|  |  |
| --- | --- |
| **密级：□绝密 □机密 □秘密** | **学校代码： 11287** |
|  | **学 号：** |



|  |
| --- |
| **硕 士 研 究 生 学 位 论 文** |

|  |
| --- |
| **一种鲁棒的基于特征子空间插值的**  **数据合成方法** |

|  |  |
| --- | --- |
| **院 系：** | **统计与数据科学学院** |
| **专 业：** | **统计学** |
| **研 究 方 向：** | **数据挖掘、机器学习** |
| **授予学位类别：** |  |
| **姓 名：** | **杜玉坤** |
| **指 导 教 师：** | **陆敏教授** |
| **答 辩 日 期：** |  |

**硕士学位论文独创性声明**

本人郑重声明：所呈交的学位论文，是我本人在导师指导下独立进行研究工作而取得的研究成果。除文中已经注明引用的内容外，本论文不包含其他个人或机构已经发表或撰写过的研究成果。其他人或机构对本研究的启发和所做的贡献均已在文中以明确方式标明或表示了谢意。本声明的法律结果由本人承担。

作者签名： 日期：

**硕士学位论文使用授权书**

研究生在校攻读学位期间论文工作的知识产权单位属南京审计大学。学校有权保存送交论文的印刷件和电子版，可以借阅或上网公布本学位论文的部分或全部内容，可以采用影印、缩印或其它复制手段保存、汇编本学位论文。学校可以向国家机关或机构送交论文的印刷件和电子版，允许论文被查阅或借阅。保密的论文在解密后遵守此规定。

本论文涉密情况：

1. 不保密（ √ ）

2. 保 密（ ），保密期限为： 年 月 日至 年 月 日

作者签名： 导师签名： 日期：

目录

[一、绪论 6](#_Toc157187724)

[1.1 研究背景及意义 6](#_Toc157187725)

[1.2 文献综述 8](#_Toc157187726)

[1.2.1 关于数据合成方法的研究 8](#_Toc157187727)

[1.2.2 针对含噪数据集处理方法的研究 9](#_Toc157187728)

[1.3 主要研究内容 10](#_Toc157187729)

[二、相关概念界定 13](#_Toc157187730)

[2.1 无监督聚类 13](#_Toc157187731)

[2.2 旅行商问题 14](#_Toc157187732)

[2.2.1 问题阐述 14](#_Toc157187733)

[2.2.2 旅行商问题求解方法 14](#_Toc157187734)

[2.2.3 TSP的实际应用 15](#_Toc157187735)

[2.2.4 TSP变体 16](#_Toc157187736)

[2.3 最小权匹配问题 17](#_Toc157187737)

[2.3.1相关概念 17](#_Toc157187738)

[2.3.3 求解方法 18](#_Toc157187739)

[2.4 卷积神经网络 19](#_Toc157187740)

[2.4.1 基本结构 19](#_Toc157187741)

[2.4.2 相关应用 19](#_Toc157187742)

[三、基于特征子空间插值的数据合成方法 21](#_Toc157187743)

[3.1 特征子空间的划分 22](#_Toc157187744)

[3.1.1 迭代层次聚类 22](#_Toc157187745)

[3.1.2 K近邻优化 25](#_Toc157187746)

[3.2 子空间插值路径排序 26](#_Toc157187747)

[3.3 相邻子空间线性回归拟合 28](#_Toc157187748)

[3.4 多阶段最小权匹配 29](#_Toc157187749)

[3.5 方法整体分析 34](#_Toc157187750)

[四、模拟实验 36](#_Toc157187751)

[4.1 模拟设计 36](#_Toc157187752)

[4.2 噪音处理效果分析 36](#_Toc157187753)

[4.3 超参数敏感性分析 36](#_Toc157187754)

[4.4 对于样本分布的影响分析 36](#_Toc157187755)

[五、实例分析 37](#_Toc157187756)

[六、结合卷积神经网络框架的进一步研究 38](#_Toc157187757)

[6.1方法概述 38](#_Toc157187758)

[6.2 预测效果分析（结合轻量型的网络） 38](#_Toc157187759)

[七、总结与展望 39](#_Toc157187760)

[参考文献 40](#_Toc157187761)

[附录 43](#_Toc157187762)

[致谢 53](#_Toc157187763)

# 一、绪论

## 1.1 研究背景及意义

在当今的数据驱动时代，数据质量对于机器学习和数据科学领域的影响日益显著。尤其是在大数据环境下，数据噪声问题成为了一个不可忽视的挑战。数据噪声指的是数据中的任何不准确、不相关或误导性信息，这可能源于多种原因，如仪器误差、数据录入错误、传输问题，甚至是数据采集过程中的系统性偏差。在大多数实际问题中，噪声的分布都是复杂且未知的，这些噪声可能导致数据分析和机器学习模型的性能大打折扣，尤其是在需要高度精准预测和分析的应用场景中，如医疗影像分析、金融市场预测、环境监测等。在这些领域，即使是微小的数据误差也可能导致严重的后果。因此，有效地识别和处理数据中的噪声，已成为提高预测模型准确性和可靠性的关键步骤[1–3]。

为了解决含噪数据集的问题，研究者们开发了多种方法，包括数据清洗、数据平滑和异常值检测等。这些方法主要集中于减轻或移除数据中的噪声，但它们通常不会从根本上提升数据质量。例如，数据清洗过程可能面临将噪声与真实的异常值判断的挑战。异常值在某些情况下可能是数据分析中的重要信息源，但数据清洗可能错误地将其视为噪声并将其移除。这种区分的困难可能导致关键信息的丢失。同样地，数据平滑技术，如移动平均或高斯平滑，虽然可以减少数据中的随机波动，但也可能导致数据过度泛化。这种过度泛化可能会模糊数据中的重要特征和细节，从而影响数据分析的准确性和深度。此外，当处理复杂和高维度的数据集时，传统的噪声处理方法往往效果不佳。在高维数据中，噪声和有用信息之间的界限可能更加模糊，使得有效地识别和处理噪声变得更加困难。这些方法在高维空间中可能无法有效区分噪声和有用信号，导致在降噪过程中可能误删有价值的数据，或者保留过多的噪声。

因此，尽管这些传统方法在某些情况下可能有效，但它们在处理复杂数据集时可能不足以应对噪声的挑战，特别是在需要保留数据中重要细节和模式的场景中。在这种背景下，本文提出了一种创新的对于含噪数据集处理的数据合成方法，基于特征子空间插值的思想，命名为RSIS（Robust Subspace Interpolation Synthesis）。这种方法旨在不损失样本原始信息的前提下，通过合成误差较小的样本，降低数据集整体的噪音水平，并减少高噪音样本的比例。RSIS方法有效减少了数据噪音的误导性影响，同时强化了数据集变量之间真实的函数映射关系。通过这两方面的作用，RSIS达到了优化样本的目的。与传统的含噪数据集噪声处理方法相比，RSIS更适用于机器学习领域的样本噪音处理，即使是在处理含有未知复杂噪音的高纬数据时，该方法也可以有效提升机器学习模型的泛化能力。

RSIS方法包含几个关键步骤。首先，通过无监督聚类的方法对原始数据集进行迭代层次聚类，该方法可以将原始数据集划分为多个样本量相近的子集，这一步骤会将原始特征空间划分为多个具有相近样本量的子空间。接着，利用聚类结果，RSIS方法引入了旅行商问题的思想对特征子空间进行排序。随后，本文结合了软参数多任务学习的机制，根据排序结果，对相邻子空间之间进行考虑了临域信息的线性拟合。最后，本文提出了一种创新的多阶段最小权匹配启发式方法，通过该方法可以得到一种较优的插值匹配策略，通过该策略对样本进行线性插值可以生成具有较小误差的样本。

目前许多研究都假设样本的噪音是同分布的，但是这种情况对于实际应用中是较为少见的，本文从理论上证明了RSIS方法即使面对不同分布的噪音样本依旧可以表现出很好的优化效果。基于噪音不同分布的场景，本文还模拟了样本中含有未知且不同分布的复杂噪音的情况，RSIS方法在实际模拟实验中依旧可以具有良好的鲁棒性，并且对样本进行显著的优化。

此外，本文还将RSIS方法结合卷机神经网络应用到计算机视觉领域，通过实验结果可知，RSIS方法同样可以有效提升图像分类模型的泛化能力。值得注意的是，RSIS方法在减少数据噪声影响的同时，还增强了数据的多样性以及代表性，这一点对于那些依赖于数据多样性的机器学习模型尤为重要。通过提供更准确、更干净且更具有多样性的训练数据，RSIS方法有助于提高机器学习模型的预测性能，从而在实际应用中发挥更大的价值。

## 1.2 文献综述

### 1.2.1 关于数据合成方法的研究

目前已有的数据合成方法数据合成的目的是通过生成非真实但统计上具有代表性的数据来增强、扩展或模拟数据集。这些方法在数据稀缺、隐私保护或模型训练等场景中被广泛应用。目前，数据合成的主要方法包括：

1.基于规则的方法。这类方法依赖于预定义的规则或算法来修改现有数据或生成新的数据实例。例如，线性插值和多项式插值用于在已知数据点之间生成新点。图像数据增强，如旋转、缩放、翻转等，也属于此类。此外，模拟法如蒙特卡洛模拟和代理模型则通过随机抽样或简化模型来模拟数据分布。

2.基于统计的方法。这些方法使用统计模型来估计数据分布，并据此生成数据。这类方法可以进一步细分为参数化方法和非参数化方法。在参数化方法中，如高斯混合模型和贝叶斯网络，数据合成过程依赖于特定的分布假设。这些模型假定数据遵循某种已知的概率分布，如正态分布或其他参数化分布，然后利用这些分布的参数来生成新的数据点。非参数化方法，如核密度估计、直方图方法以及SMOTE方法和其变体，则不依赖于固定的分布模型。它们直接从数据本身推断出数据分布的形状，而不是依赖于预先定义的数学形式。

3. 基于机器学习的方法，尤其是生成对抗网络（GANs）和变分自编码器（VAEs），通过学习数据分布来生成新数据。GANs通过对抗过程训练生成器和判别器，而VAEs使用编码器-解码器架构来学习数据的潜在表示。自回归模型，如PixelRNN和PixelCNN，通过学习数据中的像素依赖关系来生成图像。

4. 最后，基于混合方法的数据合成结合了以上方法的优点，使用多种技术来生成数据。这包括模型集成，即结合多个模型的输出来生成数据，以及多阶段生成，即先使用一种方法生成初步数据，再用另一种方法进行精细化处理。

目前已有的数据合成方法对于噪音数据集并不具有良好的鲁棒性，因为大部分数据合成方法的设计初衷并不是用于处理样本噪音的问题，如SMOTE算法，其目的是为了针对分类任务用于平衡样本数据；以及差分隐私（Differential Privacy）算法是一种旨在提供强大隐私保护的技术，同时允许对数据集进行有用的统计分析。RSIS作为一种创新的基于混合方法的数据合成方法，不仅解决了含噪数据集的特定问题，也展示了如何通过创新的合成技术来优化数据质量，进而提升机器学习模型在处理复杂数据时的性能和泛化能力。

### 1.2.2 针对含噪数据集处理方法的研究

在机器学习和数据科学的领域中，处理含噪数据集是一个重要且复杂的问题。本综述旨在探讨近年来在处理含噪数据集方面的研究进展，从传统机器学习方法到现代深度学习技术的多种策略。

在处理机器学习数据集中的噪声问题方面，研究者们做出了显著贡献。例如，Saez等人提供了对机器学习数据集中噪声问题的全面回顾，强调了数据预处理的重要性，并探讨了不同类型的噪声及其对模型性能的影响[4]。此外，Frenay和Kaban探讨了从噪声数据中学习解释性规则的技术，展示了如何从不完美的数据中提取有价值的信息[5]。Yi等人在医学图像分析中探讨了深度学习处理噪声标签的技术[6]。同时，Evans等人的研究全面探讨了使用深度神经网络从噪声标签中学习的方法，这些研究表明深度学习技术能够有效地处理复杂数据集中的噪声问题[7]。

