卟啉气体检测系统气体识别算法的研究与工程应用

重庆大学硕士学位论文

（专业学位）

学生姓名：谢新莘

指导教师：罗小刚　副教授

学位类别：工程硕士（生物医学工程领域）

重庆大学生物工程学院

二O一四年四月

Research and Engineering Application of Gas Recognition Algorithm based on Porphyrin Gas Detection System

A Thesis Submitted to Chongqing University

in Partial Fulfillment of the Requirement for the

Master's Degree of Engineering

By

Xie Xinxin

Supervised by Ass. Prof. Luo Xiaogang

Specialty：ME（Biomedical Engineering）

College of Bioengineering of

Chongqing University, Chongqing, China

April, 2014

摘 要

近年来，沙尘暴的肆虐和雾霾天气频发，容易引发人体呼吸道疾病甚至癌变。临床医疗报告已经证实，几种癌症的患者可以通过呼出气体进行疾病的早期诊断。以卟啉作为传感器的气体检测系统能够快速准确地与气体反应并产生差值图谱，作为判别气体种类的依据。由于卟啉的高灵敏度，当同种气体进行平行实验时，实验环境和待测气体浓度的细微差异都会导致卟啉差值图谱出现差异。这种现象称为实验结果发散，会影响气体识别的正确率。因此，一种智能的，能以差值图谱的数据为依据对气体进行分类的算法是保证气体检测系统高识别率的关键。

本文设计了一种应用于卟啉气体检测系统的气体种类识别算法。该算法以差值图谱的特征提取结果为处理对象，结合粗糙集的数据约简功能与 BP 神经网络的训练和识别功能，将未知气体样本进行种类识别。在对该算法的可用性和准确性进行验证后，将其以 C++语言为开发语言，分别在 Windows 系统 VC++平台和 Linux 系统的 Qt 平台编程实现，以加载到卟啉气体检测系统的 PC 系统软件和嵌入式应用程序中。具体的研究内容包括：

(1) 结合粗糙集与 BP 神经网络，利用粗糙集的数据约简功能对气体样本差值图谱的特征值进行约简，再作为 BP 神经网络的训练输入或者识别输入。避免 BP 网络陷入局部最小值，提高训练效率和识别精度。

(2) 对该算法进行验证。包括验证 BP 网络的性能如训练收敛、识别精度、泛化性等，结果表明本文所构造的网络的上述性能良好；通过对比三种气体的聚类分析结果与本文设计的算法的识别结果，可以证明该算法在实验结果发散时也能保持较好的识别率。

(3) 分别对该算法的识别程序和训练程序进行功能需求分析以及架构设计，以 VC++为平台进行编码，实现了本文设计的气体种类识别程序模块，并加载到卟啉气体检测系统在 PC 机上的运行软件中，完善了该软件的功能。

(4) 将 PC 机上的卟啉气体检测系统软件的气体种类识别模块的识别程序按照嵌入式 C 程序规范进行了改写，并移植到嵌入式检测系统应用程序中。同时提取 PC 上训练程序的训练结果并移植，使其能结合识别程序完成气体种类识别。解决了嵌入式仪器神经网络算法训练的工程化移植与应用问题。

关键词：气体识别，粗糙集，BP 网络，VC++，QT

ABSTRACT

In recent years, air pollution is getting worse. The sandstorm and haze, which contain pathogenic toxic gases, may cause respiratory disease, even cancer. The clinical medical reports have pointed out that cancer patients can get early diagnosis by detecting their exhaled gases. The gas detection system with porphyrin as sensors can react with gas quickly and accurately. Because porphyrin is a high sensitivity material of gas sensor, it may bring the discrepancies between the results of parallel experiments when there are slight changes of environment or gas concentration. This phenomenon is called "divergent problem", which can reduce the recognition accuracy. Therefore, an intelligent algorithm that can identify gases by processing data of difference maps is needed to keep good recognition rate of gas detection system.

In this study, a gas species recognition algorithm that was employed in porphyrin sensor array (PSA) gas detection system had been proposed. It was applied to identify gases by processing the differences of RGB channels of each dye spot in difference maps (the "eigenvalues"), using the rough set reduction and backpropagation neural network. After the availability and accuracy had been validated, this new algorithm would be made into a C++ program on the platforms of VC++ and Qt respectively. Then, this program had been added to the software installed in PC of PSA gas detection system, and ported to the embedded device. The content of this study mainly includes the following aspects:

(1) The necessary information of eigenvalues had been extracted by the rough set theory, and then be the input/output vector of the BP-ANN. it was helpful to avoid the local minimum value and improve the training efficiency.

(2) The convergences of training, recognition accuracy and generalization of BP-ANN had been validated well. By comparing the recognition results of cluster analysis and the new algorithm respectively, we had proved that the algorithm perform well enough to identify gases even in the divergent problem.

(3) The functional requirements analysis and architecture design for the recognition program and training program of the new algorithm had been made respectively. The programmatic implementation of the algorithm using VC++ had completed, and the algorithm was successfully added to the software of PSA system.

(4) Rewritten the recognition program for adapting the embedded C programming

standards, and ported to the embedded device. The training results generated by the training program in PC were ported to the embedded device as the parameters of the recognition program to complete the gas recognition function. The problem of how neural network training program porting and applying in embedded device was solved.

Key Words: Gas Recognition, Rough Set, BP-ANN, VC++, QT

目 录

中文摘要 I
英文摘要 II
1. 绪 论 1
1.1 引言 1
1.2 卟啉气体检测系统功能概述 1
1.3 气体种类识别算法现状 3
1.3.1 现有的气体种类识别算法 3
1.3.2 PSA 气体检测系统的信号识别问题 4
1.3.3 粗糙集与神经网络结合 5
1.4 课题的研究目的和内容 7
1.4.1 研究目的 7
1.4.2 研究意义 7
1.4.3 研究内容 7
2. 基于粗糙集和神经网络的识别算法 9
2.1 BP 神经网络结构设计 9
2.2 粗糙集 11
2.2.1 基本概念 12
2.2.2 知识的粗糙性和信息熵 13
2.3 整体算法流程 15
2.3.1 数据预处理 15
2.3.2 粗糙集约简 17
2.3.3 BP 神经网络的训练与识别 18
2.4 算法验证 18
2.4.1 聚类分析的对比验证 18
2.4.2 BP 神经网络的算法验证 20
2.4.3 粗糙集约简的算法验证 22
2.5 本章小结 25
3. 算法在 PC 系统的实现 26
3.1 编程环境：VC++ 26
3.2 软件开发流程 26
3.3 需求分析及框架设计 27

3.3.1 识别程序设计需求 28
3.3.2 训练程序设计需求 30
3.4 算法的实现 31
3.4.1 代码设计流程 31
3.4.2 数组与动态内存分配 32
3.4.3 识别程序的实现 32
3.4.4 训练程序的实现 35
3.4.5 VC++与 MATLAB 混合编程 39
3.5 本章小结 40
4. 算法在嵌入式系统的实现 41
4.1 嵌入式 Linux 和 Qt 概述 41
4.2 嵌入式系统软件设计 42
4.3 程序移植 43
4.3.1 开发环境的搭建 44
4.3.2 目标设备程序运行环境的建立 45
4.3.3 系统开发软件移植 46
4.4 本章小结 49
5. 总结与展望 50
5.1 全文总结 50
5.2 展望 51
致 谢 52
参考文献 53
附 录 57
A. 作者在攻读硕士学位期间发表的论文目录 57
B. 作者在攻读学位期间参加的科研项目目录 57
C. 识别程序与训练程序各子函数伪码 58
识别程序成员函数伪码设计 58
训练程序成员函数伪码设计 60

1 绪 论

1.1 引言

近几年来，中国的大气污染已经严重到了举国关注的程度。从肆虐的沙尘暴到阴暗的雾霾，我们拥有蓝天白云的日子越来越少。随着工业和经济的迅速发展，越来越多的废气被排到大气中。不论是工业废气还是汽车尾气，其对环境的污染程度已经越来越严重。更令人担忧的是，这些废气中通常含有或多或少的有毒气体，有些气体甚至微量就能致命。如果不能对空气中有害气体进行有效的监测，将会埋下威胁人类健康的隐患。

呼吸污染的空气会引发人们多种呼吸道疾病，严重者甚至癌变。我国是世界上肺癌发病率最高的国家。据统计，全球每年肺癌患者超过 110 万，其中的 40 万就来自中国。与其他恶性肿瘤相比，无论发病率或者死亡率，肺癌均居第一。但临床资料显示，如果能在初期进行肺癌确诊，患者的 5 年生存期可以从 15%上升至 80%[1]。可见，肺癌的早期诊断有着重大的意义。与健康人相比，肺癌患者呼出的气体中有微量的有毒气体如己醛、庚醛等[2]。通过检测呼出气体实现对肺癌病人的早期诊断，该课题已成为国家大力支持的研究热点。

不论是应用于日常生活或者临床研究，气体检测技术的需求无疑会越来越多，技术要求也越来越高[3]。传统的气体检测方法检测时间长、步骤繁琐，不能胜任对环境气体进行实时监测的工作。对于检测精度高的技术，比如色谱分析等，则需要专业人士与专业仪器才能实现，成本较高。因此，高效、低廉的检测技术具有更加良好的发展前景。

1.2 卟啉气体检测系统功能概述

在早期的气体检测系统中只有一个传感器，限制了其检测多种气体的能力。为了克服单一传感器选择性差，检测气体种类少的缺点，美国 Zaromb 等人于 1984 年提出了多点测量的技术方案[4,5]。多点测量是指结合多个传感器进行检测，相比单个传感器，多个传感器形成的阵列的响应信号是一个高维的向量，包含更多信息，能更好地识别气体[6]。在气敏传感器中，卟啉因其具有显著的分子识别效应，与气体结合颜色变化有唯一性，被认为是一种很好的气敏材料。利用卟啉及其衍生物制成的卟啉传感器阵列（Porphyrin Sensor Array, PSA）芯片在气体检测方面是一个新颖而成功的应用[7]。

卟啉气体检测系统的核心是一块 6×6 的传感器阵列芯片。该芯片由实验筛选出的 36 种气体特异性检测物质（主要是卟啉衍生物和酸碱指示剂）构成。在传感

器阵列芯片与气体反应的过程中，利用摄像头拍摄反应前后阵列的照片，计算处理得到各卟啉点反应前后的 RGB 差值，生成差值图谱。图谱中以显示的卟啉点个数及其位置作为气体种类识别的标志。理论上，每种气体的差值图谱均具有唯一性，可以作为鉴别气体种类的标准。

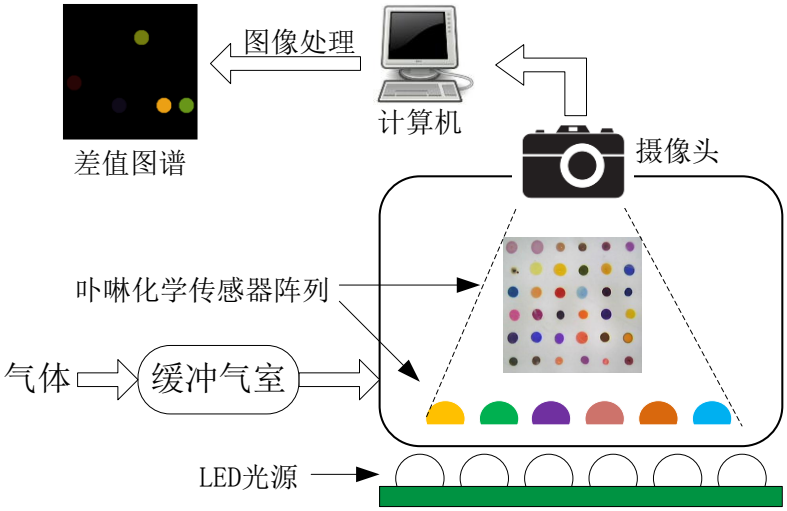

图 1.1 系统检测原理图

Fig 1.1 Structure diagram of PSA gas detection system

卟啉气体检测系统共分为四个功能模块：硬件通信、图像处理、模式识别和数据库管理[8]。硬件通信模块负责进行气体样本的物理处理和图像采集。物理处理包括基本的气体干燥、气室吹扫、控制气体流速、监测气体的温度和湿度；图像采集则包括 PSA 芯片照片的拍摄、存储、照明控制等。图像处理模块对 PSA 芯片反应前后的图像进行处理，包括图像预处理、网格划分、敏感点特征提取 3 个部分。处理后得到差值图谱和一组代表其颜色分量的数据。模式识别模块是通过差值图谱进行被测气体进行种类识别。识别算法的选取是决定卟啉气体检测系统准确率和灵敏度高低的关键。在现有系统中还没有一种智能的识别方法可以提供较高的识别精度，这也是本课题研究意义所在。所有模块返回的数据，包括检测时的环境参数、检测结果等都会被保存到数据库模块中，方便查询和统计。

本文所研究的嵌入式气体检测系统软件以 PC 版本的 PSA 系统软件为蓝本，因此也没有完善的气体种类识别功能。该系统现以 ARM11 作为处理器，因为它有良好的运算能力和用户界面接口。操作系统选择常用的分时型 Linux 操作系统，因为它能够支持多用户操作，有很好的进程独立性，普遍应用于便携式设备、网络设备等通用型电子产品。嵌入式 PSA 系统软件的开发平台选择能完美支持 C/C++ 语言开发的 Qt。

1.3 气体种类识别算法现状

1.3.1 现有的气体种类识别算法

模式识别依照事物的信息，通过一定的处理和分析手段，对事物进行描述和分类[9]。气体的模式识别，包括定性与定量的识别，是对气体进行研究的两个基本方向。本文将主要进行定性识别（即种类识别）的研究。利用何种算法在气体传感器阵列的数据中得到气体的种类信息，一直是一个国际性的研究课题。因气体传感器的种类多样，数据千差万别，应用领域广泛，种类识别的方法并无统一。每一种识别算法各有所长，应该根据具体识别对象选择使用。基于统计、遗传、神经网络等理论的气体识别算法均已有了成熟的发展[10]。

目前国际上应用最为广泛的气体种类识别算法可以分为线性分类方法和非线性分类方法[11]。K-均值聚类、支持向量机、聚类分析[12]、判别分析[13]和主元分析法等，是常用的基于统计学的线性分类方法。反向传播人工神经网络、概率神经网络[14,15]、自组织映射网络、学习向量量化和自适应共振[16]等，是属于非线性分类方法的人工神经网络。同样是非线性分类方法的还有新兴的遗传算法。

K-均值聚类是推广近邻规则的一种算法[17]。常用的距离计算函数为欧氏距离函数和马氏距离函数。Ricardo Gutierrez-Osuna[18]在电子鼻的气体分类部分证明了K-均值聚类算法有很好的分类效果。

支持向量机是统计学理论中最年轻最实用的部分，其学习风险、泛化误差、结构风险等都比较低[19,20]。支持向量机算法在解决小样本、非线性及高维模式识别中有许多优势。杜树新等[21,22]介绍了几种不同的 SVM 方法在气体模式识别中的应用，葛海峰等[23]利用 SVM 和小波结合进行气体识别研究，识别率可达 98%。

主元分析是在气体模式识别领域应用最多的算法之一[24]。其优势在于易于了解，便于实现，降维投影后的结果可以通过简单明了的图形表达出来。为保证可靠性，通常取分析结果前 2 个或 3 个累计贡献率在 75%以上的主元以作分析。魏广芬[25]等结合了主元分析与 BP 神经网络进行气体识别，提高了运算效率。

反向传播人工神经网络（BP-ANN）是在气体模式识别领域的一种主流算法。从结构上看，BP 网络模仿生物神经网络，是一个由神经元构成的自适应非线性系统，可以实现输入和输出数据之间的非线性映射[26]。Shaffer 等[27]利用 BP-ANN 进行化学传感器阵列的气体模式识别，曹明富等[28]将 BP-ANN 应用于电子鼻的气体种类与浓度分析。

自组织映射通过竞争模式选择输出，每次只有一个神经元可以输出[29]。在学习的过程中使得与成功输出的神经元有关的各连接权值向更有利于竞争的方向进行调整。该算法的目标是用低维目标空间的点去表示高维原始空间的点，并且做到最大程度地保留相关性。该算法应用于非线性输入输出的模式识别时，可以有

效的减少传感器阵列数据的冗余部分。

遗传算法是基于达尔文的生物进化论和孟德尔的遗传学基础，通过模拟进化来寻找最优解的方法[30-32]。其搜索步骤是：编码→产生初始群体→选择→交叉→变异→新群体，如果新群体达到要求，输出搜索结果，否则重复搜索步骤。该算法的优势在于有自组织、自适应和自学习的智能性、无需求导或其他辅助信息。Chien-Pang Lee[33]等结合自适应遗传算法与 K-均值聚类进行基因的筛选，获得了极高的准确度，并提高了计算速度。王鹤等[34]用遗传算法优化了 K-均值聚类算法，提高了混合气体识别的准确率。

1.3.2 PSA 气体检测系统的信号识别问题

在实际实验当中，利用 PSA 气体检测系统对同种气体进行平行实验时会产生实验结果发散的情况。卟啉是一种非常敏感的材料，实验过程中任何一点细微变化，比如气体浓度的轻微差别、实验环境温湿度的细小变化、实验仪器的微量气体残留等，都会导致卟啉颜色的改变。从而导致同种气体的平行实验得到的差值图谱却不一致的情况出现，我们称之为实验结果发散。另外，实验器材如摄像头像素不高或者光照不均匀，导致软件对图像处理时产生误差，同样也会出现实验结果发散的情况。如图 1.2 所示是三种气体：丙酮、氨气和苯的差值图谱发散的情况。由于技术的限制，要做到百分之百保证每次实验环境的绝对一致，是不现实的。因此，在进行平行实验时，发散的情况不可避免。

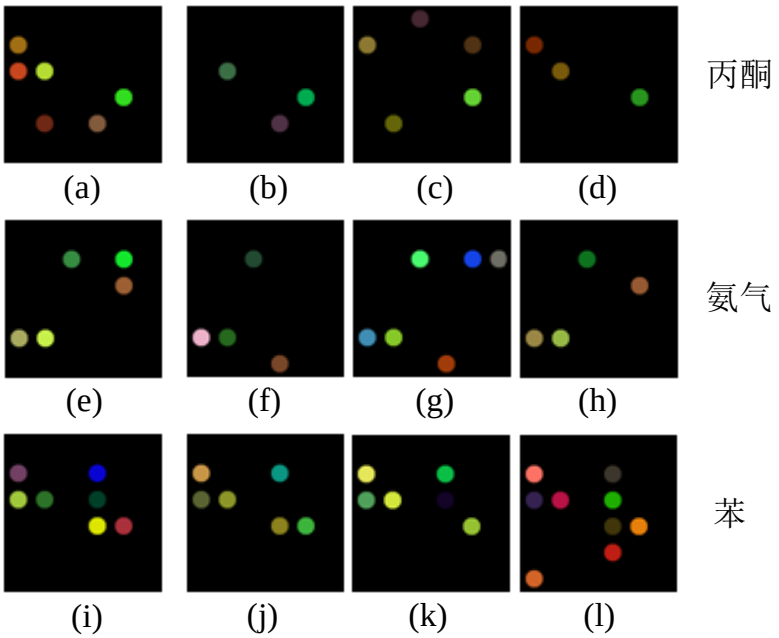

图 1.2 发散的气体差值图谱

Fig 1.2 Divergent difference maps of gases

图(a)为丙酮的理想差值图谱，(b)(c)(d)为发散差值图谱；图(e)为氨气的理想差

值图谱，(f)(g)(h) 为发散差值图谱；图(i)表示苯的理想差值图谱，(j)(k)(l) 为发散差值图谱。

对于每一组差谱图来说，虽然它们都不尽相同，但是他们都代表同种气体。在这种情况下，仅凭直接观察差值图谱来判定气体种类是不可靠的。在现有文献中，所有的气体种类识别算法均建立在同种气体平行实验的差值图谱基本一致的基础上[35-38]。如果使用简单的聚类分析，会对发散的实验结果产生误解，将本该归于同一类的差值数据归到不同的类别中。因此需要设计一种更加全面、智能的气体种类识别算法应用于 PSA 气体检测系统中，以保证系统在任何状态下都有较高的识别率。

