电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 高效支持向量机的研究与实现

专业学位类别	工 程 硕 士
学　　号	201522260305
作 者 姓 名	蒋生强
指 导 教 师	马 上　副教授

分类号 _____ 密级 _____

UDC [注1] _____

学 位 论 文

高效支持向量机的研究与实现

（题名和副题名）

蒋生强

（作者姓名）

指导教师 马 上 副教授

电子科技大学 成 都

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 工 程 硕 士

工程领域名称 电子与通信工程

提交论文日期 2018.4.12 论文答辩日期 2018.5.14

学位授予单位和日期 电子科技大学 2018 年 6 月

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号

Research and Implementation of High Efficiency Support Vector Machine

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Discipline: **Master of Engineering**

Author: **Jiang Shengqiang**

Supervisor: **Associate Professor Ma Shang**

School: **National Key Laboratory of Science and Technology on Communications**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名：蒋生强　　日期：2018 年 5 月 30 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：蒋生强　　导师签名：

日期：2018 年 5 月 30 日

摘 要

SVM 是重要且经典的统计学习算法之一，主要应用于分类及回归领域。在基于高斯核的 C 支持向量分类应用中，最优训练参数组（C，σ）对生成模型的性能起着决定性的作用。实际应用中，通常以网格搜索法和交叉验证搜索 SVM 最优训练参数组。然而，该过程繁重的计算量使得传统的搜索方式用时过长，从而限制了 SVM 在某些场合的应用。本文以提升 SVM 搜索最优训练参数组的速度为研究目标，针对 LIBSVM 在解决这个问题时耗时过长的问题，分别从算法和实现上提出了多种方法。本论文的主要工作如下：

1. 提出了共享点积矩阵（SDPM）算法。针对网格搜索法在搜索 SVM 最优训练参数组时存在的点积冗余计算的问题，提出了 SDPM 算法。该方法通过先行计算出每个样本点和整个样本集的点积并存储为点积矩阵，在后续搜索时再从点积矩阵统一读取的方式，极大地减少了搜索最优训练参数组过程中点积的计算量。理论推导表明，在指定的搜索参数下，使用 SDPM 算法需计算的点积量仅为传统方式点积计算量的 1/60。
2. 完成了基于 SDPM 算法的搜索 SVM 最优训练参数组的软件实现（SDPM-S）。SDPM-S 应用于特定的基于高斯核的 C 支持向量分类问题，并在其搜索过程中引入了 SDPM 算法。测试表明，在指定的搜索方式下，SDPM-S 的搜索速度相比于 LIBSVM 平均提高了 2 倍。
3. 提出了基于 SDPM 算法的搜索 SVM 最优训练参数组的软硬件协同架构并完成其实现（SDPM-H&S）。SDPM-H&S 主体结构由上位机和加速板卡两部分构成：其中，上位机完成数据的类别和折叠处理；加速板卡的现场可编程门阵列完成点积矩阵运算、核函数运算和交叉验证训练，加速板卡的内存条完成点积矩阵存储，加速板卡的 PCI-e 接口完成上位机和加速板卡之间的数据通信。测试表明，在指定搜索方式下，SDPM-H&S 的搜索速度相比于 LIBSVM 提高了 30 倍。

关键词：统计学习，支持向量机，最优训练参数组，共享点积矩阵，软硬件协同实现，现场可编程门阵列。

ABSTRACT

SVM is one of the most important and classical statistical learning algorithms. It is mainly used in classification and regression fields. In the C-support vector classification application based on Gaussian kernel, the optimal training parameter Combination (C, σ) plays a decisive role in the performance of the generated model. In practical applications, the SVM optimal training parameter Combination is usually searched by grid search method and cross-validation. However, the heavy computational load of this process takes too long a time with traditional search method, thus limiting the application of SVM in some occasions. This paper aims to improve the speed of SVM search for the optimal training parameter Combination and try to solve the time problem of LIBSVM, various methods are proposed in terms of algorithm and implementation. The main work of this paper is as follows:

1. A shared dot product matrix (SDPM) algorithm is proposed. In order to solve the problem of dot product redundancy in the search of SVM optimal training parameter Combination by grid search method, the SDPM algorithm is proposed. This method firstly calculates the dot product of each sample point and the entire sample Combination and stores it as a dot product matrix. The method of uniformly reading from the dot product matrix in the subsequent search greatly reduces the process of searching for the optimal training parameter Combination. The calculation of the midpoint product. The theoretical derivation shows that under the specified search parameters, the amount of dot product to be calculated using the SDPM algorithm is only 1/60 of that of the traditional dot product calculation.
2. The software implementation (SDPM-S) of searching the SVM optimal training parameter Combination based on the SDPM algorithm is completed. SDPM-S is applied to a specific Gaussian kernel-based C support vector classification problem and SDPM algorithm is introduced in its search process. Tests have shown that the search speed of SDPM-S is double the average LIBSVM search speed in the specified search mode.
3. The software-hardware collaborative framework for searching SVM optimal training parameter Combination based on SDPM algorithm is proposed and

implemented (SDPM-H&S). SDPM-H&S main structure is composed of upper computer and acceleration board. Among them, upper computer completes data classification and folding processing; accelerated board card field programmable gate array completes dot product matrix operation, kernel function operation and cross validation training. Accelerate the memory of the board to complete the dot product matrix storage, accelerate the PCI-e of the board to complete the data communication between the host computer and the acceleration board. Tests have shown that SDPM-H&S searches 30 times faster than LIBSVM in the specified search mode.

Keywords: Statistical Learning, Support Vector Machine, Optimal Training Parameter Combination, Shared Dot Product Matrix, Hardware and Software Collaborative Implementation, Field Programmable Gate Array.

目 录

图目录

表目录

缩略词表

英文缩写	英文全称	中文释义
CNN	Convolutional Neural Network	卷积神经网络
CPU	Central Processing Unit	中央处理器
DSP	Digital Signal Processing	数字信号处理
FPGA	Field Programmable Gate Array	现场可编程门阵列
LIBSVM	Library for Support Vector Machines	支持向量机库
OTPC	Optimal Training Parameters Combination	最优训练参数组合
PC	Personal Computer	个人计算机
PCI-e	Peripheral Component Interconnect express	外设组件快速互连
RAM	Random Access Memory	随机存取存储器
SVM	Support Vector Machines	支持向量机
SMO	Sequential Minimal Optimization	最小序列最优化
SDPM	Share Dot Product Matrix	共享点积矩阵
VLSI	Very Large Scale Integeration circuits	超大规模集成电路

第一章 绪论

1.1 本文研究背景

近年来，得益于超大规模集成电路（Very Large Scale Integeration circuits，VLSI）技术的发展，人工智能[1-2]在语音识别[3]、人脸检测[4]和图像处理[5]等领域掀起了一股新的研究浪潮，并取得了巨大的进展。随着谷歌公司 DeepMind 团队开发的人工智能软件 AlphaGo 在一系列国际围棋竞赛中战胜人类选手，人工智能迎来了属于自己的顶峰时代。人类在为此狂欢的同时，也对人工智能未来的发展感到担忧。不过我们相信，只要正确合理的使用人工智能，人工智能总是能为创建更好的人类社会而服务。机器学习，又称为统计学习[6-8]，作为人工智能现实应用重要的理论基础，一直是学术界和工业界的研究热点之一。机器学习是先从已知信息的数据中统计规律并构建模型，然后对未知信息的数据进行预测和分析。机器学习的基础是统计学，传统统计学假定样本数量趋于无穷，这样构建的统计学模型能无限逼近于真实模型。然而现实应用中，样本数量不可能趋于无穷，很多学者研究并提出了新的统计学习理论和方法，例如卷积神经网络（Convolutional Neural Network，CNN）[9-11]，就是解决有限样本下统计学模型的构建问题。但是卷积神经网络需要的样本数量很大，在这种情况下，以当时的科学技术条件，对巨量数据集的存储与计算均是不小的挑战，同时，实际生活中应用问题的样本数量不仅有限且样本容量往往不大。在这种情况下，如何构建有限样本容量下的统计学模型，是机器学习领域亟待解决的问题。在这种背景下，支持向量机（Support Vector Machine，SVM）[12-15]应运而生。SVM 一经提出，便以其优秀的理论和性能在模式识别、回归分析、函数拟合及风险评估等各个领域得到了广泛应用。SVM 实质是一个定义在特征空间上的间隔最大的线性分类器[16-17]，它有着严格而缜密的数学理论基础，将生活中的分类、回归及预测问题抽象为标准的凸二次规划问题，通过标准的分解算法[18]，将问题分解为优化多个小问题实现求解。SVM 包含从算法到实现的完整理论，由于其良好的推广能力和分类性能，各个领域的研究学者都在不断努力，扩展 SVM 在新领域的应用。

在通信领域，电子信息对抗[19-21]已经成为国与国之间军事技术较量的焦点之一。在电子信息的对抗中，如何在截断敌方信号后，快速完成信号调制信息的识别[22-23]，以对信号进行进一步的分析或者干扰[24]，是各国通信研究学者需要解决的重要课题。传统的调制信号识别方法是采集信号的多个特征[25-27]，通过设定阈值以一种类似于决策树[28]的方式进行，这种方式识别效果不理想且在采集的特征向量

很多的情况下，识别过程复杂且低效；另外，由于调制信号截取的信息一般不会太长，故其样本容量很小，因此，也不适合使用卷积神经网络进行识别。在样本容量不大且样本点特征向量很多的情况下，SVM 因其优异的分类性能，被许多研究学者应用于调制信号的识别中，并取得了不错的结果[29-31]。然而，由于信号调制方式多种多样，而 SVM 一次只能针对两类调制方式进行建模，且在电子信息的对抗应用中，需要迅速的建立模型并将模型应用于新信号识别中。传统的使用软件构建模型的方式便不再能满足调制信号识别领域应用的需求，因此，我们需要寻找更快的构建模型的方式，以满足调制信号识别领域快速构建模型并应用模型的需要。

1.2 SVM 国内外研究现状

SVM 的实际应用包含训练和应用两个阶段，训练阶段通过大量的测试及验证找到最好的应用模型，应用阶段则将训练阶段生成的模型用于解决实际问题。SVM 在训练阶段生成应用模型又包含两个步骤：一是通过交叉验证训练[32]和网格搜索法[33]寻找最优训练参数组（Optimal Training Parameters Combination，OTPC）；二是使用训练数据集结合 OTPC 生成应用模型。当下对于 SVM 的研究大都集中在步骤二，而步骤二实质是 SVM 的实现算法。众多 SVM 实现算法中，最经典且应用最广泛的是最小序列最优化（Sequential Minimal Optimization，SMO）算法[34]。SMO 是一种快速迭代算法，如果能缩短 SMO 的迭代时间，相应的也就缩短了 SVM 构建模型的时长。目前，有很多人从多方面提出了各种方式实现或者改进 SMO 算法并缩短训练时间。文献[35]从理论、算法和现场可编程门阵列（Field Programmable Gate Array，FPGA）实现三方面对此进行了综合阐述。在 FPGA 实现方面，对此展开研究的包括文献[36-39]。其中，文献[36]的 Markos Papadonikolakis 等人采用可扩展的 FPGA 结构分别以浮点和定点两种方式实现了非线性 SVM，其速度上较之于软件实现有较大提升，文献[37]中 Markos Papadonikolakis 等人还在 GPU 上实现了 SMO 算法并对比了与 FPGA 实现的性能。文献[38]的 Srihari Cadambi 等人以 FPGA 作协处理器，通过大量使用其中的数字信号处理（Digital Signal Processing，DSP）硬核单元，以大规模并行方式执行 SMO 算法中的乘法计算，其软硬件协同系统相比于中央处理器（Central Processing Unit，CPU）上的纯软件实现有大约 20 倍的速度提升。文献[39]的 Jhing-Fa Wang 等人针对语音识别系统，使用 Xilinx 芯片 FPGA 实现 SMO，使用嵌入式微处理器 ARM926EJ 完成预处理、语音参数提取等过程，较之于基于 C 的嵌入式 ARM 处理器，其训练速度提升了 90%，这使得 SVM 的应用摆脱了个人计算机（Personal Computer，PC）的限制，可直接应用于片上系统。在软件实现方面，许多研究者也做了很多努力[40-41]。其中，文献[40]的

A. Hussain1 通过使用特定的方法减少了训练集并实现了与 SMO 相同的性能，但仅针对二维工作集。2005 年，R.-E. Fan 等人提出了同时使用一阶导数信息及二阶导数信息选取工作集的方法，同时编写了著名的 SVM 实现库[41]LIBSVM(Library for Support Vector Machines)，LIBSVM 是公认的 SVM 领先且优秀的实现库之一。为更好的改善 SMO 性能，很多人在算法上进行改进[42-43]。其中，文献[42]提出了一种不考虑 b 值的 KKT 条件判别方法，并给出了一种选择新冲突对的计算公式。文献[43]则提出了混合集选择方法，通过一次选取大于传统的两个工作集的方式，使用 FPGA 作协处理器计算高斯核函数，同样使得 SMO 的训练过程较之于软件有较大提升。但是，这些工作都只是从硬件或者软硬件协同的方式实现或者部分实现了单次的训练过程，并且都是基于 SMO 实现的。在现实应用中，最终的应用模型对应的最佳训练参数是未知的，LIBSVM 作为先进的 SVM 实现库，完成了类别选择、最佳参数搜索、交叉验证训练及验证整个过程。然而，LIBSVM 只是一款软件实现工具，在对模型构建速度有要求的场合，参数搜索过程极大的计算量使得软件执行的时长过长，从而不能满足这些场合的需要。

1.3 本文主要研究内容

基于以上分析可见，对 SVM 的研究集中在 SVM 具体的实现过程，且均基于 SMO 实现，而 SVM 训练阶段的核心在于 OTPC 的搜索，LIBSVM 搜索 OTPC 的过程过于冗长，不能满足 SVM 对训练速度有要求的场合。为此本文研究了 SVM 搜索 OTPC 的快速实现，本文创新点如下：

A. 提出了共享点积矩阵（Share Dot Product Matrix，SDPM）算法，极大的减少了搜索 OTPC 过程点积的计算量；
B. 提出了基于 SDPM 算法的 SVM 搜索 OTPC 的软硬件协同架构并完成其实现，协同系统的搜索速度相比于 LIBSVM 可提升 30 倍；
C. 提出了幂指数分解法并结合 cordic 算法完成了限定范围内求解 e 指数的硬件实现。

本文内容安排如下：

第一章分析和介绍了 SVM 的产生背景，国内外的研究现状和发展趋势。在本章最后，给出了本文的研究内容和组织框架。

第二章主要介绍了统计学习及以统计学习为基础的 SVM 的相关理论。首先，介绍了与 SVM 相关的统计学习的基本问题和基础理论；其次，介绍了不同应用场景下的支持向量分类机，包括它们的理论和模型；最后，对 SVM 在实际应用中面临的一系列问题及其解决方法进行阐述，即 SVM 模型。

第三章主要提出了 SDPM 算法和基于 SDPM 算法的 SVM 搜索 OTPC 的软件实现和软硬件协同实现。首先，简要阐述 SVM 实际应用过程中构建应用模型时面临的问题和处理流程；其次，提出了 SDPM 算法并基于 SDPM 算法的完成了 SVM 搜索 OTPC 的软件实现；最后，完成了基于 SDPM 算法的 SVM 搜索 OTPC 的软硬件协同实现，从架构到 RTL 级进行了详细阐述。