Wu等人提出了一种新的机器学习训练方法——共同教学法，用于提高在噪声数据环境下的模型准确性。研究使用了包含非高斯噪声的ASPEN数据集，并展示了共同教学法在化学过程模型中如何提高准确性[8]。在另外一项研究中，他们还探讨了在带有噪声的数据集上使用长短期记忆（LSTM）网络进行非线性过程建模和预测控制。研究表明，标准LSTM网络在处理高斯噪声数据时表现良好，但在处理工业操作中的非高斯噪声数据时表现不佳。因此，提出了一种使用蒙特卡罗dropout方法的LSTM网络，以减少对噪声数据的过度拟合[9]。

在特定应用领域，如室内定位系统和制造过程中的数据集，噪声处理技术同样显示出其重要性。例如，Kim等人使用深度学习技术开发了基于Wi-Fi的室内定位系统，突出了在噪声环境下的应用[10]。另一方面，Oleghe等人提出了一种预测性噪声校正方法，专门用于制造过程中的数据集[11]。在处理噪声标签方面，一些研究提出了创新的方法。例如，Han等人提出了一种针对噪声标签的负学习方法，这是一种新颖的处理噪声标签的策略[12]。

Bhatia提出了一种基于降噪的混合分类器，使用遗传算法进行优化，用于植物病害预测。研究使用了番茄和大豆病害数据集，并结合支持向量机（SVM）方法进行噪声降低。通过精确度、ROC曲线下面积和F1分数等指标评估了模型性能[13]。

Qaisar等人提出了一种基于有效信号引导处理和机器学习的心电图（ECG）噪声去除和高效心律失常识别方法。使用自适应速率小波分解方案从预处理信号中提取特征，用于心律失常识别。研究结果表明，该系统能够通过分析传入信号的变化来调整其参数，有效降低系统的计算成本，并提高分类性能[14]。

Lapins等人介绍了一种弱监督机器学习方法DAS-N2N，用于抑制分布式声学传感（DAS）记录中的强随机噪声。DAS-N2N无需手动标记的干净数据进行训练，旨在将随机噪声过程映射到所选的摘要统计量，同时保留真实的底层信号。使用南极洲Rutford冰流上部署的DAS阵列的数据集，证明了DAS-N2N在抑制非相干噪声和增强自然微地震冰震事件的信噪比方面的有效性[15]。

Ali Parizad等提出了一个基于协作机器学习的框架，用于检测电力系统中导致状态变量行为偏差的网络攻击。研究采用了三种不同的基于机器学习的方法（可视化、分类和聚类），并比较这些方法在假数据注入攻击（FDIA）检测过程中的效果。结果表明，该协作优化的PCA-基于密度的机器学习技术能够检测出大多数攻击样本，并在不同情景下表现出良好的性能[16]。

也有一些研究，从数据集整体预处理以及样本采样方面进行噪音处理，如Mundra等人研究了在不平衡医疗数据分类中的机器学习方法。研究探讨了三种过采样技术和两种欠采样技术，并结合数据降维和清洗方法。结果表明，适当的采样和数据处理组合可以在大型和小型数据集中提高性能[17]。

综上所述，处理含噪数据集是一个多方面的挑战，涉及从数据预处理到模型训练的各个方面。近年来的研究表明，通过结合概率论、深度学习和信息论方法，可以有效地减少噪声数据的负面影响。这些方法的发展为机器学习和数据科学领域提供了新的工具和技术，以提高模型在复杂和不确定环境中的性能和准确性。

## 1.3 主要研究内容

本文提出了一种创新的对于含噪数据集处理的数据合成方法，基于特征子空间插值的思想，命名为RSIS方法。该方法融合了运筹优化、数据合成、机器学习多个学科技术，可以在保证不损害样本原始信息的前提下，合成大量具有较小误差的样本，从而提升机器学习模型的泛化能力。本文对RSIS方法进行了全面的模拟研究，通过生成含有复杂且未知噪音的数据集，本文探究了超参数的敏感性以及RSIS方法对于含噪数据集的优化效果。并且本文还将RSIS方法应用到计算机视觉领域进行拓展研究，创新地RSIS方法与卷积神经网络相结合，实验结果表明显著提升RSIS方法同样可以提升图像分类任务的预测效果。

本文剩余的章节安排如下：

第二章节，RSIS方法融合了运筹优化、机器学习等多个领域的技术，本章节主要阐述了后续研究涉及到的相关概念，包括对于旅行商问题的阐述，卷积神经网络的详细介绍等。

第三章节，详细阐述了RSIS方法的过程，将RSIS方法分为四个部分进行介绍。本第一部分，主要介绍了一种无监督聚类算法，该算法可以通过较少的运算达到特征空间分割的目的，得到多个具有样本量相近的特征子空间。第二部分，主要解决了针对多个特征子空间的插值排序问题，将原始问题转化为旅行商问题进行求解。第三部分，主要提出了针对两个相邻的子空间之间样本的线性拟合方法。第四部分，提出了样本匹配算法，基于加权二分图的最小权匹配问题的思想，针对两个相邻子空间之间的样本进行线性插值，通过匈牙利算法求解得到最优的样本插值匹配方案。在本章节的最后总结了RSIS方法的假设以及整体操作流程。

第四章节，对RSIS方法进行模拟实验，本文通过生成含有复杂且未知噪音的数据集，本文探究了超参数的敏感性以及RSIS方法对于含噪数据集的优化效果。

第五章节，选取了多个真实数据集，将RSIS方法集合机器学习模型对模型的泛化能力提升进行详细探究，涉及到的模型有：多层感知机、梯度提升树、随机森林、K近邻回归等。

第六章节，将RSIS方法结合CNN框架模型进行应用，探究RSIS方法对于特征提取后的张量数据的优化效果，并且考虑到了轻量型和非轻量网络，选择了ResNet、SqueezeNet、VGG等多个框架进行结合探究。

第七章节，总结了RSIS方法的优化效果，并且展望了对于RSIS方法未来的研究期望。

# 二、相关概念界定

## 2.1 无监督聚类

无监督聚类是一种重要的机器学习方法，它在数据科学、模式识别、图像分析等领域有着广泛的应用。与监督学习不同，无监督聚类不依赖于预先标注的数据，而是通过分析数据本身的特征和结构来发现数据间的潜在关系和模式。无监督聚类的目标是将数据集中的样本划分为若干个群组或簇（Cluster），使得同一簇内的样本相似度高，而不同簇间的样本相似度低。这种方法的核心在于定义“相似度”或“距离”指标，这些指标决定了数据点如何被归类。

无监督聚类在许多领域都有广泛应用，如市场细分、社交网络分析、生物信息学、图像处理等。在这些应用中，聚类帮助揭示了数据的内在结构和模式，为进一步的分析和决策提供了支持。不同的聚类算法适用于不同类型的数据集，选择合适的算法对聚类结果至关重要。在某些算法中，如K-Means，需要预先指定簇的数量，这通常需要基于数据的特性和领域知识来确定。常见的无监督聚类算有：

K-Means聚类：最常用的聚类算法之一。它通过迭代的方式优化簇内样本与簇中心的距离平方和，直到达到最优或满足停止条件。

层次聚类：通过构建一个多层次的树状结构来进行聚类。它可以是自底向上的聚合（凝聚）方法，也可以是自顶向下的分裂（分裂）方法。

DBSCAN（基于密度的空间聚类应用与噪声）：这种算法基于样本的密度，能够识别出任意形状的簇，并且对噪声点具有良好的鲁棒性。

谱聚类：使用样本数据的谱（即特征值）来进行聚类，特别适合于处理非球形的数据簇。

由于无监督聚类不依赖于预先的标签，评估聚类效果通常不如监督学习那样直观。常见的评估指标包括轮廓系数（Silhouette Coefficient）、戴维斯-布尔丁指数（Davies-Bouldin Index）和Calinski-Harabasz指数等。

在高维空间中，数据的距离度量可能变得不那么直观，这给聚类带来了额外的挑战，并且数据的噪声和异常值可能严重影响聚类结果的质量。

无监督聚类作为一种强大的数据分析工具，它能够在没有明确标签指导的情况下发现数据的内在结构。虽然它面临着选择合适算法、确定聚类数量等挑战，但通过合理的方法选择和参数调整，无监督聚类能够在各种应用中发挥重要作用。随着机器学习和人工智能技术的不断进步，无监督聚类的方法和应用也在不断地发展和完善。

## 2.2 旅行商问题

### 2.2.1 问题阐述

旅行商问题（Traveling Salesman Problem，简称TSP）是组合优化中的一个经典问题，广泛应用于物流、规划、电子设计、生物信息学等领域。它描述的问题是给定一组城市和每对城市之间的距离，求解访问每个城市一次并返回起点的最短可能路线。TSP可以用图论来表述，给定一个加权图，其中顶点表示城市，边表示城市间的道路，边的权重表示城市间的距离或成本，TSP要求找到一个最短的哈密顿回路（访问每个顶点一次并返回起点的闭合路径）。

### 2.2.2 旅行商问题求解方法

旅行商问题的求解方法可以分为两大类：精确算法和近似算法。这些方法各有优势和局限性，适用于不同规模和复杂度的TSP问题。

精确算法旨在找到TSP问题的最优解。这类算法包括：

穷举法：这是最直接的方法，该方法尝试所有可能的路径组合来找到最短路径。尽管结果准确，但穷举法的计算复杂度随城市数量的增加而呈指数级增长，因此仅适用于非常小规模的问题。

分支限界法：这种方法通过系统地枚举所有候选解决方案的子集来寻找最优解。该方法使用上下界来剪枝，从而减少需要考虑的路径数量。分支限界法比穷举法更高效，但在大规模问题上仍然有计算上的挑战。

动态规划：动态规划是解决TSP的另一种精确方法。该方法通过将问题分解为重叠的子问题，并存储这些子问题的解，来避免重复计算。动态规划适用于中等规模的TSP问题，但当问题规模增大时，所需的存储空间和计算时间也会迅速增加。

由于精确算法在大规模问题上的局限性，近似算法成为了实际应用中的首选，这些算法通常能在可接受的时间内找到一个足够好的解决方案：

启发式算法：启发式算法根据问题的特定特性来指导搜索过程。例如，贪婪算法在每一步选择当前看起来最优的选择；最近邻法选择离当前城市最近的城市作为下一个目的地；插入法则是从一个小的子路径开始，逐步添加新的城市。

元启发式算法：这类算法通过模拟自然界的过程或采用智能搜索策略来寻找解决方案。例如，遗传算法模拟自然选择和遗传机制；模拟退火模仿物质加热后冷却的过程；蚁群算法受到蚂蚁寻找食物路径的启发；粒子群优化则模仿鸟群或鱼群的社会行为。

这些近似算法的关键优势在于它们能够在合理的时间内处理大规模的TSP问题，并且通常能找到接近最优的解决方案。这些方法通过不断迭代和调整搜索策略，逐渐逼近问题的最优解。

在实际应用中，选择合适的TSP求解方法需要考虑多个因素，包括问题的规模、求解的精度要求、可用的计算资源以及解决方案的实施时间。对于小规模问题，精确算法可能是可行的；而对于大规模或实时性要求高的问题，近似算法则更为合适。总的来说，TSP的求解方法多种多样，每种方法都有其适用场景和局限性。随着计算技术的发展和新算法的不断涌现，解决TSP问题的效率和准确性不断提高，为各种实际应用提供了强有力的支持。

### 2.2.3 TSP的实际应用

TSP作为一个经典的组合优化问题，在现实世界中有着广泛而深远的应用。这些应用不仅涵盖了从物流配送到科学研究的多个领域，而且对于提高效率、降低成本、优化资源分配等方面起着至关重要的作用。

在物流和配送行业，TSP的应用尤为重要。这个领域的核心挑战是如何在最短的时间内，或者以最低的成本，将货物从一个地点运送到多个目的地。通过解决TSP，公司能够优化配送路线，减少旅行距离和时间，从而降低燃油消耗和运输成本。这对于提高配送效率、增强客户满意度以及减少环境影响都至关重要。

在旅游行业，TSP可以帮助设计覆盖多个景点的最佳旅游路线。这不仅使游客能够在有限的时间内访问更多的景点，而且还能提高旅游体验的质量。旅行社和在线旅游服务提供商通过解决TSP，能够为客户提供定制化的旅游计划，从而在竞争激烈的市场中获得优势。

