Embedded Technoloy and Intelligent Systems 嵌入式技术与智能系统, 2024, 1(2), 77-84
Published Online October 2024 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2024.12009

Hans汉斯

在微控制器上实现在设备端训练的异常检测

宋　岩，许　鹏，张　岩

恩智浦(中国)管理有限公司北京分公司，北京

收稿日期：2024年9月5日；录用日期：2024年11月21日；发布日期：2024年11月29日

摘　要

在当前嵌入式系统与人工智能技术融合的前沿领域，文章聚焦于一种基于单类支持向量机(One-Class SVM)的异常检测算法，并提供了一套完整的MCU友好的工程实现，不需要依赖于动态内存分配以及文件系统，特别适合于在资源受限的边缘设备上进行高效、实时的训练与预测。我们的方法不仅可以实现在MCU上训练和高效存储机器学习模型，还支持增量学习，从而在几乎不增加计算负担的前提下，持续改进模型对实际工况的适应能力。我们的实验装置是安装了三轴加速度传感器的震动源(如风扇)，以模拟在工作期间发出振动的工业设备。文章的方法也可以通过替换传感器和特征计算的预处理算法来实现对其它设备的监控，以适应不同的工况环境和应用的需求。

关键词

微控制器单元(MCU)，设备端训练(ODT)，支持向量机(SVM)，人工智能应用，MCXN947

Anomaly Detection on Microcontroller with On-Device Training

Yan Song, Peng Xu, Yan Zhang

Beijing Branch of NXP (China) Management, Co., Ltd., Beijing

Received: Sep. 5th, 2024; accepted: Nov. 21st, 2024; published: Nov. 29th, 2024

Abstract

This paper introduces an anomaly detection algorithm based on one-class support vector machines (SVMs) and an MCU-friendly engineering implementation. It does not rely on dynamic memory and file systems, it is particularly suitable for efficient, real-time training and prediction on resource-constrained edge devices. Our method not only enables training and efficient storage of machine learning models on MCUs, but also supports incremental learning, thus continuously improving the model's adaptability to actual operating conditions without increasing the computational burden. Our

文章引用: 宋岩, 许鹏, 张岩. 在微控制器上实现在设备端训练的异常检测[J]. 嵌入式技术与智能系统, 2024, 1(2): 77-84. DOI: 10.12677/etis.2024.12009

experimental setup is a vibration source (such as a fan) with a triaxial acceleration sensor installed to simulate industrial equipment that emits vibrations during operation. The method in this paper can also be used to monitor other devices by replacing sensors and computing features.

Keywords

Microcontroller Unit (MCU), On-Device Training (ODT), Support Vector Machine (SVM), Artificial Intelligence Application, MCXN947

1. 引言

在嵌入式系统中，特别是包含电机和较多机械结构的系统中，异常检测是确保系统稳定运行的关键。异常检测用于判断被监控的系统的当前工作状态，其最基本的需求是给出系统的状态是正常还是异常，常常以检测离群点的方式实现[1]。有很多研究对异常检测的方法和应用做了介绍，以传统机器学习和深度学习为主要方法[2] [3]。如果能评估出系统“健康值”的当前结果和历史变化，就可以此为依据，来进行异常评估。这需要采集系统运行时的状态，继而使用数据分析方法来检测出可能的异常。基于机器学习的 AI 方法近年来在异常检测中脱颖而出，在行业中，NXP、ST、ADI 等头部半导体公司在异常检测方面都多有耕耘，并且提供示例、参考方案、甚至是商业化的完整产品。

在微控制器上部署基于机器学习的算法大多需要进行离线训练，即在高性能计算平台(如个人 PC、服务器或云)上预先完成机器学习模型的训练，随后将训练好的模型部署至目标设备(如微控制器)。这种工作流程对于深度学习尤其常见，因为微控制器计算力与存储有限，难以支撑模型训练所需，而且，直接训练会大幅增加功耗。