第四章主要对比和分析了 LIBSVM、基于本文提出的 SDPM 算法的搜索 OTPC 的软件实现和软硬件协同实现三种方式下的测试结果。首先，阐述了本文的开发和测试平台；其次，分析了本文使用的测试数据集和测试前的对数据集和测试参数的预处理；最后，详细分析了不同实现方式下资源的消耗、参数误差和运行时长，指明本文工作的价值意义。

第五章是对全文的总结。

第二章 支持向量机理论

本章主要介绍了统计学习及 SVM 的基本理论。包括统计学习的基础理论、支持向量分类机的种类及 SVM 模型的相关理论。其中，统计学习理论是支持向量分类机理论的基础，而 SVM 模型是支持向量分类机在实际应用中构建模型时面临的问题及其处理流程。支持向量分类机主要包含线性可分 SVM、线性 SVM[44]和非线性 SVM[45]，其中，以非线性 SVM 应用最为广泛。第三章的 SVM 模型也是基于本章的理论得以设计和实现。

2.1 统计学习理论

2.1.1 统计学习基础

统计学习是运用计算机及网络，以一部分数据集构建统计模型并使用模型对另一部分数据集进行预测和分析[46]的一门学科。统计学习涉及计算机科学、最优化理论、概率论及统计学多个领域，并在逐步发展中形成了自己的理论体系。统计学习的研究对象是数据，并且假设同类数据是相似的，这种相似包括不同数据间的特征及数据整体的统计相关性。统计学习方法主要包括监督学习[47]和非监督学习[48]，SVM 即是一种监督学习方法。

监督学习是统计学习重要的分支，是统计学习领域研究最深入的部分，也是 SVM 理论的基础。在监督学习理论中，每一个样本点都是一个实例 x，通常由特征向量及输出表示。样本集 **T** 可表示为：

$$\mathbf{T}=\{(x_1,y_1),(x_2,y_2),...,(x_N,y_N)\}$$

$\boldsymbol{Y}$ 表示输出变量，$\boldsymbol{X}$ 表示的是输入变量，x_i 表示输入实例中的第 i 个，由特征向量构成，可记作：

$$x_i=(x_i^{(1)},x_i^{(2)},...,x_i^{(j)},...x_i^{(n)})$$

$x_i^{(j)}$ 表示第 i 个输入实例中的第 j 个特征向量。监督学习是这样一种统计学习方法：以两组数据为基础，一组长度有限且满足独立同分布，其特征向量及输出均是已知的，称为训练数据集，另一组也是长度有限且满足独立同分布，特征向量已知但是输出未知，称为测试数据集；假定学习的模型是一个函数的集合，这个函数表征的是数据的特征向量和输出之间的依赖关系，函数的集合称为假设空间；采用某种评价标准，并基于此评价标准从假设空间中选出最优的模型，使之能够依据数据的特征向量对数据的输出有最优的预测；最优模型的构建是由具体的统计学习算法完成的。监督学习系统包含学习系统和预测系统，可用图 2-1 描述:

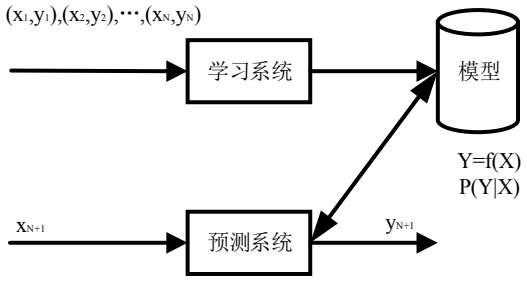

图 2-1 监督学习系统

其中，$x_1,x_2,...,x_N$ 是训练数据集，x_{N+1} 是测试数据集。学习系统从训练数据集读取数据并进行学习，然后生成模型，将生成模型应用于预测系统，预测系统读取测试数据集特征向量并进行输出，进而完成测试数据集的分析与预测。这就是监督学习系统。

2.1.2 经验风险与结构风险

监督学习的预测系统运用学习系统生成的模型，对于给定的输入 **X**，输出相应的预测值 $f(\mathbf{X})$。显然，这个预测值 $f(\mathbf{X})$ 与真实的输出值 **Y** 之间可能存在一定的差异，用损失函数来表征这种预测差异。这个损失函数是一个非负值，记作 $L(\mathbf{Y},f(\mathbf{X}))$。常用的损失函数为平方损失函数和绝对损失函数，模型性能与损失函数的值直接相关，一般来说，我们要让损失函数的值尽量小。模型的输入 $\boldsymbol{X}$ 和输出 $\boldsymbol{Y}$ 遵循联合分布 $P(\mathbf{X},\mathbf{Y})$，损失函数的期望是：Equation Chapter (Next) Section 1

$$R_{\exp}(f)=E_P[L(Y,f(X))]=\int L(y,f(x))P(x,y)dxdy \tag{2-1}$$

给定一个训练数据集 **T**，模型 $f(\mathbf{X})$ 关于训练数据集的平均损失称为经验风险，记作 R_{emp}：

$$R_{emp}(f)=\frac{1}{N}\sum_{i=1}^{N}L(y_i,f(x_i)) \tag{2-2}$$

经验风险 $R_{emp}(f)$ 是训练数据集的平均损失，期望风险 $R_{exp}(f)$ 是联合分布的期望损失，由大数定律[49]知，经验风险 $R_{emp}(f)$ 和期望风险 $R_{exp}(f)$ 在样本量 $\boldsymbol{N}$ 趋于无穷时将趋于一致，故可用经验风险估计期望风险。然而，现实应用中，训练数据集长度是有限的，用经验风险估计期望风险便不再适合。这时监督学习使用的基本策略是经验风险最小化和结构风险最小化。

经验风险最小化是假定在假设空间中，经验风险最小的模型就是最优的模型，因此经验风险最小化的模型求解即是求解如下的最优化问题：

$$\min_{f\in\Gamma}\frac{1}{N}\sum_{i=1}^{N}L(y_i,f(x_i)) \tag{2-3}$$

其中，Γ 是假设空间。

当样本容量很大时，基于经验风险最小化的学习模型性能良好，然而当样本容量很小时，经验风险最小化会产生过拟合[50]现象。防止过拟合的一种方法是使用结构风险最小化。定义正则化项或罚项来表征模型的复杂度，结构风险就是在经验风险的基础上加上这一正则化项，记作 $R_{srm}(f)$：

$$R_{\text{srm}}(f)=\frac{1}{N}\sum_{i=1}^{N}L(y_i,f(x_i))+\lambda J(f) \tag{2-4}$$

其中，$J(f)$ 是模型复杂度，其大小与模型的复杂度成反比。$\lambda\geq 0$ 是系数，是结构风险和模型复杂度的折中因子。当经验风险小且模型复杂度低时，由(2-4)知，结构风险 $R_{srm}(f)$ 小，此时的模型才能接近最优的模型。结构风险最小化是假定在假设空间中，结构风险最小的模型才是最优的模型，其最优模型求解等价于求解如下的最优化问题：

$$\min_{f\in\Gamma}\frac{1}{N}\sum_{i=1}^{N}L(y_i,f(x_i))+\lambda J(f) \tag{2-5}$$

监督学习问题实质就是经验风险函数和结构风险函数的最优化问题。SVM 的最优化问题即属于结构风险最小化，SVM 的模型求解即是结构风险函数的最优化。

2.1.3 分类和准确率

统计学习理论和方法的研究均是为其应用服务的，统计学习的两个重要应用就是分类和回归。分类是通过学习两类或两类以上的训练数据集，得到相应的模型，再利用这些模型对测试数据集进行类别判断，其实质就是将同类数据进行归类。监督学习从分类应用中学习到的模型或者决策函数被称为分类器。分类器对于新输入实例产生输出的过程即称为分类。分类问题的输出 Y 取有限个离散值，不同的输出代表了不同的类。当分类类别数为二时，称为二类分类问题，其余的称为多类分类问题。SVM 属于典型的二类分类问题，当 SVM 应用中存在多于两类的数据集时，可使用一定的方法将多类分类问题转化为两类分类问题处理。SVM 学习及分类系统可用图 2-2 描述：

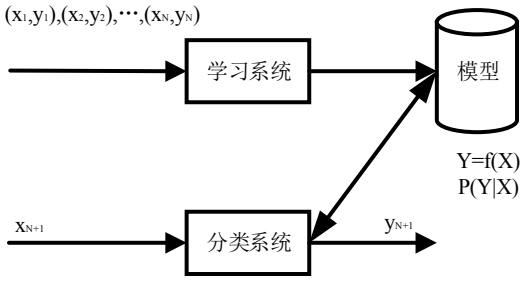

图 2-2 SVM 应用系统

当通过训练数据集构建好分类模型后，需对分类器性能进行评估，通常用分类准确率评估分类器性能。分类准确率是指在给定的一组测试数据集下，分类器输出正确的样本数与测试集总样本数的比值。对于二类分类问题，通常将我们关注的类定义为正类，另一类定义为负类。此时的分类准确率即是指分类器输入为正类且其预测输出为正类及分类器输入为负类且其预测输出为负类的的总和与分类器所有输入的比值，记作 R_{rate}：

$$R_{rate} = \frac{Num1 + Num2}{N} \bullet 100\% \tag{2-6}$$

其中，N 是训练数据集的总样本数，$Num1$ 是正类样本集预测准确的个数，$Num2$ 是负类样本预测准确的个数。分类准确率是 SVM 确定模型性能的关键指标，一般来说，我们认为某个模型的分类准确率越高，则模型性能越好。分类问题广泛应用于各个领域中，比如手写字识别[51]、文本分类[52]、垃圾邮件阻拦[53-54]及图像处理等。

2.2 支持向量分类机

SVM 是基于两类数据集构建模型，其学习目标是找到对训练数据集进行划分的最优超平面，该超平面在数学上可表示为：

$$w^T x + b = 0 \tag{2-7}$$

学习出公式(2-7)中的 w 和 b，即生成了 SVM 的模型。SVM 的学习策略是数据间隔的最大化，在数学上实质是求解凸二次规划问题[55]。所以 SVM 的学习算法就是求解凸二次规划的最优化算法。针对训练数据集的复杂情况，SVM 可分为线性可分 SVM、线性 SVM 及非线性 SVM。在实际应用中，SVM 求解的的最优化算法涉及拉格朗日理论、对偶理论和 SMO。

2.2.1 线性可分 SVM

2.2.1.1 线性可分 SVM 概念

假定在特征空间上存在线性可分的训练数据集 **T**：

$$\mathbf{T}=\{(x_1,y_1),(x_2,y_2),...,(x_N,y_N)\}$$

其中，$x_i \in R^n$，$y_i \in \{+1,-1\}$，$i=1,2,...,N$，x_i 是训练数据集的第 i 个实例。y_i 是 x_i 的类别标志，当 y_i = +1 时，表明 x_i 属于正类；当 y_i= -1 时，表明 x_i 属于负类，组合(x_i，y_i)是一个样本点。一般来说，对于线性可分的训练数据集，将两类数据分开的的超平面存在无穷多个，但是，线性可分 SVM 要求间隔最大化，此时，超平面是唯一的。

定义训练数据集线性可分，通过间隔最大化求分离超平面

$$w^*x+b^*=0 \tag{2-8}$$

以及相应的分类决策函数

$$f(x)=sign(w^*x+b^*) \tag{2-9}$$

公式(2-8)和(2-9)描述的即是线性可分 SVM，其分类示例如图 2-3 所示：

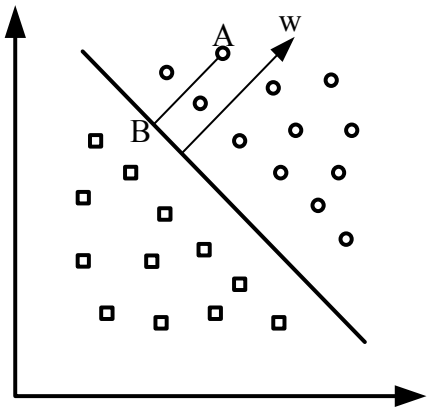

图 2-3 线性可分 SVM 示例

图 2-3 中，用“○”表示正类实例，用“□”表示负类实例。理论上说，存在着无数条直线可将两类实例正确划分。线性可分 SVM 对应着一条的直线，它离两类样本最近的点间隔最大且相等。间隔最大和 SVM 的求解将在后文进行阐述，这里先引入函数间隔和几何间隔的概念。

2.2.1.2 函数间隔和几何间隔

一旦确定了超平面 $wx+b=0$，则可用$|wx+b|$表征实例 x 到超平面的距离，通过判断 $wx+b$ 的符号与类标记 y 的符号是否一致，以确定对 x 的分类是否正确。因

此，可用 $y(wx+b)$表征分类的准确度。给定训练数据集 T 和超平面$(w，b)$，定义函数间隔 $\hat{\gamma}_i$：

$$\hat{\gamma}_i = y_i(wx_i + b) \tag{2-10}$$

定义$\hat{\gamma}$为 T 中所有样本点$(x_i，y_i)$的函数间隔的最小值：

$$\hat{\gamma} = \min_{i=1,\ldots,N} \hat{\gamma}_i \tag{2-11}$$

函数间隔只能表征分类的准确度，却不能表示实例 x 到超平面$(w，b)$的实际距离。可对 w 进行规范化，取$||w|| = 1$，则间隔是确定的。

如图 2-3，假定任意一个实例点，如 A，到超平面的实际距离以 γ_i^d 表示，B 点在超平面$(w，b)$上，是 A 在超平面$(w，b)$上的投影，则向量 BA 的方向是 w，单位向量是 $w/||w||$。设定 A 点的坐标是$(x_i，y_i)$，则 B 点可表示为：

$$x = x_i - \gamma_i^d \frac{w}{\|w\|} \tag{2-12}$$

将 B 点代入 $wx+b = 0$，有

$$w(x_i - \gamma_i^d \frac{w}{\|w\|}) + b = 0 \tag{2-13}$$

对(2-3)进行推导和化简有：

$$\gamma_i^d = \frac{wx_i + b}{\|w\|} = (\frac{w}{\|w\|})x_i + (\frac{b}{\|w\|}) \tag{2-14}$$

定义几何间隔 γ_i：

$$\gamma_i = y_i\gamma_i^d = y_i((\frac{w}{\|w\|})x_i + (\frac{b}{\|w\|})) \tag{2-15}$$

定义γ为 T 中所有样本点$(x_i，y_i)$的几何间隔的最小值：

$$\gamma = \min_{i=1,\ldots,N} \gamma_i \tag{2-16}$$

如上，几何间隔是带符号的实际距离，从函数间隔和几何间隔的定义可知，两者存在如下的关系：

$$\gamma_i = \frac{\hat{\gamma}_i}{\|w\|}$$
$$\gamma = \frac{\hat{\gamma}}{\|w\|} \tag{2-17}$$

显然，如果||w|| = 1，则函数间隔等于几何间隔，在 w 和 b 等比例缩放的情况下，函数间隔会变化而几何间隔保持不变。

2.2.1.3 间隔最大化与支持向量

A. 间隔最大化

在线性可分 SVM 中，间隔最大化是指超平面不仅要将所有正负实例分开，还要确保离超平面最近的正负实例有最大的区分度，即离超平面最近的正负实例到超平面的间距要最大且相等。硬间隔最大化在数学可描述如下：

$$\max_{w,b} \ \gamma$$
$$s.t. \quad y_i\left(\frac{w}{\|w\|}x_i + \frac{b}{\|w\|}\right) \geq \gamma \quad i = 1,2,...,N \tag{2-18}$$

由于几何间隔和函数间隔存在(2-17)的关系，(2-18)可进一步描述如下：

$$\max_{w,b} \ \frac{\hat{\gamma}}{\|w\|}$$
$$s.t. \quad y_i(wx_i + b) \geq \hat{\gamma} \quad i = 1,2,...,N \tag{2-19}$$