在电子制造行业，尤其是在电路板的设计和制造过程中，TSP被用来优化钻孔过程。电路板上的钻孔通常需要精确而高效地完成，TSP算法可以帮助确定钻孔顺序，以最小化钻头移动的总距离，从而提高生产效率并降低成本。

在自动化和机器人技术领域，TSP用于优化机器人的移动路径。无论是在制造车间的自动化装配线上，还是在自动化仓库中，通过有效的路径规划，可以确保机器人以最高的效率执行任务，减少能源消耗，并提高整体作业效率。

在科学研究领域，特别是在基因组学和蛋白质组学中，TSP被用来解决各种序列化问题。例如，在DNA测序中，TSP可以帮助确定最有效的测序路径，从而加快数据处理速度并降低成本。此外，在数据分析中，TSP有助于解决各种聚类和优化问题，从而提高分析的准确性和效率。

### 2.2.4 TSP变体

在TSP的实际应用中，也会根据不同的实际需求和约束条件产生不同的TSP变体：

对称TSP（Symmetric TSP）：这是最基本的TSP形式，其中城市间的距离是对称的，即从城市A到城市B的距离与从城市B到城市A的距离相同。这种情况通常适用于距离或成本不依赖于旅行方向的场景，如无方向限制的道路网络。对称TSP相对简单，因为它减少了需要考虑的路径数量。

非对称TSP（Asymmetric TSP）：在非对称TSP中，城市间的距离或成本是非对称的，这意味着从城市A到城市B的距离可能与从城市B到城市A的距离不同。这种情况在现实世界中很常见，例如，由于单行道、交通规则或不同的运输成本，导致往返两地的路径和成本不同。非对称TSP的复杂性更高，因为它需要考虑更多的路径可能性。

时间窗TSP（Time Windows TSP）：每个城市都有一个特定的时间窗口，在这个时间内旅行商必须到达。这种类型的TSP在物流和配送领域尤为重要，例如，送货服务可能需要在特定时间内完成。时间窗TSP增加了额外的时间约束，使问题更加复杂。

多旅行商问题（Multiple Traveling Salesmen Problem, MTSP）：在MTSP中，不止一个旅行商需要遍历所有城市，这种情况在实际中非常常见，比如多个配送车辆共同完成配送任务。MTSP的挑战在于如何有效地分配路线给每个旅行商，以最小化总成本或总旅行时间。

车辆路径问题（Vehicle Routing Problem, VRP）：VRP是TSP的一个扩展，其中包括了车辆容量限制和其他复杂的约束条件。在VRP中，一个或多个车辆从中心仓库出发，需要服务多个客户，并最终返回仓库，同时考虑车辆的载重限制、服务时间等因素。

这些TSP变体展示了TSP在解决实际问题时的灵活性和广泛性。每个变体都有其独特的特点和挑战，需要特定的算法和策略来求解。随着计算能力的提升和算法的发展，越来越多复杂的TSP变体得到了有效的解决，为各行各业的优化问题提供了有力的工具。

总之，TSP的应用范围极广，它不仅在理论上具有重要意义，而且在实际应用中发挥着巨大作用。通过不断的研究和技术创新，TSP及其变体在解决实际问题方面的潜力仍在不断扩展，为各行各业提供了强大的优化工具。

## 2.3 最小权匹配问题

### 2.3.1相关概念

最小权匹配问题（Minimum Weight Matching Problem）是组合优化和图论中的一个经典问题，广泛应用于计算机科学、运筹学、经济学等领域。在最小权匹配问题中，给定一个加权图（通常是完全图），目标是找到一个匹配，使得该匹配中所有边的权重之和最小。这个问题可以是寻找完美匹配，也可以是寻找非完美匹配，具体取决于问题的约束条件。最小权匹配问题通常在加权图中进行，其中图的每条边都被赋予一个权重，这个权重可以代表成本、距离或任何其他需要最小化的量。如果图中的每个顶点都恰好与匹配中的一条边相关联，这种匹配被称为完美匹配。这个问题的核心是在一个加权图中找到一个边的集合，使得这些边不共享任何顶点（即匹配），并且这些边的权重之和最小。

在任务分配问题中，如工厂的作业调度或办公室的任务分配，最小权匹配可以帮助找到成本最低或效率最高的分配方案。例如，将一组任务分配给一组员工，使得总工作时间最短或总成本最低;在物流行业，最小权匹配问题可以用来优化货物的配送路线。例如，将一组货物分配给一组运输车辆，以最小化总的运输距离或成本;在计算机网络或通信网络的设计中，最小权匹配可以帮助确定最佳的连接方式，以最小化布线成本或信号传输的延迟;在生物信息学和化学研究中，最小权匹配问题可以用于分子结构分析、蛋白质结构预测等领域。例如，通过匹配分子中的原子来预测化学反应的结果或分子间的相互作用;在经济学中，最小权匹配可以应用于市场配对问题，如买家和卖家之间的配对，以最大化整体市场效率或最小化交易成本。

### 2.3.3 求解方法

最小权匹配问题的求解通常涉及复杂的算法，包括：

匈牙利算法：匈牙利算法是解决最小权匹配问题，特别是完美匹配问题的经典方法。该算法基于增广路径的概念，通过不断地改进现有匹配来降低总权重。它首先从任意匹配开始，然后通过找到增广路径来交替地改变匹配中的边，直到无法进一步改进为止。匈牙利算法的效率相对较高，适用于中等规模的问题。

线性规划和整数规划：最小权匹配问题可以转化为线性规划或整数规划问题来求解。这种方法通过定义一个目标函数（即所有匹配边的权重和）和一系列约束条件（确保每个顶点只与一个匹配边相关联），来寻找最优解。这种方法在理论上可以求解任何规模的问题，但在实际应用中可能因为计算复杂度而受限。

近似算法：对于大规模的图或特殊类型的图，完全的优化可能过于复杂或计算成本过高。在这些情况下，可以使用近似算法来找到一个近似最优的解。这些算法可能无法保证找到绝对的最优解，但可以在合理的时间内提供一个质量足够好的解决方案。

总体而言，最小权匹配问题的求解方法多种多样，从经典的匈牙利算法到现代的优化技术，都为解决实际问题提供了强大的工具。同时，这个问题在各个领域都有广泛的应用，显示了其在理论和实践中的重要性。随着计算技术的发展，求解这类问题的效率和准确性不断提高，为各种实际应用提供了更多可能性。

## 2.4 卷积神经网络

### 2.4.1 基本结构

卷积神经网络（Convolutional Neural Network，简称CNN）是一种深度学习的算法，广泛应用于图像处理、视频分析、自然语言处理等领域。CNN的核心特点是能够直接对数据进行多层次的非线性处理，特别适合于处理具有空间关系（如像素）的数据。

卷积神经网络通过逐层处理输入数据，逐步提取更高层次的抽象特征。在初级卷积层，网络可能只能识别简单的特征，如边缘或颜色斑点。在更深层次，网络能够识别更复杂的特征，如形状或特定对象的部分。这种层次化的特征提取使得CNN在图像和视频分析等领域表现出色。卷积神经网络通常包含以下几种类型的层：

卷积层（Convolutional Layer）：卷积层是CNN的核心，用于提取输入数据的特征。它通过卷积核（或滤波器）在输入数据上滑动，计算卷积核与输入数据的局部区域的点积，从而产生特征图（feature map）。这个过程可以捕捉到数据的局部特征，如边缘、角点等。

激活层（Activation Layer）：激活层通常跟在卷积层之后，用于引入非线性。最常用的激活函数是ReLU（Rectified Linear Unit），它可以增加网络的表达能力，同时避免梯度消失问题。

池化层（Pooling Layer）：池化层用于降低特征图的空间维度（高度和宽度），减少参数数量和计算量，同时保持重要信息。常见的池化操作有最大池化（Max Pooling）和平均池化（Average Pooling）。

全连接层（Fully Connected Layer）：在CNN的最后几层通常会使用全连接层，它将前面层的输出展平成一维数组，然后进行全连接操作，用于分类或回归任务。

### 2.4.2 相关应用

卷积神经网络（CNN）作为深度学习的一种核心技术，已经在多个领域展现出了巨大的应用潜力和实际价值。CNN在图像识别和分类方面的应用是最为广泛和成熟的。通过学习大量的图像数据，CNN能够识别和分类各种对象，如动物、车辆、植物等。这种技术被广泛应用于数字相机的场景识别、社交媒体平台的图片分类、以及安全监控系统中的目标识别等场景。

除了识别图像中的对象，CNN还能够进行物体检测和分割。这意味着它不仅能识别出图像中的对象，还能确定对象的具体位置和轮廓。这在自动驾驶汽车的环境感知、医学图像分析（如肿瘤检测和器官分割）、卫星图像分析等领域中非常重要。

CNN也被成功应用于自然语言处理（NLP）领域。虽然传统上NLP更多地依赖于循环神经网络（RNN）和变压器模型，但CNN在处理文本数据时显示出了独特的优势，尤其是在捕捉文本中的局部特征（如短语或关键词）方面。应用包括情感分析、文本分类、机器翻译等。

在视频分析领域，CNN能够处理和分析视频数据中的时间序列信息。这包括动作识别、事件检测、视频分类等。例如，在体育赛事分析中，CNN可以识别特定的运动员动作；在安全监控中，它可以用于检测异常行为或特定事件。

CNN在医学图像分析中的应用正在快速发展。它能够帮助医生识别疾病标志，如癌症、糖尿病视网膜病变等。通过分析X光片、MRI和CT扫描等医学图像，CNN可以协助医生进行更准确的诊断。

在游戏和娱乐行业，CNN被用于增强现实（AR）和虚拟现实（VR）技术，提供更加丰富和互动的用户体验。例如，通过实时图像识别，CNN可以在AR游戏中创建交互式的游戏环境。

在零售和电子商务领域，CNN可以用于商品识别和推荐系统。通过分析用户的浏览和购买历史，CNN可以提供个性化的购物建议，增强用户体验。

无人机和机器人技术中，CNN用于导航和自主决策。无人机可以利用CNN进行实时图像分析，以避开障碍物，而机器人可以使用CNN来理解其周围的环境，进行更复杂的任务，如自动搬运和配送。

在金融服务领域，CNN被用于识别欺诈行为、分析股市趋势、自动化文档处理等。例如，通过分析交易模式，CNN可以帮助识别异常交易行为，从而预防欺诈。

# 三、基于特征子空间插值的数据合成方法

本章节主要针对RSIS方法分为四个部分进行介绍。RSIS方法主要思想是将原始特征空间划分为多个子空间，这些子空间的样本数量近乎相等，随后对子空间之间进行排序，该部分以最小化距离总和为目标将子空间插值排序问题转化为旅行商问题进行求解，最后对相邻的子空间之间的样本进行多次线性插值，通过特征子空间之间插值合成的样本具有相比原始样本更小的噪音误差，达到扩充数据、优化样本并且强化拟合效果的目的。

本章节第一部分，主要介绍了一种无监督聚类算法。该算法可以通过较少的运算达到特征空间分割的目的，得到多个具有样本量相近的特征子空间。该算法分为两个阶段，即迭代层次聚类阶段以及K近邻优化阶段。

本章节第二部分，主要解决了针对多个特征子空间的插值排序问题。以最小化距离总和为目标，将原始问题转化为旅行商问题进行求解。

本章节第三部分，主要提出了针对两个相邻的子空间之间样本的线性拟合方法。在损失函数中加入了惩罚项，在模型训练过程中考虑到了全局信息。

本章节第四部分，提出了样本匹配算法，基于加权二分图的最小权匹配问题的思想，针对两个相邻子空间之间的样本进行线性插值，通过匈牙利算法求解得到最优的样本插值匹配方案。