PSA 气体检测系统的系统软件需要植入到嵌入式版本中。由于嵌入式硬件的限制，比如不能安装如 Matlab、SPSS 等算法仿真软件、不能对算法进行实时调试、不能高效进行大量数据运算等。这些限制使得利用嵌入式开发复杂的算法需要较长的开发周期。因此需要一种效率高、周期短，同时保证算法精度的嵌入式复杂运算开发的工程应用方案。

1.3.3 粗糙集与神经网络结合

粗糙集（Rough Set）理论是波兰学者 Z.Pawlak 提出的一种新型的处理模糊和不确定知识的数学工具，能够有效地分析不精确、不可辨别、模棱两可等各种不完备的信息[39]。粗糙集的主要思想是利用已知的知识库去划分不精确或者不确定的知识。知识即对数据的划分，是一种分类的能力，而划分的依据是对象的等价关系。粗糙集与其他模糊理论最显著的区别在于无需提供对知识或者数据的主观评价，仅通过观测数据进行分析，因此对数据的不确定性描述或者处理比较客观。

同样是数据挖掘领域的分类算法，粗糙集与神经网络却各有特点。粗糙集在处理大量的，不完备且存在冗余数据的信息时，可以快速、客观地描述其内在规则，根据决策条件提取出特征属性。BP 神经网络是应用最广泛的神经网络之一，能够高效地对数据进行分类，对输入输出数据的非线性映射有非常好的逼近能力。但是过量的数据，或者过多的冗余数据，很容易降低 BP 神经网络的效率，甚至出现局部最小值。

粗糙集与神经网络的结合，能够很好地弥补神经网络算法上的不足。一方面可以改善神经网络对训练样本量大、数据冗余的情况的处理能力，提高其训练效率，另一方面也可以优化神经网络的结构或者神经元的连接权值，增强神经网络对不同数据进行处理的灵活性和适应性。国内外神经网络与粗糙集相结合的用途主要按图 1.3 所示：

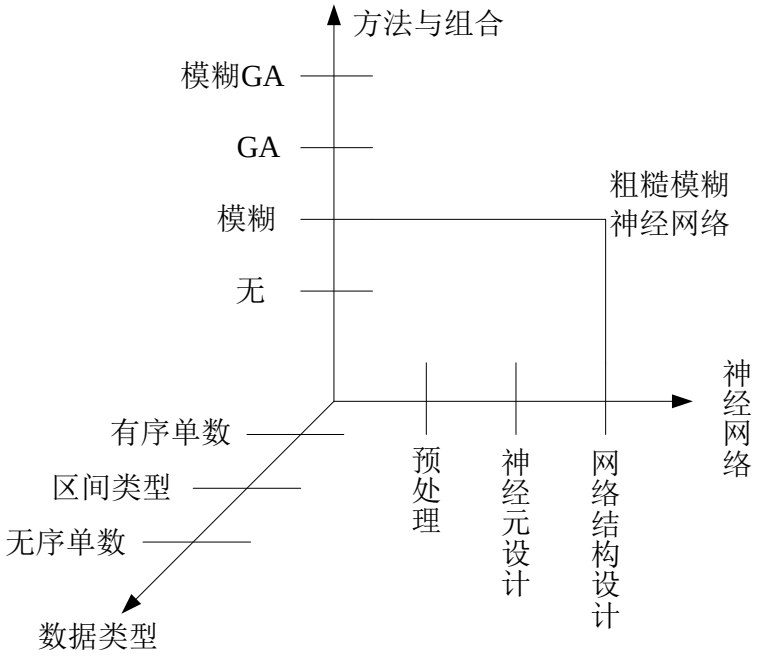

图 1.3 神经网络结合粗糙集的应用框架

Fig 1.3 Application Framework of Rough Set with Neural Network

利用粗糙集的数据约简功能对数据进行约简，可以去除大量的冗余信息。将约简后的数据进行神经网络的训练或者识别，可以提高网络训练速率。由于去除了噪声的影响，使得识别精度更高。尚志信等[40]利用粗糙集的约简功能去除网络故障诊断知识库的冗余信息，提高了 BP 神经网络对故障诊断的效率。李伟等[41]结合了粗糙集约简和遗传算法对 BP 神经网络进行优化，提高了预测精度。chen 等[42]结合粗糙集约简功能和支持向量机进行乳腺癌的诊断，取得了很好的效果。

另一个应用是利用粗糙集进行神经网络神经元的设置。粗糙集设置的神经元输出的并非一个确定值，而是一对上下界值组成的范围。使用粗糙集神经元的神经网络预测模型适用于输入或者输出为一个容错范围的情况。文莹等[43]利用粗糙集改进 BP 神经网络的神经元结构，并应用于航空电子设备故障预测，提高了预测准确率。陈锦荣等[44]利用粗糙集神经元设计了一种粗逻辑神经网络及其学习方法。

利用粗糙集设计神经网络结构。粗糙集有提取决策系统规则的能力。神经网络可以通过该规则进行结构设计，提高网络的可理解性和可用性。其中对模糊神经网络进行结构设计的研究比较多。郭志军等[45]利用粗糙集提取的决策表语义规则来初始化神经网络的参数，提高了分类效率。夏红霞等[46]利用隐含层节点与网络误差构造决策表，由粗糙集删除多余的隐层节点，达到简化网络结构的目的。

1.4 课题的研究目的和内容

1.4.1 研究目的

设计一种应用于 PSA 气体检测系统的气体种类识别算法。该算法利用系统对传感器阵列的图像处理后的数据，利用尽可能简单而有效的计算方法，对气体进行分类识别。该算法应该有较高的识别率、运算效率和可移植性。通过编程实现该算法，使其不仅可以应用于 PSA 气体检测系统上位机的安装软件中，还可以移植到该系统的嵌入式版本中，为嵌入式版本不能实现复杂识别算法开发的工程技术问题提供一个可行的解决方案。

1.4.2 研究意义

气体的种类识别对于气体检测系统而言有十分重要的意义。能成功将每一种未知气体样本进行正确的归类是一个成功的气体检测系统应该具备的功能。在实际的实验过程中，由于叶啉非常灵敏，环境的影响，操作的微小差异，气体浓度的细微变化等，都会导致差值图谱的不同，同种气体的平行实验结果有可能是发散的。此时简单的凭肉眼观察差值图谱并不能很好的将同种气体归类。因此，需要设计并实现一种智能的，能以客观数据为依据进行气体种类分析的算法。

气体识别算法研究中不乏识别精度高、理论体系完善的算法。但其实现的难易往往不在研究者的考虑范围。由于复杂的算法在利用编程语言实现时会降低仪器运行的速度和效率。如何在拥有较高精度和通过编程语言将其实现的难易程度中取得平衡，是本课题研究的意义之一。

PSA 气体检测系统需移植到嵌入式 ARM 开发板中。如 1.3.2 节所述，与 PC 机相比，嵌入式设备并不具备很好的复杂运算开发环境，开发周期长，效率低。为气体识别算法在嵌入式中实现提供一个可行的开发方案，也是为解决如何在嵌入式中开发复杂运算这一工程技术问题提供了思路。

因此本课题的研究旨在能为应用于 PSA 气体检测系统的气体种类识别方法提供一种新的思路，更好地提高气体种类识别的准确性、稳定性。同时实现该算法的程序编写，植入到系统的软件部分，并且在 PC 机和嵌入式中都有良好的运行效果和可维护性。

1.4.3 研究内容

本课题针对 PSA 芯片的差值图谱，尤其是在同种气体平行实验存在发散的情况下，总结多种气体识别算法的优缺点，分析原系统识别算法的不足之处，设计了一种识别率高、稳定性好的气体种类识别算法。而后从软件开发方面考虑，以保证运行效率为前提，将该算法通过软件编程分别加载到 PSA 气体检测系统的 PC 版本和嵌入式版本的运行软件中。该算法解决了 PSA 气体检测系统中没有智能气体种类识别算法的难题，保证了该系统软件在实际运行上的完整性和准确性。

论文的主要研究内容如下：

第一章首先对本课题的研究和应用对象——卟啉传感器阵列（PSA）气体检测系统进行了大概的介绍，包括气体检测系统的检测原理以及在进行检测实验时出现结果发散的情况。同时也介绍了现今各应用于气体传感器阵列气体识别算法的种类、原理和应用情况等，并阐述了粗糙集与神经网络结合的优势。最后提出本课题的研究内容和意义。

第二章详细地介绍了本课题所设计的气体种类识别算法的原理。从粗糙集的原理到神经网络的结构设计，为该算法奠定了理论基础。从理论上证明了算法的可行性。随后利用数学工具对本文提出的算法进行验证，结果表明该算法对气体进行种类识别具有较高的准确率和精度。

第三章详细介绍了算法在 PSA 气体检测系统 PC 端 Windows 操作系统中的实现过程。首先对 Windows 操作系统与 VC++做了概述，并介绍了系统的软件设计图以及自己要增加的部分，然后从软件开发方面以功能需求为基础详述软件设计的过程，最后从软件编程的角度介绍数据传递的方式和优化选择。

第四章详细介绍了算法在 PSA 气体检测系统嵌入式版本的实现过程。与 PC 不同的是，嵌入式版本的开发环境为 Linux 下的 Qt 语言编程。详细介绍了如何将算法从 PC 版本移植到嵌入式系统中。

第五章为全文总结，列出了整个研究工作中存在的不足以及可以改进的地方。并且对算法程序的后续优化工作提出了一些设想。

2 基于粗糙集和神经网络的识别算法

本课题设计了一种应用于叶啉传感器的识别算法，该算法借鉴了生物芯片的数据处理方法。两者相似的地方在于都是圆点阵列，处理的对象都是阵列图像。生物芯片在对阵列图像进行处理后，往往使用有监督的方法或无监督的方法将大量的基因数据进行处理，分离出需要的少量数据[47,48]。人工神经网络是应用得比较广泛的一种有监督的分类方法[49]，适用于处理线性鉴别分析难以处理的数据。

BP 神经网络是应用得比较多的人工神经网络，但是有着不能避免的缺点：易受冗余数据的影响，产生局部最小值而不是全局最小值等。而粗糙集则是一种能很好地挖掘数据间相关信息，去除冗余信息的数学工具，因此两者结合对 BP 神经网络有扬长避短的效果。

2.1 BP 神经网络结构设计

在生物神经系统中，神经元可以收发不同数量的生物信息，经过累计达到某个临界值时发送给大脑。多个神经元可以并发运行。大脑通过调节神经元之间连接的数目和强度进行学习。人工神经网络是以神经系统结构为模型，以设计高效的机器学习算法为目标的一种接近人类思维方式的算法。人工神经网络能够使机器通过训练学习外部环境，拥有大规模并行处理数据、自适应学习过程、分布式存储等能力。BP 神经网络是目前应用最为广泛和成功的神经网络之一[50]。

BP 神经网络在 1986 年由 Rumelhart 和 McCelland 提出，是一种具有三层或三层以上的阶层型前馈神经网络。该算法按误差逆传播算法训练，网络结构如图 2.1 所示。典型的三层 BP 网络包括输入层、隐含层（中间层）和输出层。网络上下两层之间每个神经元都相互连接，但是每层之间的神经元却无连接。相连的神经元都具有连接权值和阈值，这是整个网络的特性所在。通过改变这些权值和阈值，可以调节网络的识别准确率。

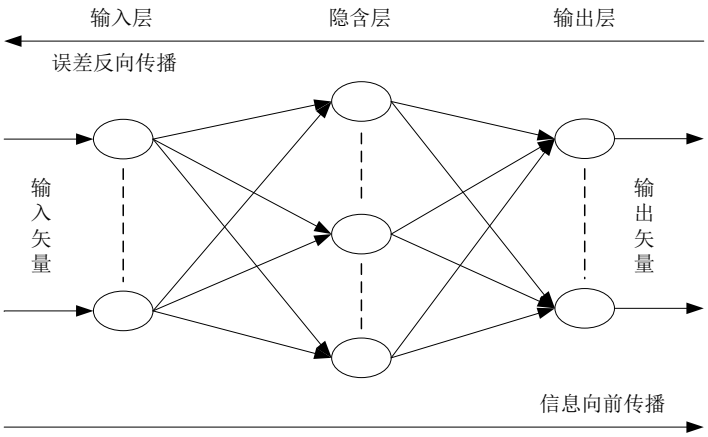

图 2.1 BP 神经网络的结构图

Fig 2.1 Structure diagram of BP neural network

BP 神经网络的学习可以分为三个步骤。首先，数据顺序经过网络各层直至输出完成“顺传播”；然后训练目标与实际输出的误差以反方向逐层修正每层神经元的连接权值，完成“误差逆传播”；“顺传播”与“逆传播”交替进行，直到全局误差发生收敛并小于预定误差，从而完成“记忆训练”和“学习收敛”过程。BP 网络运算流程如图 2.2 所示。

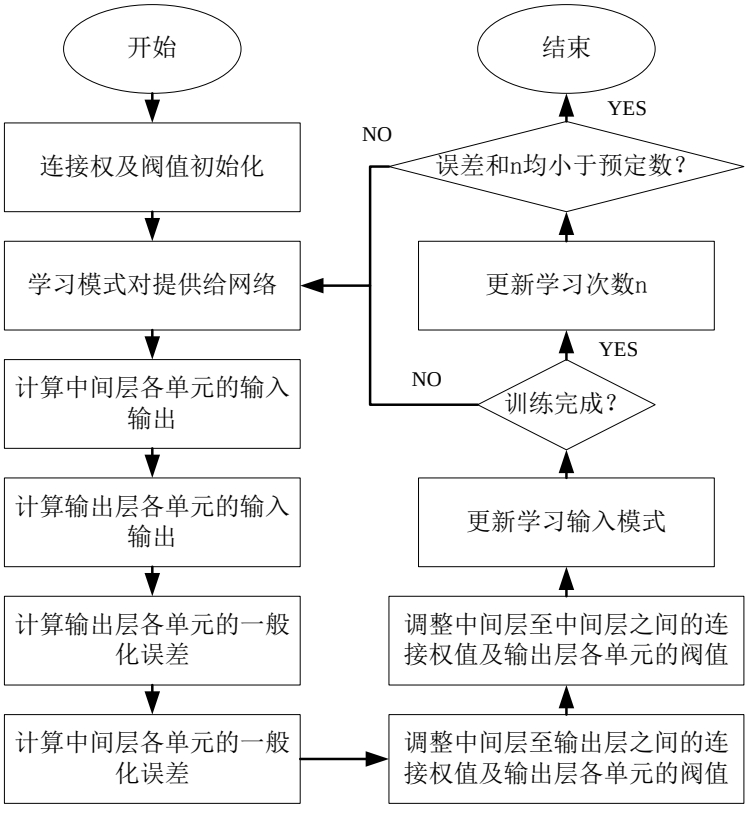

图 2.2 BP 神经网络的算法流程图

Fig 2.2 Algorithm flowchart of BP neural network

在使用 BP 神经网络前先要设计网络结构。比如选择输入层和输出层的节点数、隐含层数与每层节点数、激励函数等。

本文选取 sigmoid 函数作为激励函数：

$$f(x)=\frac{1}{1+\exp(-x)} \tag{2.1}$$

隐含层节点数的选择关系到神经网络的训练时间和训练结果的准确性。但是目前并没有准确的统计出隐含层节点数与训练效率的数学关系，也没有一个统一的选择指标。比较常用的做法是参考一些比较有效的经验公式选定一个数值范围，取该范围内的数值逐一实验，直到选到最合适的隐含层数。本文中隐含层节点数的选择参照以下三个经验公式：

$$M<I-1 \tag{2.2}$$

$$M=\log_2 I \tag{2.3}$$

$$M=\sqrt{I-O}+a \tag{2.4}$$

其中 M 代表隐含层节点数，I 代表输入层节点数，O 代表输出层节点数，a 代表 1 到 10 之间的任意整数。

传统的 BP 神经网络理论有着不可避免的缺陷。过量冗余数据易造成神经网络训练过度，陷入局部最小的状态，增加训练次数。训练次数的增多会导致学习效率下降，收敛速度减慢，从而使训练的时间增加。PSA 气体检测系统每次处理得到的差值图谱含有大量的冗余数据。为保证 BP 神经网络的训练效率和精度，需要在训练之前先进行数据的降维处理。因此，选择一种合适的降维算法，提取特征数据，最大程度地保持数据的信息，是非常必要的。

2.2 粗糙集

粗糙集理论采用等价关系将研究对象粒化为若干互斥的等价类，作为描述对象任意概念的基本信息粒子。粗糙集把知识作为集合的一部分，并作为分类的依据。一个对象是否属于一个集合可以分为三种情况：肯定属于、可能属于、肯定不属于。对于对象的任意概念，粗糙集定义了两个等价类的并集：下近似和上近似来逼近这一概念[51]。所谓近似，是指在一个集合U 中，其中一个分割中的集合的并集对其他的任意子集（同属于U ）的逼近。设 x 为U 中的一个对象，$X\in U$，R 是U 上的一个等价关系。

集合 X 的下近似（Lower approximation）定义为：

$$\underline{apr}(X)=\{x\,|\,[x]_R\subset X\} \tag{2.5}$$

上近似（Upper approximation）定义为：

$$\overline{apr}(X)=\{x|[x]R\cap X\neq\phi\} \tag{2.6}$$

相应的，x 与 X 的三种关系可以表述为正域 $POS(X)$、边界域 $BND(X)$ 和负域 $NEG(X)$：

$$POS(X)=\underline{apr}(X) \tag{2.7}$$

$$BND(X)=\overline{apr}(X)-\underline{apr}(X) \tag{2.8}$$

$$NEG(X)=U-\overline{apr}(X) \tag{2.9}$$

粗糙集理论定义在经典的等价关系和等价类基础上，适合处理离散型非确定数据，多应用于数据约简、规则生成及预测等方面。

2.2.1 基本概念

定义 1 决策系统。设 $S=\{U,A,V,f\}$，论域 $U=\phi$，A 为属性集，V 为属性 $a\in A$ 的值域，$f:U\times A\to V$ 为信息函数。若 A 由条件属性集合 C 和决策属性集合 D 组成，且 C、D 满足 $C\cup D=A$，$C\cap D=\phi$，则称 S 为决策系统。通过在 U 上定义两个等价关系 R_C 和 R_D，根据这两者之间元素的相互关系可以得到一系列用于对未知数据进行分类的决策标准。

定义 2 不可分辨关系。对于决策系统 $S=(U,C\cup D)$，$B\subseteq C$ 且 $B\neq\phi$，B 中所有等价关系的交集称为 B 上的一种不可分辨关系。用 $IND(B)$ 表示不可分辨关系。以此为理论基础，可以引出上面提到的正域与边界域的一般规则表达式。

正域：若 $x\in POS\pi_{R_C}(\pi_{R_D})$，则有 $[x]_{R_C}\to_p D_i$，$D_i\in\pi_{R_D}$，$[x]_{R_C}\subset POS\pi_{R_C}(\pi_{R_D})$；

边界域：若 $x\in BND(X)$，则有 $[x]_{R_C}\to_B D_i$，$D_i\in\pi_{R_D}$，$[x]_{R_C}\subset BND\pi_{R_C}(\pi_{R_D})$。

在大多数的数据处理中，存在着许许多多的属性。有些属性可以相互决定，有一定的等价关系，同时也会存在一些对决策没有任何作用的属性，称为冗余的属性。去除冗余的属性不会影响分类的效果，而且能提高分类的速率，因此，粗糙集理论引入了约简的功能。