如前所述，函数间隔的值是可以任意改变的，这种改变不影响目标函数的求解。因此，可取 $\hat{\gamma} = 1$ 简化分析，同时，由于最大化 1/||w||和最小化 $1/(2\|w\|^2)$ 是等价的。故线性可分 SVM 的最优化问题可表示为如下形式：

$$\min_{w,b} \ \frac{1}{2}\|w\|^2$$
$$s.t. \quad y_i(wx_i + b) - 1 \geq 0 \quad i = 1,2,...,N \tag{2-20}$$

如果求得(2-20)的解 w^* 和 b^*，即可得到分割超平面 $w^*x + b^* = 0$ 和分类决策函数 $f(x) = sign(w^*x + b^*)$，也就得到了线性可分 SVM 的模型。可通过理论证明，线性可分 SVM 的最大间隔超平面是唯一的。

B. 支持向量

在训练数据集线性可分的情况下，那些距离超平面最近的实例点被称为支持向量。也就是说，支持向量是使式(2-20)的约束条件等号成立的样本点，即 $y_i(wx_i+b)$-

1=0 所对应的 x_i。在这种定义下，在支持向量中间，是没有任何实例点的，只有支持向量决定着超平面，决定着线性可分 SVM 的模型。其它实例点对模型的建立不起作用，故增加或移除非支持向量实例点，对模型没有影响。支持向量的个数一般不多，故线性可分 SVM 的模型是由少数重要的样本点确定的。

2.2.2 线性 SVM

2.2.2.1 线性 SVM 的概念

线性可分 SVM 要求训练数据集线性可分，实际应用中，多数情况下训练数据集是线性不可分的。处理线性不可分的训练数据集，需在线性可分 SVM 的基础上进行扩展。线性 SVM 的示例如图 2-4 所示：

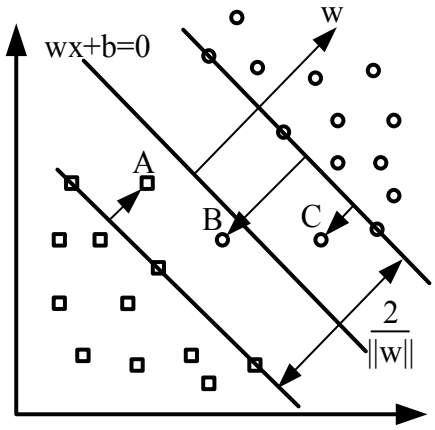

图 2-4 线性 SVM 示例

定义线性不可分训练数据集，通过间隔最大化求得超平面：$w^*x+b^*=0$ 和相应的分类决策函数：$f(x)=sign(w^*x+b^*)$，称为线性 SVM。

线性不可分意味着训练数据集中存在一些奇异点，如图 2-4 中的 A、B、C 三个实例点所示，A、B、C 即是奇异点，如果去掉这些奇异点，剩下的绝大部分数据线性可分。在线性不可分的训练数据集中，这些奇异点的函数间隔不满足大于或者等于 1 的约束条件，此时，需对(2-20)的约束条件进行修改。修改方式是引入松弛变量 $\xi_i \geq 0$ ，使每个实例点的函数间隔加上松弛变量后大于等于 1，如此，约束条件变为 $y_i(wx_i+b)\geq 1-\xi_i$ ，同时，对每个松弛变量，引入一个代价。这样，目标函数由线性可分 SVM 的 $1/(2\|w\|^2)$ 变为：

$$\frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}\xi_i \tag{2-21}$$

其中，$C>0$ 称为惩罚系数。最小化目标函数(2-21)意味着间隔要尽量大而$1/(2\|w\|^2)$要尽量小，同时，要使被误分类样本点的个数尽量少。线性 SVM 的最优化问题描述如下：

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y_i(wx_i+b) \geq 1-\xi_i, i=1,2,...,N \\ & \xi_i \geq 0, i=1,2,...,N \end{aligned} \tag{2-22}$$

线性 SVM 是对线性可分 SVM 的扩展，其包含了线性可分 SVM，因而具有更广泛的应用。

2.2.2.2 线性 SVM 学习的对偶算法

设定线性 SVM 的最优化问题(2-22)为原始问题，显然，式(2-22)的约束条件满足 *Slater* 条件，这样，便可以利用拉格朗日的对偶性，通过求解原始问题的对偶问题求出原始问题的最优解。这样做的目的是对偶问题更容易求解，且能引入核函数[57]，从而将 SVM 进一步推广到非线性分类问题。

构建拉格朗日函数：

$$L(w,b,\xi,\alpha,\mu)=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}\xi_i-\sum_{i=1}^{N}\alpha_i(y_i(wx_i+b)-1+\xi_i)-\sum_{i=1}^{N}\mu_i\xi_i \tag{2-23}$$

其中，$\alpha=(\alpha_1,\alpha_2,...,\alpha_N),\mu=(\mu_1,\mu_2,...,\mu_N)$是拉格朗日乘子向量，$\alpha_i\geq 0,\mu_i\geq 0$。由此，原始问题（2-22）可描述为：$\min_{w,b,\xi}\max_{\alpha,\mu}L(w,b,\xi,\alpha,\mu)$，其对偶问题为$\max_{\alpha,\mu}\min_{w,b,\xi}L(w,b,\xi,\alpha,\mu)$。因此，要求解对偶问题，先求$L(w,b,\xi,\alpha,\mu)$关于$w,b,\xi$的极小值，再求$L(w,b,\xi,\alpha,\mu)$关于$\alpha,\mu$的极大值。

求 $L(w,b,\alpha)$ 关于 w,b,ξ 的极小值 $\min_{w,b,\xi}L(w,b,\xi,\alpha,\mu)$：将拉格朗日函数$L(w,b,\xi,\alpha,\mu)$分别对$w,b,\xi$求导有：

$$\begin{aligned} \nabla_w L(w,b,\xi,\alpha,\mu) &= w-\sum_{i=1}^{N}\alpha_i y_i x_i = 0 \\ \nabla_b L(w,b,\xi,\alpha,\mu) &= -\sum_{i=1}^{N}\alpha_i y_i = 0 \\ \nabla_{\xi_i} L(w,b,\xi,\alpha,\mu) &= C-\alpha_i-\mu_i = 0 \end{aligned} \tag{2-24}$$

得

$$\begin{aligned} & w=\sum_{i=1}^{N}\alpha_i y_i x_i \\ & \sum_{i=1}^{N}\alpha_i y_i=0 \\ & C-\alpha_i-\mu_i=0 \end{aligned} \tag{2-25}$$

将式(2-25)带入式(2-23)，有

$$\min_{w,b,\xi} L(w,b,\xi,\alpha,\mu)=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i.x_j)+\sum_{i=1}^{N}\alpha_i \tag{2-26}$$

由 $\min_{w,b,\xi} L(w,b,\xi,\alpha,\mu)$ 求 α 的极大，即可得到对偶问题：

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i.x_j)+\sum_{i=1}^{N}\alpha_i \\ s.t. \quad & \sum_{i=1}^{N}\alpha_i y_i=0 \\ & C-\alpha_i-\mu_i=0 \\ & \alpha_i\geq 0 \\ & \mu_i\geq 0, i=1,2,...,N \end{aligned} \tag{2-27}$$

利用式(2-27)的等式约束消去 μ_i，约束条件可改写为：$0\leq\alpha_i\leq C$。将目标函数由极大转化为求极小，可将式(2-22)描述的线性 SVM 的最优化问题进一步描述为：

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i.x_j)-\sum_{i=1}^{N}\alpha_i \\ s.t. \quad & \sum_{i=1}^{N}\alpha_i y_i=0 \\ & 0\leq\alpha_i\leq C \end{aligned} \tag{2-28}$$

继续求解式(2-28)，得到 $\alpha^*=(\alpha_1^*,\alpha_2^*,...,\alpha_N^*)$，则可依据式(2-25)得到 w^*，选择满足 $0<\alpha_j^*<C$ 的任一 α^* 的分量 α_j^*，求得 $b^*=y_j-\sum_{i=1}^{N}y_i\alpha_i^*(x_i.x_j)$。则可计算出超平面和分类决策函数，求得线性 SVM 的模型。

2.2.3 非线性 SVM

2.2.3.1 非线性 SVM 学习算法

现实应用中的分类问题，很多是非线性的。如图 2-5 所示，无法用任何直线（线性模型）将两类实例点分开，只能通过椭圆曲线（非线性模型）才能实现分类的目的。

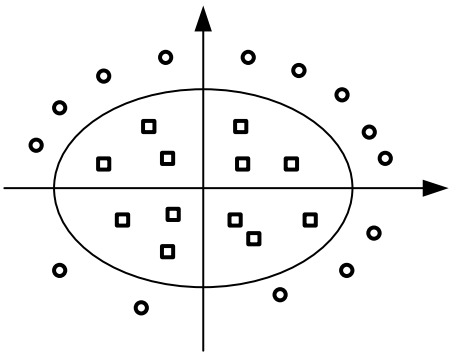

图 2-5 非线性 SVM 示例

处理非线性分类问题需要用到核技巧，核技巧是一种非线性变换，核技巧的引入，可将非线性分类问题转化为线性分类问题求解。因此，非线性分类问题的求解通常包含两个过程：首选选择一种变换方式将数据由原始空间映射到新空间；然后在新空间里使用线性分类方法处理训练数据集并学习模型。如果存在一个从输入空间 $\aleph$ 到特征空间 **H** 的映射 $\phi(x)$，使得对于所有的 $x,z\in\aleph$，函数 $K(x,z)$ 满足：

$$K(x,z)=\phi(x).\phi(z) \tag{2-29}$$

称 $K(x,z)$ 为核函数，$\phi(x)$ 为映射函数，$\phi(x).\phi(z)$ 是 $\phi(x)$ 与 $\phi(z)$ 的内积。实际应用中，通常只定义 $K(x,z)$ 而不显式的定义 $\phi(x)$。核技巧应用于 SVM，就是将线性 SVM 的目标函数和分类决策函数中的内积 $(x_i\bullet x_j)$ 用核函数 $K(x,z)=\phi(x).\phi(z)$ 来代替，进而完成非线性 SVM 转化为线性 SVM 进行求解。非线性 SVM 的学习算法描述如下：

A. 选择适当的核函数 $K(x,z)$ 和惩罚系数 C，构造并求解最优化问题

$$\begin{aligned}\min_{\alpha}\quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i\bullet x_j)-\sum_{i=1}^{N}\alpha_i \\ s.t.\quad & \sum_{i=1}^{N}\alpha_i y_i=0 \\ & 0\le\alpha_i\le C, i=1,2,...,N\end{aligned} \tag{2-30}$$

B. 选择 α^* 的分量 α_j^*，满足 $0<\alpha_j^*<C$，计算

$$b^*=y_j-\sum_{i=1}^{N}y_i\alpha_i^*K(x_i\bullet x_j) \tag{2-31}$$

C. 构造决策函数

$$f(x)=sign\sum_{i=1}^{N}y_i\alpha_i^*K(x_i\bullet x_j)+b^* \tag{2-32}$$

当 $K(x,z)$是正定核[58]函数时，式(2-32)是凸二次规划问题，解是唯一存在的。

2.2.3.2 常用的核函数

在长期的应用实践中，人们总结了几种常用的核函数。非线性 SVM 在选取核函数时，并没有特定的选取规则，往往因具体问题而异。如下列出了几种常用的核函数：

A. 多项式核函数

$$K(x,z)=(x\bullet z+1)^p \tag{2-33}$$

其中，p 代表核函数是 p 次多项式。

B. 高斯核函数

$$K(x,z)=\exp(-\frac{\|x-z\|^2}{2\sigma^2}) \tag{2-34}$$

高斯核函数又称为高斯径向基核函数，应用较为广泛。本文设计的系统所选取的核函数即为高斯径向基核函数。

2.3 最优训练参数组搜索理论

使用 SVM 解决实际问题时，有如下三个问题需要考虑：首先，SVM 是二类分类问题，但现实生活中处理的数据通常大于两类，此时需将多类分类问题转化为两类分类问题；其次，模型的构建除了需要训练数据集，还需要选择较好的训练参数组，不同参数组的选择深刻的影响着最后生成模型的性能；最后，则是选择何种具体的算法实现 SVM，即得到式(2-30)的最优解。本节将从这三方面就 SVM 的应用模型展开阐述。

2.3.1 多类分类问题

SVM 属于二类分类模型，当存在多类（假定为 V 类）数据需进行分类时，常用的解决方法有两种：其一是 one-to-all 形式，即在进行多类数据的分类时，将某一类训练数据指定为正类，其余所有的训练数据指定为负类。完成一次分类后，再重新指定某一类训练数据为正类，其余所有的训练数据为负类，直到所有每个单独的类均被指定为一次正类并完成相应的分类。在总类别数为 V 的情况下，只需进

行 V 次分类即可。显然该方法需进行的分类次数不多，然而，该方法的缺点是得到的分类模型性能很差；其二是 one-to-one 形式，即将训练数据两两进行组合，这样，一个数据的组合只包含两类训练数据，将每个组合逐一分类直到完成所有组合的分类。在总类别数为 V 的情况下，该方法需进行 $(V \bullet (V-1))/2$ 次分类，相比于 one-to-all 分类方式，增加了需分类的次数，但得到的分类模型性能更好。本文第三章的设计在处理 SVM 多类分类问题时采用的是 one-to-one 形式。

2.3.2 最优训练参数组

在训练确定的两类数据的非线性 SVM 模型时，如果选择高斯径向基函数为核函数，则 SVM 训练应用模型需要两个关键的训练参数：一是惩罚系数 C，该参数决定了对分类错误的容忍度，同时也是训练生成的模型的系数值的上限；二是高斯核函数中的参数 σ，该参数影响系统的过拟合问题，适当的 σ 对生成最佳的模型起着重要的作用。在本文的设计中，我们称性能最好的模型所对应的训练参数组为最优训练参数组合（OTPC）。实际应用中，OTPC 在大多数情况下是未知的，在真正使用训练数据集构建模型前，需选择合适的方式先找到模型的 OTPC。传统的搜索 OTPC 的方式是网格搜索法和交叉验证。

2.3.2.1 网格搜索法

网格搜索法是这样一种搜索训练数据集 OTPC 的方式：首先在有限的实数区间内，分别对训练参数 C 和 σ 取离散值；然后将这些值两两组合得到不同的训练参数组集合；最后将这个集合中的训练参数组结合训练数据集分别进行独立的交叉验证训练，得到所有训练参数组下交叉验证训练的准确率后，找出最高准确率对应的一组训练参数组合，该参数组合即是训练数据集的 OTPC。

2.3.2.2 交叉验证

交叉验证是训练数据集不充分的情况下常用的数据折叠方式，其中以 s 折交叉验证的使用最为广泛。其方法是：首先将训练数据集随机的分成 s 份互不重叠、数量相等的子数据集；然后将 s-1 个子集的数据结合训练参数组完成交叉验证训练并生成模型，用余下的子数据集验证模型并统计该子数据集的准确率；将这一过程重复进行，直到每个子集都分别参与了训练和验证，最后统计出所有训练数据在该训练参数组下的准确率。使用网格搜索法统计到所有训练参数组对应的准确率后，选出最高准确率对应的训练参数组结合整个训练数据集构建模型作为最终的应用模型。交叉验证的基本思想是数据的重复使用，通过不断地把训练数据集细分为更小的训练集和测试集，进而完成交叉验证训练、测试和模型构建。