本章节第五部分，总结了RSIS方法的假设以及整体操作流程。

RSIS方法整体流程图如图3.1所示：

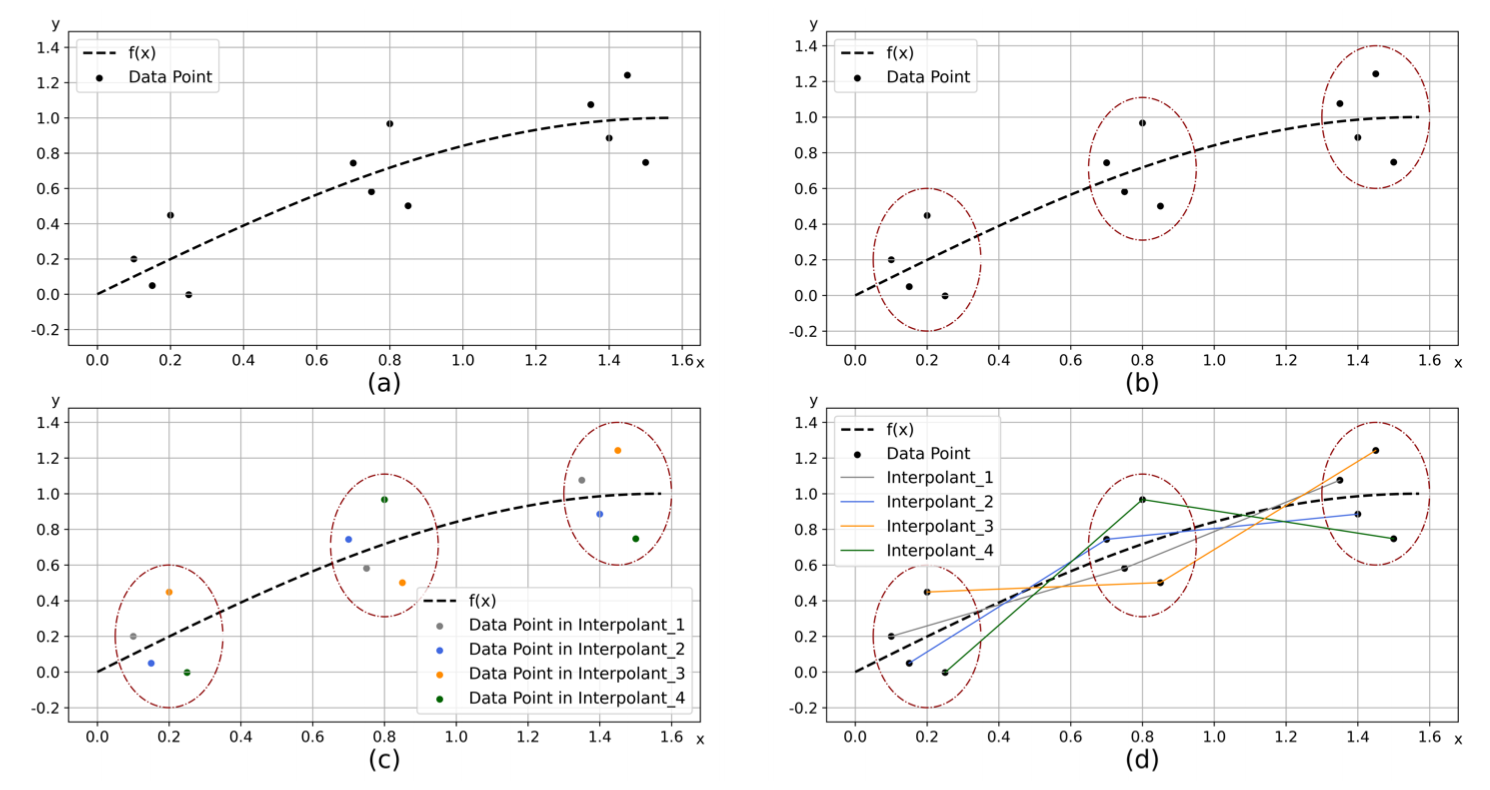


图3.1 RSIS方法整体流程图

图（a）中通过无监督聚类算法将原始特征空间划分为多个具有相近样本的特征子空间；图（b）中仅考虑了一维特征空间的情况对TSP问题进行求解，得到子空间插值排序方案；图（c）对相邻排序的子空间之间进行线性拟合；图（d）中基于线性拟合的结果，对相邻的子空间之间的样本进行线性插值，合成误差较小的样本。

## 3.1 特征子空间的划分

在RSIS方法中，首先需要对原始数据集进行聚类，本文提出了一种无监督聚类算法，该方法可以将特征空间划分为多个等量样本的子空间，该算法分为两个阶段，迭代聚类阶段以及K近邻优化阶段。

### 3.1.1 迭代层次聚类

迭代层次聚类过程的设计主要基于贪心策略的思想，通过多次循环迭代进行聚合聚类的无监督算法, 需要先给定超参数，超参数的直观解释是原始特征空间划分的子空间的数量。

首次迭代过程，给定样本特征的有序集合, 每次迭代过程中都应先对于集合中所有元素计算距离矩阵, 易得距离矩阵中的最小值 并且定义集合, 作为第一个初始的簇，根据公式（3.3）计算簇心：

， （3.1）

其中，为集合的簇心，为当前集合中的样本数量。首次迭代的最后，将从集合中剔除，, 并且更新集合.

在随后的迭代过程中，对于集合计算距离矩阵，以及距离矩阵最小值. 对于分为三种情况进行讨论：都是样本；一个是样本另一个是簇心；两个都是簇心。

在都是原始样本的情况下，定义集合生成新的簇并计算簇心, 将从集合中剔除并且将添加到集合中，即.

在是样本，是簇心的情况下，更新, 将从集合中剔除并且根据公式（3.1）重新计算并更新集合中的簇心，即. 对于是簇心，是样本的情况同理。

若，皆为簇心，则将两个簇进行合并，并且更新

每次迭代结束后，如果，则将从中剔除，并且引入LOF离群点检测[19]算法对集合中的样本进行离群点检测，剔除掉多余的样本点，将作为一个完整的簇不参与后续迭代。一方面保证每个簇的样本数量为，另一方面确保每个簇的样本在特征空间中不会过于分散以此进一步优化聚类的效果。最后将通过LOF算法检测出来的多余的样本点重新添加到集合中，. 若集合，则停止迭代。该部分具体的迭代流程详见图3.2。

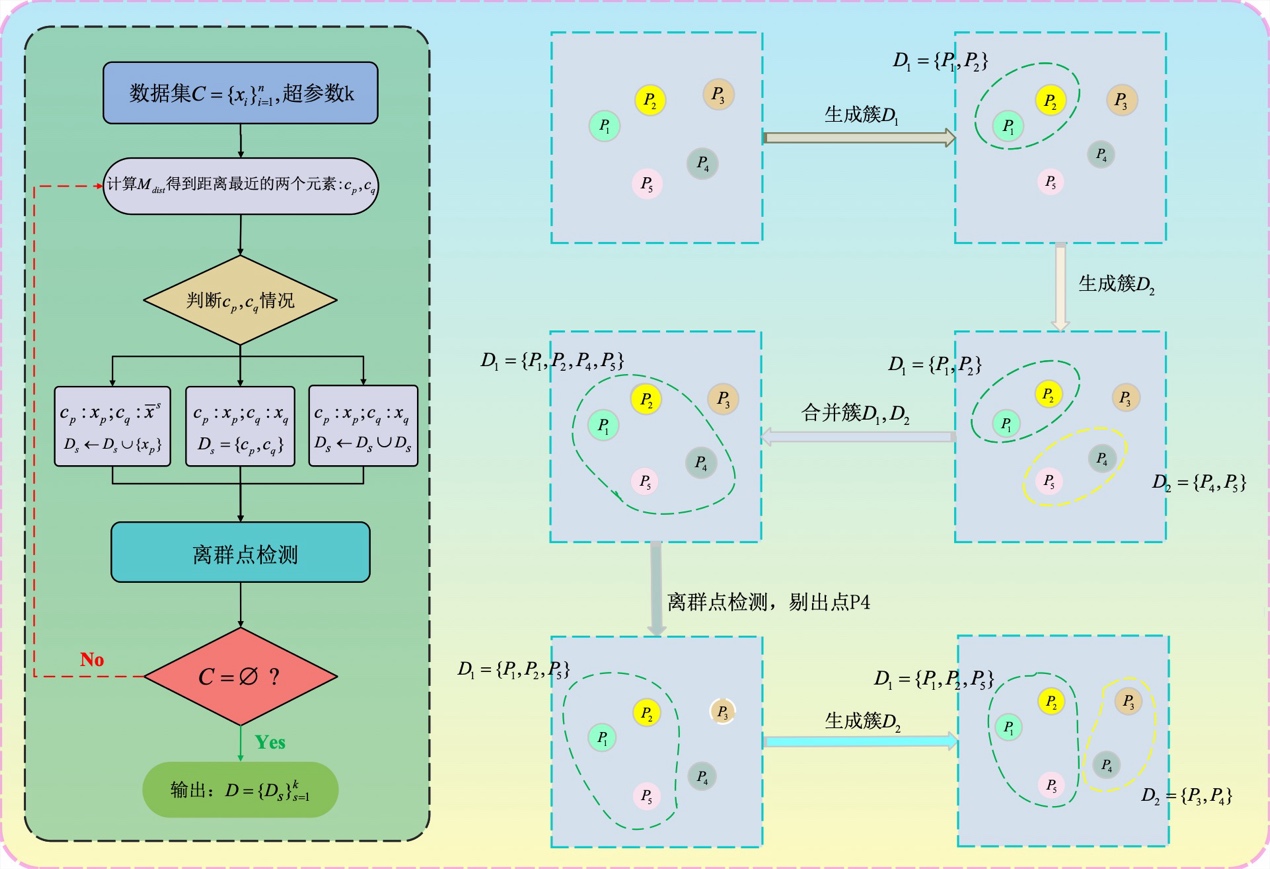


图3.2 迭代层次聚类过程

综上，该聚类过程可以将原始数据集划分为个具有等量样本和1个不超过个样本的子集，即, , 且将原始特征空间划分为个子空间.

表3.1 迭代层次聚类过程

|  |
| --- |
| 算法1: 迭代层次聚类 |
| 输入：特征数据集；超参数  输出： |
| While ：  计算的距离矩阵  得到特征空间中距离最近的两个元素  对于分为三种情况进行处理  若两个都是样本：  定义新的集合  计算簇心，并将其添加到集合中  将样本从集合中剔除  若一个是样本另一个是簇心：  将样本添加到集合中  重新计算并且更新集合中的簇心  将样本从集合中剔除  若两个都是簇心：  合并集合  重新计算并且更新集合中的簇心  将簇心从集合中剔除  如果：  通过LOF算法筛选集合中的多余样本  将多余的样本添加的集合中  将簇心从集合中剔除 |

在迭代层次聚类过程中，由于需要计算特征集合的距离矩阵并且找到距离最近的两个元素，这会大大提升算法的复杂度，这一环节可以使用K-d树（K-dimensional tree）方法进行优化。构建K-d树的时间复杂度通常是，在K-d树中删除或添加一个点以及查找最近两个元素的平均时间复杂度为，但随着迭代的进行，样本量会逐渐减少，因此实际的查找时间会略有下降。在不考虑LOF离群点检测将簇中多余的样本重新放回数据集的情况下，迭代总次数为次，因此总体时间复杂度应当会接近。

### 3.1.2 K近邻优化

K近邻（K-Nearest Neighbors, KNN）方法是一种基于实例的学习算法，它不显式地进行模型学习，而是直接使用训练数据进行预测。KNN方法在特征空间中对数据点进行分类或回归，其核心思想是在特征空间中找到与新样本最接近的K个已知数据点（即“邻居”），并根据这些邻居的信息来预测新样本的输出。

由于迭代层次聚类的过程是基于贪心策略的思想，这就会导致在迭代后期的聚类的可能存在不合理的情况，影响整体的聚类效果，因此本文设计了K近邻优化过程对聚类结果进行微调，一些样本的簇分配可能会改变，但整体上保持了簇的均匀性，以提高聚类效果。

本文给定KNN方法近邻数量的超参数为. 通过上述的迭代层次聚类过程，可以得到一个初始的聚类结果，对于所有样本易求得特征空间中距离最近的个样本，随后将每个样本所属的簇更新为这近邻的个样本中占比最多的簇（如图3.3所示）。需要注意的是，在这一过程中，会变更小部分样本所属的簇，导致每个簇的样本数量有小幅度变动，样本较为分散的簇可能会消失，但整体上会保证大多数簇的样本数量近乎相等。通过以上的迭代层次聚类以及K近邻优化两个阶段，可得到聚类结果, 其中，是K近邻优化后簇的数量。