定义 3 约简。在决策系统 $S=(U,C\cup D)$ 中，C 为条件属性集合，如果存在一个非空集合 C'，满足 ① $IND(C',D)=IND(C,D)$；② 不存在 $C''\subset C'$ 使 $IND(C'',D)=IND(C,D)$，则称 C' 为 C 的约简。C 可以有多个约简，其集合记为 $red(C)$。

定义 4 核。在 C 中所有必要关系组成的集合称为 C 的核，用 $core(C)$ 表示。并且 $core(C)=\cap red(C)$。核是所有约简所具有的共同特征。

2.2.2 知识的粗糙性和信息熵

粗糙集理论认为，知识就是和论域相关的一个模式划分，拥有粒度性质。在对两种知识粒度进行比较时，粗糙性是一个衡量的标准。其定义为代数学中的等价关系和集合包含关系[52]。

设 $K=(U,P)$ 和 $K_1=(U,Q)$ 是两个知识库，$U/IND(P)$ 表示不可区分关系 $IND(P)$ 在集合 U 的划分，如果 $U/IND(P)\subseteq U/IND(Q)$，则称知识 P 比知识 Q 较细（finer）或者知识 Q 比知识 P 较粗（coarser），记为 $P\prec Q$。其中符号 $\subseteq$ 是表示对于任意的非空集合 $A\in U/IND(P)$，总能找到另一个非空集合 $B\in U/IND(Q)$，同时满足 $A\subseteq B$。

在论域中，一种划分就是一组基础概念。任意的一个元素或对象是否属于特定一个属性（即落入这种划分之中）是具有概率分布意义的。也就是说，在粗糙集理论中，知识可以被看成是随机变量。

设 P 和 Q 为论域 U 上两个已知的知识，根据粗糙性的定义，从代数学的角度把 P、Q 当作论域中的两个随机变量集合。它们的概率分布可以表示为：

$$[X;p]=\begin{bmatrix}X_1,X_2,\cdots,X_n\\P(X_1),P(X_2),\cdots,P(X_n)\end{bmatrix} \tag{2.10}$$

$$[Y;p]=\begin{bmatrix}Y_1,Y_2,\cdots,Y_m\\P(Y_1),P(Y_2),\cdots,P(Y_m)\end{bmatrix} \tag{2.11}$$

其中 $X=U/IND(P)$，$Y=U/IND(Q)$。

$$p(X_i)=\frac{cardX_i}{cardU},\quad i=1,2,\cdots,n \tag{2.12}$$

$$p(Y_i)=\frac{cardY_i}{cardU},\quad i=1,2,\cdots,m \tag{2.13}$$

符号 $cardE$ 表示集合 E 的基数。

P 与 Q 的联合概率分布定义为：

$$[XY;p]=\begin{bmatrix}X_1\bigcap Y_1,\cdots,X_i\bigcap Y_j,\cdots X_n\bigcap Y_m\\p(X_1Y_1),\cdots,p(X_iY_j),\cdots,p(X_nY_m)\end{bmatrix} \tag{2.14}$$

其中 $p(X_iY_j)=\dfrac{card(X_i\bigcap Y_j)}{cardU}$，$i=1,2,\cdots,n$，$i=1,2,\cdots,m$。

从知识的概率分布和联合概率分布可以引申出其条件概率分布。若知识 P 为已知，则知识 Q 的条件概率分布可以定义为：

$$p(Y_j\mid X_i)=\frac{p(X_iY_j)}{p(X_i)},\quad i=1,2,\cdots,n,\quad i=1,2,\cdots,m \tag{2.15}$$

根据信息论，在条件概率已知的条件下，可以定义出知识的信息熵、条件熵（相对信息熵）和互信息的代数表达式。

知识 P 的信息熵：

$$H(P)=-\sum_{i=1}^{n}p(X_i)\log p(X_i) \tag{2.16}$$

知识 Q 相对于知识 P 的条件熵：

$$H(Q\mid P)=-\sum_{i=1}^{n}p(X_i)\sum_{j=1}^{m}p(Y_j\mid X_i)\log p(Y_j\mid X_i) \tag{2.17}$$

知识 P 与知识 Q 的互信息：

$$I(P;Q)=H(Q)-H(Q\mid P) \tag{2.18}$$

以知识的粗糙集为理论基础将知识的概率分布代数化，并且结合信息论引申出信息熵、条件熵和互信息的代数表达式，逐步建立知识与信息之间的联系。根据概率及熵函数的性质，可以得出下面的定理：

定理 1 设 $K=(U,P)$ 和 $K_1=(U,Q)$ 是论域 U 中的两个知识库，D 是 U 中的决策集合，如果 $P\prec Q$，表示下列式子成立：

(1) $H(P)\geq H(Q)$ 或者 $I(P;D)\geq I(Q;D)$

(2) $H(P)>H(Q)$ 并且 $H(Q\mid P)=0$

(3) $I(P;D)>I(Q;D)$ 并且 $H(Q\mid P)=0$

也就是说，知识粗糙性的粗细与其提供的信息成反比，粗糙性越细的知识，提供的信息越多。

定理 2 在决策表 $S=(U,C\cup D)$ 中，U 为论域，C 和 D 分别代表条件集和决策集。对任意的 $B\subseteq C$，$B\neq\phi$，若 B 是 C 中相对于 D 的一个约简则应该满足下列条件：

(1) $I(B;D)=I(C;D)$

(2) 对于任意的 $p\in B$，都有 $H(D\mid B)=H(D\mid B-\{p\})$

粗糙集在神经网络中应用得最成熟的方面就是对训练样本进行预处理。一般情况下，神经网络的输入训练样本往往含有或多或少的冗余数据。利用粗糙集的数据约简功能，可以筛选出包含信息最多的数据，过滤掉多余的数据，这样就减少了噪声对神经网络的影响。经过降维后的训练样本使得神经网络的训练时间缩短，减少了出现局部最小值的概率，从而提高神经网络的识别效率。本文中粗糙集的数据约简功能对神经网络数据进行降维的步骤如下：

(1) 由差谱图数据构造决策表，制定条件属性及决策属性；

(2) 使用基于信息熵的粗糙集分析方法寻找其最小约简和核；

(3) 保留约简后特征属性的值，特征属性的个数作为网络的输入节点数；

(4) 将约简后的数据进行神经网络训练、识别；

(5) 重复 2-4 步骤，直到识别精度不再提高。

2.3 整体算法流程

本文提出了一种结合粗糙集与 BP 神经网络的气体种类识别算法。尽管 BP 神经网络的分类精度高，对噪声数据不敏感，但是在训练大量数据或者过量冗余数据时易造成神经网络训练过度，识别效率低等情况。而粗糙集的约简功能可以快速删除冗余数据，最大程度保留原数据中的有用信息，与神经网络结合使用可以补偿其训练速度慢，适应性差，容易得出局部最小值等缺点。

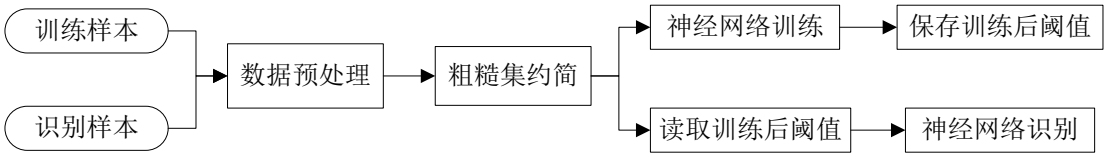

图 2.3 整体算法流程图

Fig 2.3 Diagram of the algorithm designed in this study

由整体算法流程图可以看出，本文的算法设计成三部分。第一部分包括选择实验数据和对其进行基本的预处理，预处理程序包括数据第一次降维和归一化；第二部分将预处理后的数据根据粗糙集的约简理论构造决策表，选择条件属性和决策属性，得到与决策属性最密切相关的条件属性，删除其余属性，即数据第二次降维；第三部分将降维后的数据利用 BP 神经网络进行训练和识别。通过比较输入数据降维前和降维后 BP 网络的识别效率和准确率，可以验证本文算法有比较高的识别精度。

2.3.1 数据预处理

当气体经过卟啉化学传感器阵列，发生反应时，系统的图像传感器将采集到 n 个时间点的 PSA 图像，包括反应前的图像与反应过程中每隔相同时间采集到的反应图像。通过图像特征提取算法计算出每一幅反应图像的差值编码，该差值编码为气体在 n 个时间点与 PSA 芯片上的 36 个点反应时每个点的 RGB 分量差值，其维度为 1×36×3=1×108。在实验中，每个点的颜色变化随着时间的推移而逐渐稳定，这是一个实验的数据中最有代表性的部分。为了实现对气体种类的识别，选择反应饱和时的数据作为识别算法的输入数据。根据实验经验，反应在进行 5-6 分钟后进入稳定状态，因此选择每个实验中该段时间的差值图谱数据进行处理。

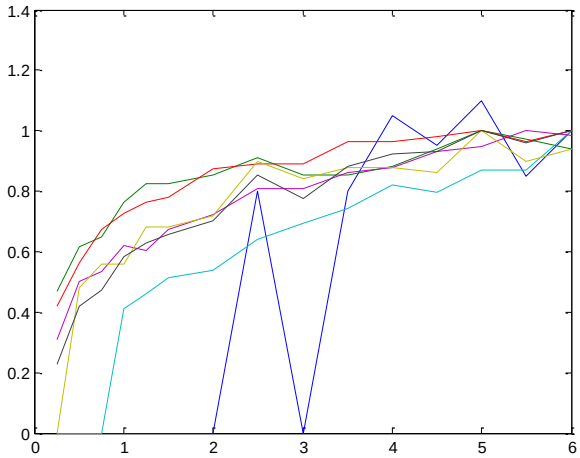

图 2.4 各卟啉点随时间颜色变化趋势

Fig 2.4 Trends of color change of porphyrin points

每一幅 PSA 图像的 RGB 差值向量就是其差值图谱的数字化表示，是通过 PSA 系统的图像处理算法对差值图谱进行特征提取得到的。每个 PSA 芯片上的点的颜色变化都以 R、G、B 三个分量的变化表示出来，即 ΔR、ΔG、ΔB。在实验中，我们更关注一个卟啉点整体的颜色变化程度。因此，我们采用欧氏距离公式将一个卟啉点在三维空间的 ΔR、ΔG、ΔB 分量转换成一个一维空间的值 d，使其总体变化幅度更加明确地表示出来。

$$d = \sqrt{\Delta R^2 + \Delta G^2 + \Delta B^2} \tag{2.19}$$

经过欧氏距离公式的处理后，单幅差值图谱的差值编码的维度从 1×108 下降到 1×36。实现了第一次数据降维。

在实际实验中，由于不能保证每次实验的环境、样本浓度、仪器状态等都在严格相等的状态下，得到的实验数据值均反映的是当时实验环境的结果，不同实验间的数据值难免存在误差。为了尽可能地消除不同实验环境之间的差异，需要对实验数据进行归一化操作。将所有数据转化成范围在 0 和 1 之间的小数，有了同样的表示标准，不同实验间的数据结果才更有可比性。在本文中，每个样本数据均通过公式(2.20)进行线性归一化处理。

$$x_k = (x_k - x_{\min}) / (x_{\max} - x_{\min}) \tag{2.20}$$

其中 x_k 代表每一个差值向量内的元素，x_{max} 是绝对值最大的元素，x_{min} 是绝对值最小的元素。

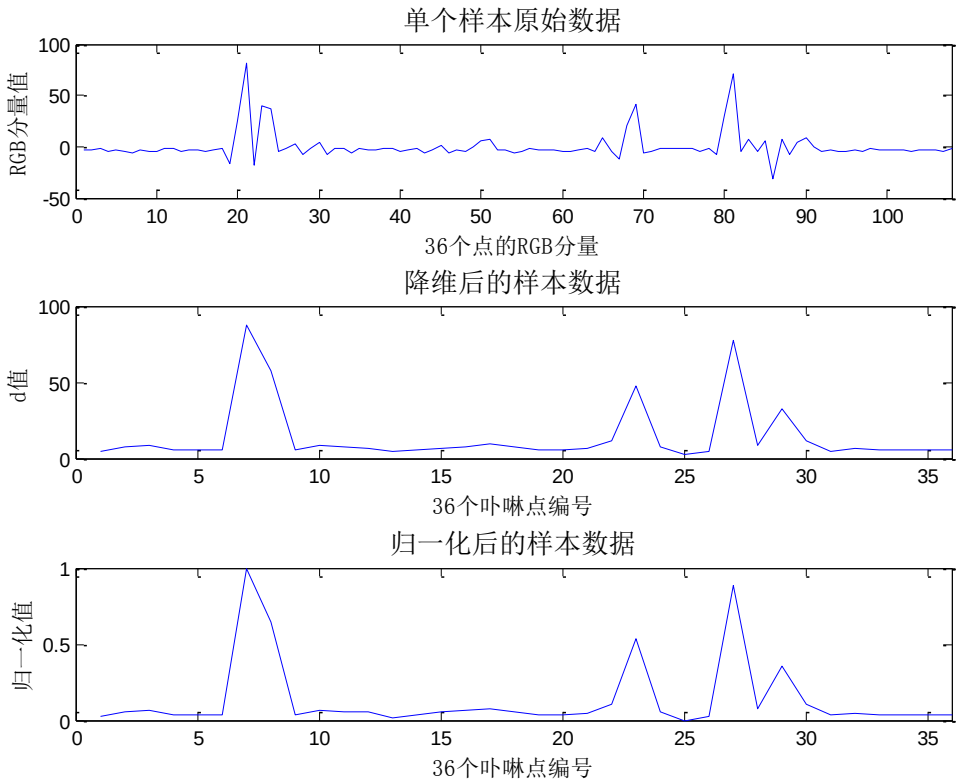

图 2.5 样本数据预处理过程

Fig 2.5 Pre-procession of a sample data

图 2.5 展示了单个气体样本的 RGB 差值向量在经过数据预处理后维度和值的变化。可见，数据预处理使得发生颜色变化的卟啉点更加突出，但是各点间的相对变化保持稳定。

2.3.2 粗糙集约简

根据粗糙集约简的定义可知，知识的任意一个约简对知识的划分能力是相同的。在论域中，知识可以有不同的约简。这些约简的交集称为知识的核，核对于知识而言是唯一的。粗糙集对知识进行约简的过程就是粗糙集寻找知识特征的过程，约简的结果即要寻找的特征体现。由于核是知识约简的交集，可以推论出核是知识特征体现的本质，是所有知识进行约简的基础。

粗糙集约简，从理论上来说，就是减少决策表的条件属性的数量，在不破坏决策表所有信息的基础上，使条件属性的数量最少。粗糙集的属性约简的方法有很多，大体上可以分为静态约简和动态约简的方法。静态约简包含的大多是启发式的约简方法。其基本思路是将重要度高的属性添加到核中，组成约简集合。信息熵、可分辨矩阵、正区域等方法均被应用于粗糙集的属性约简中。动态约简的方法针对的是动态变化的信息系统，在静态算法理论的基础上加入了增量式动态更新约简集合的功能，以约简不断变化的信息系统[53]。

CEBARKCC 算法是粗糙集基于信息熵的静态属性约简算法之一。该算法以决策表核属性集为起点，自底向上选择使 $H(D|B\{a\})$ 最小的非核条件属性 a 添加到核属性集中，直到满足 $H(D|B)=H(D|C)$。

在决策系统 $S=(U,C\cup D)$ 中，C 为条件属性集合，D 为决策属性集合，CEBARKCC 算法的计算步骤如下：

(1) 计算 D 相对 C 的条件熵 $H(D|C)$ 和互信息熵 $I(C;D)$；

(2) 计算 C 相对于 D 的核属性 B;

(3) 若核属性 B 的绝对值不为 0，计算条件熵 $H(D|B)$，$I(B;D)$；

(4) 若 $I(C;D)\neq I(B;D)$，计算决策属性集合 D 条件属性 C 的每一个子属性 $c_i\in C-B$ 的条件熵 $H(D|B\cup\{c_i\})$；

(5) 逐次选择使 $H(D|B\cup\{c_i\})$ 最小的非核条件属性 c_i 添加到和属性集中，直到 $I(C;D)=I(B;D)$。

2.3.3 BP 神经网络的训练与识别

在章节 2.1 对 BP 神经网络的介绍中，仔细地列出了进行网络结构设计的参考信息。为了更好地发挥神经网络的性能优势，输入训练或者识别的数据也应该遵从一些规则。主要从两个方面考虑：样本数据的数量和质量。

一般来说，训练样本的数量越多，BP-ANN 对其内在规律越逼近。但是正如上述章节所说，过多的训练样本会降低 BP-ANN 的学习效率，还有陷入得到局部最小值的风险。因此训练样本数量的选择应该根据神经网络的结构设计决定。网络的映射关系越复杂，结构规模越大，准备的训练样本数量就越多。由于本文中设计的 BP 神经网络结构并不复杂，是基本的三层结构，虽然有数据降维的处理，但是应该避免过大的数据量。

在选择样本的时候，应该选择有代表性的样本，而且各类样本数据之间做到数目均衡。就如 2.3.1 而言选择反应稳定的特征差值向量作为样本。而每种气体参与训练的样本数量也应该一致。在输入训练时要将不同类别的样本进行交叉输入。

决定网络训练时间的因素除了有训练样本的数量之外，还包括初始权值的设计。因为网络权值的初始化决定了网络训练从误差曲面的哪一个点开始。常用的设计权值初始化的原则包括选择足够小的权值，或者使初始值为 1 和-1 的权值数相等。本文的 BP 网络权值初始化使用赋值随机数的方法。

2.4 算法验证

2.4.1 聚类分析的对比验证

聚类分析是一种研究样本类别问题的统计方法，分类标准和分类算法都非常丰富。聚类分析的标准通常为两类：系数和距离。相似系数是指用来描述样本之

间性质相近程度的系数，当两个样本之间性质越接近，其相似系数的绝对值越接近 1，相反，两者的相似系数绝对值越接近 0。相似系数越大，这两个样本被归为一类的可能性就越大。距离是指从空间的角度将样本定义为高维空间的一个点，两个样本之间在此空间的距离决定了他们被归为一类的可能性。距离越小，性质越接近。

欧氏距离是用得最多的距离聚类方式。其聚类原理是根据每个样本在空间中相互间距离的远近来归类。欧氏距离应用于对阵列图像数据进行处理时，识别速度快，分类效果也较好。但是当存在发散样本时，欧氏距离对其分类的能力将被限制。

在本次实验中，选择图 1.2 所示的 3 组气体发散实验共 12 个样本数据作为数据源，每个样本均选择时间点为 6min 的样本数据，组成一个 12×108 的向量，经过数据预处理后（即降为 12×36 的向量）作为欧氏距离聚类的输入。

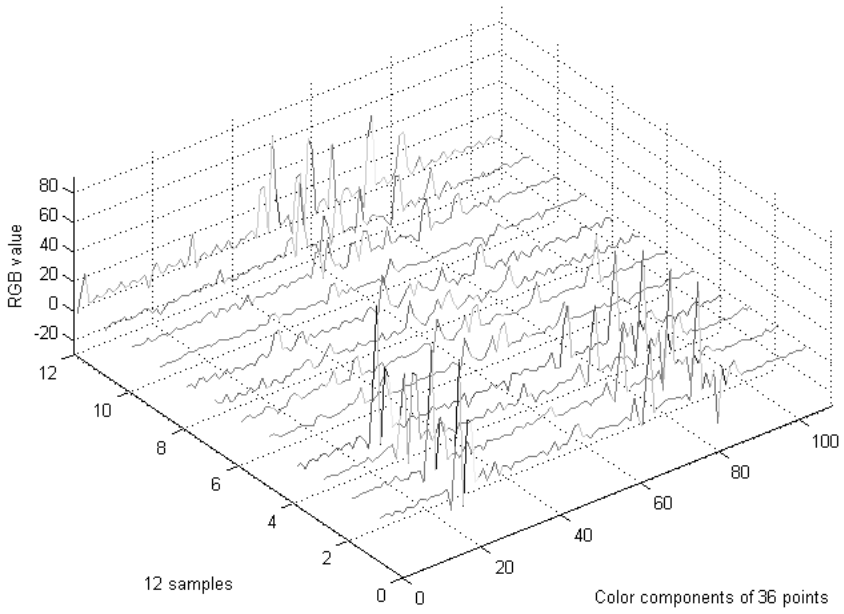

图 2.6 12 个发散样本的数据

Fig 2.6 Data of 12 divergent samples

为了方便区分，每个样本均被编号。将图 1.2 中的样本(a)、(e)、(i)分别命名为 ACE、NH3、BEN，表示丙酮、氨气和苯的理想样本数据；将(b)、(c)、(d)分别命名为 A1、A2、A3，表示丙酮的发散样本数据；将(f)、(g)、(h)分别命名为 N1、N2、N3，表示氨气的发散样本数据；将(j)、(k)、(l)分别命名为 B1、B2、B3，表示苯的发散样本数据。本文利用 SPSSInc18.0 软件的系统聚类功能对 12 个样本数据进行欧氏距离聚类。