2.3.3 SVM 实现算法

在 SVM 的应用中，在确定了训练数据集、训练参数组和数据的折叠方式及折叠次数后，最后就是进行具体的交叉验证训练以生成折叠数据的模型并验证该模型的性能。非线性 SVM 的单次交叉验证训练在数学上是解决问题(2-30)，核函数形式如式(2-34)。有很多算法用于解决问题(2-30)，其中最传统的是最小序列最优化算法（SMO），由于 SMO 每次迭代均需更新阈值和误差，后面的研究者又对 SMO 进行了改进，提出了基于 SMO 的改进 SVM 实现算法。

2.3.3.1 SMO 算法

SMO 算法是一种迭代算法，其每次迭代都会选择并优化两个工作集，使得目标函数(2-30)的下界降低，直到所有的变量都满足 KKT 条件，算法收敛，目标函数(2-30)的下界保持不变，同时得到(2-30)的最优解。在每次迭代中，使用特定的选择算法选择两个变量进行优化而其他变量保持不变，则(2-30)的多变量优化问题变成了两个变量的二次规划问题，这个二次规划问题可使用解析方法求解。SMO 算法通过多次迭代将原最优化问题分解为子问题同时求解子问题，最终实现原问题的求解。SMO 算法的样本点满足的 KKT 条件表示如下：

$$\begin{aligned} &\alpha_i = 0 <=> y_i g(x_i) - 1 \geq 0 \\ &0 < \alpha_i < C <=> y_i g(x_i) - 1 = 0 \\ &\alpha_i = C <=> y_i g(x_i) - 1 \leq 0 \end{aligned} \tag{2-35}$$

其中，$g(x_i) = \sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) + \mathrm{b}$。

2.3.3.2 改进的 SVM 实现算法

SMO 算法在每次迭代中更新样本误差前均需计算阈值，且其工作集选择算法仅使用了一阶导。为进一步提升 SMO 算法的性能，很多研究者从各方面对 SMO 算法进行了改进，并取得了较好的结果。总的来说，改进的 SVM 实现算法主要包含四个过程：搜索工作集索引 i；搜索工作集索引 j；更新拉格朗日系数；更新梯度。改进的 SMO 实现算法是第三章交叉验证训练硬件设计的理论基础，而交叉验证训练是搜索 SVM 最优训练参数组的重要组成部分。

A. 不考虑 b 值的 KKT 条件判别法

Keerthi 等提出了一种新的 KKT 条件判别方法，这种方法无需在每次迭代时更新阈值，只需在迭代收敛后计算一次阈值即可。假定

$$F_i = \sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) - y_i \tag{2-36}$$

有

$$\begin{aligned} y_i(F_i + b) &= y_i(\sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) + b) - 1 \\ &= y_i g(x_i) - 1 \end{aligned} \tag{2-37}$$

则 SMO 算法中的 KKT 条件(2-35)可进一步改写为

$$\begin{aligned} &\alpha_i = 0 <=> y_i(F_i + b) \geq 0 \\ &0 < \alpha_i < C <=> y_i(F_i + b) = 0 \\ &\alpha_i = C <=> y_i(F_i + b) \leq 0 \end{aligned} \tag{2-38}$$

引入记号：

$$\begin{aligned} &I_0 = \{i : 0 < \alpha_i < C\}, when\ F_i = -b \\ &I_1 = \{i : y_i = +1, while\ \alpha_i = 0\}, when\ F_i \geq -b \\ &I_2 = \{i : y_i = -1, while\ \alpha_i = C\}, when\ F_i \geq -b \\ &I_3 = \{i : y_i = +1, while\ \alpha_i = C\}, when\ F_i \leq -b \\ &I_4 = \{i : y_i = -1, while\ \alpha_i = 0\}, when\ F_i \leq -b \end{aligned} \tag{2-39}$$

则式(2-39)可描述为

$$\begin{aligned} &i \in I_0 \cup I_1 \cup I_2 <=> F \geq -b \\ &i \in I_0 \cup I_3 \cup I_4 <=> F \leq -b \end{aligned} \tag{2-40}$$

当 KKT 条件满足时，对 $\forall i \in I_0 \cup I_1 \cup I_2, \forall j \in I_0 \cup I_3 \cup I_4$，有 $F_i \geq -b \geq F_j$，假定：

$$\begin{aligned} b_{low} &= \max\{-F_i : i \in I_0 \cup I_1 \cup I_2\} \\ b_{up} &= \min\{-F_j : j \in I_0 \cup I_3 \cup I_4\} \end{aligned} \tag{2-41}$$

有 $b_{up} \geq b_{low}$，由此，可直接判断 $b_{up} \geq b_{low}$ 是否成立，而不用再每次更新 b 值判断所求解是否满足 KKT 条件。不等式 $b_{up} \geq b_{low}$ 是改进的 SVM 实现算法工作集索引选取算法的基本条件。

B. 工作集索引选取算法

2005 年，R.-E. Fan, P.-H. Chen, and C.-J. Lin 等人基于(2-41)及条件$b_{up} \geq b_{low}$，提出使用二阶导数信息选取工作集索引的方法。本文第三章的工作集选择算法也是基于本部分进行设计和实现。

定义对于所有的 *t*，*s*：

$$\begin{aligned} &a_{ts} = K_{tt} + K_{ss} - 2K_{ts}, \quad b_{ts} = -y_t G_t + y_s G_s > 0, \\ &and \\ &\bar{a}_{ts} = \begin{cases} a_{ts} & if \ \ a_{ts} > 0 \\ \tau & otherwise \end{cases} \end{aligned} \tag{2-42}$$

其中，*G* 是梯度，即导数信息。

工作集索引 *i* 选择方法如下：

$$i \in \arg\max_t \left\{ -y_t G_t \mid t \in I_{up}(\alpha^k) \right\} \tag{2-43}$$

其中

$$I_{up}(\alpha) = \{t \mid \alpha_t < C, y_t = 1 \ or \ \alpha_t > 0, y_t = -1\} \tag{2-44}$$

工作集索引 *j* 选择方法如下：

$$j \in \arg\min_t \left\{ -\frac{b_{it}^2}{\bar{a}_{it}} \mid t \in I_{low}(\alpha^k), -y_t G_t < -y_i G_i \right\} \tag{2-45}$$

其中

$$I_{low}(\alpha) = \{t \mid \alpha_t < C, y_t = -1 \ or \ \alpha_t > 0, y_t = 1\} \tag{2-46}$$

当计算得到工作集索引 *i*，*j* 后，返回工作集索引集 $B = \{i, j\}$ 即可。工作集索引选取算法是 SVM 实现算法的重要组成部分，只有得到工作集的索引，才能在本次迭代中对模型系数α_i, α_j进行优化，以进一步减小目标函数(2-30)的值。

C. 更新拉格朗日系数

当得到工作集的索引集 B 后，即可更新对应的拉格朗日系数α_i, α_j。在更新拉格朗日系数前，需先计算出系数的改变量 *delta*，再由原拉格朗日系数增加或减掉 *delta*。另外，由式(2-30)的约束条件知，拉格朗日系数值在 0 到 *C* 的范围内，故更新拉格朗日系数后，还需检查更新后的新系数值是否符合限定的范围，如果符合则继续完成后续实现，如果不符合则需进行必要的修改。

D. 更新梯度

梯度更新是改进的 SMO 实现算法在单次迭代过程中的最后一部分。由于 G 的长度为 N，可采用循环的方式逐个进行更新即可，其更新算法如下：

$$\begin{aligned} &Loop\ over\ all\ possible\ k,\ k\ is\ loop\ variable \\ &G[\mathrm{k}]^{new} = G[\mathrm{k}]^{old} + Q_i[\mathrm{k}]\bullet delta\_\alpha\_i + Q_j[\mathrm{k}]\bullet delta\_\alpha\_j \end{aligned} \tag{2-47}$$

其中，$Q_t = y_t K_t$，K 是工作集对应的核函数向量，由式(2-34)计算得出。$delta\_\alpha\_i$ 和 $delta\_\alpha\_j$ 是拉格朗日系数的实际改变量，由更新拉格朗日系数部分计算得到。

E. 收敛条件

改进的 SVM 实现算法的收敛条件如下：

$$G_{i\_max} + G_{j\_max} < eps \tag{2-48}$$

其中，G_{i_max} 是工作集 i 索引对应的梯度最大值，G_{j_max} 工作集 j 索引对应的梯度最大值，如果 G_{i_max} 和 G_{j_max} 的和值在一定的允许误差 eps 内，则本次迭代收敛，求得(2-30)的最优解。

2.4 本章小结

本章介绍了统计学习的基础理论及 SVM 相关的基本理论。统计学习理论介绍了统计学习的基本问题、监督学习、结构风险与经验风险及分类和准确率。支持向量分类机包含线性可分 SVM、线性 SVM 和非线性 SVM，其处理的对象分别是线性可分训练数据集和线性不可分训练数据集，非线性 SVM 需引入了核函数，以处理非线性数据。在 SVM 的实际应用中，训练数据集往往不只两类，且 OTPC 往往是未知的，这都需要采取合适的方式进行处理，本章最后对 SVM 的实现算法 SMO 进行了阐述，同时为提升 SMO 性能，对 SMO 的改进算法进行了详细介绍。

第三章 基于 SDPM 算法的搜索 SVM 最优训练参数组实现

本章提出了共享点积矩阵（Share Dot Product Matrix，SDPM）算法；阐述了搜索 SVM 模型的 OTPC 时面临的问题、基于 SDPM 算法的 SVM 模型的软件实现思想及流程、基于 SDPM 算法的 SVM 模型的软硬件协同实现的设计方法及兼具效率和资源的考量，对软硬件系统的硬件设计进行了详细描述。本章设计极大的提升了 SVM 搜索 OTPC 的速度性能，不仅能应用于 SVM 搜索 OTPC，还能用于 SVM 通过训练数据集快速地构建模型。对本章设计的 SVM 模型的性能及评估将在第四章进行讨论和比较。

3.1 引言

3.1.1 SVM 搜索 OTPC 时面临的问题

在 SVM 构建应用模型的训练阶段，当训练数据集的类别数很大时，采用 one-to-one 的形式会涉及大量的类别组合；在选定的某个类别组合（只包含两类训练数据的情况）下，训练参数的组合方式也是多种多样；当选定某组训练参数后，还需将训练数据集进行折叠，这又涉及大量的交叉验证训练；而每次交叉验证训练又是由多次迭代构成，且每次迭代都包含了相当多的计算量。

总的来说，在以高斯函数为核函数的 C-支持向量[59]分类应用中，传统软件在实现 SVM 应用模型时面临的问题是：软件搜索 SVM 最佳应用模型的 OTPC 用时过久。实际应用中，由于 SVM 最佳应用模型的 OTPC 是未知的，故只能通过网格搜索法结合交叉验证训练搜索得到。网格搜索法实质是以一种遍历方式的方式完成每组训练参数的交叉验证训练，而交叉验证训练是将数据进行折叠后再分别训练，更是极大的增加了需要训练的总次数，从而使得软件搜索 OTPC 需要很长的时间；传统硬件在实现 SVM 应用模型面临的主要问题是：目前硬件实现单次交叉验证训练过程均是基于 SMO 算法，尚未找到基于改进 SMO 算法的硬件实现。这两个问题降低了 SVM 搜索 OTPC 的速度性能，限制了 SVM 在某些场合的应用。

3.1.2 本章实现方案概述

为提升 SVM 搜索 OTPC 的速度性能，本章分别从算法和实现上进行了探索：算法上，本章提出了 SDPM 算法，以减少搜索 OTPC 过程中点积的计算量；实现上，本章首先基于 SDPM 算法完成了 SVM 搜索 OTPC 的软件实现和仿真，相比于 LIBSVM 实现了一定的搜索速度提升。为进一步提升搜索性能，本章基于 SDPM

算法完成了 SVM 搜索 OTPC 的软硬件协同实现，特别是交叉验证训练模块，本章在硬件上对改进的 SMO 算法进行了详细实现，同时对协同系统进行了大量的测试和分析，相比于 LIBSVM 实现了搜索速度的最大提升。本章接下来的部分将对 SDPM 算法和两种实现方式分别进行阐述。

3.2 基于 SDPM 算法的搜索 OTPC 的软件实现

3.2.1 SDPM 算法

SVM 使用网格搜索法搜索 OTPC 时，在每组训练参数组合下的交叉验证训练都是独立的。每次交叉验证训练都包含大量的迭代过程，在单次迭代过程中，先使用特定的工作集选择算法选取一组待优化工作集索引(i，j)；再按式(2-34)计算核函数 $K(x_i, x_j)$并以链表存储的形式存储计算结果，进而完成后续训练。链表存储的方式能有效的减少存储空间，但在网格搜索法搜索 OTPC 时，在不同训练参数组下的交叉验证训练过程中，其不同的迭代轮次，很可能会重复选到相同的工作集索引(i，j)，观察式(2-34)可知，这会造成$(x_i - x_j)^2$的重复计算，进而增加搜索过程中点积的计算量。

3.2.1.1 SDPM 算法原理

网格搜索法在一次迭代过程中，在选取到工作集索引(i，j)后依据训练数据集中的 x_i，x_j 临时计算$(x_i - x_j)^2$进行存储然后完成后续训练。SDPM 算法则是在处理器读取到所有的训练数据集并完成预处理后，先直接计算出每一个实例点和所有实例点（包括自己）的点积并存储为点积矩阵，在后面的训练和迭代过程中，当选定工作集索引(i，j)后，无需再计算$(x_i - x_j)^2$，直接从点积矩阵中读取相应的点积向量完成后续训练。该算法只需在搜索之前计算点积并存储为点积矩阵，后续的搜索过程在需要点积向量时，统一从点积矩阵中读取向量结果，此即本章提出的 SDPM 算法。SDPM 算法由于要存储点积矩阵，故会占用一定的存储空间。然而，理论计算表明在使用网格搜索法搜索 OTPC 时，使用 SDPM 算法能极大地减少点积的计算量，进而缩短搜索 OTPC 的时长，提升搜索的速度性能。

3.2.1.2 SDPM 算法复杂度分析

A. 存储空间

假定训练数据集的总长度为 N，存储器内部点积数据的存储位宽为 W，则所需的存储空间$T = W \times N^2$，设定 N=20000，W=16Bits，则 T=800MB。显然，当下存储卡的存储大小是远远大于 800MB 的。

B. 点积计算量

假定训练数据集的总长度为 N，惩罚系数 C 的可选值个数为 m，高斯核函数的参数 σ 的可选值个数为 n，交叉验证的折叠次数为 s，每次训练平均迭代次数因子为 γ，平均迭代次数因子指一次交叉验证训练过程中，迭代的平均次数与训练数据集总长度的比值。Equation Chapter (Next) Section 1

则传统方法搜索 OTPC 时点积 $(x_i - x_j)^2$ 的计算次数 $Num1$ 约为

$$Num1 = (m \times n) \times k \times (N \times \frac{s-1}{s} \times \gamma) \times (2 \times N) \tag{3-1}$$

使用 SDPM 算法搜索 OTPC 时点积 $(x_i - x_j)^2$ 的计算次数 $Num2$ 约为

$$Num2 = k \times (N \times \frac{s-1}{s})^2 \tag{3-2}$$

设定 N=20000，m=n=16，s=5，γ=0.1，分别代入式(3-1)和式(3-2)，则 $Num1 = 8.2 \times 10^{10}$，$Num2 = 1.3 \times 10^{9}$，可见，前者的点积计算量约为后者的 60 倍，因此，使用 SDPM 算法能极大的减少搜索 OTPC 过程中点积的计算量。