图3.3 近邻优化流程图

总的来说，K近邻优化是一个用于改进迭代层次聚类结果的过程，它通过找到每个样本的最近邻居并将样本重新分配到占比最多的簇，以实现聚类结果的微调，从而在保证了簇的均匀性的同时提高了聚类效果。

## 3.2 子空间插值路径排序

通过对原始数据进行聚类可以得到多个子集，每个子集对应不同的特征子空间。RSIS方法是对两个相邻的子空间之间的样本进行插值合成数据，该部分主要针对多个子空间插值路径的设计进行分析。

插值路径的设计思想是从一个子空间出发，经过一次所有的子空间并最终返回起始点，目标是最小化总路径距离。可以转换为旅行商问题（Travelling Salesman Problem, TSP）进行求解，将该问题表示为一个加权图，其中每个子空间样本的簇心可以表示为顶点，簇心之间的直线路径可以表示为加权图的边，路径的距离可以表示为边的权重。通过上文的聚类算法可得到一个初始的插值路径有序集合，设定目标函数：

（3.2）

旅行商问题是组合优化和理论计算机科学中的一个经典问题，也是一个典型的NP-hard问题，这意味着没有已知的多项式时间复杂度算法可以解决所有TSP实例，随着特征图顶点数量的增加，求解问题所需的时间和资源以指数级增长。在实际应用中，TSP被用来解决物流、规划、芯片制造等领域的问题[20–23]，但是直接解决TSP是非常复杂的，许多研究设计了多种启发式算法和近似算法来找到足够好的解[24–26]，虽然这些解不一定是最优的，这些方法通过不同的策略在可接受的时间内寻找到一个近似最短路径。为了保证能在可接受的时间内得到足够好的解决方案，本文首先使用贪婪算法快速找到一个初始解，然后用3-opt方法来优化并得到最终解。最终可以得到一个新的排序好的集合. 在本文的后续部分，本文将对于序号相邻的两个子集所对应的特征子空间定义为相邻特征子空间。

以上是针对多维度特征空间的情况，若特征空间是一维的，考虑到一维空间的特殊性，插值路径的设计依旧要求经过一次所有的子空间，但不需要重新回到起点，可以直接使用贪心策略进行求解，在一维情况下，这种方法得到的解一定是最优解。首先，找到插值路径的起点子空间：

（3.3）

其中，是特征集合的最小值。对于后续簇的定义如下：

（3.4）

除非另有特别说明，本文后续的研究主要集中在多维特征空间情况下进行探讨。

## 3.3 相邻子空间线性回归拟合

给定含噪数据集, 其中 假设, 其中为的去噪真实值, 为的真实值, RSIS方法假设关系式为一个连续的函数, 即对于任意的, 任意的, 存在使得, 当且时, 总有.可得：

（3.5）

其中，为中所含的噪音，为中所含的噪音，为等式误差项。公式（3.5）可转化为：

（3.6）

通过上述的聚类以及子空间排序算法，可以将数据集划分为多个有序子集, , 以及将特征空间划分为多个子空间. 对于两个相邻的特征子空间，由于RSIS方法假设关系式为一个连续的函数,则可以将近似拟合为一个线性的函数，公式（3.3）可转化为：

, (3.6)

其中，为相邻特征子空间与中线性拟合的误差项，。若两个子空间之间的距离足够近并且子空间的测度趋向于0时，则线性拟合误差.

可以考虑使用OLS、Lasso回归或者局部加权线性回归等方法[27,28]对两个相邻子空间之间的进行估计，但是考虑到RSIS的假设为一个连续的函数，仅仅根据两个子空间的样本直接进行线性拟合并不能充分考虑到全局的信息。本文借鉴了软参数多任务学习框架的思想[29–32]，同时对多个子空间之间的线性函数进行估计，并且在损失函数的设计添加了L1正则项，以此使得每个线性函数的估计都考虑了全局性信息：

（3.7）

其中，为参数，为正则项系数，根据子空间之间的距离自适应调整，在后文的模拟实验以及实例分析中定义=1。

根据**Theorem 1.**可知，是一个凸函数，因此结果必然收敛且局部最优解就是全局最优解。

## 3.4 多阶段最小权匹配

接下来需要对相邻的两个子空间的样本进行线性插值合成新的样本，插值的过程是在两个样本之中进行的，RSIS方法生成的数据要充分考虑到所有的样本信息，插值规则设计如下：

1. 插值的两个样本应当来自不同的子空间；

2. 每个样本至少进行一次插值；

3. 插值次数为；

4. 所有样本参与插值的次数必须是均匀的，即每个样本最多插值次。

需要给定另外一个超参数，该超参数直观解释是单位距离插入的样本数量,取值的大小会直接影响到RSIS方法合成样本的数量。对于两个来自相邻且不同子空间的样本, 合成的样本为. RSIS方法假设是样本变量是连续的变量，插值公式如下：

（3.8）

（3.9）

插值公式（3.8），（3.9）设计的核心思想是自适应的根据样本在特征空间的距离去控制生成样本的数量，并且插入的样本都是线性且在两个样本中都是等距的，由于在RSIS方法中假设所有变量都是连续的，因此插值公式的涉及并不考虑变量出现间断或者跳跃的情况。

在实际应用中，对于的取值应当考虑到特征空间的维度以及每个维度的量纲，因为在高纬空间中，样本之间的距离会呈指数级增加，如果值设置过大，会导致合成的样本数量增多，不仅会占用大量内存，造成样本信息的冗余，也会增加没有必要的计算。

由于数据集包含噪音，通过插值合成的样本会受到原始样本噪音的影响也会包含噪音。不失一般性，假设, 根据RSIS方法的插值规则，一共存在 种插值匹配策略可供选择。同样不失一般性，不妨假设特征空间的维度，*,*可以通过公式（3.8）计算所有样本插值后的总体误差：

， （3.10）

其中，是经过两点的线性函数，是在趋向于无穷大时对进行插值合成样本的总误差。进一步的，通过公式（3.10）可以求得合成样本的平均误差，见公式（3.11）.

， （3.11）

如果用穷举法计算所有策略的平均误差，由于公式（3.10），（3.11）涉及到复杂的积分运算，并且如果子空间样本数量过多，这会消耗大量的计算资源，若特征空间的维度较高，则需要的计算资源会更大。因此，如何仅仅耗费少量的计算资源的情况下快速且高效选择一种较优的插值匹配策略是需要RSIS方法重点探讨的问题。

**Theorem 2.** 对于两个相邻的子空间与对应的子集,. 考虑关系式。假设线性拟合误差，则有.

*Proof of Theorem 2.* 由于假设线性拟合误差，根据公式（3.6）可得：

不妨假设. 由于为线性函数，根据公式（3.10）以及（3.11）可得：

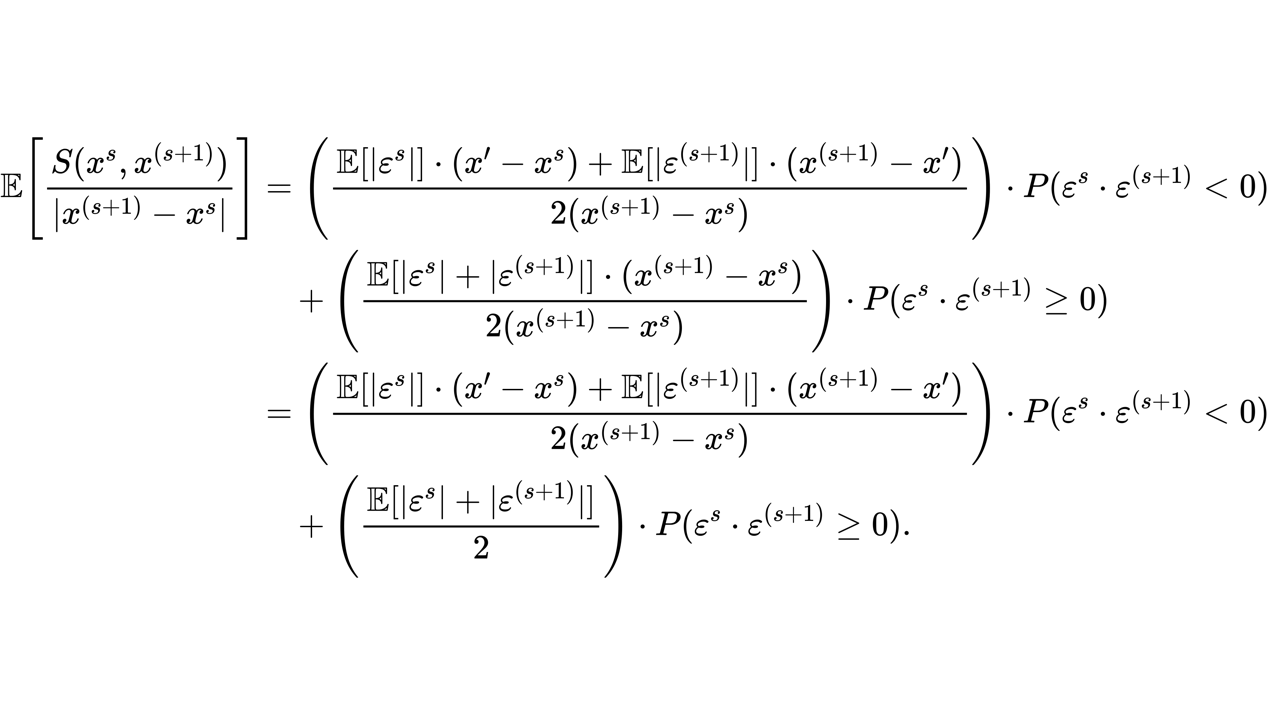
当时，, 使得.

根据迭代期望定律(LIE)可得:

可根据基础的几何面积计算来简化，当时，, 使得. 可得：

当时：

带入原式可得：



由于 并且：

可得

通过**Theorem 2.**可知，在假设线性拟合误差的条件下，即使面对噪音不同分布的情况，对任意两个子空间的样本进行线性插值，则插值合成数据的平均噪音期望小于这两个样本的平均噪音期望。通过**Theorem 3**可知，造成这种结果的主要是因为对于两个样本的噪音，若, 则有，若，则有.

**Theorem 3.** 令 假设线性拟合误差，则 .

*Proof of Theorem 3.* 当时，由于线性拟合误差,可得：

易得：

.

通过以上证明不难看出，在满足假设条件的情况下，不论两个样本之间插值数量多少，总有：

.

可以根据子空间之间线性拟合函数估计样本的噪音，通过公式（3.6）可得：

. (3.12)

求得子空间之间的样本误差，下一步只需要考虑如何根据样本的误差信息去确定样本之间的匹配策略，因为不同误差的样本之间进行插值合成的数据噪音也会不同，可以转换为二分图最小权匹配问题进行分析。近些年来，最小权匹配问题在物流、网络设计、资源分配等领域有广泛应用[33,33,34]；也有部分研究将最小权匹配问题扩展到更复杂的网络中，例如动态网络或随机网络，研究如何在变化的环境中找到最优解[35–37]；对于那些计算上不可行求解精确结果的大型问题，一些研究可能会寻找能够快速得到近似解的算法[38–40]。

根据插值规则，需要对每个样本至少进行一次插值，并且约束了插值的总次数。经过K近邻优化后子空间之间的样本数量大多数情况并不是完全相等的，所以这不能视作一个最小权完美匹配问题。这个问题是一个加权匹配问题的变体，同时融合了平衡匹配和最小权匹配的特点，因为它需要最小化匹配的总权重，并且约束了样本匹配次数的均匀性。目前针对这种问题并没有一个现成的标准算法，因此需要对传统的最小权匹配问题进行改进以适应此目标场景。

依旧延续上文的假设, 在这种情况下，对于集合的每个考虑进行一次或者两次匹配，本文提出了一种启发式的多阶段的最小权匹配方法对该问题进行分析。需要注意的是，这种方法并不能保重所得结果是一个最优解，但是但它可以在极短的时间内得到一个合理的近似解。这种方法的优点在于它相对简单，并且对于大规模问题来说可以提供较好的效率，如果需要找到最优解，可能需要使用更复杂的优化方法，例如整数线性规划（ILP）来精确求解问题，但通常需要更多的计算资源和时间。

对于一个加权二分图，和一个权重函数, 其中是两侧顶点的集合，是边的集合，表示可能的匹配。由于，定义权重函数. 考虑目标函数：

, （3.13）

s.t. ，

*,*

其中为二元决策变量。

在第一个阶段中，在不考虑目标函数（3.13）约束条件的情况下，使用匈牙利算法直接对二分图进行求解，最终可得二分图的一个最大匹配. 此时对于每个都参与了一次匹配，中个顶点参与了一次匹配，还有个顶点并未参与匹配，对于未参与匹配的顶点定义为.