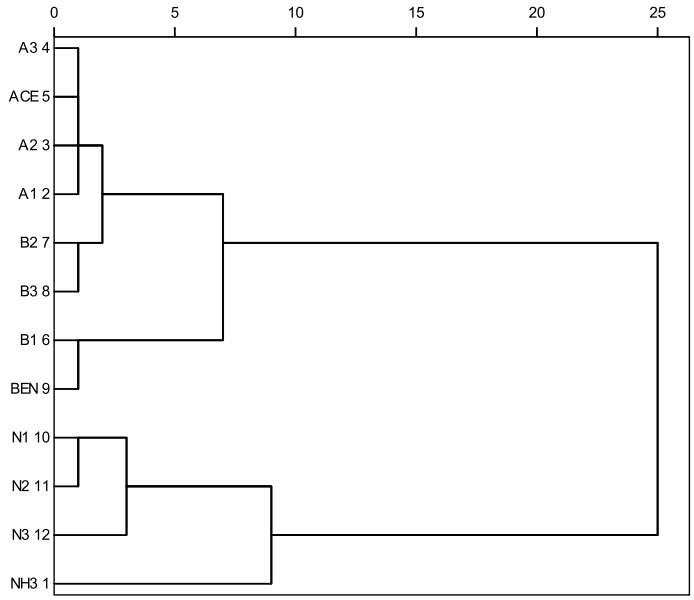

图 2.7 欧氏距离聚类结果

Fig 2.7 Results of Euclidean distance clustering

从聚类的结果可以看到，丙酮的理想样本 ACE 与发散样本 A1、A2、A3 被归为一类，苯的发散样本中只有 B1 与理想样本 BEN 归为一类，B2、B3 并没有被归类，而氨气的理想样本 NH3 虽然与发散样本 N1、N2、N3 被归为一类，但是两者的欧氏距离差距较大。因此，我们可以证明，在气体发散样本存在的情况下，利用欧氏距离聚类进行气体种类识别是不太准确的。

使用欧氏距离聚类有着明显的不足之处：在发散样本中，同种气体样本的差值图谱会有颜色或者显色点位置的变化，反映在空间中则是不同纬度值的变化。若简单地以距离作为分类标准，则不会得到令人满意的分类效果。

2.4.2 BP 神经网络的算法验证

一个实验样本包含 36 个卟啉点的 RGB 分量的欧氏距离值，因此输入节点数定为 36。训练目标为三类气体的标号。同样，为了方便，将氨气、丙酮和苯 3 种气体的类别变化分别编号为 1 号、2 号、3 号。并且以气体编号作为训练目标，即输出层的节点数设置为 1。隐含层节点数根据公式(2.2)(2.3)(2.4)确定出范围[6,35]，经多次实验比对选择隐含层的值为 8。BP 神经网络的学习率决定了输出误差对神经元各连接权值的修改幅度，通常取范围为[0,1]的小数。学习率过大不容易找到全局最小值，过小则训练次数增多，训练进度变慢。同样的，过小的预期误差也会增加训练的次数和时间。本文选择选择学习率为 0.3，预期误差为 0.0001。

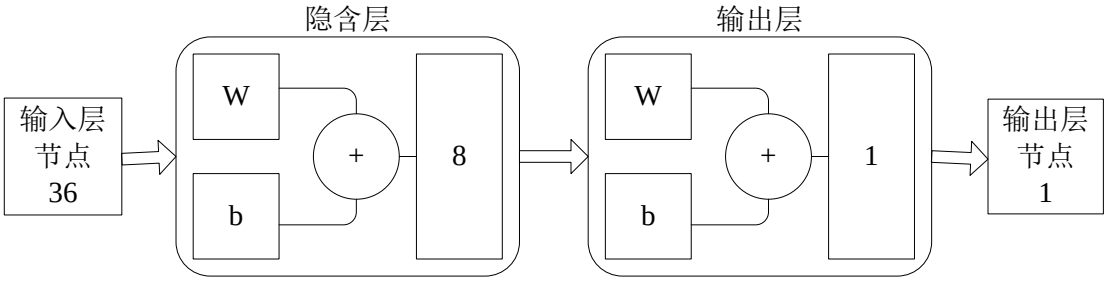

图 2.8 BP-ANN 训练网络结构

Fig 2.8 Structure of BP-ANN training network

本文在 3 种气体的平行实验数据中各挑选了 10 个理想样本中时间点为 5-6min 的数据，经过数据预处理后，组成一个 30×36 的向量作为训练数据。所谓的理想样本，是指在进行平行实验后，得到的差值图谱变色点的位置基本一致的样本。训练目标则定为每个样本对应的气体种类号，是一个 1×30 的向量。将训练样本和训练目标都输入到设定好的 BP 神经网络中，其训练误差收敛曲线如下图所示。当学习效率为 0.3，原定误差为 0.0001 时，需要训练 321 次可达到原定误差。

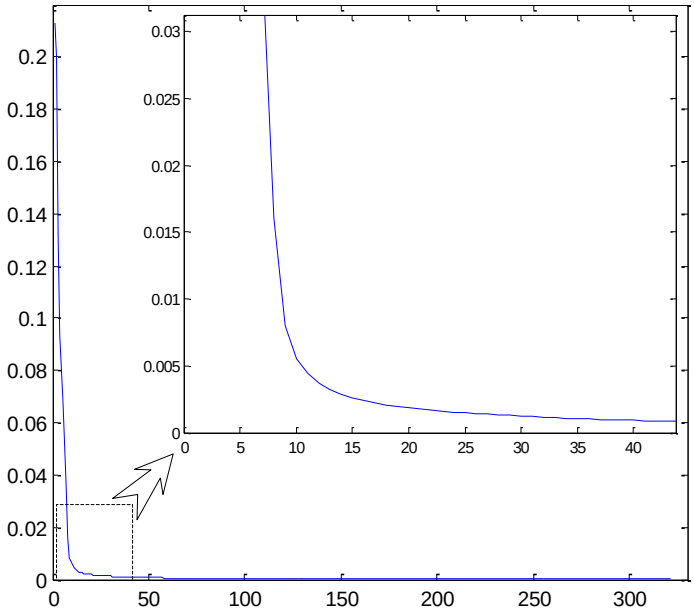

图 2.9 BP-ANN 训练误差收敛曲线

Fig 2.9 Curve of BP-ANN training error convergence

为了检验训练网络的泛化性，我们在训练样本中每种气体样本选出 3 个样本，共 9 个样本作为识别样本 1。另外，我们将图 1.2 所示的 3 组发散样本经过同样的数据约简后作为识别样本 2，两组识别样本的维度都是 9×36。将这两组识别样本都用训练好的神经网络进行识别，识别结果如下图所示。

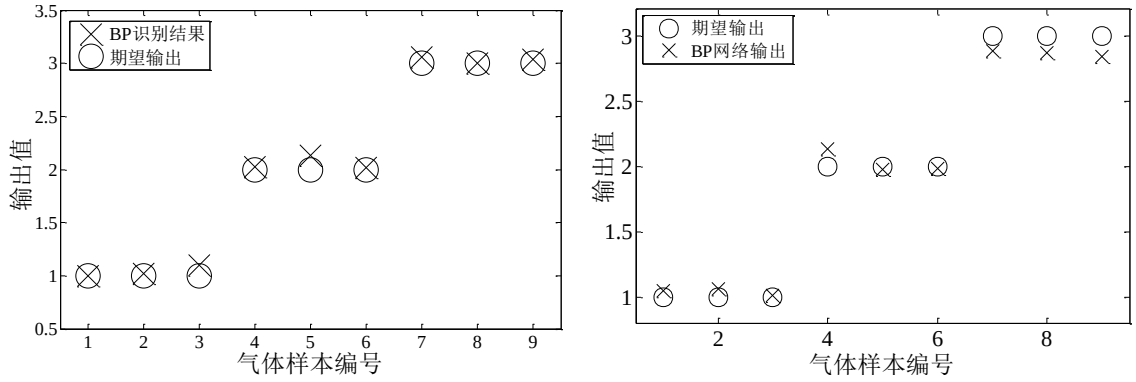

图 2.10 (a)样本 1 的识别结果；(b)样本 2 的识别结果

Fig 2.10 (a) Results of sample 1 (b) Results of sample 2

表 2.1 3 种气体 BP 网络识别输出与预期输出对比

Tab.2.1 Comparison between BP-ANN outputs and expected outputs of three gases

编号	样本 1 识别结果	样本 2 识别结果	期望输出
1	1.0136	1.0496	1
2	1.0307	1.0637	1
3	1.0864	1.0133	1
4	2.0096	2.1330	2
5	2.1180	1.9723	2
6	1.9953	1.9807	2
7	3.0139	2.8794	3
8	2.9961	2.8708	3
9	3.0135	2.8420	3

从结果可以看出，相比于欧氏距离聚类，BP 神经网络能很好地区分 9 种气体发散样本的种类。不论是对训练样本的识别还是对发散样本的识别，BP 神经网络都表现数量较好的识别效果，也证明了本文设计的 BP 神经网络的泛化性良好。

2.4.3 粗糙集约简的算法验证

由粗糙集中决策系统的定义可知，用差谱图数据构造决策表，U 表示所有的差谱图数据值，C 表示条件属性，即 36 个点经数据预处理后的值，D 表示决策属性，即每种气体的种类编号。本文根据 PSA 芯片上的 36 个点的排列位置为每个点都标上编号。每一个点在所有实验样本中的值是属性集 C 的一个子集，以该点的编号命名。不论是训练样本还是识别样本，均根据 2.3.2 小节介绍的约简步骤进行约简处理。

图 2.11 PSA 芯片上每个点的编号

Fig 2.11 Codes of dye spots in PSA chip

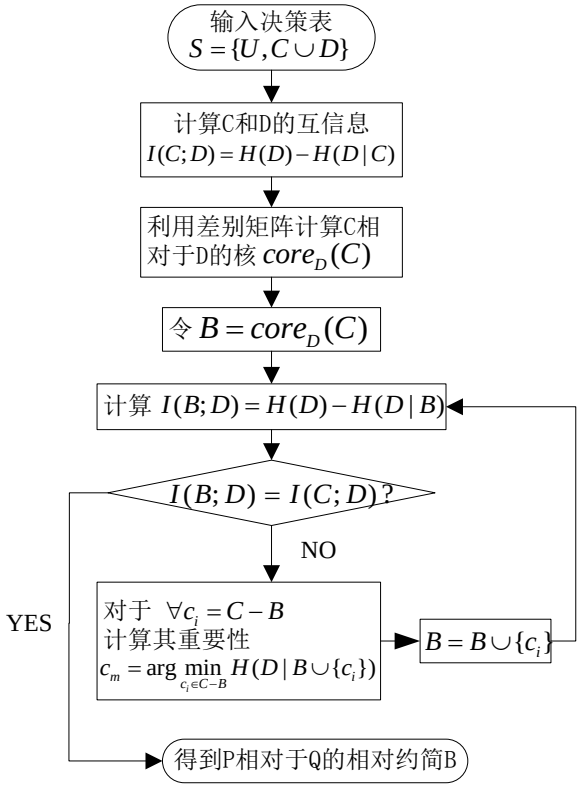

图 2.12 粗糙集约简流程图

Fig 2.12 Flowchart of rough set reduction algorithm based on information entropy

经过粗糙集约简后得到的特征属性为{17,22,16,9,8}。表示 PSA 芯片上编号为 17,22,16,9,8 的五个点。只保留这五个点的值，组成 30×5 的向量作为 BP 神经网络的训练样本。识别样本数据同样降为 9×5 的向量。利用 BP 神经网络对数据进行训练和识别。

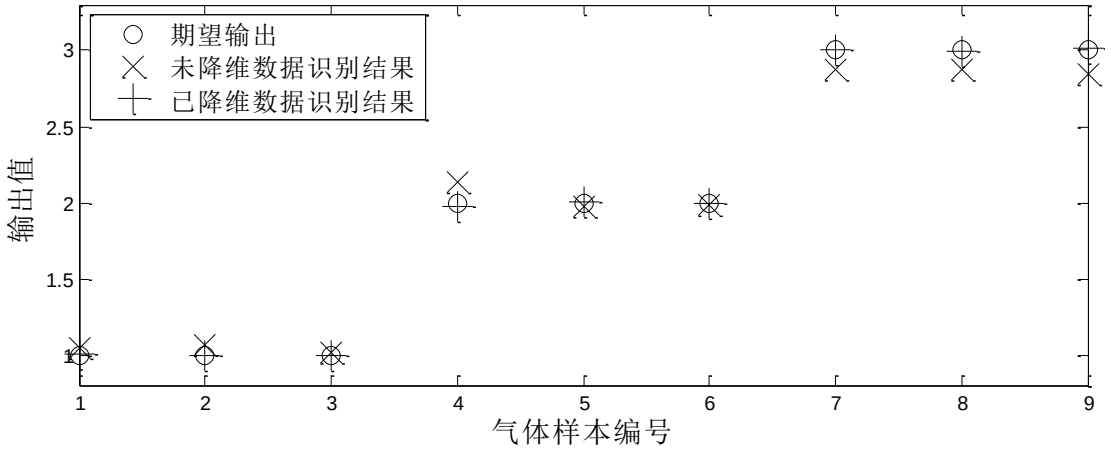

图 2.13 数据约简后 BP 网络输出结果

Fig 2.13 Outputs of BP-ANN that has data reduced

表 2.2 数据约简后的 BP 网络识别输出与预期输出对比

Tab.2.2 Comparison between expected outputs and BP-ANN outputs that has data reduced

编号	识别结果	期望输出
1	1.0015	1
2	0.9999	1
3	0.9990	1
4	1.9728	2
5	2.0009	2
6	1.9973	2
7	3.0011	3
8	2.9995	3
9	3.0151	3

本文将约简前后 BP 神经网络的识别结果与期望输出的差值称为识别误差，它可以反映识别效果的好坏。输出误差越小，表示识别的精度越大，误判为另一种气体的可能性就会相应减小。

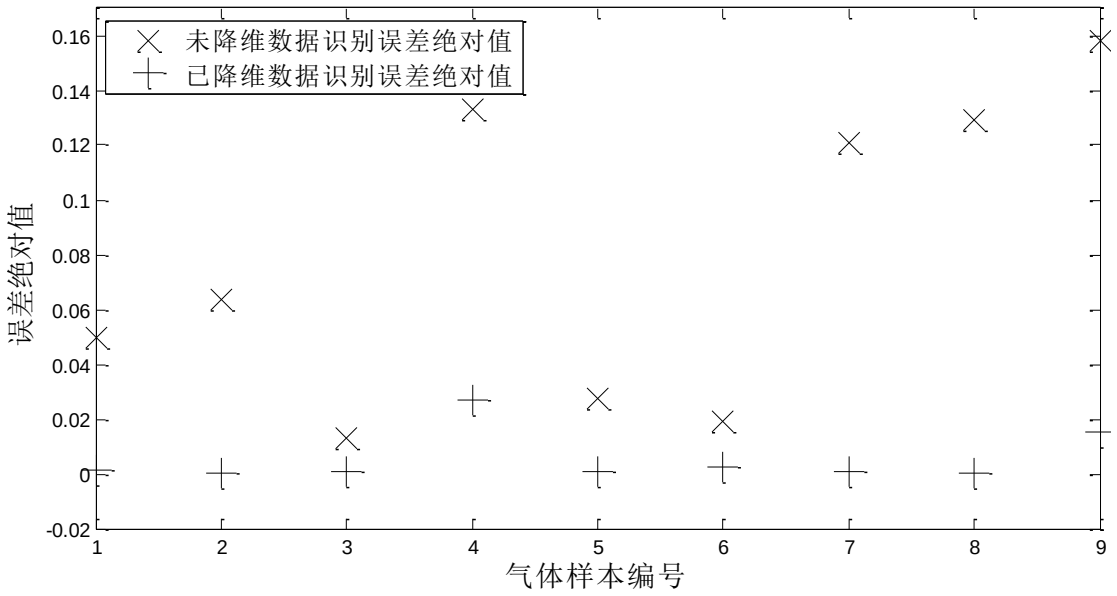

图 2.14 数据约简前后 BP 网络识别误差对比

Fig 2.14 Comparison of BP-ANN identification errors before and after data reduction

图 2.14 可以证明，BP 神经网络以降维前数据作为输入时，识别误差最大值为 0.16；以降维后数据作为输入时，识别误差最大值为 0.02，下降了 87.5%。可见，利用粗糙集对神经网络输入数据进行约简处理后，不仅减少了冗余数据，降低了 BP 神经网络训练与识别的数据量，同时提高了识别精度。因此，结合粗糙集和 BP 神经网络的气体种类识别算法可以很好地实现气体种类的识别。

2.5 本章小结

本章介绍了 BP 神经网络结构设计的方法，并结合本课题的实验仔细地选择合适的设计参数。同时介绍了粗糙集中应用于本课题研究的一些概念、定义和定理。本章还详细地介绍了本课题结合 BP 神经网络和粗糙集进行气体种类识别算法的整体流程。

本章对本文提出的算法进行了验证。通过对实验结果进行分析，在对发散样本进行种类识别时，简单的欧氏距离聚类识别的误差较大，而 BP 神经网络能够将所有发散气体样本进行正确归类。说明 BP 神经网络无论在识别精度上还是算法结构上都比欧氏距离聚类更适用于发散气体样本的种类识别。将数据利用粗糙集进行进一步约简和提取后，由于只保留了特征属性，减少了其他冗余数据的干扰，使得 BP 神经网络的运算效率得到提高，从而提高了识别精度。经过粗糙集约简后的 BP 网络识别误差从最大的 0.16 降到了 0.02 以下。因此，对于 PSA 系统的检测结果，特别是存在发散样本的情况下，本文提出的算法具有较高的识别精度。

3 算法在 PC 系统的实现

在前面的章节中，详细地介绍了本文设计的气体种类识别算法的理论基础和处理流程，并用实验数据验证了该算法的有效性。本文根据算法的计算步骤，在VC++的开发环境下，对算法进行编程实现，并加载到 PSA 气体检测系统的 PC 版本的系统软件中。

3.1 编程环境：VC++

早期的计算机系统软件通过汇编语言实现程序的编写和运行。汇编语言可以对硬件直接操作，运行速度快，但是程序可移植性和可读性并不好。C 语言不仅克服了汇编语言这一缺陷，还保留了其可以对硬件、内存地址等进行直接操作的优点。C++通过把 C 语言的数据和对数据的操作封装在类、对象等机制，同时增加派生、继承、重载、多态性等特征，实现了 C 语言面向对象开发软件的功能[54]。

VC++（Microsoft Visual C++）是在 Windows 下开发人员使用的面向对象的可视化集成编程的专业 C++ SDK（Standard Software Develop Kit，专业软件开发平台），可以进行 C、C++、CLI 等编程语言的程序开发。在 2007 年 11 月，Microsoft 发布了 VC++9.0 版本，该版本集成在 Visual Studio 2008 中，是目前为止最稳定的 VC++版本。VS2008 具有以下四个特点：