3.2.2 基于 SDPM 算法的软件实现

SVM 特别适合在小样本容量下学习并构建模型。在小样本容量下，搜索 OTPC 时使用 SDPM 算法，能显著的减少交叉验证训练过程中点积的计算量，进而缩短搜索时长、提升搜索性能。本节将阐述基于 SDPM 算法的 SVM 模型的软件实现。

3.2.2.1 软件设计思想

在进行任何软件系统的设计时，层次设计思想都是非常重要的。在 SVM 构建模型的训练阶段，在搜索 OTPC 时，如何准确合理的处理多类别组合、多参数组合、数据折叠及交叉验证训练并结合 SDPM 算法，是非常重要的问题。本章在解决这个问题时，即采用的是层次设计法：首先，将搜索的大过程划分为几个层次的小过程，这样可以简化问题；然后，找到不同层次之间的联系和衔接，使得处理过程按序进行；最后，定义好不同层次的输入和输出，即实现不同层次的功能。完成搜索 OTPC 的层次设计后，再进行代码编写、仿真和测试，最终完成基于 SDPM 算法的搜索 OTPC 的软件实现。

3.2.2.2 软件实现流程

软件实现的具体流程如图 3-1 所示：

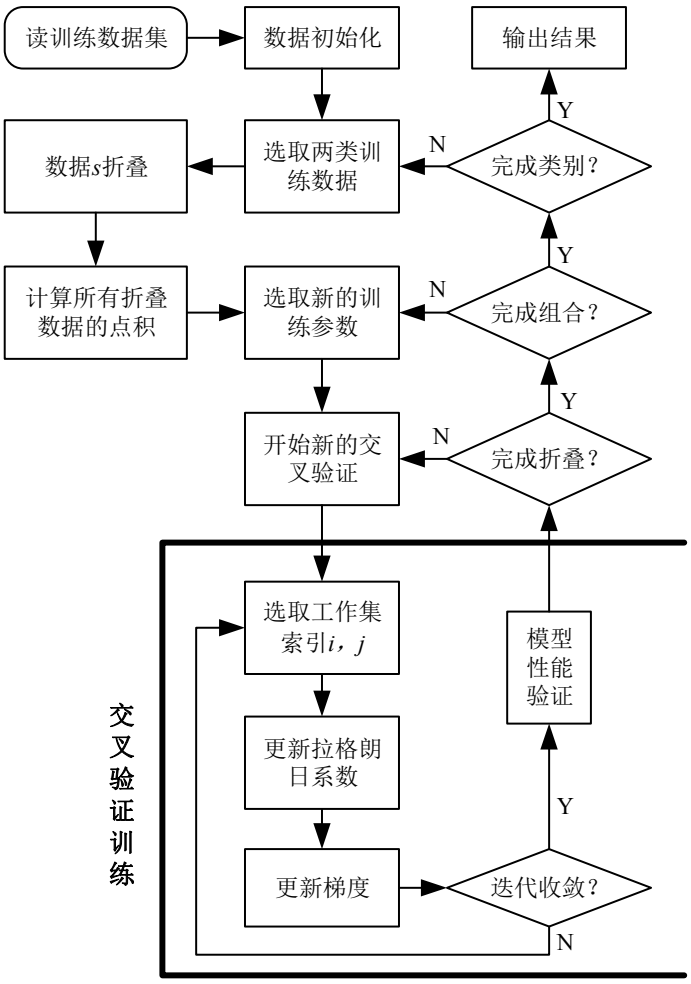

图 3-1 SVM 模型的软件实现框图

基于 SDPM 算法的 SVM 模型的软件实现分如下几个步骤完成：

A. 处理器从外置存储器（例如硬盘、移动硬盘等）读取所有的训练数据集到内存并保存，然后完成数据的初始化；

B. 如果是多类数据，则按照 one-to-one 的方式从训练数据集中选择两类数据，如果只有两类数据，则直接读取数据，准备搜索这两类数据的 OTPC；

C. 将 B 中的训练数据集 *s* 折叠，并计算每份折叠数据的点积矩阵并存储，先行计算训练数据的点积矩阵并存储，这是 SDPM 算法的基础；

D. 利用指定的训练参数 C 和 σ 的范围确定训练参数组的集合，再从这个集合中选择一组训练参数，训练数据集将在该组训练参数下完成交叉验证训练并统计准确率；

E. 实现数据的交叉验证训练并统计数据的准确率。交叉验证训练过程包含训练和测试两部分，单次交叉验证的数据集也被分为了训练数据集和测试数据集两部

分：训练过程由多次迭代实现，在单次迭代过程中，首先使用特定的工作集选择算法选取工作集索引 i，j，然后更新工作集索引 i，j 对应的拉格朗日系数，最后更新所有的梯度值。测试过程则是在训练过程结束后，先通过拉格朗日系数计算阈值 b，再利用拉格朗日系数和 b 构建该份训练数据集下的模型，最后使用测试数据集验证模型的性能并统计测试数据集的准确率。

F. 当某份折叠数据对应的交叉验证训练结束后，即开始另一份折叠数据的交叉验证训练，直到完成所有折叠数据的交叉验证训练，同时得到该组训练参数下模型的准确率并保存；

G. 完成某组训练参数组合下模型的准确率统计后，重新选定一组训练参数组合，完成 E、F，直到所有训练参数组集合中的参数组均完成了训练，得到其对应的准确率。

H. 完成两类数据的搜索 OTPC 的过程后，从训练数据集中重新选定两类训练数据集，完成 C、D、E、F，直到完成所有训练数据类别组合下的 OTPC 搜索。最后，保存结果并输出。

3.3 基于 SDPM 算法的搜索 OTPC 的软硬件协同实现

基于 SDPM 算法的搜索 OTPC 的软件实现极大的减少了搜索过程中点积的计算量。然而 SVM 搜索 OTPC 并构建应用模型的过程涉及极其巨大的计算量，基于 SDPM 算法的软件实现在搜索 SVM 的 OTPC 时，其执行时间还是太久，依旧不能满足 SVM 对搜索速度的需求。为最大限度地提升 SVM 搜索 OTPC 的速度性能，本章设计了一套基于 SDPM 算法的软硬件协同架构，显著提升了 SVM 的 OTPC 搜索性能，进而完成模型构建并满足应用的需要。

3.3.1 系统整体框架设计

本章设计的软硬件协同架构如图 3-2 所示。系统的主体结构由一台基于 x86 架构处理器的 PC 和加速板卡构成。加速板卡包含一片 Xilinx 的 Virtex 7 系列 FPGA 芯片，承担训练数据的数据接收、点积计算、核函数运算及具体的交叉验证训练过程；还包含一块 DDR3 内存条，用于存储折叠数据经点积计算后生成的点积矩阵。PC 机和加速板卡之间通过外设组件快速互连（Peripheral Component Interconnect express，PCI-e）协议传输训练数据和模型参数，FPGA 与 DDR3 内存条之间通过集成的 DDR3 控制 IP 核控制数据进行传输。整个系统数据的流向是： PC 机输送训练数据至加速板卡，在加速板卡内部完成数据处理，最后加速板卡输送模型参数至 PC 机。

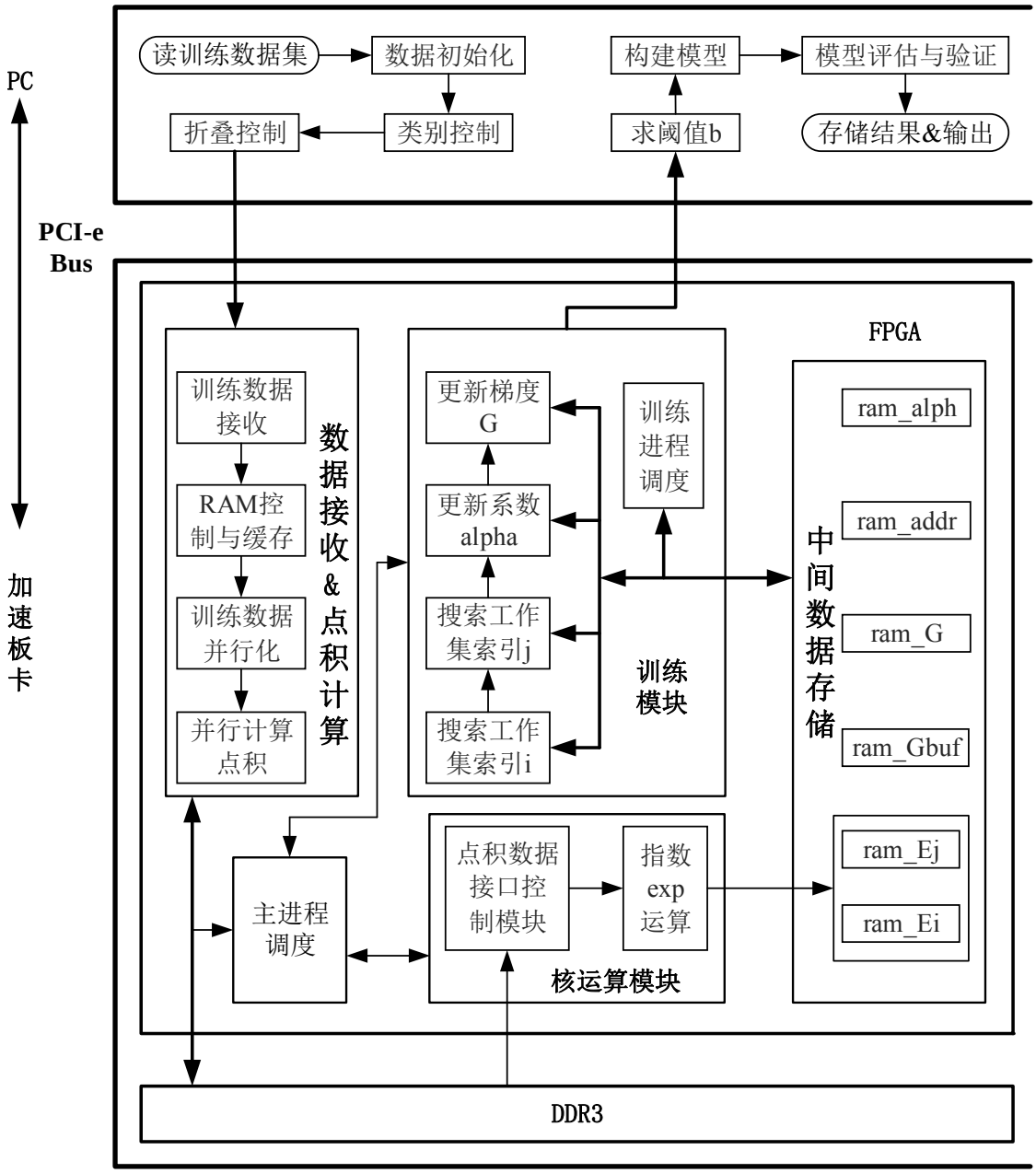

图 3-2 软硬件协同系统主架构

3.3.1.1 系统软件设计

系统的软件设计在结构上分成两部分：分别是训练数据输入 FPGA 前和 FPGA 输出模型参数后。前者主要完成训练数据的读取和预处理，而后者主要完成模型的验证及最终结果的存储。本部分设计详述如下：

A. 在训练数据输入 FPGA 前

PC 机首先从硬盘或移动硬盘读取所有的训练数据集到内存；接着完成所有训练数据集的初始化；然后完成数据的类别控制，以确定此次需搜索 OTPC 的训练

数据集；最后将训练数据集 *s* 折叠，准备好第一份折叠数据输入至 FPGA。PC 机上的数据初始化意味着将训练数据集归一化并按 16Bits 定点化，其定点格式为 1Bit 符号位和 15Bit 小数位；类别控制主要针对包含多类训练数据集的应用，当训练数据集大于两类时，采用 one-to-one 形式从多类数据集中选出两类数据用于训练，如果数据只有两类，则直接将两类数据用于训练；折叠控制是先将类别控制模块输出的训练数据进行 *s* 折叠，再将其中一份折叠后的训练数据输出至 FPGA。

B. 在 FPGA 输出模型参数后

FPGA 执行运算后生成的模型实质是一组拉格朗日系数向量，FPGA 将这组系数向量通过 PCI-e 回传至 PC 机。PC 机首先根据系数向量求解验证模型的阈值 *b*；其次通过系数向量和阈值 *b* 依据式(2-32)构建决策函数；最后结合折叠控制模块输出的测试数据对模型进行验证，以得到该测试集在特定训练参数组合下的准确率并存储。当得到所有训练参数组所对应的准确率后，比较并输出准确率最高的一组参数组合，即 OTPC，该 OTPC 也是此次参与训练的数据集对应的 OTPC。完成两类数据的 OTPC 搜索后，再重新选定两类训练数据，搜索其 OTPC，直到搜索出所有类别组合对应的 OTPC 并输出。

3.3.1.2 系统硬件设计

系统的硬件设计主要指通过FPGA内部的资源，运用VHDL描述逻辑实现SVM搜索 OTPC 过程的部分算法。在 FPGA 内部逻辑模块中，主进程调度模块负责全局调度，控制 FPGA 内部逻辑时序的有序进行；数据接收和点积计算模块完成两个功能，一是接收通过 PCI-e 从 PC 机传来的按指定定点格式编码的训练数据，将数据按序存储至 FPGA 内部的随机存取存储器（Random Access Memory，RAM），二是完成所有训练数据的点积计算并输出至 DDR3 存储为点积矩阵；指数运算模块负责从 DDR3 中读取由训练模块选定的工作集索引所对应的两路点积向量，利用这些点积向量依据式(2-34)完成核函数运算并存储至 FPGA 内部 RAM；交叉验证训练模块则是完成具体的交叉验证训练过程，并通过 PCI-e 向 PC 机输出训练结果。

3.3.2 主进程调度

主进程调度是 FPGA 内部实现全局有序运转的控制核心。主进程调度在实现上实质是一个 Mealy 状态机，主进程调度的状态转移图可用图 3-3 描述，总共包含四种状态（idle，kernel，wait_ddr，train），状态机通过接收特定的输入指令控制其状态的有序转移，同时输出相应的指令实现对 SVM 搜索 OTPC 过程的稳定控制，最终得到训练数据集的 OTPC。

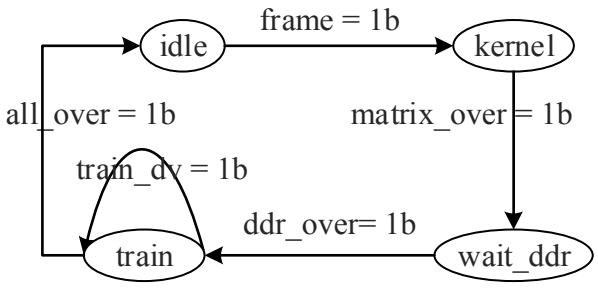

图 3-3 主进程调度状态转移图

3.3.2.1 闲置状态

闲置状态(idle)表示系统暂时没有搜索训练数据集 OTPC 的任务。当系统完成初始化或者接收到交叉验证训练模块输出的交叉验证训练结束的指令(all_over = 1b)后，即处于闲置状态。在此状态下，主进程调度一旦接收到 PCI-e 控制核输出的启动新搜索的命令(frame = 1b)后，便输出相应的激活指令到数据接收和点积计算模块，使之完成新训练数据集的接收及点积计算，并将计算结果存储至 DDR3 形成点积矩阵，同时，主进程调度自身进入核状态(kernel)。