在第二个阶段中，定义一个新的加权二分图，，依旧使用匈牙利算法对二分图求解，并得到最大匹配. 由于，因此在中对于每个顶点都参与了一次匹配，中个顶点再次参与了一次匹配。最终将与合并得到。是二分图中的一个匹配，使得中的每个顶点参与一次或两次匹配，且而中的每个顶点仅参与一次匹配，满足约束要求。

以上是对特定的假设情况介绍的二阶段最小权匹配方法，这也是在子空间之间样本匹配中最常见的一种情况。若两个子空间样本数量差距过大，这个问题则需要分为个最小权匹配阶段进行求解（见表3.2）。对相邻的子空间之间通过多阶段最小权匹配算法求得样本插值匹配策略，随后根据公式（3.8），（3.9）进行插值合成新的具有更小噪音的样本，RSIS方法整体完成。

表3.2 多阶段最小权匹配

|  |
| --- |
| 算法2: 多阶段最小权匹配 |
| 输入：顶点集；  线性回归函数.  输出：插值匹配 |
| 估计样本的噪音  定义二分图  定义权重函数  For :  使用匈牙利算法对二分图求解得  更新, 其中为中已匹配的的顶点集  更新二分图 |

## 3.5 方法整体分析

RSIS方法共有三个假设：为一个连续的函数；线性拟合误差；所考虑的样本中的变量是连续的。该方法分为多个步骤，整体操作流程以及特征空间一维情况下的直观效果图见表3.3以及图3.5。

表3.3 RSIS方法

|  |
| --- |
| 算法3: RSIS方法 |
| 输入：数据集  超参数  输出：插值合成后的数据集 |
| 构建特征数据集  通过聚类算法对样本进行聚类得到  构建加权图转换使用贪心算法得到一个初始插值路径解  使用3-opt方法来优化并得到最终解  对子空间之间进行线性拟合得到  For :  估计中样本的噪音  利用多阶段最小权匹配方法得到插值匹配策略  根据匹配策略对样本进行插值合成新的数据  将新合成的数据添加到数据集中 |

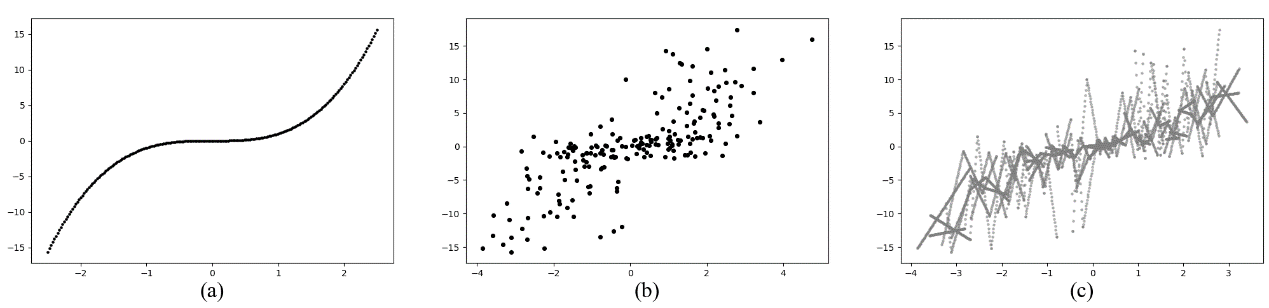


图3.5 一维特征空间下RSIS方法处理效果图

如上图所示，生成实际函数关系为的样本，样本量为200. 随后对样本添加高斯噪声。令k=6，η=100，对噪音数据使用RSIS方法进行数据合成处理，样本量增加至3808。

# 四、模拟实验

## 4.1 模拟设计

## 4.2 噪音处理效果分析

## 4.3 超参数敏感性分析

## 4.4 对于样本分布的影响分析

# 五、实例分析

使用MLP模型，给定n个测试数据集，n个折线图2条线，分别是每个数据集使用RSIS方法前后的收敛效果。

# 六、结合卷积神经网络框架的进一步研究

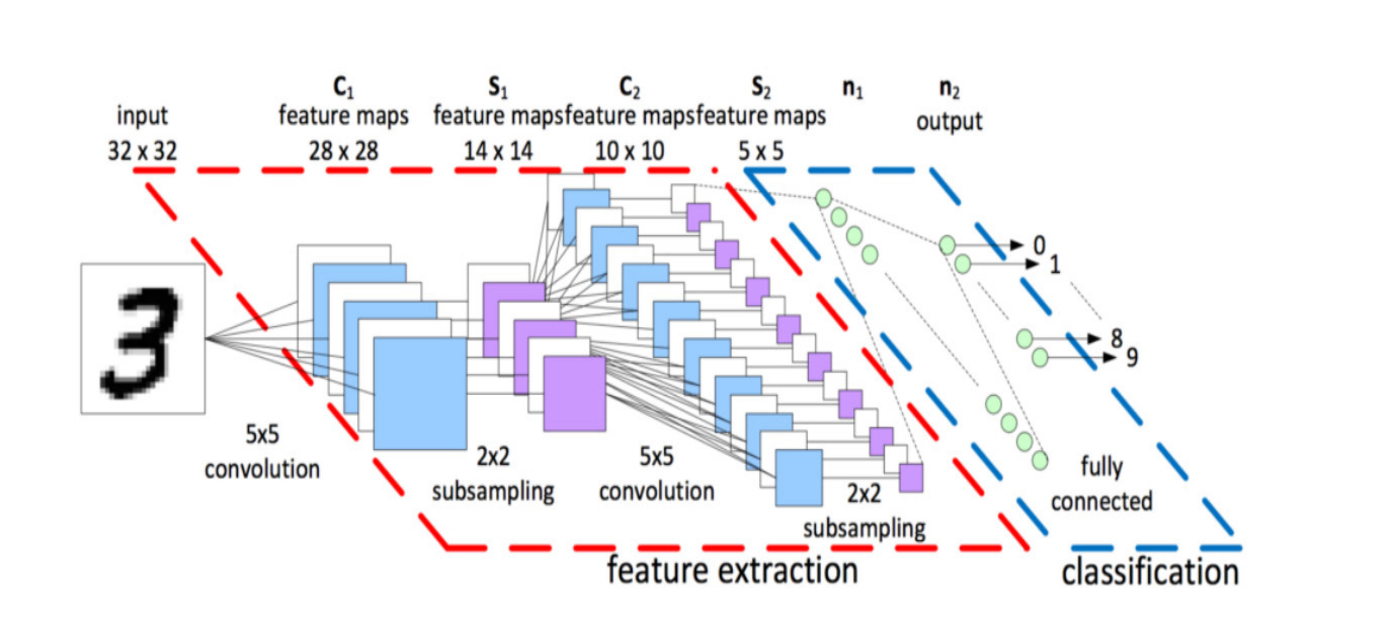
随着深度学习技术的发展，自从CNN在图像识别和分类方面取得重大突破以来，从最初的LeNet到更复杂的架构如AlexNet、VGG、ResNet等，CNN已成为处理图像数据的标准工具，被广泛应用于各种视觉任务，包括对象检测、图像分割、实时视频处理等。这些应用不仅展示了CNN在处理复杂视觉信息方面的强大能力，也推动了从自动驾驶到医疗诊断等多个领域的技术发展。

上文的研究主要针对特征矢量数据使用RSIS进行数据合成，本章节重点探究RSIS方法结合CNN框架在计算机视觉领域的应用研究。

## 6.1方法概述

RSIS方法针对向量数据进行数据合成，但是无法直接用于图像数据，因此RSIS方法存在一定的局限性。CNN作为优秀的特征提取模型，本文将RSIS方法与传统的深度学习CNN框架相结合，提出了一种创新的CNN框架，旨在增强图像数据的特征表示和模型泛化能力。该框架首先利用CNN的卷积和池化层对输入图像进行深度特征提取，然后通过全局池化层将这些特征转换为一维向量表示。在此基础上，应用RSIS方法来生成新的向量表示。见图6.1。

具体来说，对于一组RGB图像数据集合，其中为图像数值化的张量数据。设CNN的特征提取函数，其中表示图像的尺寸以及通道数，表示提取特征的维度。通过CNN的特征提取操作，图像被转换为一维的特征向量. 将集合使用RSIS方法进行数据合成得到，其中为添加了新的合成数据后的特征向量集的样本量。最后将所有样本放入全连接网络中进行迭代训练。训练好的模型可以用于多种应用，如图像分类、对象识别或其他相关的计算机视觉任务。该方法的关键优势在于通过数据合成增强了数据集的代表性和多样性，从而提高了模型在处理新场景和变化时的泛化能力和鲁棒性。



向下的箭头

f e+RSIS+Class

图6.1 框架结构图

## 6.2 训练策略的设计

将RSIS方法融入CNN框架中，传统的模型训练策略并不适用，本文设计了一种契合该框架的模型训练策略，分为两个阶段（见图6.2）：

在第一个阶段中，与传统的训练方式一样，依旧将图像进行数据增强处理，随后放入模型中进行前向传播计算损失函数。最后对损失函数求梯度进行反向传播操作，更新模型参数，直至模型收敛为止。

在第二阶段中，将原始图像放入第一阶段训练好的特征提取器中得到每个图像的特征向量数据集，对特征向量数据集使用RSIS方法合成新的数据，并将新的数据再次放入全连接层中对其进行训练直到收敛。需要注意的是，这一阶段仅仅只针对全连接层的参数进行更新，特征提取层并不参与训练过程。

## 6.2 实例分析

一个噪音数据集，手写字体，彩图，彩图的灰度图

resnet vgg 两个轻量网络

迭代过程图：仅针对一次实验的情况，共四个图，一个数据集，四个网络，三个样本量

预测效果表：原定不变，但是训练流程变动一下

第一阶段模型M1

第二阶段模型M2

对于验证集数据I

占比图，

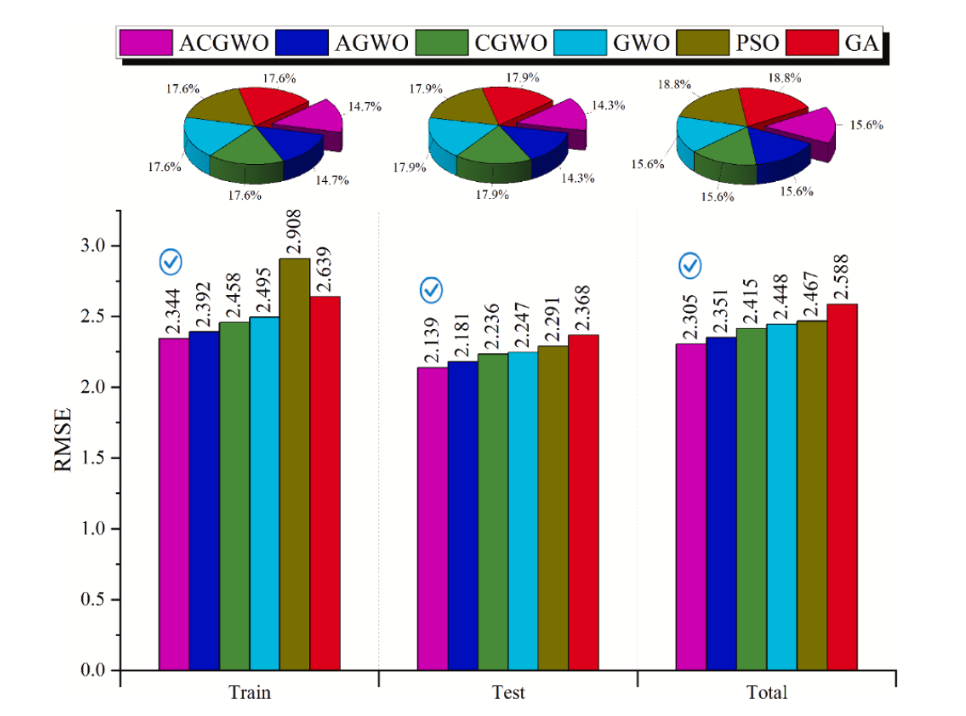
原始模型对，改进模型错 M1（I） is right， but M2（I）wrong :P1

原始模型错，改进模型对 M2（I） is right， but M1（I）wrong :P2

都错 P3

都对 P4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Resnet | Vgg |  |  |
| 噪音数据集 | 一个饼图 |  |  |  |
|  |  |  |
|  |  |  |



# 七、总结与展望

# 参考文献

[1] ALIZADEHSANI R, ROSHANZAMIR M, HUSSAIN S, 等. Handling of uncertainty in medical data using machine learning and probability theory techniques: a review of 30 years (1991–2020)[J/OL]. Annals of Operations Research, 2020. DOI:10.1007/s10479-021-04006-2.