(1) 领先的数据库工具，具有对 Microsoft SQLServer、DB、Oracle 等数据库的完全往返工程功能；
(2) 灵活的软件建模，支持标准建模语言的图表符号，为 Microsoft Studio.NET 的创建提供了基础；
(3) 高效的体系结构，允许开发人员和定义程序的初始工具包、策略以及体系指导，即使是经验不足的开发人员也可以构建复杂的应用程序；
(4) 集成的源代码控制，包括集成测试功能、存根页、自动化、回归测试、分析瓶颈、版本控制、跟踪更改、并行开发等功能。

3.2 软件开发流程

对于绝大多数的仪器而言，软件是其主要的价值体现[55]。开发出来的软件质量的好坏关系着这个仪器的运行状态。作为保证软件质量的必要条件，一套规范化的开发流程必不可少。在软件工程学的角度，软件的开发可以分为五部分：

(1) 需求分析。包括用户需求（在本文中应该是系统功能设计要求）以及开发中遇到的所有的需求，是一个回答做什么的过程。需求分析就是对用户需求进行正

确的理解，浓缩，再利用软件开发语言表达出来，然后建立软件的逻辑模型的过程。需求分析的主要方法包括结构化分析方法、数据流程图和数据字典等；

(2) 软件设计。包括概要设计（也称为系统设计）和详细设计两个阶段。概要设计是一个将软件分解成模块的过程。在这个过程中，每个模块代表一个系统功能。在这个过程中，还包括设计系统的结构和流程、功能和接口的设计等。详细设计是在概要设计的基础上进行每个模块的程序流程、算法、数据结构、类层次划分和调用关系等过程。详细设计的工作细致而复杂，目的是保证软件的所有需求可以完全分配；

(3) 代码编写。这是将虚拟的设计变成可执行程序的过程。在这个阶段应该实现开发语言、开发环境、编程风格的选择，注意代码的可维护性，可重用性以及可扩展性，以保证开发出来的软件产品的质量。

(4) 软件测试。测试的目的是尽可能地发现软件产品的错误，确认每个功能的正确实现。测试的关键是设计完整的测试用例。测试的方法多样，不同的测试方法需要的测试用例并不一样。经典的测试有两种：黑盒测试和白盒测试。

(5) 软件维护。在软件产品开发完成并交付使用后，实际情况的变化会产生其他的需求，需要对软件产品进行适当修改、纠正错误等操作以使其适应新的需求。软件的维护可以提高软件的扩展性能，节约成本。因此在设计和编码阶段应该同时考虑代码的可维护性。

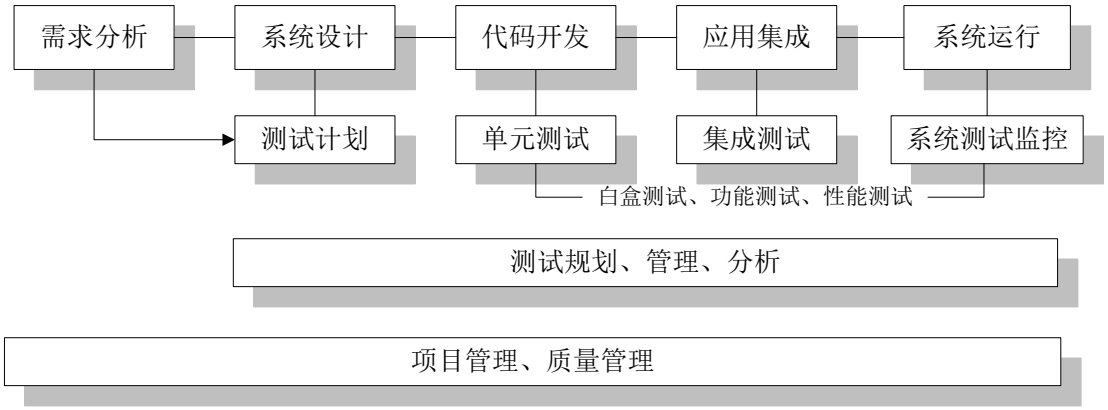

图 3.1 软件开发流程图

Fig 3.1 Software Development Flowchart

3.3 需求分析及框架设计

需求分析和框架设计对软件开发过程至关重要。对要开发的对象进行需求分析，是一个可以充分理解对象功能，对其有一个总体性认识的过程。框架设计则是将这个总体认识进行细化，结合开发人员对实际情况的思考，由形象到具体部

署整个开发流程的过程。两者是软件开发的基础。全面的需求分析结合详细的框架设计，是保证软件产品的良好质量的关键。

需求分析可以分为功能型需求和非功能性需求[56]。功能性需求（Functional Requirement）指的是开发对象需要通过软件实现的功能。对于本文的开发对象，即本文提出的气体种类识别算法而言，其功能需求是该算法各部分计算公式的编程需求的综合。非功能性需求（Non-Functional Requirement）是指对软件开发产品质量的要求和限制。由于本文的开发对象多是实现数据运算和数据存取功能，因此非功能性需求可以表现在运算效率、操作流程等方面的优化。

3.3.1 识别程序设计需求

在功能需求上，原有的叶啉气体检测系统可以分为四个功能：硬件通信、图像处理、模式识别和数据管理。模式识别功能可以实现对气体样本的定性和定量分析。因此，本文设计的算法程序应该加载在模式识别部分。

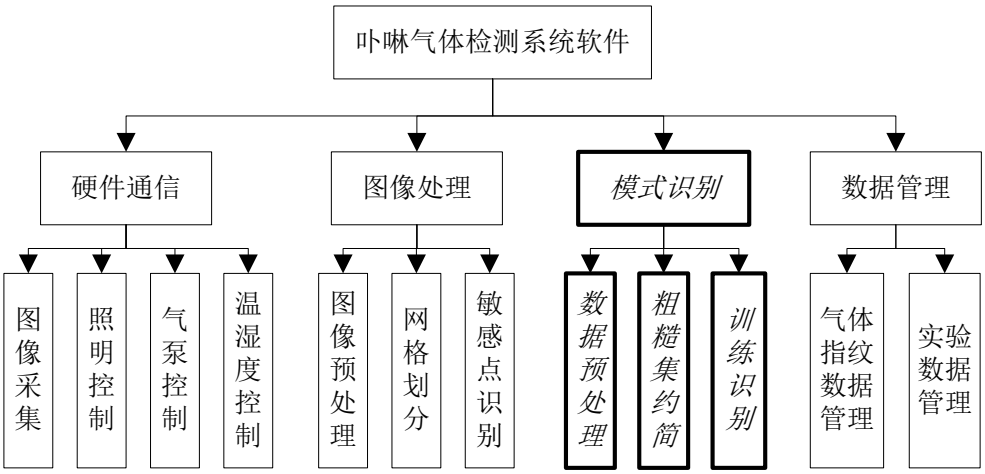

图 3.2 系统软件功能图

Fig 3.2 Diagram of software function

图像处理部分的主要功能是提取 CCD 图像采集设备采集的图像中每个点的像素值（即 RGB 分量），将同一个点反应前后的像素值相减，得到一组差值变量，这组差值变量即模式识别的输入。本文提出的种类识别算法的程序设计分为两部分实现：对实验数据的训练和识别。神经网络识别部分会加载到原系统的图像处理程序之后，在图像处理完成之后直接运行。加载位置如下图所示：

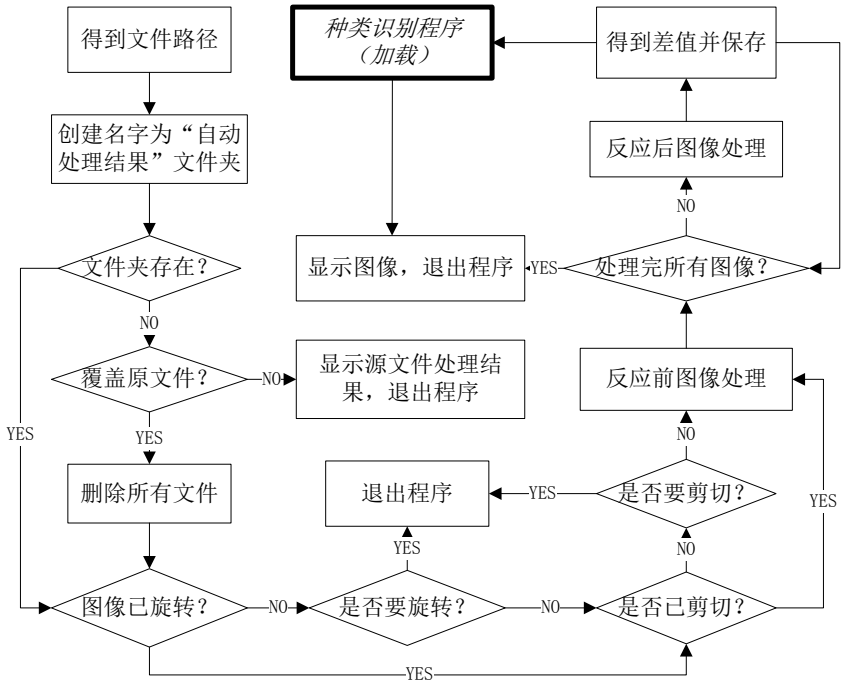

图 3.3 模式识别程序加载示意图

Fig 3.3 Schematic diagram of image processing

根据 2.1 节和 2.3 节介绍的内容，还可以整理出新的气体种类识别算法的神经网络识别部分具体的处理步骤。

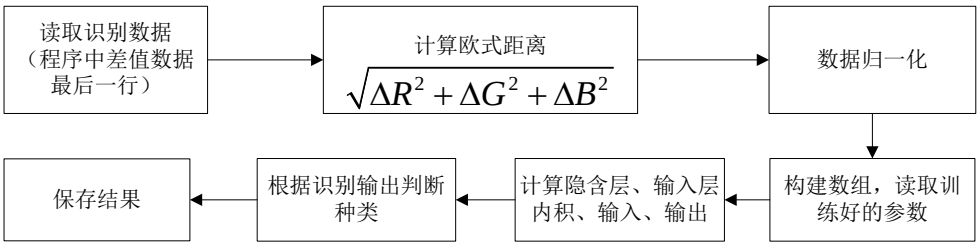

图 3.4 识别程序流程图

Fig 3.4 Schematic diagram of recognition

根据整理出来的处理流程，可以将识别程序设计分为六个模块：读取识别数据、计算欧氏距离、数据归一化、读取训练参数、计算隐含层和输出层的输入输出、判断种类、保存结果。

识别程序的功能型需求：

(1) 读取识别数据。在识别时选择反应后期的数据作为识别输入。在图像处理程序中，处理后的数据存放在一个数组中。可以直接利用该数组作为处理对象。

(2) 计算欧氏距离。根据公式(2.19)可知，此处需要用到计算绝对值、平方、开方

的函数。依照尽量使用库函数的原则，本文选择了 C 语言计算标准库的函数，这些函数定义在头文件 math.h 中。

(3) 数据归一化。本函数实现的功能有：选择数组中的最大值和最小值，并根据公式(2.10)计算数组元素中归一化后的值。

(4) 读取训练参数。实现打开保存阈值、权值和训练目标的文件，构造数组的功能。

(5) 计算隐含层和输出层的输入输出。是识别程序的主体部分，利用读取的阈值权值根据激励公式计算识别对象的各层内积，得到识别结果。

(6) 判断种类。根据识别结果与训练目标的差判断所属气体种类。

(7) 保存结果。识别结果的保存对读写的速度要求较低，但对存储空间、数据管理等方面的要求较高。因此，对于识别结果，数据库是比较理想的存储媒介。

识别程序的非功能型需求：识别程序处理的数据量小、计算步骤简单，但是运行次数多，对运算速度要求高。

3.3.2 训练程序设计需求

气体种类识别算法的准确度高低直接取决于训练精度，与识别程序相比，不论是程序开发还是运算方法，训练程序都比较繁琐。其设计需求可以分为四部分：界面设计、读入数据格式设计、程序流程设计、阈值保存和输出格式设计。

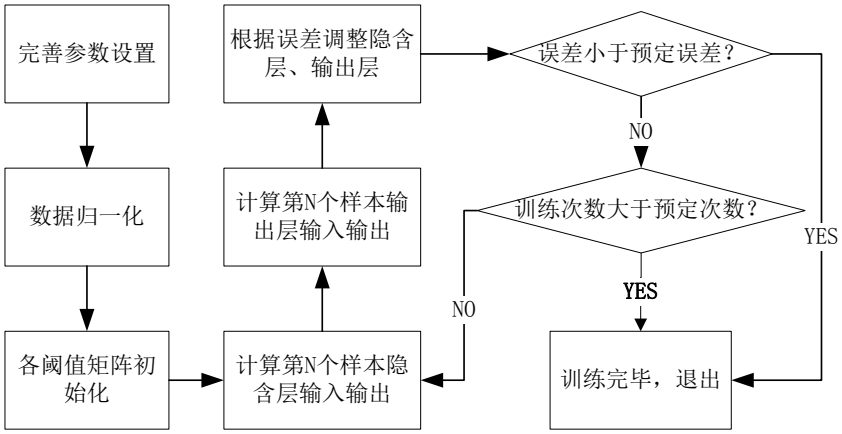

图 3.5 训练程序流程图

Fig 3.5 Schematic diagram of BP training

训练程序的功能型需求：

(1) 界面设计。包含能够读取文件和触发训练功能的按钮、训练需要的参数、训练状态和结果的展示。软件开发对于界面设计有着最基本的要求，即易用性、规范性和安全性。

(2) 训练程序需要规定其输入和输出数据的格式，包括训练数据、目标数据的输入

方式、各节点的阈值权值的保存形式等。训练程序的输入数据数量众多，输出的阈值和权值相对较少。大量的训练数据便可以存在数据管理方便的文件中，比如 Excel 表格、Access 文件等。

(3) 程序流程设计。定义需要调用到的子函数功能（与识别程序基本一致），在 VC++平台上建立训练程序的工程。

(4) 阈值保存和输出格式设计。阈值和权值需要实现程序之间的传递。考虑程序读写的效率，可以选择保存为一些读取简单的文件类型，比如系统初始化文件 ini 文件或者 txt 文件等。

训练程序的非功能型需求：训练程序处理的数据量庞大，由于训练的精度直接影响到识别精度，因此运算复杂，需要占用大量内存，运算时间也会增加。但是对运算速度的要求并不如识别程序高。

3.4 算法的实现

在 3.1 节中提到了在现行的 VC++版本中，VC++9.0 是最为稳定的版本（集成于 VS2008 中），并具有领先的数据库工具、灵活的软件建模、高效的体系结构和集成的源代码控制等优点。因此，本算法选择 VC++9.0 版本作为开发平台。

3.4.1 代码设计流程

代码设计是使用编程语言中有限的基本控制结构来实现算法逻辑的过程。通过对基本控制结构进行组合和嵌套，可以实现逻辑复杂的结构。为了增加程序代码的可读性、可维护性和可执行性，代码编写需要遵循特定的标准。

(1) 在源程序中有大量的标识符，包括模块名、变量名、常量名、子函数名等等。每个标识符都有其功能，这应该从它们的名字上反映出来。如识别程序、训练程序中各子函数名可以翻译成英文，避免用拼音命名。

(2) 源程序中应该保持一定数量的注释，这是开发人员与读者之间通信的重要手段。所以在识别程序和训练程序编写代码的过程中，可以在每个子程序前加入解释该函数功能的注释。

(3) 首先考虑功能的正确性与代码的可维护性，而后考虑性能的优化。在设计中要尽可能使用开发环境提供的库函数和方法。如本课题在进行计算时使用 math.h 定义的计算函数，存取文件时使用 ofstream 类中的成员函数，连接数据库时使用原系统软件已有的类的函数等。

(4) 编码之前使用伪码来描述程序的流程，再翻译成编程语言。各模块的功能需要尽可能单一，并确保每一个模块的独立性。本课题先将各函数的结构用伪码表示出来，再根据伪码编写代码。

(5) 数据结构的选择要以有利于程序简化为前提，并且应该从数据出发编写程序。

在识别和训练程序中均使用数组作为保存数据训练的容器。

(6) 复杂的大程序要进行分块编写和测试，之后集成。本课题中首先完成每个模块的接口定义，然后对每一个模块进行开发和测试，最后进行联调。

3.4.2 数组与动态内存分配

数组是同种数据类型的变量按顺序进行存储形成的集合。一个数组内的所有元素的存储地址是连续的，当数组的元素个数确定之后，数组的长度不会改变。因此，只要知道数组第一个元素的地址，就可以方便读取任意位置的数据，访问速度快。在本文中，读入内存中的运算数据均是保存在数组中。

数组的构造方法有静态和动态之分。静态数组是指在定义的时候就指定了长度的数组，动态数组是在分配存储空间时才给出数组的具体长度。在程序运行时，静态数组的元素个数是不变的，动态数组的元素个数通过变量决定。静态数组采用数组定义语句向系统申请内存空间；动态数组采用函数 malloc()、new()等函数来向系统申请内存空间。从使用方法上来说，静态数组和动态数组是一样的[57]。

在训练程序中，由于用户输入的训练样本数并不是固定的，因此读取数据时不能通过定义静态数组来实现。在识别程序中，存储特征变量的数组大小取决于 PSA 芯片叶啉点的个数。为了适应不同数目的叶啉点，识别程序采用动态数组来存储特征变量。

在处理数据的量方面，训练程序远大于识别程序。为了保持代码的可读性，很好地区别出不同训练样本，训练程序使用了二维数组作为存储容器。二维数组可以看作是一个特殊的一维数组，在实际的内存中，二维数组与一维数组一样，存在地址连续的内存中。使用二维数组可以清楚地将每个训练样本，以及一个训练样本中的每个点值以行列的形式表示出来。

静态数组，或者局部变量数组，申请的内存地址位于栈空间，动态数组的地址申请的内存地址位于堆空间。栈区（stack）使用一级缓存，内存空间由编译器自动释放和分配，堆区（heap）使用二级缓存，内存空间由程序员申请和释放，在程序结束的时候由 OS 回收。因此，对于动态数组，在使用完成该数组后，或者子程序结束时，应该使用 delete 函数将该动态数组的指针所指区域进行手动释放。不释放的地址不会被后面定义的变量使用，很容易造成内存泄露。由于计算过程中原始指针的值会改变，而 delete 释放的是以该指针所指向的地址作为初始地址的内存空间。所以在训练和识别程序中，动态数组的指针均应该备份，并且在使用完毕后通过 delete 释放备份指针所指向的内存。

3.4.3 识别程序的实现

下面列出最主要的气体识别函数的伪码设计，其余函数的伪码设计详见附录。

函数名：气体识别函数
输入：保存从差值图谱提取得到的 RGB 差值的数组指针
输出：程序运行状态变量
开始运行：
调用 new[]，构建动态 double 数组 sample, rgbdata;//保存 RGB 差值, 欧式距离值
备份两数组指针;
FOR 循环，将 RGB 差值复制到数组 sample;
调用计算欧式距离函数;
调用归一化函数;
调用约简点选择函数;
读取配置文件中粗糙集约简保留的点个数和位置坐标;
根据点的个数动态生成数组;
将数组相对应的位置数据保存成新数组 rgbdatanew;
FOR 循环，i=0，i 等于行列乘积停止
动态生成二维数组指针 W**;//用于保存输入层到隐含层的权值矩阵
FOR 循环，i=0，i 等于行列乘积停止
动态生成二维数组指针 V**;//用于保存隐含层到输出层的权值矩阵
分别动态生成保存隐含层输入输出和阈值、输出层的输入输出和阈值的数组;
生成动态数组保存识别结果;
读取 txt 文件，为每个数组赋值;
FOR 循环，i=0，i 等于隐含层数停止
计算隐含层内积的输入输出;
FOR 循环，i=0，i 等于输出层数停止
计算输出层内积的输入输出;
delete 各数组指针;
调用函数判断气体所属类别;
调用函数保存识别结果以及识别的差值数据;
return;