尽管离线训练在微控制器机器学习中占据主导地位，但在设备端进行训练也有其独特的优势，尤其是在应用类别五花八门而工作条件变化多端的工业异常检测领域，在这种场景下，设备端训练使模型能够根据实时收集的数据动态调整，提高对新异常的检测能力。这对于环境变化频繁或未知异常类型的情况尤为重要。除此以外，每台设备的工作条件和环境可能不同，设备端训练能够针对特定设备进行模型优化，提高异常检测的精确度和针对性。

为了实现设备端的训练和推理，软件框架和示例具备以下特点：

1) 少样本训练。最好是 100 笔以内的样本即可训练，以满足 MCU 上的内存限制。

2) 自动标注或无需标注。传统的监督学习需要标注，但是嵌入式系统一般不包含用于标注的工具。

3) 小巧精悍和易于解释的 ML 算法，如 SVM。

4) 轻量级的训练引擎。传统的训练引擎需要的代码量大，依赖动态内存和文件系统，使用容量大的数据类型，往往需要避免使用这些大的数据类型或者有必要对其进行优化。

5) 简练而高效的数据处理。包括利用传感器的 FIFO，通用和易于实现的特征提取算法，使用环形缓冲存储预处理后的特征而非原始数据等。

2. 硬件装置和整体框图

基于上述的目标和约束，我们选择使用单类支持向量机(One-Class SVM, OCSVM) [4] [5]机器学习算法，

并且深度重构和优化 Libsvm 库而得到对 MCU 友好的 Libsvmcu。并且使用可调转速的风扇为装置，来模拟在工作期间可能出现多种正常的振动模式的工业设备。我们搭建了异常检测装置用于验证，如图 1 所示。

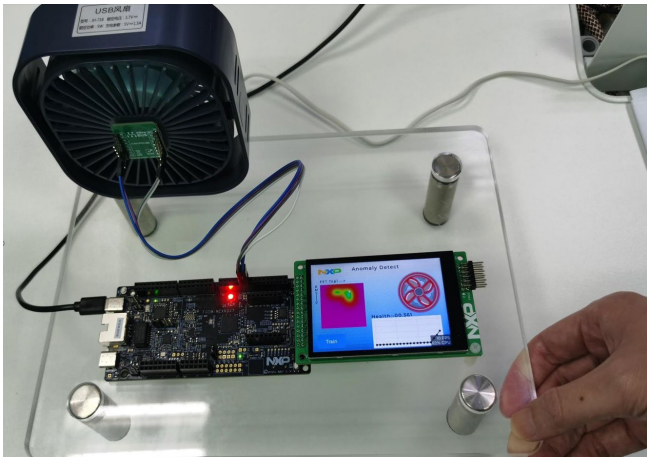


Figure 1. Anomaly detection device
图 1. 异常检测装置

上图中扮演工业装备的是一个有 3 个转速档位可调节的 USB 风扇。在扇叶罩外表面贴装了小型三轴加速度传感器，它使用 I2C 接口和 MCU 主板通信。MCU 主板是 FRDM-MCXN947，它上面包含 NXP 的 MCXN947 微控制器，并且扩展了 3.5 英寸 LCD 屏幕作为图形用户界面，还使用一个三色 LED 来显示当前是正常还是异常状态。整个系统的结构框图如图 2 所示。