[2] GUO Y, WANG W, WANG X. A Robust Linear Regression Feature Selection Method for Data Sets With Unknown Noise[J/OL]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(1): 31-44. DOI:10.1109/TKDE.2021.3076891.

[3] JESMEEN M, HOSSEN J, SAYEED S, 等. A survey on cleaning dirty data using machine learning paradigm for big data analytics[J]. Indonesian Journal of Electrical Engineering and Computer Science, 2018, 10(3): 1234-1243.

[4] SAEZ J A, LUENGO J, HERRERA F. Dealing with noise problem in machine learning data-sets: A systematic review[J]. Procedia Computer Science, 2019, 159: 244-253.

[5] FRÉNAY B, KABÁN A. Classification in the presence of label noise: a survey[J]. IEEE Transactions on Neural Networks and Learning Systems, 2013, 25(5): 845-869.

[6] YI K, WU J. Probabilistic end-to-end noise correction for learning with noisy labels[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 7017-7025.

[7] EVANS R, GREFENSTETTE E. Learning explanatory rules from noisy data[J]. Journal of Artificial Intelligence Research, 2018, 61: 1-64.

[8] WU Z, RINCON D, LUO J, 等. Improving Machine Learning Modeling of Nonlinear Processes Under Noisy Data Via Co-teaching Method[C]//2021 American Control Conference (ACC). IEEE, 2021: 4660-4666.

[9] WU Z, RINCON D, LUO J, 等. Handling noisy data in machine learning modeling and predictive control of nonlinear processes[C]//2021 American Control Conference (ACC). IEEE, 2021: 3345-3351.

[10] KIM Y, YIM J, YUN J Y, 等. NLNL: Negative Learning for Noisy Labels[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 101-110.

[11] OLEGHE O. A predictive noise correction methodology for manufacturing process datasets[J]. Journal of Big Data, 2020, 7(1): 1-27.

[12] HAN J, LUO P, WANG X. Deep self-learning from noisy labels[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 5138-5147.

[13] BHATIA A, CHUG A, SINGH A P, 等. A hybrid approach for noise reduction-based optimal classifier using genetic algorithm: A case study in plant disease prediction[J]. Intelligent Data Analysis, 2022, 26(4): 1023-1049.

[14] QAISAR S M, DALLET D. ECG noise removal and efficient arrhythmia identification based on effective signal-piloted processing and machine learning[C]//2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC). IEEE, 2021: 1-6.

[15] LAPINS S, BUTCHER A, KENDALL J, 等. DAS-N2N: Machine learning Distributed Acoustic Sensing (DAS) signal denoising without clean data[J]. arXiv preprint arXiv:2304.08120, 2023.

[16] PARIZAD A, HATZIADONIU C J. Cyber-attack detection using principal component analysis and noisy clustering algorithms: A collaborative machine learning-based framework[J]. IEEE Transactions on Smart Grid, 2022, 13(6): 4848-4861.

[17] MUNDRA S, VIJAY S, MUNDRA A, 等. Classification of imbalanced medical data: An empirical study of machine learning approaches[J]. Journal of Intelligent & Fuzzy Systems, 2022, 43(2): 1933-1946.

[18] ZOR M, BOSTANCI E, GÜZEL M S, 等. Analysis of Interpolation-Based Image In-Painting Approaches[M]//Advanced Sensing in Image Processing and IoT. CRC Press, 2022: 153-170.

[19] BREUNIG M M, KRIEGEL H P, NG R T, 等. LOF: identifying density-based local outliers[C]//Proceedings of the 2000 ACM SIGMOD international conference on Management of data. 2000: 93-104.

[20] DÜNDAR H, SOYSAL M, ÖMÜRGÖNÜLŞEN M, 等. A green dynamic TSP with detailed road gradient dependent fuel consumption estimation[J/OL]. Computers & Industrial Engineering, 2022, 168: 108024. DOI:10.1016/j.cie.2022.108024.

[21] GHEYSARI K, KHOEI A, MASHOUFI B. High speed ant colony optimization CMOS chip[J/OL]. Expert Systems with Applications, 2011, 38(4): 3632-3639. DOI:10.1016/j.eswa.2010.09.017.

[22] YU Y, LIAN F, YANG Z. Pricing of parcel locker service in urban logistics by a TSP model of last-mile delivery[J/OL]. Transport Policy, 2021, 114: 206-214. DOI:10.1016/j.tranpol.2021.10.002.

[23] AKHAND M A H, AYON S I, SHAHRIYAR S A, 等. Discrete Spider Monkey Optimization for Travelling Salesman Problem[J/OL]. Applied Soft Computing, 2020, 86: 105887. DOI:10.1016/j.asoc.2019.105887.

[24] WANG C, MA B, SUN J. A co-evolutionary genetic algorithm with knowledge transfer for multi-objective capacitated vehicle routing problems[J/OL]. Applied Soft Computing, 2023, 148: 110913. DOI:10.1016/j.asoc.2023.110913.

[25] TOAZA B, ESZTERGÁR-KISS D. A review of metaheuristic algorithms for solving TSP-based scheduling optimization problems[J/OL]. Applied Soft Computing, 2023, 148: 110908. DOI:10.1016/j.asoc.2023.110908.

[26] MICHALAK K. Feasibility-Preserving Genetic Operators for Hybrid Algorithms using TSP solvers for the Inventory Routing Problem[J/OL]. Procedia Computer Science, 2021, 192: 1451-1460. DOI:10.1016/j.procs.2021.08.149.

[27] LIU C, LI B, VOROBEYCHIK Y, 等. Robust linear regression against training data poisoning[C]//Proceedings of the 10th ACM workshop on artificial intelligence and security. 2017: 91-102.

[28] CLEVELAND W S. Robust locally weighted regression and smoothing scatterplots[J]. Journal of the American statistical association, 1979, 74(368): 829-836.

[29] LEE H B, YANG E, HWANG S J. Deep asymmetric multi-task feature learning[C]//International Conference on Machine Learning. PMLR, 2018: 2956-2964.

[30] MRINI K, DERNONCOURT F, YOON S, 等. A gradually soft multi-task and data-augmented approach to medical question understanding[C]//Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2021: 1505-1515.

[31] DUONG L, COHN T, BIRD S, 等. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser[C]//Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers). 2015: 845-850.

[32] MISRA I, SHRIVASTAVA A, GUPTA A, 等. Cross-stitch networks for multi-task learning[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 3994-4003.

[33] PANDA S K, NANDA S S, BHOI S K. A pair-based task scheduling algorithm for cloud computing environment[J/OL]. Journal of King Saud University - Computer and Information Sciences, 2022, 34(1): 1434-1445. DOI:10.1016/j.jksuci.2018.10.001.

[34] SharkID A Framework for Automated Individual Shark Identification.pdf[Z].

[35] ZUKHRUF F, BALIJEPALLI C, FRAZILA R B, 等. Algorithms for restoring disaster-struck seaport operations considering interdependencies between infrastructure availability and repair team assignments[J/OL]. Computers & Industrial Engineering, 2023, 175: 108894. DOI:10.1016/j.cie.2022.108894.

[36] DRENT C, KEIZER M O, HOUTUM G J V. Dynamic dispatching and repositioning policies for fast-response service networks[J/OL]. European Journal of Operational Research, 2020, 285(2): 583-598. DOI:10.1016/j.ejor.2020.02.014.

[37] ENAYATI S, MAYORGA M E, RAJAGOPALAN H K, 等. Real-time ambulance redeployment approach to improve service coverage with fair and restricted workload for EMS providers[J/OL]. Omega, 2018, 79: 67-80. DOI:10.1016/j.omega.2017.08.001.

[38] AKBARZADEH B, MAENHOUT B. A decomposition-based heuristic procedure for the Medical Student Scheduling problem[J/OL]. European Journal of Operational Research, 2021, 288(1): 63-79. DOI:10.1016/j.ejor.2020.05.042.

[39] ANGRIMAN E, MEYERHENKE H, SCHULZ C, 等. Fully-dynamic Weighted Matching Approximation in Practice[M/OL]. arXiv, 2021[2024-01-23]. http://arxiv.org/abs/2104.13098.

[40] DUAN R, PETTIE S. Linear-Time Approximation for Maximum Weight Matching[J/OL]. Journal of the ACM, 2014, 61(1): 1-23. DOI:10.1145/2529989.