3.3.2.2 核状态

核状态(kernel)是 SDPM 算法应用于 SVM 搜索 OTPC 的过程所必经的状态之一，是 SDPM 算法的实际体现。SDPM 算法需提前将训练数据集的点积计算出来并存储为点积矩阵，后续搜索 OTPC 的过程均共用该点积矩阵。核状态即是主进程调度控制数据接收和点积计算模块完成点积计算的一种状态。在此状态下，当主进程调度接收到数据接收和点积计算模块输出的表示其完成所有点积计算的指令(matrix_over = 1b)后，便输出相应的控制指令，关闭数据接收和点积计算模块。同时，主进程调度进入 DDR3 状态(wait_ddr)。

3.3.2.3 DDR3 状态

由于数据接收和点积计算模块从输入训练数据，完成点积计算到输出至 DDR3 进行存储存在延时。因此，数据接收和点积计算模块完成所有点积计算和 DDR3 完成所有点积数据的存储并不是同步的，这之间的时间差映射到主进程调度的状态即是 DDR3 状态(wait_ddr)。当 DDR3 完成所有点积数据的存储后，会向主进程调度输出存储完成的指令(ddr_over = 1b)，主进程调度接收此指令后，便向训练模块输出开始交叉验证训练的指令，同时，主进程调度进入训练状态(train)。

3.3.2.4 训练状态

训练状态(train)是主进程调度控制交叉验证训练模块有序执行交叉验证训练过程的一种状态。由于有多组训练参数需进行交叉验证，故主进程调度会在训练状态保留很长一段时间。当训练模块完成某组训练参数的交叉验证训练后，会向主进程调度发送指令(train_dv = 1b)，主进程调度接收此指令，输出另一组新的训练参数组到训练模块，启动训练模块开始新的交叉验证训练，但主进程调度本身保持训练状态不变。直到主进程调度接收到训练模块发送的完成所有训练参数组交叉验训练的指令(all_over = 1b)，便输出相应的控制指令，关闭训练模块。同时，主进程调度再次进入闲置状态(idle)。完成搜索 OTPC 的一次折叠训练。

3.3.3 数据接收和点积计算

SDPM 算法表明某两类训练数据集下搜索 OTPC 的所有交叉验证训练过程均从同一个点积矩阵读取点积向量用于自身训练。该点积矩阵便是由数据接收和点积计算模块生成并存储在 DDR3 中。数据接收和点积计算模块包含两个设计：一是完成从 PC 机传来的训练数据的接收并按指定格式存储于 FPGA 内部 RAM 中；二是从 FPGA 的 RAM 中读取训练数据，计算点积并输出至 DDR3 中进行存储。数据接收和点积计算模块整体实现框图如 3-4 所示：

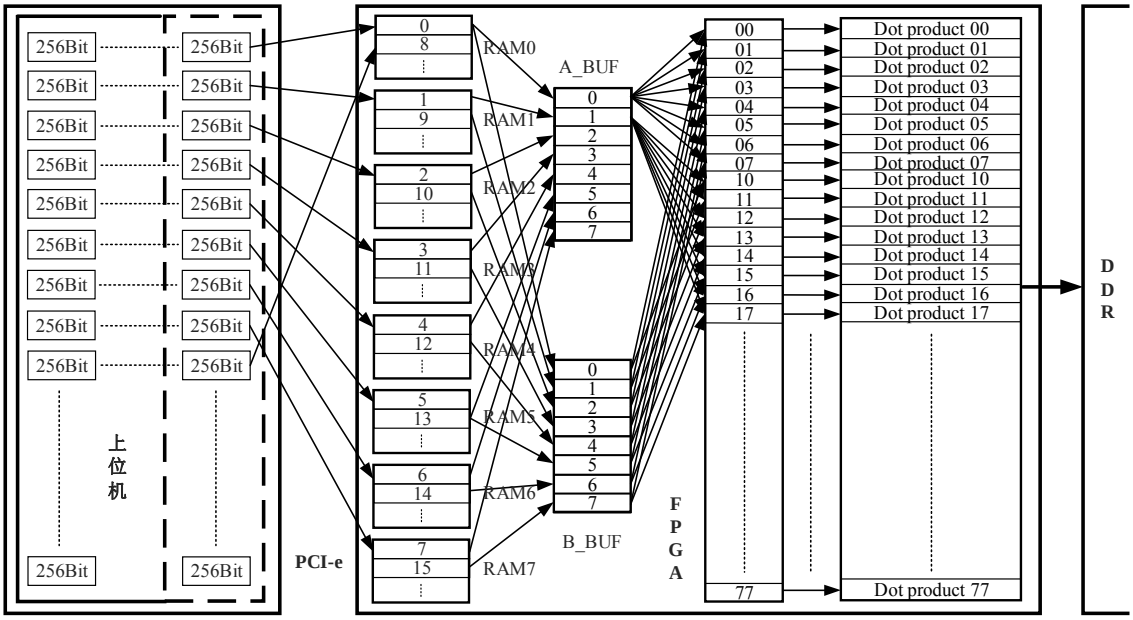

图 3-4 数据接收和点积计算整体实现框图

3.3.3.1 数据接收和存储

数据点积的计算规模是 N^2，硬件计算点积时采用 64 并行的结构。这样一次可计算点积矩阵的 64 个点。为实现这个目的，训练数据在输入至 FPGA 后，按指定顺序存储在 8 个 RAM 中。假定训练数据集可抽象为一个二维矩阵存储于 PC 机，

其存储方式如图 3-4 左边部分所示。设定训练数据集的最大长度为 20480，单个样本点维度为 100，由于采用定点计算，实例点的每个特征均以 16Bits 定点格式表示，100 维度总计 1600Bits，而图 3-4 中每个样本点的实际位宽为 2048Bits（图 3-4 中未标出），增加的 448Bit 是比特值为 0 的向量，用以方便 PCI-e 传输及 FPGA 存储。PCI-e 的用户时钟频率为 250MHz，单时钟传输位宽为 256Bits。在 FPGA 接收端，总共使用 8 个 Block RAM 用于存储定点化后的训练数据集，每个 RAM 的宽度为 1600Bit，深度为 20480/8=2560。

假定数据的起始编号为 0，数据编号分别为 0~20479；假定 RAM 的起始编号为 0，RAM 编号分别为 0~7；假定 RAM 地址的起始编号为 0，RAM 地址编号分别为 0~2559。训练数据集在 RAM 中的存储方式如下所述：

A. FPGA 的数据接收缓存接收到训练数据后，先将位宽为 2048 Bits 的样本点截断为 1600Bits，这样可以去掉无效的填充数据；
B. 第一轮：将第 0 个样本点存入第 0 个 RAM 的第 0 个地址，将第 1 个样本存入第 1 个 RAM 的第 0 个地址中，以此类推，直到将第 7 个样本点存入第 7 个 RAM 的第 0 个地址中；第二轮：将第 8 个样本点存入第 0 个 RAM 的第 1 个地址中，将第 9 个样本点存入第 1 个 RAM 的第 1 个地址中，以此类推。
C. 重复 A、B，直到所有的训练数据分别存储到 8 个 RAM 中。这样，每个 RAM 存储数据的编号间隔都是 8。之所以按这种方式存储，是为点积的 64 并行计算考虑。

3.3.3.2 点积的并行计算

当训练数据按指定格式存储于 FPGA 的 8 个 RAM 后，数据点积计算模块通过特定的读取方法将训练数据读取出来，通过不同组合方式可生成 64 路数据，然后以 64 路并行的方式计算各路数据的点积并输出至 DDR3 进行存储。训练数据点积计算模块的读取方法为：通过给每个 RAM 相同的地址便能一次读到 8 个训练点。在完成一次并行计算时，通过给定两个地址 A1，A2，便能读到两组训练数据 D1，D2，分别存储在缓存 A_BUF 和缓存 B_BUF 中，将 D1 内的每一个训练点都与 D2 内的每一个训练点两两组合，分别进行点积计算，则可实现 $T=C_8^1\times C_8^1=64$ 并行。对于单路数据，点积计算实质是实现如下的表达式：

$$dot=(x-z)^2 \tag{3-3}$$

因此，对于具有 100 维度的样本点来说，兼顾时间和资源两重考虑，计算这样一个点积分成两部分进行：第一部分，先完成 20 个维度的点积计算，具体实现时，先

将样本点对应的 20 个维度分别相减，再将结果平方，最后将平方后的结果想加即可得到 20 个维度的点积；第二部分，依次完成第一部分所述的 20 个维度的点积计算，再将结果存储并累加 5 次，总共可得到 100 个维度的点积。*dot* 的量化位宽为 26Bits。

3.3.4 核函数运算

在以高斯核为核函数 SVM 训练中，交叉验证训练模块需要工作集的核函数向量才能完成交叉验证训练。然而，从 DDR3 中读取到的数据仅仅是点积向量，需将点积向量通过计算变为核函数向量，硬件上实现这一计算过程的便是核函数运算模块。本模块实现框图如 3-5 所示：

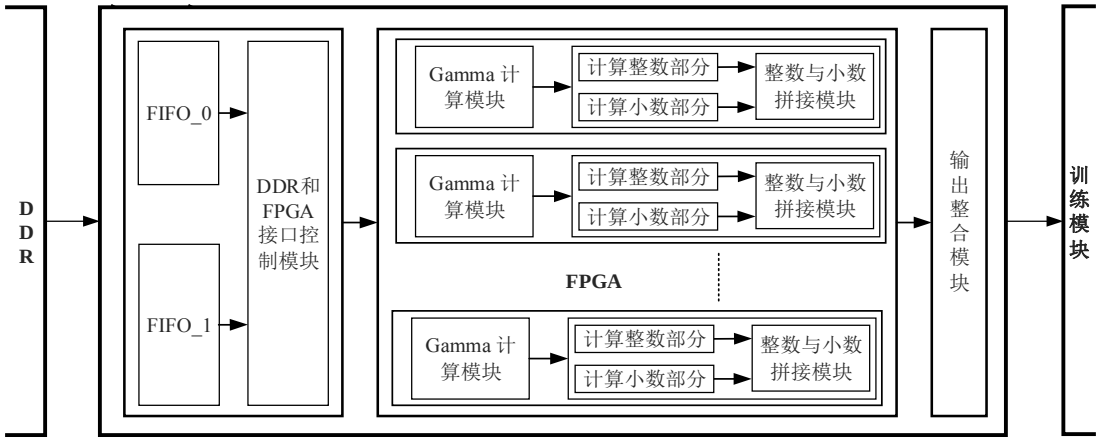

图 3-5 核函数运算实现框图

由于 DDR3 和 FPGA 内部所使用的时钟不同，故 DDR3 传过来的数据需先经过异步 FIFO 缓存。为了方便数据传输，点积向量中添加了冗余比特，故 DDR 和 FPGA 接口控制模块需将冗余比特去除，将真正的点积数据输入后续的模块进行计算。综合硬件速度与资源考虑，指数运算采用 64 并行的电路结构，这样，便可一次计算出 64 个核函数的值。得到 64 个核函数的值后，整合数据输出至交叉验证训练模块完成后续的训练。数学上，单路指数运算模块是计算如下的表达式：

$$y = \exp(-\sigma \times dot) \tag{3-4}$$

3.3.4.1 σ 计算

σ 计算即计算 $res = \sigma \times dot$，在单路的核函数运算电路中，*dot* 表达式是如式(3-3)所示，由数据接收和点积计算模块完成其运算并存储于外置存储芯片 DDR3。*dot* 的量化位宽为 26Bits，其定点格式为 10Q16，即 1Bit 符号位，9Bits 整数位和 16Bits

小数位。而 σ 的取值范围为$[2^{-16},2^{15}]$，其量化位宽为 32Bits，定点格式为 16Q16，即 1Bit 符号位，15Bits 整数位和 16Bits 小数位。在硬件里将两者作乘法，其结果 *res* 的位宽为 58Bits。由于 *res* 始终大于或者等于 0，故 *y* 值小于或者 1，而指数值始终大于 0，故 *y* 的取值区间为[0,1]，可将其量化位宽定为 16Bits，除去 1Bit 符号位，小数位位宽为 15Bits。且有 $e^{-16} < 2^{-16}$，故取 *res* > 16 时，*y* 直接取值为 0。硬件计算时，将 *res* 的 58Bits 截断为 21Bits，定点格式为 5Q16。

3.3.4.2 幂指数分解法

对指数的运算我们采取如下数学表达式(3-5)完成：

$$exp(res) = cosh(res) + sinh(res) \tag{3-5}$$

而我们要计算的是 *exp*(-*res*)且

$$cosh(-res) = cosh(res) \quad sinh(-res) = -sinh(res) \tag{3-6}$$

故

$$exp(-res) = cosh(res) - sinh(res) \tag{3-7}$$

当 $res \geq 16$ 时 *res* 直接截断为（0b& 0xFFFFF）。硬件实现上，使用 cordic 算法的 IP 核分别计算 cosh(res)和 sinh(res)，然后将两者结果相减，即可计算 exp(-res)。然而，cordic 算法 IP 核对于输入输出数据的范围有限定，其输入 *Phase_in* 被限定在[*-pi/4,pi/4*]。而本模块幂指数输入的范围为[0,16)，远远大于 *Phase_in* 的输入范围，为解决这个问题，将指数运算进一步分解为如下计算式(3-8)：

$$\begin{aligned} exp(-res) &= exp(-\mathrm{int}) \bullet exp(-fraction) \\ exp(-fraction) &= exp(-fraction\_1) \bullet exp(-fraction\_1) \end{aligned} \tag{3-8}$$

其中

$$\begin{aligned} res &= int + fraction \\ fraction &= 2 \bullet fraction\_1 \end{aligned} \tag{3-9}$$

将指数运算的幂拆分为整数部分和小数部分分别计算。由于整数部分的取值仅有 16 种情况，可通过硬件里的选择器来完成。将小数部分除以 2 后，由于 0 < *fraction_1* < 0.5，满足 *cordic* 算法 IP 核对输入数据范围的要求，可通过调用 IP 核求解。最后将整数部分与小数部分相乘作为单路核函数计算模块最终的输出。

3.3.5 交叉验证训练

SDPM 算法在一定程度上减少了点积的计算量。即便如此，SVM 还是需要通过大量的交叉验证训练才能搜索到 OTPC。因此，如果能缩短单次交叉验证训练所需的时间，则能在很大程度上提升整体的速度性能。在本文的设计中，交叉验证训练由交叉验证训练模块完成，单次的交叉验证训练包含多次迭代，当迭代结果收敛时，则一次交叉验证训练结束；在单次迭代中，包含一个训练进程主调度器和四个分进程，主调度器通过控制总线命令和分配总线数据，保证一次迭代的有序进行，四个进程分别为搜索工作集 *i* 索引进程，搜索工作集 *j* 索引进程，更新拉格朗日系数 α 进程及更新梯度 *G* 进程。该模块总框图如图 3-6 所示：