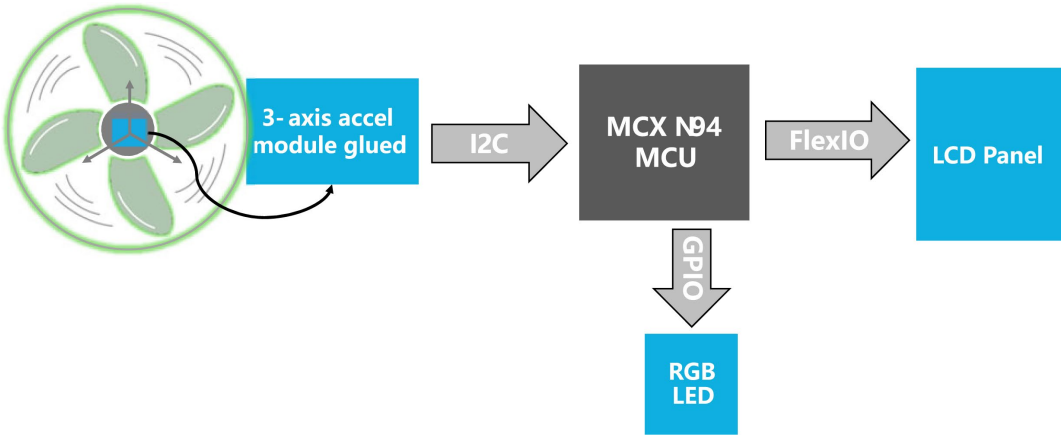


Figure 2. Block diagram of anomaly detection
图 2. 异常检测框架图

3. 训练和推理框架-Libsvmcu

为了实现基于支持向量机(SVM)的在设备上训练和预测，我们基于著名的开源支持向量机库 Libsvm [5]，进行多项重构和定制化，实现了 Libsvmcu 库。Libsvm 在 MATLAB，sckit-learn 等众多知名框架中得到了广泛应用，但是部署到 MCU 上还相对有些重。我们改版的 Libsvmcu 和原版的 Libsvm 相比做了以下修改。

1) 去掉对堆内存的依赖。原版 Libsvm 中的所有动态生长的数据结构、软件缓存、以及临时不定长

度的 buffer 都使用 C++的 new/delete 或 malloc/free。我们通过大量调整分配和释放的顺序以符合后进先出的方式，去掉了对堆内存的依赖，使它们可以存储在单块静态预分配的内存中。这对于可靠性、确定性、平均性能都有明显的改进。

2) 实现支持就地执行(XIP)的模型序列化机制。在确保所有动态产生的数据块都位于单块内存的基础上，我们适当调整 Libsvm 中模型对象类中的字段顺序并添加辅助字段，把所有指针和二级指针字段都收集在一起。以此为基础，实现了对模型对象的重定位，使得位于内存中的模型对象可以直接序列化到微控制器内置的 99999++Flash 存储器中，并且支持在 Flash 上直接就地使用(XIP)，无需再加载到 RAM 中。这种技术甚至支持把无需使用封装格式打包的模型对象通过网络传输到另外的机器上，再经过简单的重定位后即可直接在 RAM 或烧写到 Flash 的地址中使用。有了这套基于重定位的序列化/反序列化技术，我们直接移除了 Libsvm 中原先使用的基于文件 I/O 的模型存储和加载机制。

3) 支持稠密向量。Libsvm 原先只支持稀疏化存储的训练样本的标量基本元素，以便支持高维数据的稀疏性，例如对于 100 维样本中只有几个非零元素的情况。然而，这样的代价是需要占用额外 4 字节保存各标量基本元素的位置。这对于在嵌入式系统中更常出现的低维且各维数据都有意义的情况却是没有必要的额外开销。我们添加了对常规数据的支持，减少了一半的存储空间来保存训练样本和学习出来的支持向量。

4) 减少核矩阵缓存的内存开销。Libsvm 使用双向链表实现一个基于最近最少使用(LRU)替换策略的核矩阵缓存(Kernel Cache)，这个核矩阵是各训练向量在经过核函数高维映射后得到的在相应的再生核希尔伯特空间中的距离矩阵，缓存线的容量和数量都是训练向量的个数。原来的 LRU 缓存使用标准的双向链表来存储缓存线的头部并且动态申请和释放内存。我们根据在 MCU 上 on-device training 时大部分情况不超过 256 个训练样本的特点，以及结合前文提到的栈式内存管理机制，改造成基于 8 位索引和一次性分配缓存最大容量的策略。

5) 支持 32 位和 16 位浮点数。Libsvm 原先只支持 64 位双精度浮点数。我们把它改造成支持 32 位浮点数，并且在小范围使用 16 位浮点类型。Libsvm 采用 C++实现，C++的运算符重载对于我们实现 16 位浮点类型提供了极大的语法帮助，使 Libsvm 主体代码几乎无需改动。