# 附录

附录1 RSIS方法代码实现

|  |
| --- |
| import warnings  # 忽略所有警告  warnings.filterwarnings('ignore')  import torch  from torchvision import datasets, transforms  import torchvision.models as models  import torchvision.transforms as transforms  from PIL import Image  from tqdm import tqdm  import pandas as pd  import numpy as np  import torch  from torch import nn  from tqdm import tqdm  import matplotlib.pyplot as plt  import itertools  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.svm import SVR  from sklearn.metrics import mean\_squared\_error  from sklearn.metrics import mean\_absolute\_error  from torch.utils.data import TensorDataset, DataLoader  from collections import OrderedDict  from sklearn import preprocessing  import math  from sklearn.neighbors import LocalOutlierFactor  from sklearn.linear\_model import LinearRegression  from scipy.optimize import linear\_sum\_assignment  from scipy.spatial import KDTree  from sklearn.metrics import accuracy\_score  from sklearn.neighbors import KNeighborsClassifier  from scipy.spatial import distance\_matrix  from tsp\_solver.greedy import solve\_tsp  from sklearn.preprocessing import OneHotEncoder  def efficient\_min\_distance\_method(samples,p):  tree = KDTree(samples)  min\_distance = np.inf  closest\_pair = (0, 0)  for i in range(samples.shape[0]):  distances, indices = tree.query(samples[i], k=2, p=p) # k=2 返回最近的两个点（包括自身）  if distances[1] < min\_distance: # distances[1] 是第二近的点（即除了自身外最近的点）  min\_distance = distances[1]  closest\_pair = (i, indices[1])  return closest\_pair  def K\_Space(cluster\_nums, samples, p=2):  n = samples.shape[0]  num\_each\_cluster = n // cluster\_nums  clusters = [[] for \_ in range(cluster\_nums+1)]  cluster\_centers = []  samples\_remaining = samples.copy()  add\_cluster\_index = 0  clusters\_ = []  time = 0  while samples\_remaining.shape[0] + len(cluster\_centers)>1:  #打印每次迭代的效果    # print(f"第{time}次循环 samples\_reminding:{samples\_remaining.shape[0]} 临时类数量:{len(cluster\_centers)} 产出类数量:{len(clusters\_)}")    matrix = np.vstack([samples\_remaining,[i for i in cluster\_centers]]) if len(cluster\_centers)>0 else samples\_remaining  #首先计算合并的矩阵中距离最近的两个元素  index1, index2 = efficient\_min\_distance\_method(matrix,p=p)  #求得两个元素是样本还是簇心  label1 = 0 if index1 < samples\_remaining.shape[0] else 1 # 0表示样本,1表示簇心  label2 = 0 if index2 < samples\_remaining.shape[0] else 1    #样本点之间的合并  if add\_cluster\_index < len(clusters) and label1+label2 == 0:  clusters[add\_cluster\_index].extend([samples\_remaining[index1], samples\_remaining[index2]])  cluster\_centers.append(np.mean(clusters[add\_cluster\_index], axis=0))  samples\_remaining = np.delete(samples\_remaining, [index1, index2], axis=0)  add\_cluster\_index += 1  # print(f"样本合并创建新类 ")    #当簇的数量足够时,如果还是两个样本,则将两个样本作为一个整体加入距离簇心最近的类中  elif add\_cluster\_index == len(clusters) and label1+label2 == 0:  two\_samples\_mean = np.mean([samples\_remaining[index1], samples\_remaining[index2]],axis=0)  # print(cluster\_centers)  # print(clusters)  # print(clusters\_)  if add\_cluster\_index == 0 and len(clusters) == 0:  clusters.append([])  clusters.append([])  clusters[add\_cluster\_index].extend([samples\_remaining[index1], samples\_remaining[index2]])  cluster\_centers.append(np.mean(clusters[add\_cluster\_index], axis=0))  samples\_remaining = np.delete(samples\_remaining, [index1, index2], axis=0)  add\_cluster\_index += 1  # print(f"类数已充足,增加类数")  else:  cluster\_centers\_matrix = np.vstack([i for i in cluster\_centers])  tree = KDTree(cluster\_centers\_matrix)  \_, indices = tree.query(two\_samples\_mean, k=1, p=p)  clusters[indices].extend([samples\_remaining[index1], samples\_remaining[index2]])  cluster\_centers[indices] = np.mean(clusters[indices], axis=0)  samples\_remaining = np.delete(samples\_remaining, [index1,index2], axis=0)  # print(f"两个样本归纳新类")    #样本点添加到类中  elif label1+label2 == 1:  index\_samples = index1 if label1 == 0 else index2  index\_cluster = index1-samples\_remaining.shape[0] if label1 == 1 else index2-samples\_remaining.shape[0]  clusters[index\_cluster].append(samples\_remaining[index\_samples])  cluster\_centers[index\_cluster] = np.mean(clusters[index\_cluster], axis=0)  samples\_remaining = np.delete(samples\_remaining, index\_samples, axis=0)  # print(f"一个样本归纳新类")    #类与类之间的合并  else:  index1 = index1 - samples\_remaining.shape[0]  index2 = index2 - samples\_remaining.shape[0]  if len(clusters[index1])+len(clusters[index2]) <= num\_each\_cluster:#如果两个类中样本合并的数量不足饱和  clusters[index1] = clusters[index1]+clusters[index2]  cluster\_centers[index1] = np.mean(clusters[index1], axis=0)  del clusters[index2]  del cluster\_centers[index2]  add\_cluster\_index -= 1  #使用lof算法剔除多余样本点  else:  points = np.vstack([clusters[index1]+clusters[index2]])  del clusters[max(index1,index2)]  del clusters[min(index1,index2)]  del cluster\_centers[max(index1,index2)]  del cluster\_centers[min(index1,index2)]  lof\_model = LocalOutlierFactor(n\_neighbors=int(num\_each\_cluster/2),p=p)  lof\_model.fit\_predict(points)  score = abs(lof\_model.negative\_outlier\_factor\_)  clusters\_.append(list(points[np.argsort(score)[:num\_each\_cluster]]))  samples\_remaining = np.vstack([samples\_remaining,points[np.argsort(score)[num\_each\_cluster:]]])  add\_cluster\_index -= 2  clusters.append([])  # print(f"两个类合并")    time += 1  #剔除已经饱和的簇  for i in range(add\_cluster\_index):  if len(clusters[i]) >= num\_each\_cluster:  clusters\_.append(clusters[i])  del clusters[i]  del cluster\_centers[i]  add\_cluster\_index -= 1  break  #将最后剩余的样本点归纳为一个类  if samples\_remaining.shape[0] + len(cluster\_centers) == 1:  if len(clusters) == 0:  break  clusters\_.append(clusters[0])    return clusters\_  #计算聚类效果  def calculate\_indicator(samples,label):  clusters\_center = []  for i in range(np.max(label+1)):  center = np.mean(samples[np.where(label==i)],axis=0)  clusters\_center.append(center)  clusters\_center = np.array(clusters\_center)  m = 0  for i in range(samples.shape[0]):  distance = np.linalg.norm(samples[i,:]-clusters\_center,axis=1)  if np.argmin(distance) != label[i]:  m += 1    return m/samples.shape[0]    def list\_transform\_array(clusters):  x\_train = []  y\_train = []  max\_depth = int(len(clusters)/3)  #将原本聚类结果完全转化为列表,导出label  for i in range(len(clusters)):  for j in range(len(clusters[i])):  x\_train.append(clusters[i][j])  y\_train.append(i)  #将列表格式的数据转化为数组  x\_train = np.vstack(x\_train)  y\_train = np.array(y\_train)  return x\_train,y\_train    def fine\_turning(x\_train,y\_train):  n,m = x\_train.shape  nums\_class = np.max(y\_train)+1  #计算优化前的指标  indicator = calculate\_indicator(x\_train,y\_train)  # print(f"优化前聚类效果指标:{indicator}")  #选择机器学习模型进行预测,调整聚类结果  knn = KNeighborsClassifier(n\_neighbors=int(n/nums\_class))  knn.fit(x\_train, y\_train)  predictions = knn.predict(x\_train)  # print(f"优化样本占比:{round((1-accuracy\_score(y\_train, predictions))\*100,5)}%\n优化后聚类效果指标:{calculate\_indicator(x\_train,predictions)}")    c = predictions  while next(x for x in range(len(c)+1) if x not in c) <= c.max():  missing\_number = next(x for x in range(len(c)) if x not in c)  c = np.array([x - 1 if x > missing\_number else x for x in c])    return x\_train,c  def caculater\_cluster\_index(samples,cluster\_label):  cluster\_center = []  for i in range(max(cluster\_label)+1):  cluster = samples[np.where(cluster\_label == i)]  cluster\_center.append(np.mean(cluster,axis=0))  distance\_matrix\_ = distance\_matrix(cluster\_center,cluster\_center)  path = solve\_tsp( distance\_matrix\_, endpoints = (0,0) )  return path  def transform(b,c,index):  conclusion = []  for i in range(max(index)+1):  number = index[i]  conclusion.append(np.vstack(b[np.where(c==number)]))    return conclusion  def match1(data, regular=True):  # 输入的data需要时簇心排序好的list  '''  input: list data 里面是每一类的array（簇心已排序好）  torch: 是否考虑上一条线 默认为True  output: list Para 里面是相邻两类的拟合直线参数  '''  if not regular:  Para = []  for i in range(len(data)-1):  model = LinearRegression()  d = np.concatenate([data[i],data[i+1]])  model.fit(d[:,:-1], d[:,-1])  para = np.append(model.coef\_,model.intercept\_)  Para.append(para)    else:  # 拟合第一条线  Para = []  model = LinearRegression()  d = np.concatenate([data[0],data[1]])  # model.fit(data[0][:,:-1], data[0][:,-1])  model.fit(d[:,:-1], d[:,-1])  para = np.append(model.coef\_,model.intercept\_)  Para.append(para)  # 簇心  center = []  for i in range(len(data)):  center.append(np.mean(data[i][:,:-1],axis=0))  for i in tqdm(range(1,len(data)-1)):  p = torch.randn(len(Para[0]), requires\_grad=True)  p\_ = torch.tensor(Para[i-1]).float()  d = torch.tensor(np.concatenate([data[i],data[i+1]])).float()  epoch\_num = 3000  optimizer = torch.optim.Adam([p],lr=0.01)  # lam = 0.5 # 惩罚项系数  temp\_min = ((np.array(center)[1:]-np.array(center)[:-1])\*\*2).min()  temp\_max = ((np.array(center)[1:]-np.array(center)[:-1])\*\*2).max()  lam = (temp\_max - (center[i]-center[i-1])\*\*2) / (temp\_max-temp\_min) \* 0.5  lam = lam[0]  for epoch in range(epoch\_num):  optimizer.zero\_grad()  L = ((torch.matmul(d[:,:-1],p[:-1].reshape(-1,1))+p[-1]-d[:,-1].reshape(-1,1))\*\*2).mean() + lam\*((p-p\_)\*\*2).mean()  L.backward()  optimizer.step()  Para.append(p.detach().numpy())  return Para  def match2(A,B):  '''  input: list A 第一类每个点的误差  list B 第二类每个点的误差  output: array row\_ind 第一类的匹配点索引  array col\_ind 第二类的匹配点索引  total\_cost 总成本  '''  # 提供的成本矩阵  cost\_matrix = np.array([[abs(a + b) for b in B] for a in A])  # 获取原始矩阵的行数和列数  rows, cols = cost\_matrix.shape  # 扩展成本矩阵为方阵  if rows == cols:  extended\_matrix = cost\_matrix  else:  size = max(rows, cols)  extended\_matrix = np.full((size, size), np.max(cost\_matrix) + 1) # 使用最大值+1来扩充成本矩阵  extended\_matrix[:rows, :cols] = cost\_matrix  # 应用匈牙利算法  row\_ind, col\_ind = linear\_sum\_assignment(extended\_matrix)  # 初始化总成本  total\_cost = 0  # 仅使用原始矩阵范围内的匹配来计算总成本  total\_cost = extended\_matrix[row\_ind, col\_ind].sum()  total\_cost = total\_cost - (np.max(cost\_matrix) + 1)\*np.abs(len(B)-len(A))  if len(A) > len(B):  for j in range(np.abs(len(B)-len(A))):  index = row\_ind[np.where(col\_ind==len(A)-1-j)[0][0]]  col\_ind[index] = np.argmin(cost\_matrix[index,:])  total\_cost += np.min(cost\_matrix[index,:])  elif len(A) < len(B):  for j in range(np.abs(len(B)-len(A))):  index = row\_ind[np.where(row\_ind==len(B)-1-j)[0][0]]  row\_ind[index] = np.argmin(cost\_matrix[:,col\_ind[index]])  total\_cost += np.min(cost\_matrix[:,col\_ind[index]])  # 输出结果  return row\_ind, col\_ind, total\_cost  def interpolation(samples, y, k, eta, regular=True):  a = K\_Space(k,samples)  b,c = list\_transform\_array(a)  b,c = fine\_turning(b,c)  index = caculater\_cluster\_index(b,c)  conclusion = transform(b,c,index)    for i in range(len(conclusion)):  indices = np.array([np.where((samples == row).all(axis=1))[0][0] for row in conclusion[i]])  conclusion[i] = np.concatenate([conclusion[i], y[indices]],axis=1)  data = conclusion  Para = match1(data, regular)  Row\_ind, Col\_ind, Total\_cost = [], [], []  for i in range(len(Para)):  A = list((np.matmul(data[i][:,:-1], Para[i][:-1].reshape(-1,1)) + Para[i][-1] - data[i][:,-1].reshape(-1,1)).reshape(-1))  B = list((np.matmul(data[i+1][:,:-1], Para[i][:-1].reshape(-1,1)) + Para[i][-1] - data[i+1][:,-1].reshape(-1,1)).reshape(-1))  row\_ind, col\_ind, total\_cost = match2(A,B)  Row\_ind.append(row\_ind)  Col\_ind.append(col\_ind)  Total\_cost.append(total\_cost)  samples\_add = np.concatenate([samples,y],axis=1)  for i in range(len(Row\_ind)):  for j in range(len(Row\_ind[i])):  point1 = data[i][Row\_ind[i][j]]  point2 = data[i+1][Col\_ind[i][j]]  # distance = np.sqrt(((point2-point1)\*\*2).sum())  distance = np.linalg.norm(point2-point1,ord=2)  num\_samples = int(np.floor(eta\*distance))  sample\_points = np.linspace(point1, point2, num\_samples+ 2)[1:-1]  samples\_add = np.concatenate([samples\_add,sample\_points])    return data, Para, samples\_add |

# 致谢