气体识别函数共调用了欧式距离计算函数、归一化函数、约简点选择函数、气体种类判别函数、保存识别结果函数、保存样本数据函数以及读取训练参数函数。下面简述各函数的功能：

计算欧氏距离函数：将原始的差值数值根据欧氏距离公式计算出每个点的欧式距离值，数据维度由1×108降为1×36。

归一化函数：找出全部 36 个点的欧式距离值中的最大值和最小值，根据归一化公式将每个数转成 0 到 1 之间的小数。

约简点选择函数：根据读入的粗糙集选出来需要保留的点的位置，将归一化后的数值进行重组，只留下需要保留的点的值。

气体种类判别函数：首先将识别出来的结果与各种类气体的训练目标相减，找到最小的差值，以此判断气体属于哪种类型。如果差值小到一定的范围，该样本气体将会被保存，成为修正训练程序的训练样本。同时根据（1-差值/100）%计算识别样本与训练目标的相似度，表示该样本与所判断的气体种类的接近程度。

保存识别结果函数：通过调用原系统已有的连接 Excel 文件类的功能函数实现。将识别结果存入 Excel 文件中[58]，与特征值向量和差值图谱保存在一起。

保存样本数据函数：通过调用原系统已有的连接 Access 数据库的功能函数实现，将特征差值、相应的训练目标保存到 Access 数据库中。

读取训练参数函数：使用 C++的 ifstream 类中的成员函数实现 txt 文件的打开、读取和关闭操作。

程序代码开发完成后，对其进行编译，修改编译错误。之后进行链接，开始运行程序。点击主界面的“图像处理”按钮，进入图像处理界面。在图像处理界面中点击加载项目按钮，加载实验处理得到的 PSA 照片。对照片进行剪裁、旋转等处理后，点击自动处理按钮，程序进行图像处理和对处理结果进行识别。识别的结果如下图所示：

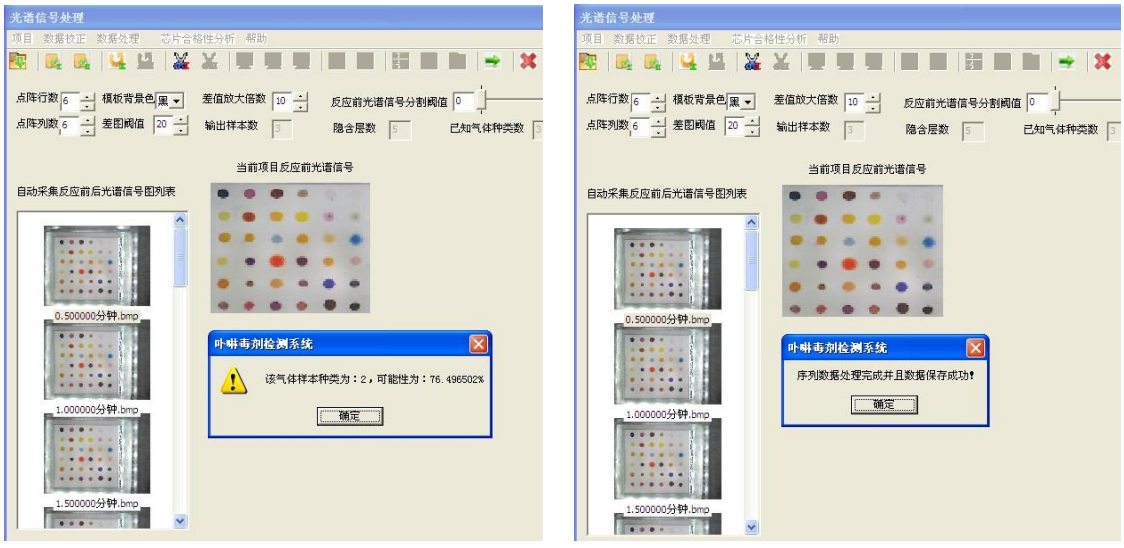

图 3.6 (a)识别程序运行效果图 (b) 识别程序保存数据效果图

Fig 3.6 (a) Interface of running recognition program (b) Interface of saving data

识别程序运行后，结果以对话框的形式传递给操作人员。传递的信息包括识别出来的气体样本种类编号，以及与该种气体的接近程度（或者概率）。气体的样本种类编号由训练目标的排列顺序决定。同时，识别结束后会将识别结果与图像处理结果一起存放在 Excel 表格中。

将图 1.2 所示的 9 组发散样本依次作为 PSA 系统软件中识别程序的输入。参照 2.4 小节的样本编号方法，1-3 号的为氨气，4-6 号为丙酮，7-9 号为苯；氨气、丙酮和苯的气体种类编号分别为 1、2、3。运行识别程序，按照 2.3 节介绍的整体算法流程进行数据处理，得到的识别结果如下：

表 3.1 加载在 PSA 系统中的识别程序的识别结果

Fig.3.1 recognition results of the recognition program added in the PSA system

发散样本编号	识别出的气体种类	相似度
1	1	94.634%
2	1	97.613%
3	1	93.668%
4	2	76.596%
5	2	73.883%
6	2	78.174%
7	3	66.444%
8	3	72.650%
9	3	72.202%

表 4.1 表明，算法在 PSA 系统软件中实现后对 9 组发散样本的识别正确率为 100%，体现了该算法程序良好的识别正确率。每个发散样本与其被判定所属的气体种类的相似度均为 60%以上。

3.4.4 训练程序的实现

结合 3.3.2 节的训练程序算法流程设计和功能需求分析结果，可以将训练程序分为两部分：交互界面设计和算法实现。

图 3.7 训练程序界面

Fig 3.7 Interface of training program

整个界面分为三个部分：选择训练文件、设置参数、状态显示。单击“浏览”会弹出一个名为“选择样本”的对话框，用户可以选择需要训练的样本数据文件。由于 Access 数据库和 txt 文件的数据均可以方便地转化成 Excel 表，因此这里选择 Excel 作为训练程序的样本数据文件类型。指定文件类型的功能通过“选择样本”对话框中的文件类型过滤功能实现。

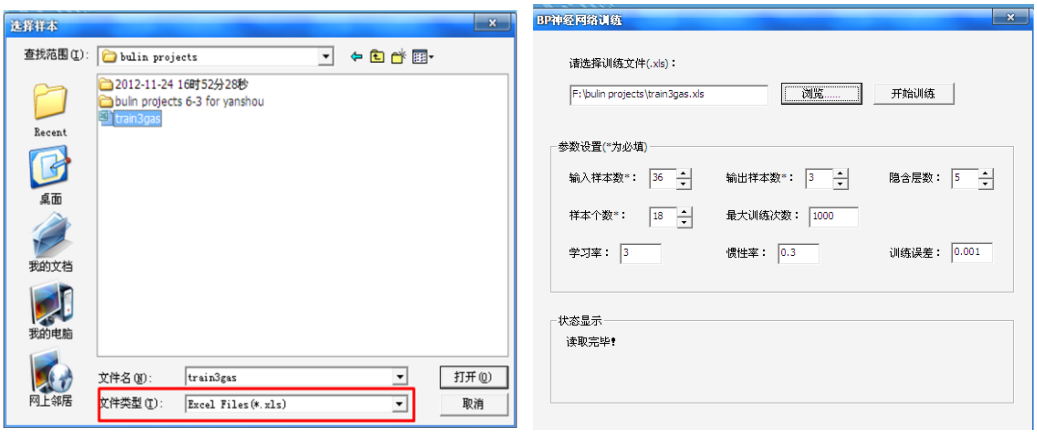

图 3.8 (a) 选择样本对话框 (b) 数据读取完毕界面

Fig 3.8 (a) Interface of choosing the file of sample data, (b) Interface of loading data

训练文件选择完毕，“浏览”按钮旁边的文本框将显示样本数据文件所在路径。打开保存样本数据的文件后，程序将自动读取数据内容，并保存至内存数组中。待全部数据读取完毕，在“状态显示”栏将显示“读取完毕!”字样。

在参数设置部分，共有 9 个参数可以手动设置。为了方便用户，在对话框初始化时，将先读取配置文件中上次对话框运行时保存的各参数值，并显示出来。因此用户可以方便地根据本次训练样本的特点修改相应的参数值即可，避免了用户每次启动训练界面要输入大量参数的麻烦。

设置参数完成后，单击“开始训练”按钮，训练程序启动。训练结束后，在在“状态显示”栏将显示“训练完毕!”，并显示训练的输出误差以及训练次数。同时，训练程序将训练好的权值、阈值、训练目标、用户设置的训练参数等参数分别保存至 txt 文件和 ini 文件中，供识别程序使用。

图 3.9 训练完成后界面

Fig 3.9 Interface of training finished

训练程序有六个主要的成员函数，分别为读取数据函数、粗糙集约简函数、开始训练函数、初始化数组函数、神经网络训练函数和保存参数函数。以下列出训练控制函数“开始训练”和训练执行函数“神经网络训练”的伪码设计，其他成员函数的伪码设计详见附录。

函数名：开始训练
输入：样本数据
输出：训练好的参数
开始运行：
UpdateData(TRUE);
检测读取文件 FLAG 是否为 1;
检测是否有参数输入为空;
调用粗糙集约简函数;
调用约简点选择函数;
调用数组初始化函数;
DO
FOR i=0，i 小于样本个数运行
调用训练函数;
调用得到训练误差函数;
训练次数+1;
IF 训练次数大于预定训练次数
将与状态显示的文本框关联的变量赋值“训练过程出错!”;
break;
while（训练误差大于预定误差）
将训练状态更新到界面;
打开 txt 文件，保存训练参数;
return:

函数名：神经网络训练
输入：样本编号
输出：训练好的数据
开始运行：

```
FOR i=0; i 小于隐含层节点数运行
    FOR j=0; j 小于输入层节点数运行
        sigma+=W[i][j]*traindata[m][i]; //求隐层内积
    X[j]=-sigma-sita[j]; //求隐层净输入
    M[j]=1(1+exp(-x[j])); //求隐层输出
同理，计算输出层内积、净输入、输出;
FOR j=0; j 小于隐含层节点数运行
    //计算调整量和误差
    Out_Gra[j]=(traindata[m][m_innode+j]-C[j])*C[j]*(1-C[j]);
trainer[m]+=fabs(traindata[m][m_innode+j]-C[j])*fabs(traindata[m][m_innode+j]-C[j]);
trainer[m]/=2;
FOR j=0;j 小于隐含层节点数运行
    FOR i=0;i 小于输出层节点数运行
        d_add+=V[j][i]*Out_Gra[i];
    Mid_Gra[j]=M[j]*(1-M[j])*d_add;
FOR i=0; i 小于输出层节点数运行
    FOR j=0; j 小于隐含层节点数运行
        V[i][j]+=m_alfa*M[j]*Out_Gra[i];//学习率参与调整权值
    Gama[i]+=m_fi*Out_Gra[i];//输出层输出阈值
FOR i=0;i 小于隐含层节点数运行
    FOR j=0;j 小于输入层节点数运行
        W[i][j]=m_fi*traindata[m][i]*Mid_Gra[j];//调整权值
    sita[j]+=m_fi*Mid_Gra[j];
    return;
```

读取数据函数通过连接 Excel 文件，使用 FOR 循环遍历文件中每行数据，并保存到指定变量所在的内存中。

约简点选择函数与识别程序的同名函数的功能相似。得到训练误差函数是将单个训练样本数组的各元素的训练误差求平均值，以改平均值作为训练误差。

初始化数组函数的作用是用生成动态数值的方式构造训练用到的保存各层的阈值、权值和训练结果的函数。构造数值完成后完成各数值指针的备份。在训练之前，输入层、隐含层、输出层的权值都将赋予随机数作为初始化。

保存参数函数主要是构造 MFC 已有的文件类生成 txt 文件，并将训练好的阈值、权值、训练参数、约简结果都保持到相应的 txt 文件中。

3.4.5 VC++与 MATLAB 混合编程

在面对复杂的数值运算时，VC++并没有优势。由于缺少完善的数值计算函数库，使得 VC++在实现复杂度较高的算法时，代码开发难度也较大。Matlab 软件是一款应用十分广泛的专业的数学软件。它包含了大量高度集成的数值运算函数库，有十分强大的数值分析、信号处理、矩阵运算等功能。在执行复杂的运算时，Matlab 无疑比 VC++有更高的效率和精度。VC++与 Matlab 结合，可以提高 VC++开发算法的计算精度。

在本文设计的算法中，粗糙集约简是比较关键的一步。与神经网络相比，粗糙集算法的运算流程比较复杂。单纯用 C 语言实现需要手动编写大量的运算函数，增加程序的复杂度。因此，本文选择利用 Matlab 的函数库来实现粗糙集约简部分。

VC++与 Matlab 混合编程的方式有多种。包括直接调用 Matlab 引擎、调用 Matlab 生成的 DLL 或者 exe 等文件、调用 Matlab 编程生成的 C/CPP 文件等。此外，还有方便 VC++与 Matlab 混合编程的中间工具如 comtool、Mideva 等。上述方法中，直接调用 Matlab 引擎并不被推荐，调用 Matlab 编程生成的 C/CPP 文件不需要进行任何转换，理论上是最为安全的方法，但是该 C/CPP 代码的可读性和可维护性都较低。所以本文选择调用 Matlab 生成的 DLL 的方式。该方式简单、快捷，虽然有不能脱离 Matlab 运算环境的要求，但是保证了运算精度。

Matlab 生成名为 libreduct 的 DLL 文件：

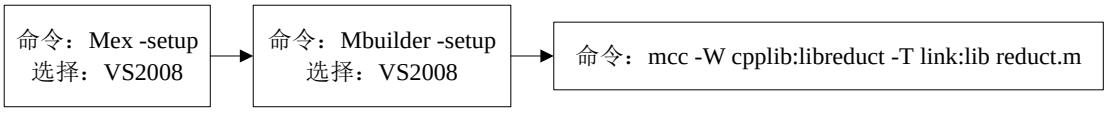

图 3.11 生成 DLL 文件流程图

Fig 3.11 Operation of making dynamic link library

VC++调用库 libreduct.lib 的环境配置：

(1) 项目属性页/配置属性/C-C++/常规/附加包含目录：Matlab 内的 include 目录；
(2) 项目属性页/配置属性/链接器/常规/附加库目录：Matlab 内的 lib 目录；
(3) 配置项目属性页/配置属性/链接器/输入/附加依赖性：libreduct.lib、mclmcrrt.lib、mclmcr.lib
(4) libreduct.lib、libreduct.h、libreduct.dll 复制到项目目录下，并添加到项目中。

函数名：粗糙集约简

输入：样本数据 sample

输出：约简后的样本数据

开始运行：

```
libreductInitialize();//初始化 libreduct 库，若失败，返回错误
TRY
    mwArray mSample (样本个数，输入节点数，mxDOUBLE_CLASS);
    //将数组转换成 matlab 矩阵形式
    mSample.setData(sample, 样本个数*输入节点数);
    mwArray mData (1，输入节点数，mxDOUBLE_CLASS);
    procreduction(1,mData,mSample);//调用粗糙集约简函数 procreduction
    生成动态数组 data，长度为输入节点数，用以保存返回数据;
    mData. GetData(data,输入节点数);
    libreductTerminate();//终止 lib 库
    mclTerminateApplication();//终止 MCR
CATCH( const mwException& e)
    return;
打开 txt 文件，保存约简保留点的个数和位置;
return;
```

3.5 本章小结

本章详细介绍了本文设计的算法在 PC 机环境下利用 VS2008 为开发平台的实现过程。本章的软件开发依照标准的开发流程，首先对要设计的算法进行了功能需求分析，对需要使用的数据结构和存储方式进行了初步的设计。然后按照代码设计流程进行代码编写。另外，根据程序界面设计的要求，设计了训练程序的界面。本文并没有将算法的实现局限于一种开发平台中，而是将比较复杂的粗糙集处理部分利用专业的数据处理工具 Matlab 生成，以动态库的方式被 VC 调用。

4 算法在嵌入式系统的实现

前章详细地介绍了本文设计的气体种类识别算法在 VC++开发平台下的实现过程。在本章中将介绍如何把 VC++下的程序移植到嵌入式的检测设备中。嵌入式检测设备以 Linux 为运行环境，以 Qt 为软件的运行和开发平台。本章将从环境的搭建、代码的移植、系统运行结果等方面详细说明在嵌入式系统中本文算法的实现过程。

4.1 嵌入式 Linux 和 Qt 概述

以前的嵌入式应用开发用的是汇编语言代码，所有的硬件驱动程序以及接口程序都需要开发者编写。而 Linux 的核心与公用程序都可以放入嵌入式设备的硬件资源中。嵌入式 Linux 操作系统是嵌入式操作系统中应用最广泛的[59]。与其他嵌入式操作系统相比，Linux 有许多优势：

(1) 代码开源，服务定制。开发人员可以修改系统代码以实现自定义功能，也可以通过重编译系统内核来定制服务。
(2) 兼容多硬件平台，有良好的可移植性。这使得应用软件可以在标准平台上开发好后移植到具体的硬件上，提高了开发质量。
(3) 有强大的网络功能，支持 TCP/IP 协议，以及更高层的 ftp、http、telnet 等协议。
(4) 成熟完善的 C 语言编辑器。有 GCC、G++等 C 语言编译器，可以方便地编译.c 和.cpp 文件，支持 C 语言开发
(5) 方便的文件管理。Linux 支持多种文件系统，包括 FAT32、EXT32、虚拟文件系统、连接文件系统等。
(6) 完备的开发支持。Linux 系统具有完整的开发工具链，庞大的开发人员基础，因此系统开发技术更加成熟。

由于 Linux 具有可定制的优点，通过修改和裁剪内核可以保留需要的功能，删除不需要的功能，修改可利用的功能，可以最大限度地压缩操作系统的大小。本文要移植的嵌入式检测设备保留了 USB2.0 驱动及 SD 卡驱动等必要驱动，对 Vedio4Linux2 摄像头驱动和 RS232 串口驱动进行了参数修改，同时删除了一些不必要的参数。

Linux 系统支持 GUI 应用软件开发平台。广泛地应用于 Linux 系统上的 GUI 开发应用软件工具包括：OpenGUI、MiniGUI、MicroWindows 和 Qt/Embedded。本文要移植的嵌入式检测设备的运行软件利用 Qt 开发完成。Qt 是一个跨平台的 C++图形用户界面库，与 VC 一样，都是面向对象的开发语言，易于扩展，并且允

许组件编程[60]。与其他开发工具相比，Qt 有如下优点：

(1) 优良的跨平台特性。Qt 可以支持多个操作系统的应用软件开发，包括：Windows，Windows CE / Mobile，Linux/X11，Embedded Linux，Mac，Symbian，Maemo/MeeGo，QNX 等等。

(2) 良好的封装机制。Qt 的模块化程度很高，重用性好，使得开发过程十分方便。以信号/槽（signals/slots）的回调机制代替 callback，简化了各元件之间的协调工作。

(3) 丰富的 API。Qt 提供了 500 多个 C++类和丰富的模板，整合了数据库、OpenGL 库、XML 库以及 WebKit 库，甚至还包括了正则表达式的处理功能。

(4) 有丰富的开发文档，为开发人员提供快捷的帮助。

Qt Creator 是一款轻量级跨平台集成开发环境 (IDE)。其设计目标是使开发人员利用 Qt 框架进行快速、便利的开发。Qt Creator 集成了 Qt Designer、图形化的 GDB 调试前端，集成 qmake 构建工具等，通过对编译环境和环境变量进行简单的设置，可以实现开发软件在目标设备上运行。本课题研究的嵌入式检测设备运行软件便是基于此 IDE 开发的。