经过我们裁剪优化后的 libsvmcu 运行库，实现 One-Class SVM 仅需要十余 KB 程序存储器的空间，并且一般训练时间少于 1 秒。

4. 为应用 SVM 算法的特征设计

SVM 是传统机器学习方法，需要基于原始信号合理设计特征方可使用。在我们的装置中，是模仿很多工业设备在工作期间会产生的机械振动。我们利用三轴加速度传感器采集数据，通过深入分析加速度信号，提取了根均方值(RMS)和傅立叶变换后辐值最高的频率所对应的位置(FFTTop1)作为关键特征。这些特征不仅能够有效反映设备运行状态，还具备较强的鲁棒性和计算效率。

为了计算上述的(RMS, FFTTop1)特征，我们先收集一定长度的原始传感器读数，当前的设定是 256 笔(a_x, a_y, a_z)信号组成的形如(3, 256)的原始信号，其中 3 表示 3 个方向轴。然后，对每轴分别计算 RMS 和 FFTTop1，最后把各轴的 RMS 和 FFTTop1 取平均。

在计算 RMS 时，为了抵消掉重力加速度的影响，我们对各个轴上的数据做了滑动平均处理。

这里我们想强调，这种双特征的设计方法只是众多特征设计中的一种，经过实验观察发现对于我们的硬件装置还是很好用的。但是，SVM 算法可以支持远远更高维的特征空间，用户如果使用不同的传感器，或者如果发现默认的这两个特征不足以表达更复杂的数据模型，还需要具体问题具体分析地设计相应的特征。在包括 SVM 的经典机器学习算法中，特征工程也是非常重要的环节。

5. 应用层软件架构设计

在实现完整的在设备端训练的异常检测应用时，我们使用了 FreeRTOS 来简化应用逻辑的划分和调度。整体来看，使用了 3 个任务。下图 3 展示了任务的划分与关系。

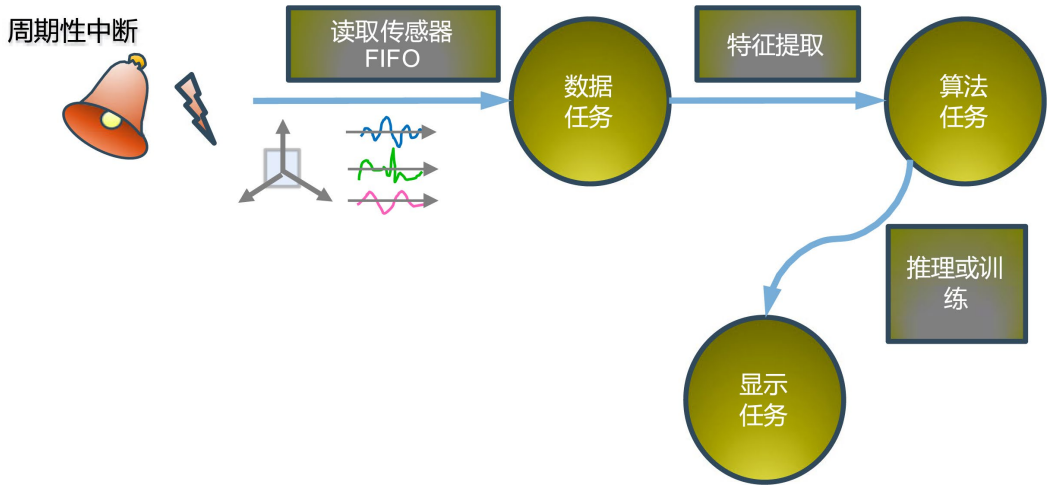


Figure 3. Software block diagram
图 3. 软件框图

数据任务：在它的主循环中，每次在循环开始时获取任务开始的时钟计数。接着，检查传感器的 FIFO 缓冲区数据量，计算要读取的数据量，并尝试从传感器读取数据存入临时缓冲区。对于读取到的数据，遍历不同维度进行处理，累加到当前均值变量并存储到另一个缓冲区，同时记录已填充的数据量。当该数据量达到特定长度(满足一个 FFT 窗口的长度)时，进行一系列数据处理操作，包括计算每个维度的均值并更新直流偏移指数移动平均，对数据进行减去直流偏移后计算均方根值(RMS)。接下来，对这段信号应用汉宁窗，然后执行快速傅里叶变换，找出幅值最大的频率所在的位置(FFTTop1)。处理后的 RMS 和 FFTTop1 就是提取的特征，并存入特征数组，并通过 FreeRTOS 的队列发送出去给算法任务。之后，对数据进行滑动处理，将数据向左移动一个固定的长度并更新相关变量。最后，获取任务结束时钟计数，根据任务起始执行时间与指定时间的比较结果调整任务延迟时间，确保数据任务的执行以固定的节奏进行。数据任务的周期必须短于传感器的 FIFO 装满所需的时间。

