赖于硬件，在 300K 大小的数据集上实验的搜索时间与 LMI 原实验相比仍有一定差距，但是在超参数的选择上结果能够保持一致，与原实验的相似。另外，还在原实验未测试过的 100K 大小的数据集上验证了改进 LMI 的效果，从图 5 可见学习回合数和超参数的选择与 300K 的实验结果一致，搜索时间有明显的减少。

（1）复现结果 1：size=300K, n_buckets_prc=7, epochs=205, n_categories=122, lr=0.009, build time=1263.3835325, search time=26.646454

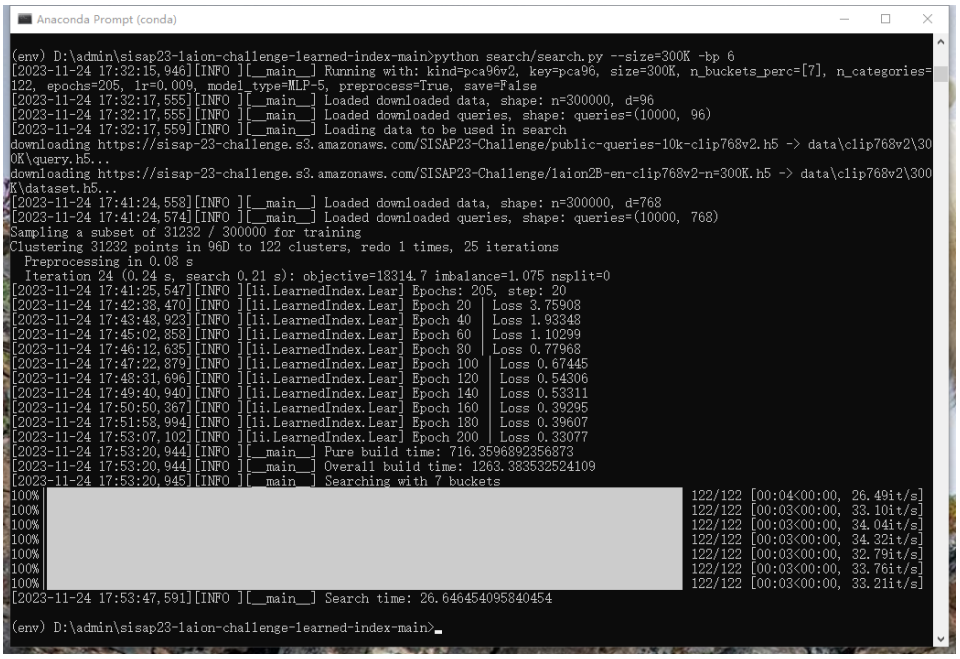

图 4 实验结果 1

（2）复现结果 2：size=100K, n_buckets_prc=7, epochs=205, n_categories=122, lr=0.009, build time=295.4309738，search time=10.076256

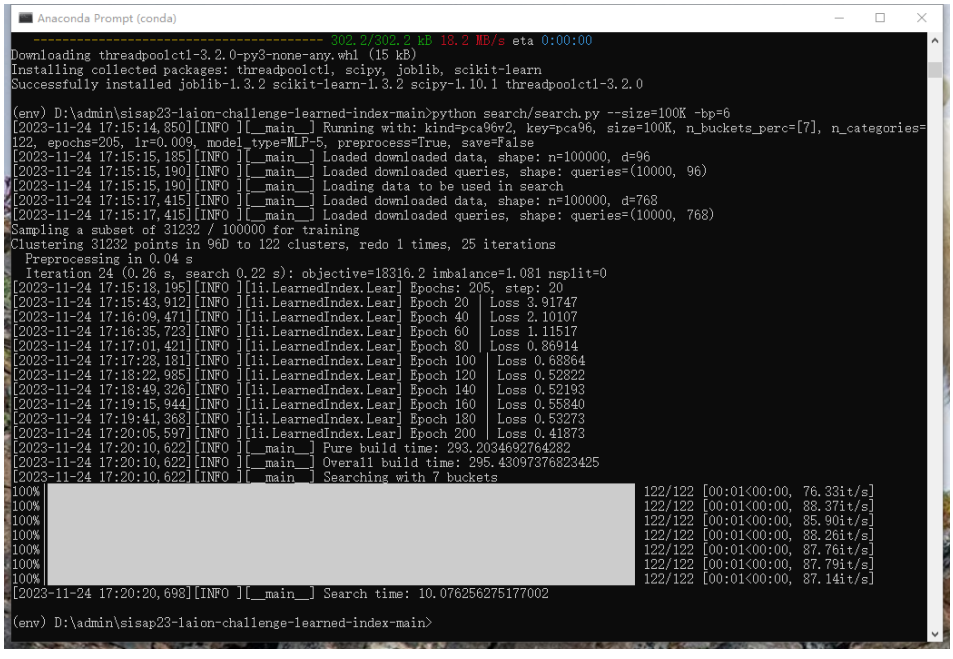

图 5 实验结果 2

6 总结与展望

总结来看，在求解 90%以上召回率的近似最近邻搜索的约束下，该索引方法确实能够显示出快速的导航时间。本课题在原始 LMI 的基础上对神经网络的选择进行了改进，从实验结果上看，在超参数、学习回合数上能够得到较好的实验结果，但是搜索时间受硬件影响更大，优化程度有限，在未来可继续在分类器的索引结构和神经网络结构之间的权衡等方面进行优化。

参考文献

[1] Slanináková T, Procházka D, Antol M, et al. SISAP 2023 Indexing Challenge–Learned Metric Index[C]//International Conference on Similarity Search and Applications. Cham: Springer Nature Switzerland, 2023: 282-290.

[2] Ciaccia P, Patella M, Zezula P. M-tree: An efficient access method for similarity search in metric spaces[C]//Vldb. 1997, 97: 426-435.

[3] Novak D, Batko M, Zezula P. Metric index: An efficient and scalable solution for precise and approximate similarity search[J]. Information Systems, 2011, 36(4): 721-733.

[4] Antol M, Ol'ha J, Slanináková T, et al. Learned metric index—proposition of learned indexing for unstructured data[J]. Information Systems, 2021, 100: 101774.