4.2 嵌入式系统软件设计

嵌入式 PSA 气体检测系统与 PC 版本一样，有硬件通信、信号采集、图像处理、模式识别、数据库管理等功能。所有的功能均在 ARM/Linux 操作系统上，利用 Qt 为开发平台开发实现。用户通过 LCD 触摸屏操控系统的启动与运行。在嵌入式版本中，本文设计的气体种类识别算法加载在图像处理进程之后的模式识别功能模块中。

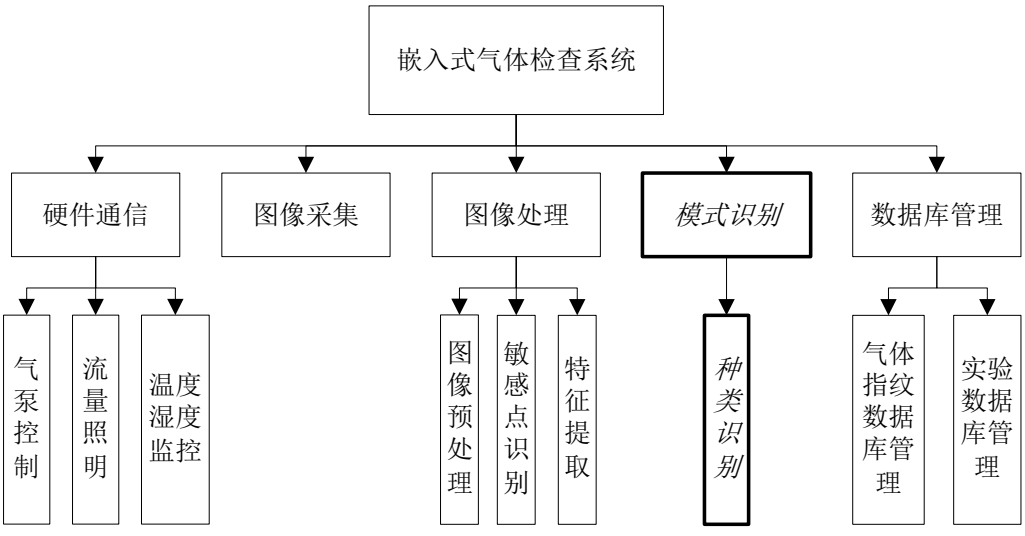

图 4.1 嵌入式叶啉气体检测系统功能结构图

Fig 4.1 Structure of embedded gas detection systems

在内核存储性能、外部设备功能、接口数量等硬件指标上，嵌入式设备远不如 PC 设备。硬件性能决定软件功能，开发适用于嵌入式系统的应用程序必须比 PC 机上运行的程序更加轻量级。对于数值运算的程序，由于嵌入式系统 CPU 运算速度的限制，程序的算法不可过于复杂，处理数据不能过多，否则将增加 CPU 的负荷，影响运行速度。

本文设计的气体种类识别算法中，训练程序因为包含了粗糙集和神经网络的运算公式，比识别程序要复杂的多。同时，训练程序由于要大量的训练样本，运行时所占用的系统内存也比识别程序要大的多，甚至会超过嵌入式内核的负载。再次，由于训练算法的使用次数有限，而识别算法才是检测系统最常用到的部分，因此本文只将识别程序移植到嵌入式气体检测系统中。

由于没有对样本进行训练的功能，为了保证嵌入式设备的识别精度，必须对识别程序在运行时读取的阈值权值进行不定时地更新。更新的来源便是运行在 PC 设备的训练程序。

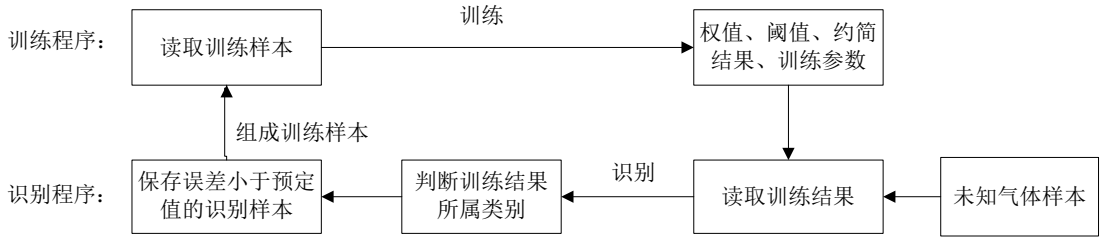

图 4.2 训练程序与识别程序的交互示意图

Fig 4.2 Schematic diagram of interacting between training program and recognition program

在识别程序中，识别完成后有一个判断气体所属类别的过程，通过计算识别结果与训练目标的差值，找到差值最小的训练目标，该目标对应的气体类型即识别样本所对应的类型。当该差值小于一定值时，我们便认为该识别样本与其被判定的类型足够匹配，因此可以将其值保留下来作为训练样本。在训练过程中，每个训练样本与其对应气体种类越匹配，训练出来的精度就会越高，而识别的精度也会相应提高。越高的识别精度意味着越少的识别错误，以及越多的高度匹配样本作为训练样本。因此形成了一个良好的、不断修正的训练识别运算循环。

4.3 程序移植

交叉编译是在嵌入式设备上运行软件常用的实现方式。交叉编译是指在一台宿主机（通常是 PC 机）上进行软件开发，然后通过主机编译器生成可执行文件，最后将该文件烧写到嵌入式设备上运行的过程。在本课题研究中，嵌入式气体检

测系统运行软件由宿主机在 Linux Fedora 9 操作系统下使用 Qt 下开发、编译、链接完成。嵌入式目标机选择广州友善之臂计算机有限公司提供的 Tiny6410 开发版，软件升级支持三星 2 代 MLC Nand Flash（MLC2），标准配备 256MRAM +2GBMLC2。

4.3.1 开发环境的搭建

常用构建交叉工具链的方法包括分布编译、网络下载、使用 Crosstool 工具。采用分布编译的方式构建交叉工具链的方法易错而且复杂，使用 Crosstool 工具构建交叉工具链比第一种方法简单，但耗时长。因此本文采用从友善之臂提供的网址上下载已制作完成的交叉编译工具链进行构建。具体步骤如下：

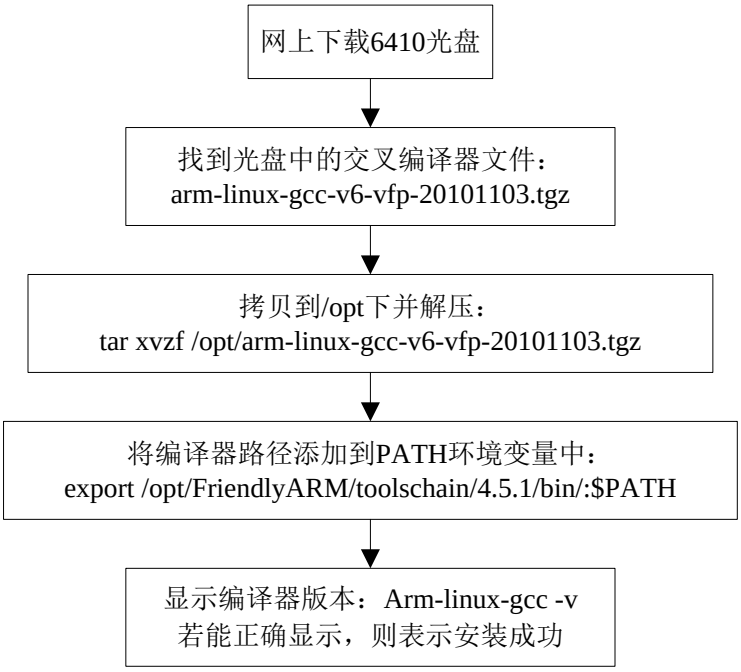

图 4.3 安装交叉编译器流程

Fig 4.3 Steps of cross-compiler installation

气体检测系统人机界面能够运行在嵌入式系统中，需要分别在宿主机安装 Qt。Qt Creator 是针对宿主机开发的 IDE，安装在操作系统为 Linux 的宿主机下。安装步骤如下：

(1) 下载 Qt Creator 的安装文件保存到 Linux 的自定义安装目录下，修改该文件的权限：chomd u+x 文件名，将其修改为可执行文件

(2) 执行该安装文件：./文件名，实现 Qt Creator 的安装。

Qt 是面向嵌入式应用开发的一个非桌面系统。4.7 版本是比较新的版本，有非常强大的类库，对于程序的加载非常方便。在宿主机上安装 Qt4.7 步骤如下：

(1) 在 6410 光盘下的 Qt4.7.0 的源代码包 qt-everywhere-opensource-src-4.7.0.tar.gz 拷贝到/opt 下并解压：

tar xvzf /opt/qt-everywhere-opensource-src-4.7.0.tar.gz

(2) 交叉编译 Qt4.7.0 的源代码：

echo yes | ./configure -prefix /opt/Qt4.7 -opensource -embedded arm -xplatform qws/linux-arm-g++ -no-webkit -qt-libtiff -qt-libmng -qt-mouse-tslib -qt-mouse-pc -no-mouse-linuxtp -no-neon

(3) 安装 Qt4.7.0：

make && make install

表 4.1 Qt4.7.0 交叉编译命令主要参数说明

Table 4.1 Description of important parameters of cross-compiler command

参数	说明
-embedded arm	将编译针对 arm 平台的 embedded 版本
-xplatform qws/linux-arm-g++	使用 arm-linux 交叉编译器进行编译
-qt-mouse-tslib	使用 tslib 库来驱动触摸屏
-prefix /opt/Qt4.7	Qt4.7 最终的安装路径

4.3.2 目标设备程序运行环境的建立

Qt 应用程序在嵌入式设备中运行之前，需要为该设备配置软件环境。主要包括三个镜像文件：Bootloader 引导程序、Linux 内核以及文件系统。文件系统的移植还包括 Qt 嵌入式共享库的移植。

在友善之臂提供的镜像文件中已经包含了 Bootloader 引导程序，名称为 supervivi-128M。移植过程十分方便，可以通过菜单搭载烧写环境进行烧写。移植到嵌入式设备的 Linux 内核版本为 2.6.32 版本，由于 Linux 内核有可定制的特点，在烧写之前可以先删除不必要的服务和驱动，压缩内核空间。然后对内核文件进行编译生成内核映像文件，将该文件搭载到 supervivi 中进行烧写。

文件系统的移植包括运行程序、库文件、设备文件和用户配置文件等文件的移植。与内核文件一样，文件系统在移植之前也可以进行删减操作。本文中嵌入式设备采用的文件系统为 yaffs，在友善之臂提供的压缩文件名字为 rootfs_qtopia_qt4-20110304.tar。采用的共享库版本为QTE4.7.0，路径为宿主机下的目录/opt/Qt4.7。文件系统的移植步骤如下：

(1) 将文件系统压缩包解压，将宿主机中的 QTE 共享库路径下的文件拷贝到“文件系统的解压路径/lib”目录下，完成共享库的移植。

(2) 将宿主机中触摸屏支持库路径下的文件拷贝到“文件系统的解压路径/usr/local/tslib”目录下，并修改“文件系统的解压路径/etc/”的“profile”文件，设置系统开机启动的环境变量，完成触摸屏支持库的移植。

(3) 通过 mkyaffsimage 生成 yaffs 文件系统的映像文件，将该文件搭载到 supervivi 中进行烧写。

4.3.3 系统开发软件移植

当目标设备的软件运行环境设置好以后，开发软件移植流程只需要几个步骤，如图 4.4 所示。

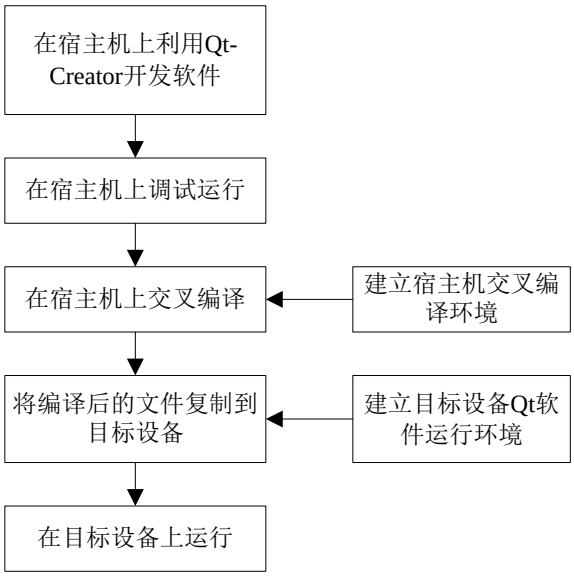

图 4.4 开发软件移植流程图

Fig 4.4 Flowchart of software transplantation

进行嵌入式设备开发软件的移植首先要在宿主机上进行开发。宿主机是通过 Windows XP 平台上的 VMware 虚拟机软件实现的。它可以在一台 PC 机上同时运行多个操作系统并方便切换。在每个操作系统中均能配置所需的软硬件环境。通过在 VMware 上安装 Linux Fedora 9 操作系统和 Qt 开发软件，设置内存为 1024MB，硬盘大小为 20GB，完成宿主机的开发环境搭建。

如图 4.1 所示，本文的识别程序移植在 PSA 气体检测系统嵌入式运行软件的图像处理部分之后。该图像处理模块共有三个处理步骤：图像预处理（灰度化、滤波等），网格划分和特征提取。当图像特征提取完成后运行识别程序。由于 Qt 软件支持大量的 C++标准库，对 C++开发有良好的兼容性，而识别程序并没有涉及到界面设计，主体代码均为 C++代码，因此识别程序的代码开发与在 VC++上的开发步骤基本一致。

运行虚拟机软件，打开设置好的操作系统电源，成功登陆后点击运行 Qt Creator，打开气体检测系统程序代码。在图像处理类程序中定义识别子程序及需要调用的其他子程序，包括计算欧氏距离函数、归一化函数、约简点选择函数、气体种类判别函数、保存样本数据函数、读取训练参数函数。其中保存样本数据函数利用 QFile 类将样本数据保存到 txt 文件中，读取训练参数函数设置了读取训练参数 txt 文件的路径，在运行时必须保证该路径下有相应的 txt 文件。

根据嵌入式系统对代码运行效率比 PC 机更严格的特点，对代码进行了优化，主要在以下几个方面：

(1) 长度小的变量类型所占内存少，可以提高运行效率。因此将识别代码中 double 类型的变量换成了 float 类型，虽然会对精度有一定的影响，但是保证了运算效率。

(2) 减少使用 C 语言标准库函数，用 Qt 封装的库函数代替。如在读取训练参数函数中将打开 txt 文件的 open、close 函数换成 Qt 封装的打开文件 QFile 类的打开和读取函数，将 string 类型变量换成 QString 类型变量。

(3) 使用位操作来代替除法，也是提高运算效率的一种方法。比如应用在判断气体种类函数中计算相似度的部分。

程序添加完毕后，点击 Qt 界面的 Build 按钮对整个程序进行交叉编译。如无语法错误，编译后将生成一个可执行文件。由于 Tiny6410 开发板支持 SD 卡启动，因此可以将该文件与保存训练程序得到的参数的 txt 文件一起拷贝到嵌入式设备中，将这些 txt 文件放在读取训练参数函数指定的目录下，并在超级终端窗口输入命令运行编译好的执行文件。

图 4.5 嵌入式系统程序欢迎界面

Fig 4.5 The program welcome interface

图 4.6 图像采集界面

Fig 4.6 Interface of image acquisition

在欢迎界面中单击“信号采集”按钮，系统将自动采集 PSA 芯片反应前后的照片，采集完毕后进入图像处理程序进行处理。处理结果作为识别程序的输入。

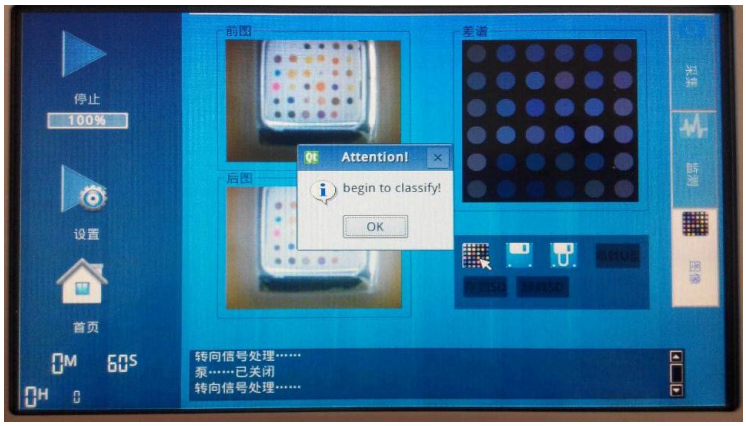

图 4.7 识别程序开始界面

Fig 4.7 Interface of recognition program began

图 4.8 识别结束界面

Fig 4.8 Interface of recognition program finished

与 PC 系统中的识别程序一样，嵌入式版本的识别程序经过计算欧氏距离、归一化、读取训练参数、计算各层内积后得到识别结果。识别结果以对话框的形式表现出来，包括判断气体样本所属气体种类的编号，以及是该种类的可能性。识别完成后，对于可能性高的样本将保留其图像处理的特征向量，作为训练样本。这些数据保存在 SD 卡由程序指定路径下的 txt 文件中，需要训练时将 SD 卡取出插入 PC 设备内，使用 Excel 文件将 txt 文件的数据导入，开始训练。

4.4 本章小结

本章详细介绍了本文设计的气体种类识别算法在嵌入式系统的实现过程。针对原嵌入式气体检测系统程序的运行流程，以及嵌入式设备的硬件特点，设计了嵌入式与宿主机数据交互流程。并且通过环境的搭建，将识别程序代码成功移植到了嵌入式检测系统中。

5 总结与展望

5.1 全文总结

本文设计了一种适用于 PSA 气体检测系统的气体种类识别算法。该算法结合了粗糙集的数据约简功能与 BP 神经网络的识别功能，以未知气体样本的差值图谱被特征提取处理后得到的颜色变化差值为处理对象，通过对这些差值进行降维、训练和识别，作为判断未知气体样本种类的方法。经实验证实，无论识别的实验数据是理想状态下还是识别发散状态，该方法均比简单的聚类分析有更好的识别效率和精度。在验证了该算法的可用性和准确性之后，本文详细介绍了该算法在 Windows 操作系统的 VC++平台的开发过程，以及将其移植到 Linux 操作系统的 Qt 平台的过程。

本文已完成的工作可归结如下：

(1) 本文分析了粗糙集算法与神经网络相结合的理论基础，设计了一种结合粗糙集与 BP 神经网络的气体识别算法及数据预处理的方法。通过与聚类分析的发散实验数据分类结果对比，证实了 BP 网络应用于气体模式识别的优越性；通过对比对训练样本与识别样本的识别结果，证实了 BP 网络设计良好的泛化性；通过对比粗糙集约简前后的识别结果，证实了本文算法良好的识别效率和识别正确率。

(2) 本文以 C++为编程语言，以 VC++为开发平台，将设计的算法分为识别和训练两部分，对其进行功能需求分析和框架设计，再分别编程实现，并加载至原 PSA 气体检测系统的运行软件中。识别程序加载到软件的图像处理部分，训练程序则封装成类，作为一个独立的功能模块供检测软件系统调用。

(3) 本文搭建了以 Linux 为操作系统，以 Qt-Creator 为集成开发平台的嵌入式开发环境，并详细依照嵌入式 C 语言开发要求优化了识别算法代码，成功将识别算法移植到嵌入式设备并运行。

(4) 本文同时设计了嵌入式设备与宿主机进行训练与识别数据交互的方案，实现了气体种类识别算法的工程应用。克服了嵌入式 ARM 开发板没有算法仿真功能、没有运行复杂算法基础的缺点，通过将 PC 机上的训练程序的训练参数移植到嵌入式中并完成识别程序的方法，成功地实现了嵌入式 ARM 开发板上气体种类识别的功能，改善了嵌入式 PSA 气体检测系统对气体进行检测和模式识别能力。