算法任务：在算法任务的主循环里，其功能涉及数据收集、模型训练和基于模型的预测及状态控制等多个方面。

首先，代码处于一个无限循环中，不断地从一个队列中接收数据。每次循环开始，通过 FreeRTOS 的队列 API 函数从来自数据任务的特征队列中获取特征数据并保存，然后设置标志表示有新的特征数据可用。

接下来，根据应用的不同状态进行不同的操作。当处于收集状态时，会逐步将接收到的特征数据存入一个特征数组中。如果数据积累到一定数量，会进行模型训练。倘若开启了增量训练，则首先获取已有模型中的支持向量，然后使用新的数据和已有的支持向量重新训练模型。训练完成后，更新应用状态为等待返回。

当处于预测状态且有训练好的模型时，首先获取当前风扇状态。然后通过调用模型预测函数使用模型，对接收的特征数据进行预测，得到预测结果。根据预测结果更新系统的当前健康值，并根据健康值与特定阈值的比较来判断风扇所处的状态。如果风扇状态发生变化，还会控制红绿指示灯的亮灭，以直

观地显示风扇整体是处于正常(绿灯)还是异常(红灯)。

数据任务和算法任务的配合流程如同生产者和消费者，它们的整体工作步骤如下图 4 所示。

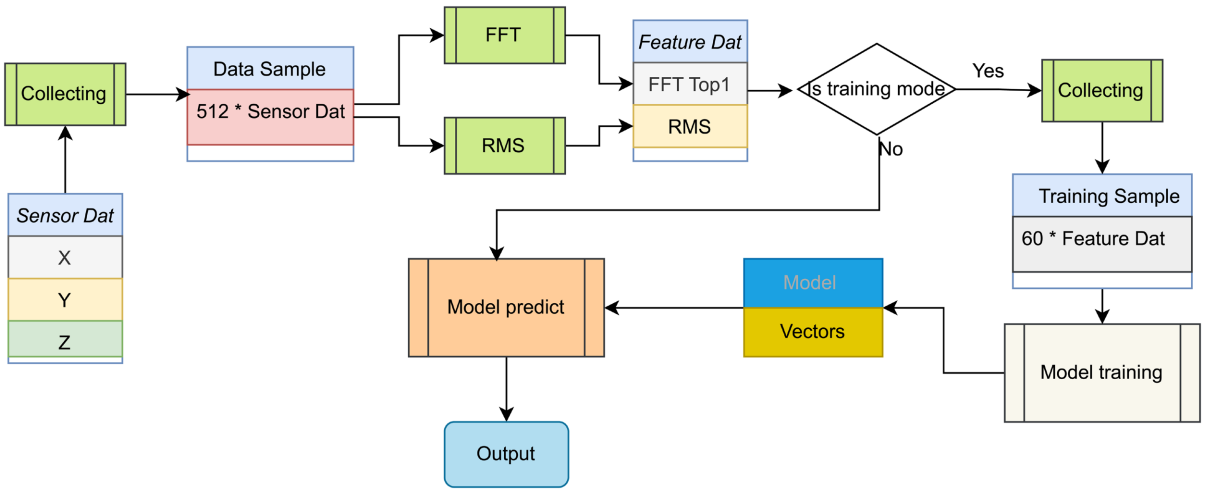


Figure 4. Workflow regarding to data task and algo task
图 4. 数据任务与算法任务的工作流

显示任务：显示任务的功能比较繁杂，它实现了一个基于 LVGL 的 GUI 应用。主要包含了检测界面、训练界面，以及参数设置界面。显示任务定期执行，它直接通过共享变量读取当前最新的状态和数据，并与其它任务没有通信。显示任务的功能不再展开叙述，唯有一个注意事项是它的优先级必须低于其它应用任务，避免它导致数据任务不能及时得到调度执行，这可能会丢失传感器数据。在我们的调试期间这曾经是一个晦涩的 bug。