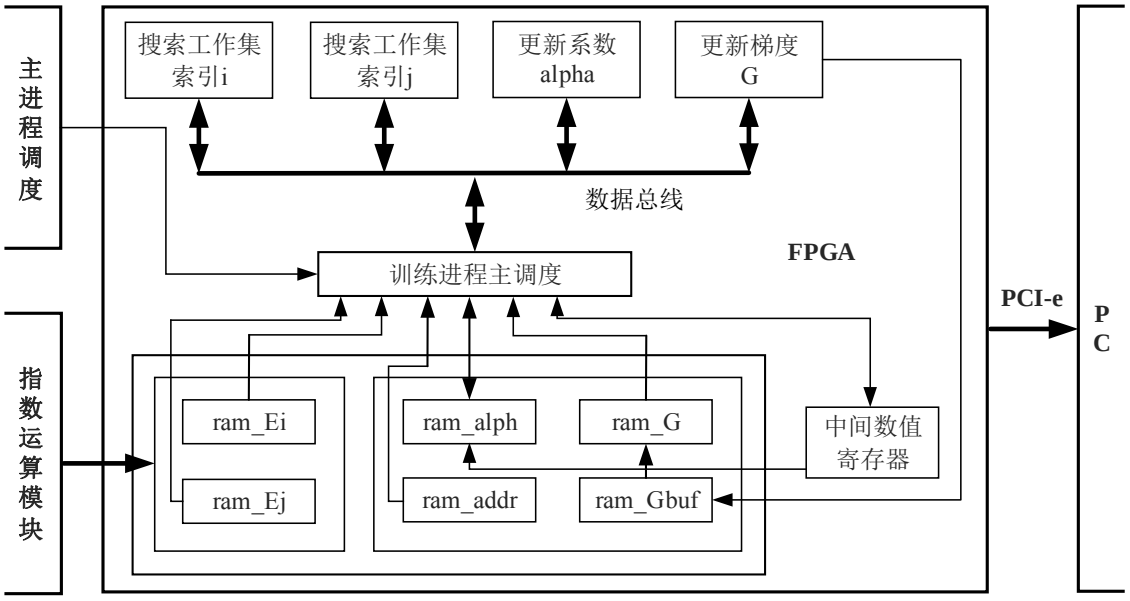

图 3-6 训练模块实现框图

交叉验证训练进程受主进程调度控制，当系统完成新训练数据集的点积计算及存储后，主进程调度会向交叉验证训练模块发送开始交叉验证训练的指令，随着训练指令一起发送的，还有新的训练参数组。此次交叉验证训练便是在该训练参数组下进行。

3.3.5.1 训练进程主调度

训练进程主调度是交叉验证训练模块的控制核心。它控制一次迭代有序进行，同时检查迭代收敛条件，如果不满足收敛条件则重新开始新的迭代，直到满足收敛条件，停止迭代，完成一次交叉验证训练。硬件实现上，训练进程主调度核心是一个状态机，通过发送和接收不同的指令完成状态的转移。同时，该模块还会向交叉

验证训练进程的其它子模块传输中间数据并保存这些模块的输出结果。其状态转移图如图 3-7 所示：

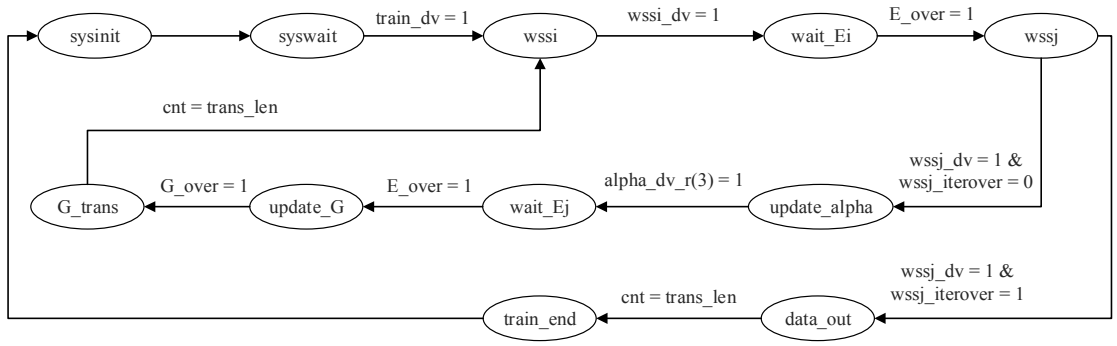

图 3-7 训练进程主调度状态转移图

训练进程主调度状态及其转移规则阐述如下：

A. 系统初始化(sysinit)：当交叉验证训练模块第一次接收到训练指令或者完成一次交叉验证训练后，即进入初始化状态。初始化状态主要完成寄存器的初始化，为新的训练作准备。

B. 系统等待(syswait)：当训练模块完成寄存器的初始化后即进入系统等待状态，系统等待状态是真正开始训练之前的状态。

C. 搜索工作集索引 *i* 状态(wssi)：当训练进程主调度接收到由主进程调度发送的开始训练的指令(train_dv = 1)，即进入搜索工作集索引 *i* 状态。同时输出控制指令到搜索工作集索引 *i* 模块，使之完成工作集索引 *i* 的的搜索。当搜索工作集索引 *i* 模块搜索到 *i*，会发送搜索完成的指令(wssi_dv = 1)。训练进程主调度接收此指令，其状态转移至等待接收 *i* 核函数向量的状态(wait_Ei)。

D. 等待接收 *i* 核函数向量状态(wait_Ei)：当搜索到工作集的 *i* 索引后，核函数运算模块即从 DDR3 读取索引 *i* 对应的点积向量进行计算并存储，该过程映射到训练进程主调度即是等待接收 *i* 核函数向量状态。当训练进程主调度接收到核函数模块发送的核函数向量存储完成的指令(E_over = 1)，其状态转移至搜索工作集索引 *j* 状态(wssj)。

E. 搜索工作集索引 *j* 状态(wssj)：当训练进程主调度进入搜索工作集索引 *j* 状态，便输出控制指令到搜索工作集索引 *j* 模块，使之完成工作集索引 *j* 的的搜索。当搜索工作集索引 *j* 模块搜索到 *j*，会发送搜索完成的指令(wssj_dv = 1)及迭代是否完成指令(wssj_iterover)。如果(wssj_dv = 1& wssj_iterover = 0)（，表明迭代未完成，训练进程主调度进入更新拉格朗日系数状态(update_alpha)；如果

(wssj_dv = 1& wssj_iterover = 1)，表明迭代已完成，训练进程主调度进入数据输出状态(data_out)。

F. 更新拉格朗日系数状态(update_alpha)：训练进程主调度进入该状态后，输出控制指令至更新拉格朗日系数模块，使之完成拉格朗日系数的更新，当更新拉格朗日系数模块完成所有拉格朗日系数的更新后，向主进程调度发送指令(alpha_dv_r(3) = 1)，主进程调度进入等待接收 *j* 核函数向量状态(wait_Ej)。

G. 等待接收 *j* 核函数向量状态(wait_Ej)：当搜索到工作集的 *j* 索引后，核函数运算模块即从 DDR3 读取索引 *j* 对应的点积向量进行计算并存储，该过程映射到训练进程主调度即是等待接收 *i* 核函数向量状态。当训练进程主调度接收到核函数模块发送的核函数向量存储完成的指令(E_over = 1)，其状态转移至更新梯度值状态(update_G)。

H. 更新梯度值状态(update_G)：在单次迭代过程中，当搜索完工作集索引 *i*，*j*，更新好拉格朗日系数 alpha，最后就是更新梯度值。其对应的训练进程主调度状态即是更新梯度值状态，训练进程主调度转移至该状态后，输出控制指令至更新梯度值模块，使之完成梯度值的更新，当更新梯度值模块完成所有梯度值更新后，发送相指令(G_over = 1)到训练进程主调度，训练进程主调度进入转移梯度值状态(G_trans)。

I. 转移梯度值状态(G_trans)：观察式(2-47)可知，梯度值的更新是在梯度旧值的基础上加上一个变化量得到新值。硬件实现上，如果只设置一个 RAM 保存梯度值，则必须先从这个 RAM 取梯度旧值，然后进行更新，再把更新后的新值放回 RAM。显然，这不能构成流水，将极大的影响梯度的更新效率，进而降低 SVM 搜索 OTPC 的速度性能。为实现流水线结构更新梯度值，可设置两个 RAM，一个保存梯度旧值（RAM_A），一个保存梯度新值（RAM_B），这样，从 RAM_A 中读取梯度旧值先进行更新，然后存入 RAM_B，最后将 RAM_B 中的梯度值转移至 RAM_A，实现梯度值的更新与保存。从 RAM_B 中转移梯度值至 RAM_A，其对应的训练进程主调度即是转移梯度值状态，当完成所有的梯度值转移后，训练进程主调度再次进入搜索工作集索引 *i* 状态(wssi)。

J. 数据输出状态(data_out)：迭代收敛后，交叉验证训练模块将模型系数输出至 PC 机，完成后续验证模型的生成和测试。数据输出状态是训练进程主调度将模型参数回传至 PC 机的状态。当回传完成后，训练进程主调度进入训练结束状态(train_end)。

K. 训练结束状态(train_end)：训练结束状态是数据输出状态和系统初始化状态之间的过渡状态。

3.3.5.2 搜索工作集 *i* 索引

搜索工作集 *i* 索引在数学上的形式表达如式(2-43)，I_{up} 满足条件(2-44)。*G* 长度为 *N*。在考虑其硬件设计时，由于搜索工作集 *i* 索引是在满足条件 I_{up} 的子集里选择 $((-1)\times y_t \times G_t)$ 的最大值，且 α 的长度为 *N*。因此，可使用并行结构加流水线实现。为简化硬件设计，两类训练数据（假定数据的起始编号为 0）是交替输入至搜索工作集索引 *i* 模块的，第一类数据（数据编号为偶数）固定其 $y_t=+1$，此时只需考虑 $((-1)\times G_t)$ 的最大值，第二类数据（数据编号为奇数）固定其 $y_t=-1$，此时只需考虑(G_t)的最大值。并行结构中单路基本电路的输入只包含两个样本点的相关信息，具体实现时，基本电路的设计考量为：将两个样本点的梯度值(G_t[0]，G_t[1])作为一组输入，同时输入其索引(*addr*[0]，*addr*[1])，通过选择器和比较器可找到两个工作集索引里满足条件 I_{up} 下的较大值 *o_addr_i*，同时保留梯度的较大值 G_{i_max} 作为最后判断迭代是否收敛的条件变量之一。基本电路可用图 3-8 描述：

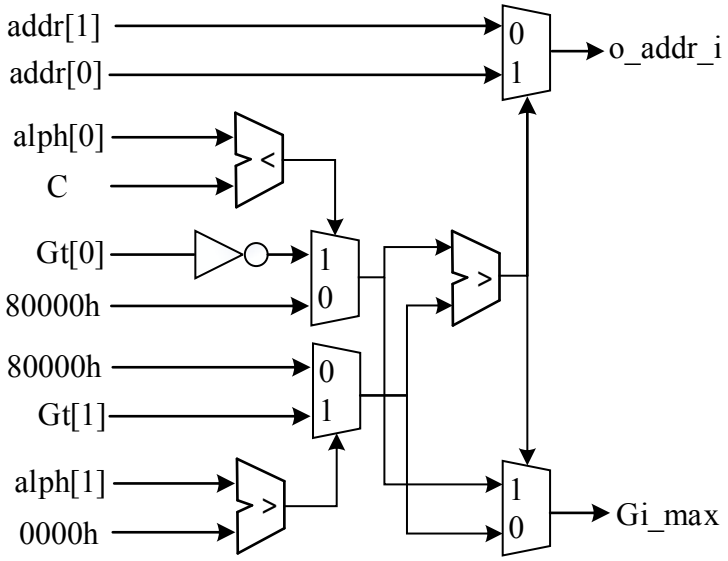

图 3-8 搜索工作集 *i* 索引基本电路

图 3-8 所描述的基本电路结构可一次计算两个工作集索引值里的较大值，将其扩展到 64 并行，则经过第一级计算后，可从 128 个工作集里找到 64 个待定工作集索引。将 64 个工作集索引两两比较作为第二级计算，可得到 32 个待定工作集索引。后级采用类似的比较方法，最终会以菊花链结构找到初始 128 个工作集里的待定工作集索引。再采用流水线结构，可找到最终的工作集 *i* 的索引并输出。

3.3.5.3 搜索工作集 *j* 索引

搜索工作集 *j* 索引在数学上的形式表达如式(2-45)，其中，a_{it} 和 b_{it} 如式(2-42)所示，I_{low} 满足条件(2-46)。K_{ts} 表示训练数据里第 *t* 个训练点和第 *s* 个训练点经核函

数计算后的值，K_{tt}、K_{ss} 分别表示第 t 个训练点和第 s 个训练点与自身经核函数计算后的值，$(-y_i \bullet G_i)$即搜索工作集 i 索引模块的输出 G_{i_max}。硬件设计时，同样使用并行结构加流水线实现，其数据输入格式及 y 值的处理与搜索工作集 i 索引模块相同。搜索工作集 j 索引的目标值计算涉及平方运算与除法运算，硬件实现较为复杂，可将计算过程进行分解，分成多个模块实现。具体实现时，基本电路由两部分构成：第一部分先实现约束条件 I_{low}，由两个比较器和选择器实现，其输出为 *alph_flag0*，*alph_flag1*，后续节拍通过判断它们是否为 1 以判断输入是否满足约束条件 I_{low}；同时，第一部分还实现了满足 I_{low} 下的 b_{it}， 即 $b_i[0]$，$b_i[1]$；还从保存核函数值的 RAM 里读出了 $E_i[0]$，$E_i[1]$。搜索工作集 j 索引模块基本电路的第一部分具体电路可用图 3-9 描述：

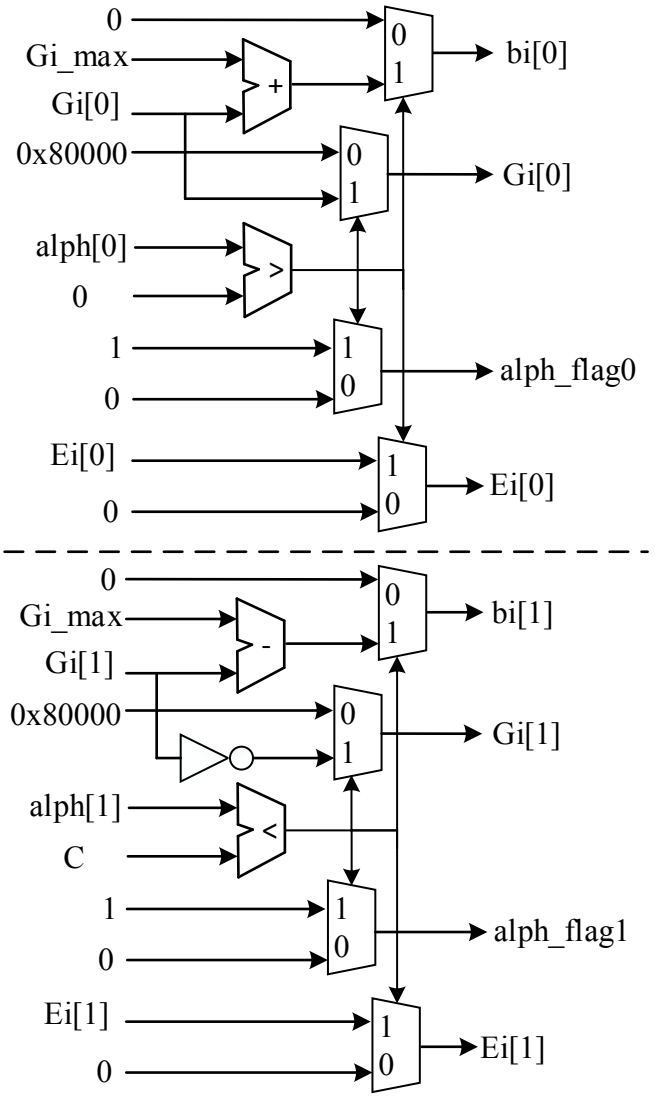

图 3-9 搜索工作集 j 索引基本电路第一部分

实现约束条件 I_{low} 及 b_{it} 后，即可进一步计算本模块的目标值，基本电路的第二部分即是计算该目标值，然后得到 j 索引。显然，目标值的计算是在输入满足约束条件 I_{low} 的情况下进行的，故本部分需要较多的选择器。约束条件式(2-46)具有最高优先级，式(2-45)里的约束条件具有次优先级。由于 K_{tt}、K_{ss} 的实际值为 1，可化简 $a_{ts}=2\times(1-K_{ts})$，本模块只是比较目标值大小，倍数关系可进一步忽略，最终只需计算 $a_{ts}=(1-K_{ts})$ 即可。最后，由于硬件量化为有限位宽，中间过程的计算值需作防溢出处理，第二部分电路结构可用图 3-10 描述：