5.2 展望

本文所设计的气体种类识别算法能够应用于 PSA 气体检测系统，在理想状态下和发散状态下均能够识别气体样本所属类别，判断其相似度。在算法的设计和实现过程中仍有许多不足之处，希望在以后的研究中能得到进一步完善。

在算法设计方面，BP 神经网络具有训练和识别的功能，是算法的主体部分。目前，BP 神经网络的结构设计并没有一个统一的标准，隐含层节点数、隐含层数、权值阈值的初始化等参数设置均需要根据实验经验确定，这给网络的训练精度和识别结果带来了不确定的影响。在以后的研究中应该寻找优化 BP 神经网络结构的方法，更加深入地探索不同网络结构设计对结果的影响，以找到最适合 PSA 气体检测系统的网络结构，为识别准确率提供多一份保证。

在算法实现方面也存在可改善的地方。首先是训练程序的样本数据在本文中保存在 Excel 表格内，但随着日后样本数的增加，Excel 文件将不再适用。因此可以考虑将训练程序与数据库连接起来，并且实现能够手动选择需要训练的样本数据，方便日后进行大量数据的存取。在识别程序中，本文设计的代码结构也存在不足。为了方便各成员函数之间的数据传递，本文程序中定义了大量的动态数组和全局变量，在使用完毕后需要手动释放内存空间。在释放过程中由于数量过多容易造成遗漏，导致内存泄露等问题。因此需要设计一个自动释放内存的功能函数代替手动释放。另外，训练程序与识别程序实现数据交换的载体是 txt 文件，该文件方便易用，但容易被篡改。出于安全性考虑，可以对文件进行加密效果，比如修改文件扩展名等。

致 谢

本文的研究工作是在许多人的帮助下完成的。感谢我的导师罗小刚副教授。从课题的展开到完成，不论是宏观的设计框架还是细节的取舍，罗老师都提出了宝贵的意见，倾注了大量精力和心血。在工作中，罗老师严谨的学术作风、务实的工作态度为我树立了榜样，对我影响深远。

感谢课题组各位老师对我研究工作的支持和指导。感谢课题组所有成员营造的互通有无、相互交流、相互帮助的良好的科研氛围。感谢课题组的徐永红师姐对我进行实验指导，让我可以顺利完成气体检测实验，得到宝贵的实验数据。

感谢已毕业的刘平师姐、徐峰师兄在繁忙的工作中抽空为我解答软件开发过程中遇到的难题，并提供了宝贵的开发经验、方法和思路。

感谢实验室张佳佳、梁灿师妹、卜祥南师弟为我提供的帮助。在与张佳佳一起完成课题的过程中他为我提供了许多有用的建议，梁灿师妹和卜祥南师弟则在搭建嵌入式环境时给予我无私的协助。

感谢我的父母和家人。他们理解并支持我的研究工作，为我默默付出，让我无后顾之忧地完成学业。他们给予我精神上和物质上的支持是我无价的财富。

感谢评阅论文和参与答辩的各位专家、教授，感谢你们提出的宝贵意见。

谢新苹

二〇一四年四月 于重庆大学

参考文献

[1] 操乐杰. 肺癌筛查和早期诊断进展[J]. 临床肺科杂志, 2007, 12(6):531-532.

[2] 卢崇蓉, 胡燕婕, 陈恩国, 邱园华, 应可净. 肺癌患者呼出气体中特征性挥发性有机化合物的研究[J]. 中华和结合呼吸杂志, 2010, 33(2): 104-108.

[3] Cotte R I, Justes D R, Nanita S C, Noll R J, Mulligan C C, Sanders N L, Cooks R G. Analysis of gaseous toxic industrial compounds and chemical warfare agent simulants by atmospheric pressure ionization mass spectrometry[J]. Analyst, 2006, 131(4): 579-589.

[4] Orellana G, Moreno-Bondi M C. Frontiers in chemical sensors [C]. Wolfbeis O S. Springer Series on Chemical Sensors and Biosensors. Germany: Springer-Verlag Berlin Heidelberg. 2005: 93-132.

[5] 闫嘉. 基于电子鼻技术的人体伤口感染检测系统构建及算法研究[D]. 重庆: 重庆大学, 2012.

[6] 廖海洋, 吕建波, 侯长军等. 多点测量式毒气阵列检测系统研究与光谱分析[J].光电工程, 2009, 36(1): 68-72.

[7] MICHAEL C J, JENNIFER B P, DANIEL P B, et al. Colorimetric Sensor Arrays for Volatile Organic Compounds [J]. Anal. Chem., 2006, 78(11):3591-3600.

[8] 汪德暖. 基于嵌入式气体检测系统软件控制及算法研究[D]. 重庆: 重庆大学, 2011.

[9] 孙即祥. 现代模式识别[M]. 长沙：国防科技大学出版社, 2002.

[10] 罗磊. 基于模糊模式识别的光谱分类研究[D]. 河南：郑州大学, 2006.

[11] 张覃轶, 谢长生, 黄永彬. 气体传感器阵列常用模式识别算法[J]. 传感器技术, 2005, 24(5): 1-4.

[12] Schreyer S K, Mikkelsen S R. Chemometric analysis of square wave voltammograms for classification and quantitation of untreated beverage samples [J]. Sensors and Actuators B, 2000, 71:147-153.

[13] Claude D, Martine L, Maryam S. Discrimination and identification of a refrigerant gas in a humidity controlled atmosphere containing or not carbon dioxide: application to the electronic nose [J]. Sensors and Actuators B, 2004, 98:46-53.

[14] Dutta R, Hines E L, Cardner J W, et al. Tea quality prediction using a tin oxide-based electronic nose: an artificial intelligence approach [J]. Sensors and actuators B, 2003, 94:228-237.

[15] Shaffer R E, Rose-Pehrsson S L, McGill R A. A comparison study of chemical sensor array pattern recognition algorithms [J]. Analytica Chimica Acta, 1999, 384:305-317.

[16] Cosimo D, Pietro S, Lorenzo V. Odor discrimination using adaptive resonance theory [J]. Sensors and Actuators B, 2000, 69:248-252.

[17] Yolanda G M, Concepcion C O, Jose L P P, et al. Electronic nose based on metal oxide semiconductor sensors and pattern recognition techniques: characterization of vegetable oils [J]. Analytica Chimica Acta, 2001, 449:69-80.

[18] Ricardo G O. Pattern Analysis for Machine Olfaction: A Review [J]. IEEE SENSORS JOURNAL. 2002, 2(3):189-202.

[19] Cristianini N, Taylor J S. 支持向量机导论[M]. 李国正等译. 北京：电子工业出版社, 2004.

[20] 张学工. 关于统计学习理论与支持向量机[J]. 自动化学报, 2000, 26(1):32-41.

[21] 张浩然, 韩正之, 李昌刚. 支持向量机. 计算机科学, 2002, 29(12): 135-137.

[22] 杜树新, 吴铁军. 模式识别中的支持向量机方法. 浙江大学学报（工学版）. 2003, 37(5)：521-527.

[23] 葛海峰, 林继鹏, 刘君华, 丁晖. 基于支持向量机和小波分解的气体识别研究. 仪器仪表学报. 2006, 27(6):573-578.

[24] Andrea B, Paola S, Marcela, et al. Electronic nose based discrimination of a perfumery compound in a fragrance [J]. Sensors and Actuators B, 2003, 92:222-227.

[25] 魏广芬, 唐祯安, 余隽. 基于主成分分析和 BP 神经网络的气体识别方法研究[J]. 传感技术学报. 2001, 4:292-297.

[26] 史忠植. 神经网络[M]. 北京：高等教育出版社, 2009: 48-63.

[27] Shaffer R E, Rose-Pehrsson S L, Mcgill R A. Improved probabilistic neural network algorithm for chemical sensor array pattern recognition [J]. Anal Chem, 1999, 71:4263-4271.

[28] 曹明富, 陈星, 王永清, 应可净, 徐凤娟, 王平. 基于虚拟气体传感器阵列的新型肺癌检测电子鼻实验研究[J]. 中国生物医学工程学报, 2008, 1(27)：102-106.

[29] 李啸, 闫卫平, 郝应光. 气体传感器阵列信号处理的混合神经网络[J]. 传感器技术, 2000, 19（4）：20-22.

[30] Takahiro H, Haruhiko K, Takashi O. Recognition of human activities by gas sensor response using genetic algorithm [J]. Scripta Technica, 2001, 85:50-61.

[31] Frank D, Birgit K, Gunter G. Genetic algorithms and neural networks for the quantitative analysis of ternary mixtures using surface Plasmon resonance [J]. Chemometrics and Intelligent Laboratory Systems, 2003, 65:67-81.

[32] Duda R O, Hart P E, Stork D G. Pattern classification (2nd Edition) [M]. New York: Wiley, 2001.415-446.

[33] Chien-Pang Lee, Wen-Shin Lin, Yuh-Min Chen, Bo-Jein Kuo. Gene selection and sample classification on microarray data based on adaptive genetic algorithm/k-nearest neighbor

method. Expert Systems with Applications, 2011, 38: 4661–4667.

[34] 王鹤, 周东祥, 宋委远, 吴筝. 基于遗传神经网络的混合气体识别研究[J]. 华中科技大学学报（自然科学版）, 2007, 35(9)：118-128.

[35] LIANG F, CHRISTOPHER J M, JONATHAN W K, et al. Colorimetric Sensor Array for Determination and Identification of Toxic Industrial Chemicals [J]. Anal. Chem. 2010, 82(22):9433–9440.

[36] SUNG H L, LIANG F, JONATHAN W K, et al. An Optoelectronic Nose for Detection of Toxic Gases [J]. Nat Chem. 2009, 13(1): 562–567.

[37] PATRICK K, JEFFREY D A, JAMES R C, et al. Towards the development of a portable device for the monitoring of gaseous toxic industrial chemicals based on a chemical sensor array [J]. Sensors and Actuators B, 2008(134): 307–312.

[38] THICHAMPORN E, RADEEMADA M, CHAIWAT P, et al. Polydiacetylene paper-based colorimetric sensor array for vapor phase detection and identification of volatile organic compounds [J]. J. Mater. Chem. 2012(22):5970–5977.

[39] PAWLAK Z, SKOWRON A. Rudiments of rough sets [J]. Information Sciences, 2007, 177(1): 3-27.

[40] 尚志信, 周宇, 叶庆卫, 王晓东. 基于粗糙集和 BP 神经网络算法的网络故障诊断模型研究[J]. 宁波大学学报：理工版, 2013, 26(2): 45-48.

[41] 李伟, 何鹏举, 杨恒, 陈明. 基于粗糙集和改进遗传算法优化BP神经网络的算法研究[J]. 西北工业大学学报, 2012, 30(4): 601-605.

[42] Hui Ling Chen, Bo Yang, Jie Liu, Da You Liu. A support vector machine classifier with rough set-based feature selection for breast cancer diagnosis[J]. Expert Systems with Applications. 2011, 38(7): 9014-9022.

[43] 文莹, 肖明清, 胡雷刚, 方甲永. 基于粗糙集神经网络的航空电子设备故障预测研究[J]. 计算机测量与控制, 2010, 18(4):c807-809.

[44] 陈锦荣, 丁世飞. 一种粗逻辑神经网络的构造和学习方法[J]. 计算机科学. 2011, 38(12):c236-238.

[45] 郭志军, 何昕, 魏仲慧, 张伟华, 梁国龙. 一种基于粗糙集神经网络的分类算法[J]. 2011, 28(3): 838-841.

[46] 夏红霞, 王惠营, 胡磊. 基于粗糙集的神经网络结构优化方法[J]. 2010, 247(5): 49-51.

[47] 史蒂夫·拉塞尔, 丽萨·梅多斯, 罗斯林·拉塞尔. 生物芯片技术与实践（中文版）[M]. 肖华胜, 张春秀, 武雪梅, 华友佳, 刘寒梢, 译. 北京：科学出版社, 2010.

[48] Sherlock G. Analysis of large-scale gene expression data[J]. Briefings in Bioinformatics, 2001, 2(4):350-362.

[49] Chiang J H, Chao S Y. Modeling human cancer-related regulatory modules by GA-RNN hybrid algorithms[J]. BMC Bioinform, 2007, 8:91.

[50] 王岩, 陈向东, 蒋亚东, 赵静. 基于独立分量分析和 BP 网络的电子鼻模式识别[J]. 传感器与微系统, 2006, 25(9): 36-39.

[51] An Incremental Attribute Reduction Algorithm for Decision Information Systems Based on Rough Set [A]. Proceedings of 2008 International Symposium on Distributed Computing and Applications for Business Engineering and Science Volume Ⅱ[C], 2008.

[52] Qian Y H, Liang J Y, Pedrycz W, Dang CY. An efficient accelerator for attribute reduction from incomplete data in rough set framework[J]. PATTERN RECOGNITION, 2011, 44(8):1658-1670.

[53] 杨传健, 葛浩, 汪志圣. 基于粗糙集的属性约简方法研究综述[J]. 计算机应用研究, 2012, 29(1): 16-20.

[54] Stanley B.Lippman 等著. C++ Primer 中文版[M]. 李师贤等译. 北京：人民邮电出版社, 2006.

[55] Steve McConnell 著. 快速软件开发[M]. 席相林等译. 北京：清华大学出版社, 2008.

[56] 余金山. 实时 UML Rational Rose RealTime 建模暗里剖析[M]. 北京: 电子工业出版社, 2007: 109-110.

[57] Mark Allen Weiss 著. 数据结构与算法分析[M]. 冯舜玺译. 北京：机械工业出版社, 2004.

[58] 刘平. 基于叶啉化学传感器的气体检测系统软件研究[D]. 重庆: 重庆大学, 2010.

[59] 倪继利. Qt 及 Linux 操作系统窗口设计[M]. 北京: 电子工业出版社, 2006.

[60] 韦东山. 嵌入式 Linux 应用开发完全手册[M]. 北京: 人民邮电出版社, 2008: 2-8.

附 录

A. 作者在攻读硕士学位期间发表的论文目录：

[1] Xiaogang Luo1, Xinxin Xie et al. Research on gas recognition algorithm based on porphyrin chemical sensor[J]. Physiological Measurement. (SCI 期刊，已送审)

B. 作者在攻读学位期间参加的科研项目目录：

[1] 国家科技支撑计划项目（2012BAI19B03）负责气体识别算法设计及实现.

C. 识别程序与训练程序各子函数伪码：
识别程序成员函数伪码设计

函数名：计算欧式距离函数

输入：RGB 差值数组

输出：欧式距离值数组

开始运行：

FOR 循环，i=0，i 等于行列乘积停止

$$rgbdata[i] = \sqrt{sample[i*3]^2 + sample[i*3+1]^2 + sample[i*3+2]^2}$$

delete 各数组指针;

返回 rgbdata;

函数名：归一化函数

输入：欧式距离值数组

输出：归一化小数数组

开始运行：

FOR 循环，i=0，i 等于行列乘积停止

找出数组中的最大值和最小值;

FOR 循环，i=0，i 等于行列乘积停止

$$rgbdata[i] = (rgbdata[i] - \min) / (\max - \min)$$

delete 各数组指针;

返回 rgbdata;

函数名：约简点选择

输入：数组 rgbdata

输出：约简后数组 new;

开始运行：

读取保存约简点位置的信息，得到约简点个数 n;

生成动态数组 locate，new，长度为 n; //保存约简点位置、约简后的数组

读取保存约简点的位置，存入 locate;

int j=0;

FOR i=0; i 小于 rgbdata 长度运行

IF i=locate[j];

j++;

new[j]= rgbdata[i];

ELSE continue;

返回 new;

函数名：判断气体类别函数

输入：识别结果数组

输出：气体所属类别及概率

开始运行：

生成动态数组 target，error;//保存训练目标，保存误差

读取 txt 文件中的训练目标;

FOR i=0，i 小于输入层节点数×气体种类数运行

计算识别结果与训练目标的差值总和;

计算识别差值的平均值;

平均值保存到数组 error 中;

FOR i=0，i 小于气体种类数运行

找出 error 中的最小值 min;

计算概率：(1-min)×100%;

IF 概率>阈值

调用保存样本数据函数保存特征差值、训练目标;

调用保存识别数据函数;

返回概率;

函数名：保存识别数据

输入：识别结果数组

输出：无

开始运行：

构造原系统连接 Excel 文件的类的对象;

连接 Excel;

FOR i=0, i 小于输出层节点数运行

写入识别结果;

关闭 Excel;

返回;

函数名：保存样本数据

输入：原始特征差值数组，训练目标

输出：无

开始运行：

构造原系统连接数据库的类的对象;

连接数据库;

FOR i=0, i 小于特征差值数组长度+输出层节点数运行

写入特征差值数组;

写入训练目标;

关闭数据库;

返回;

训练程序成员函数伪码设计

函数名：保存训练参数

输入：需要保存的参数数组

输出：无

开始运行：

构造 ofstream 类对象 fw; fw.open(“w.txt”);

FOR i=0; i 小于隐含层节点数运行//保存输入层到隐含层的权值

FOR j=0; j 小于输入层节点数运行

fw<<W[i][j]<<” ”;

fw.close(); //关闭文件

构造 ofstream 类对象 fv; fv.open(“v.txt”);

FOR i=0; i 小于隐含层节点数运行//保存隐含层到输出层的权值

FOR j=0; j 小于输出层节点数运行

fv<<V[i][j]<<” ”;

fv.close();

构造 ofstream 类对象 fsita; fsita.open(“sita.txt”); //保存隐含层阈值

FOR i=0; i 小于隐含层节点数运行

fsita<<sita[i]<<” ”;

fsita.close();

返回;

函数名：读取数据

输入：样本数据文件路径

输出：运行状态信息

开始运行：

构造 CFileDialog 对象 opentest;

opentest.m_ofn.lpstrTitle=弹出对话框标题栏名称;

opentest.m_ofn.lpstrTitle=过滤数据文件的格式;

opentest.DoModal();

检索是否安装有 Excel 驱动 "Microsoft Excel Driver (*.xls);

sDsn.Format(_T("ODBC;DRIVER={%s};DSN=' ';DBQ=%s"), sDriver, sFile);

TRY

打开数据库;

设置查询语句;

FOR i=0; i 小于输入训练输入节点运行

读取文件数据保存至数组中;

关闭数据库;

将与状态显示的文本框关联的变量赋值“读取完毕!”;

UpdateData(FALSE); //状态显示至界面

读取文件完成 FLAG=1;

CATCH(CDBException,e)

return;

函数名：初始化数组

输入：无

输出：初始化各数组

开始运行：

FOR i=0; i 小于输入节点数运行

生成动态二维数组指针 W**; //保存输入层到隐含层权值

FOR i=0; i 小于隐层节点数运行

生成动态二维数组指针 V**; //保存隐含层到输出层权值

分别动态生成保存隐含层输入输出和阈值、输出层的输入输出和阈值的数组;

生成动态数组保存识别结果;

FOR i=0; i 小于隐含层节点数运行

FOR j=0; j 小于输出层节点数运行

$$V[i][j]=(double)(rand()/(double)RAND_MAX)-0.5$$

FOR j=0; j 小于输出层节点数运行

$$gama[j]=(double)(rand()/(double)RAND_MAX)-0.5$$

return;

函数名：约简点选择

输入：样本数据 data

输出：约简后数组 new

开始运行：

读取保存约简点位置的信息，得到约简点个数 n;

生成动态数组 locate，长度为 n; //保存约简点位置

生成动态二维数组 new[样本个数][n]; //约简后的数组

读取保存约简点的位置，存入 locate;

FOR k=0, k 小于样本个数运行

int j=0;

FOR i=0; i 小于样本长度运行

IF i=locate[j];

j++;

new[k][j]= data[i];

ELSE continue;

返回 new;

函数名：得到训练误差

输入：单个样本训练误差数组 trainerr[]

输出：训练误差

开始运行：

定义 double 类型 err;

FOR 循环，i 小于 trainerr 运行;

err=trainer[]所有元素的平均值;

返回 err;