6. 更多讨论

6.1. 增量学习

在很多系统中都包含多个正常的子状态。例如，我们的装置中包含了一个可以调速的风扇，在风扇处于关机、1 档、2 档、3 档的状态下，都属于正常。于是，我们为了考虑这种应用的实际需要，也支持一种增量训练的机制。

在每次训练时，如果系统中已经包含了之前训练得到的支持向量，我们会把它们读取出来和新获得的样本一起重新训练，并产生一个新的支持向量的集合，并且用新模型替换旧模型。这样可以支持多个正常子状态。

6.2. One-class SVM 中的超参数调节

在本文使用的 one-class SVM 用于异常检测时，有两个对结果影响非常大的超参数，Gamma 和 Nu。Gamma (γ)是核函数的一个参数，它决定了数据映射到高维空间后的分布情况。在径向基函数(RBF)中，较大的 Gamma 值意味着较小的决策边界，使模型更关注训练数据的局部特征；而较小的 Gamma 值意味着较大的决策边界，从而产生一个更平滑的模型，对训练数据中的局部波动不太敏感。Nu(ν)是一个用户定义的参数，表示错误数据点比例的上限和边界的下限。这个参数有助于控制支持向量的比例以及决策边界的宽松度。简单来说，较小的 Nu 值使模型更倾向于忽略更多的异常值，导致决策边界更宽；相反，较大的 Nu 值使模型更倾向于包含更多的数据点，导致决策边界更窄。

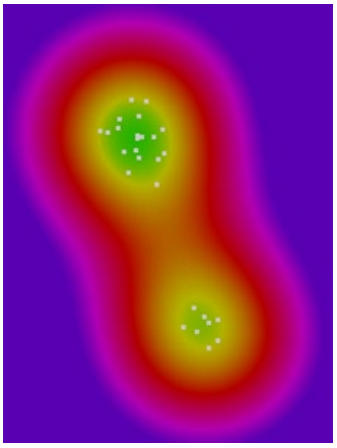
在大多数演示设置中，默认值(Gamma: 50, Nu: 0.1)通常在敏感性和容忍度之间实现合理的平衡。

Nu 值较大时，模型对数据变化更敏感，但非常大的 Nu 值(如≥0.4)可能会使模型将一些远离平均值的训练样本视为异常。因此，如果想要高敏感性和快速响应的演示，可以使用较大的 Nu 值(如 0.1 到 0.4)；但如果希望演示对随机或意外干扰更具容忍度，则应使用较小的 Nu 值(如 0.03 到 0.1)。

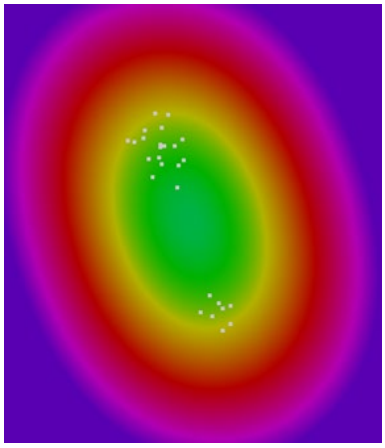
Gamma 值较大时，模型倾向于将训练数据处理为多个聚类(每个训练样本的有效范围变小)。所以，如果正常状态包含多个子状态(例如风扇关闭和风扇开启都是正常状态)，那么应该使用较大的 Gamma (如 20 到 200)；但较大的 Gamma 值也会使模型对训练数据中的随机性不太稳健，因此，如果演示环境受到严重干扰(如正在工作的计算机的振动、附近风扇引起的空气振动、附近走动的人)，则应使用较小的 Gamma 值(如 5 到 20)。

为了便于演示与调节超参数，我们在液晶显示屏上对模型的决策边界进行二维彩色等高线可视化，横轴表示 FFTTOP1 幅值所对应的频率，纵轴表示 RMS。等高线中，绿色-蓝色的颜色区域是正常区域，而黄色、红色和紫色的颜色区域是异常区域。表 1 显示了超参数对异常决策边界的重大影响。

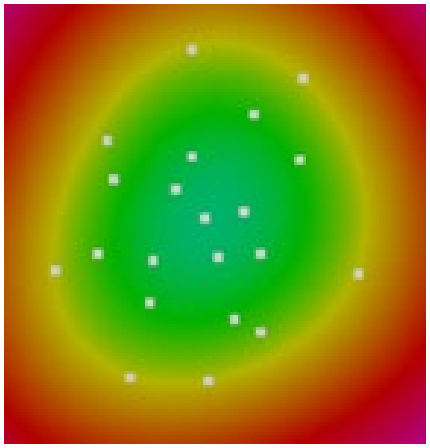
Table 1. Hyperparameter tuning (Gamma and Nu)
表 1. 超参数的整定(Gamma 和 Nu)



Gamma = 50, Nu = 0.1
Optimal Gamma and Nu that makes the model can detect 2 sub normal states.

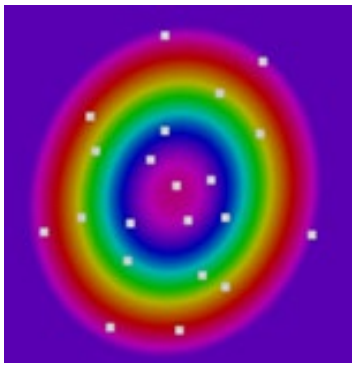


Gamma = 10, Nu = 0.1
Gamma is so small that the decision boundary is too smooth to distinguish the two sub normal states.

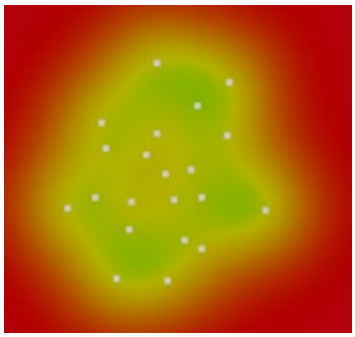


Gamma = 50, Nu = 0.1
Model can tolerate some random noises.

续表



Gamma = 50, Nu = 0.4
Nu is too large that the model output big range of numbers, and wrongly treats some points that are far from center as abnormal.



Gamma = 300, Nu = 0.1
Gamma is too large that the model clusters too much, makes the decision boundary is not smooth and tends to overfit.

7. 结语

在微控制器上部署异常检测的常见方法是使用有监督机器学习在 PC 上预训练模型。然而，一些异常检测的问题受个体差异影响大，或者难以获得异常情况的样本。为了解决这一问题，本文介绍了一种基于单类支持向量机(SVM)算法的设备上训练的异常检测方法。该方法利用了 SVM 算法在小样本和高维数据上的优越性能，通过深度重构 Libsvm 库，如内存管理和模型序列化等，得到最终的 MCU 友好的 Libsvmcu 库，并开发了一套软件应用框架在设备上进行训练，实现了对风扇异常的有效检测。同时，该方法还支持增量学习，使得模型能够随着时间的推移不断更新，以适应嵌入式系统的变化。

本文中配合使用的示例已开源：https://github.com/nxp-appcodehub/dm-on-device-training-fan-anomaly-on-mcxn947。

参考文献

[1] Chandola, V., Banerjee, A. and Kumar, V. (year) Anomaly Detection: A Survey. *ACM Computing Surveys*, **41**, 1-15.

[2] Pimentel, M.A.F., Clifton, D.A., Clifton, L. and Tarassenko, L. (2014) A Review of Novelty Detection. *Signal Processing*, **99**, 215-249. https://doi.org/10.1016/j.sigpro.2013.12.026

[3] Hodge, V.J. and Austin, J. (2004) A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, **22**, 85-126.

[4] Chang, C.-C. and Lin, C.-J. (2024) LIBSVM—A Library for Support Vector Machines. https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[5] Tax, D.M.J. (2001) One-Class Classification. Doctoral Thesis, Delft University of Technology.