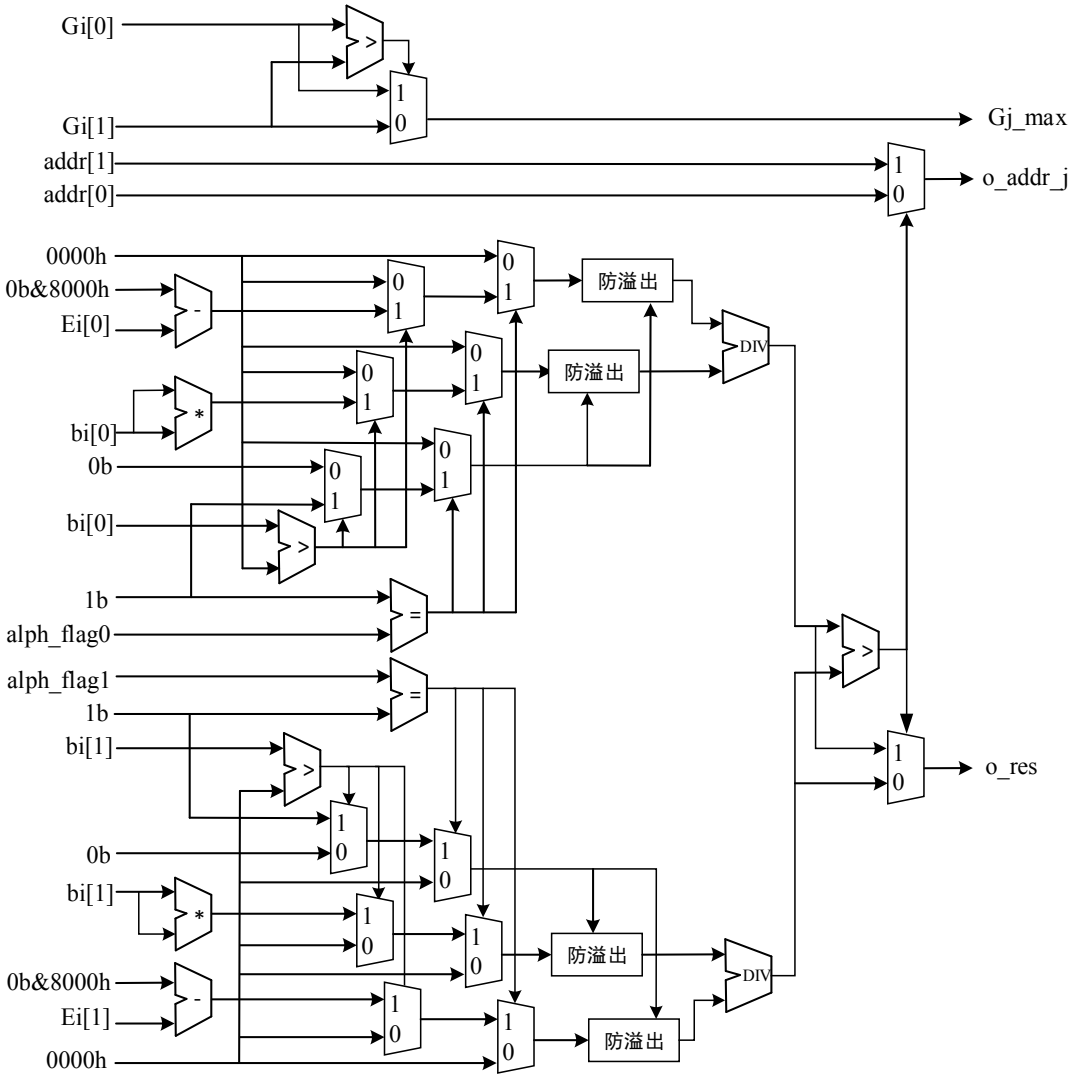

图 3-10 搜索工作集 j 索引基本电路第二部分

图 3-9 和图 3-10 构成搜索工作集 j 索引模块的基本电路，其包含三个输出，两个输入里目标值的较大者 *o_res*，其对应的工作集索引 *o_addr_j*，梯度的较大值 G_{j_max}。同搜索工作集 i 索引模块一样，将此基本电路扩展为 64 并行，然后两两组

合比较它们的 *o_res*，可得到 32 组结果向量(*o_res*，*o_addr_j*，G_{j_max})后续采用类似的比较方法，最终会以菊花链及流水线结构，找到最终的 *j* 索引 *o_addr_j*。

3.3.5.4 更新拉格朗日系数

当搜索到工作集的 *i* 索引及 *j* 索引后，即可更新 *i* 索引和 *j* 索引分别对应的拉格朗日系数。由式(2-30)知，由于拉格朗日系数是在 0 到 *C* 的范围，故本模块的实现分为两部分：第一部分是计算需更新的拉格朗日系数的改变量；第二部分是更新拉格朗日系数并检查更新后的拉格朗日系数值是否符合限定的范围，如果不符合则须做一定修改。在考量本模块的第一部分的硬件设计时，由于改变量 *delta* 和修正量 *sum_diff* 的计算方式因 y_i ，y_j 是否相等而有差异，因此，具体实现时，需同时考虑 $y_i = y_j$ 和 $y_i \neq y_j$ 两种情况下 *delta* 和 *sum_diff* 的硬件实现。当 $y_i = y_j$ 的情况，*delta* 和 *sum_diff* 计算方法如式(3-10)：

$$\begin{aligned} delta &= (\frac{G[i]-G[j]}{quad} \\ sum\_diff &= alpha\_i + alpha\_\text{j} \end{aligned} \tag{3-10}$$

当 $y_i \neq y_j$ 的情况，*delta* 计算方法如(3-11)：

$$\begin{aligned} delta &= (\frac{-G[i]-G[j]}{quad}) \\ sum\_diff &= alpha\_i - alpha\_\text{j} \end{aligned} \tag{3-11}$$

其中：

$$quad = 2\times(1-Ei[j]) \tag{3-12}$$

alpha_i 和 *alpha_i* 分别是工作集索引 *i* 和 *j* 对应的拉格朗日系数。E_i 是指数运算模块输出的第 *i* 个核函数向量，其存储于 FPGA 内部的 RAM 中，长度为 *N*， $E_i[j]$是 E_i 中的第 *j* 个值。当搜索工作集 *j* 索引的过程结束后，如果此次迭代没有收敛，则 *alpha_i*、*alpha_j* 及 $E_i[j]$由训练进程主调度器从中间数据寄存器或内部 RAM 中读取输出至本模块。由于硬件实现时，乘以 2 相当于将结果左移一位，故式(3-12)中的乘以 2 没有具体的硬件实现，只需在后面的结果中标定好定点格式下小数点的位置即可。在更新拉格朗日系数时，修正量 *sum_diff* 是为第二部分检查改变后的拉格朗日系数值是否符合(2-30)作准备。本模块基本电路第一部分的实现框图如图 3-11 所示：

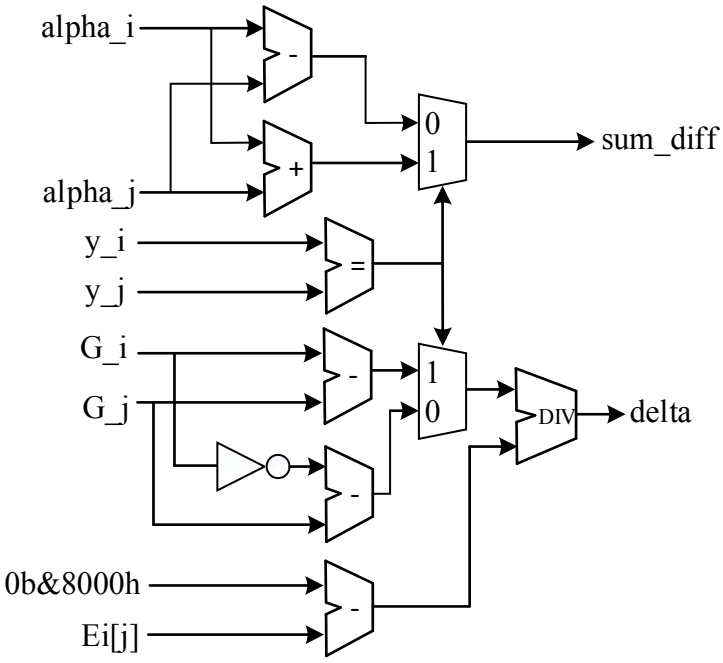

图 3-11 更新拉格朗日系数模块基本电路第一部分

当得到改变量 *delta* 和修正量 *sum_diff* 后，即可更新拉格朗日系数。其更新算法比较简单，可表示如下：

$$
\begin{aligned}
&if \quad y_i = y_j \ \ then \\
&\qquad \alpha\_i^{new} = \alpha\_i^{old} - delta \\
&\qquad \alpha\_j^{new} = \alpha\_j^{old} + delta \\
&else \\
&\qquad \alpha\_i^{new} = \alpha\_i^{old} + delta \\
&\qquad \alpha\_j^{new} = \alpha\_j^{old} + delta
\end{aligned}
$$

更新好拉格朗日系数后，该模块第二部分就是检查更新后的拉格朗日系数是否在限定的范围内。如果在限定范围内，则不做任何修改，如果超出限定范围，须对新系数值作修改，使之符合约束要求。同样，由于硬件采用有限比特的位宽表示数据，故需做防溢出处理。最后，该模块需向主进程调度输出拉格朗日系数真正的改变量，为更新梯度进程作准备。

3.3.5.5 更新梯度

每次迭代时，在更新完拉格朗日系数后，即可利用工作集索引对应的核函数向量及拉格朗日系数的真正改变量 $delta\_\alpha\_i$ 及 $delta\_\alpha\_j$ 更新梯度值。其更新算

法如式(2-47)所示，出于流水线设计考虑，梯度的更新包括两部分：一是从保存梯度旧值的 RAM 中读取梯度旧值完成更新，并保存在新的梯度 RAM 中；二是从保存梯度新值的 RAM 中读取梯度新值转移至保存梯度旧值的 RAM 中，以备下一次的更新。由于梯度 G 的长度为 N，本模块依旧采用并行基本电路加流水线结构实现。其基本电路为两路结构，即基本电路一次可更新两个梯度值。具体设计时，先通过核函数 K_i 和 K_j 的值计算出 Q_i 和 Q_j 的值，再分别与 $delta\_\alpha\_i$ 及 $delta\_\alpha\_j$ 相乘，再将结果相加，最后加上原梯度值即可。其基本电路结构如图 3-12 所示：

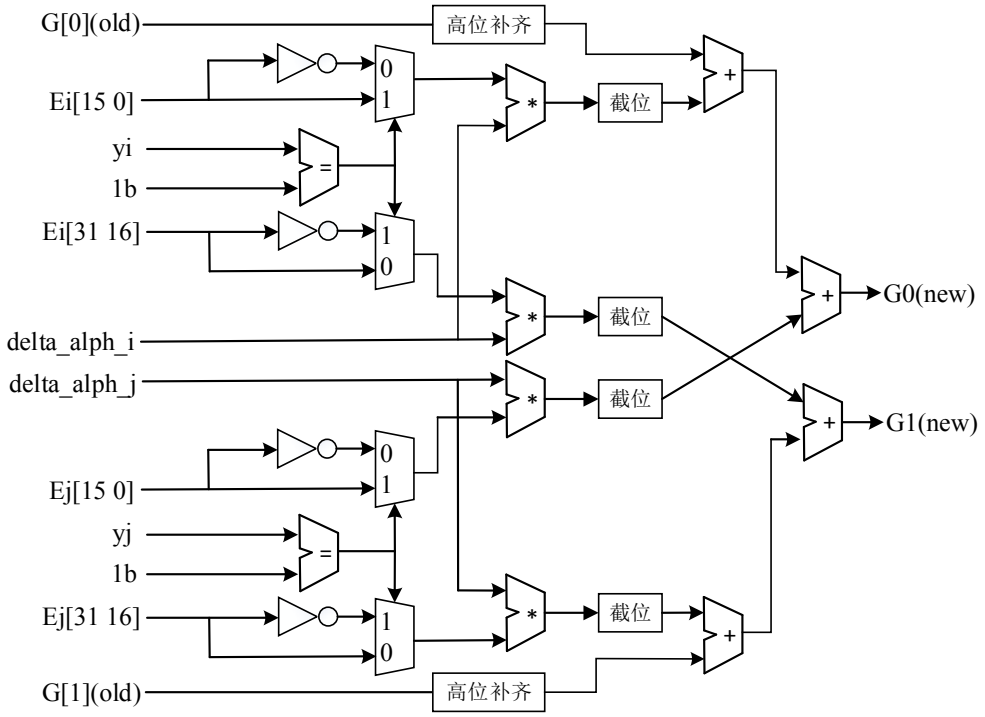

图 3-12 更新梯度模块基本电路

将更新梯度模块基本电路 64 并行，即可一次完成 128 个梯度值的更新，使用流水线结构，最终可实现所有梯度值的更新。当完成所有梯度值的更新后，梯度值的转移比较简单，硬件上使用两个同步 FIFO 即可实现。

3.4 本章小结

本章提出了 SDPM 算法并基于 SDPM 算法分别对 SVM 模型的软件及软硬件协同实现进行了详细阐述。本章首先介绍了传统实现搜索 OTPC 时面临的问题，由此引出 SDPM 算法；然后描述了基于 SDPM 算法的 SVM 模型的软件实现思想及流程，SVM 模型的软件实现是 SVM 模型软硬件协同实现的仿真基础；最后详细描述了基于 SDPM 算法的 SVM 模型的软硬件协同实现，重点阐述了交叉验证

训练的硬件实现。本文设计的 SVM 模型不仅能应用于 SVM 搜索 OTPC，还能用于 SVM 通过训练数据集构建模型，具有很大的应用价值。

第四章 测试结果与评估

本章主要介绍了基于 SDPM 算法的软硬件协同系统的开发及测试平台、系统的测试数据源及测试预处理，最后就资源消耗、参数误差及运行时间三方面分别从单次交叉验证训练和搜索 OTPC 过程两个层面上对测试结果进行了详尽的分析和评估。同时，本章还对基于 SDPM 算法的软件实现也进行了仿真及评估，并将其测试结果与软硬件协同系统的测试结果进行比较。总的来说，本文设计的协同系统虽会耗费一定的软硬件资源，但同时也使得 SVM 搜索 OTPC 的速度性能得到最大限度提升，从而扩展了 SVM 在更多场合的应用。

4.1 开发及测试平台

4.1.1 软件开发及测试平台

本文首先基于 SDPM 算法完成 SVM 搜索 OTPC 在软件上的仿真与测试，测试平台为基于英特尔核 i5 的台式机，CPU 运行频率为 3.3GHz，内存 8GB，操作系统为 Wondows7，单线程执行指令，用户层仿真代码基于应用软件 VS2013 进行开发和测试。

4.1.2 软硬件开发及测试平台

为进一步缩短 SVM 搜索 OTPC 的时间，本文提出并设计了一套软硬件协同架构，其在硬件上包括一台台式机和一块协处理器板卡，台式机 CPU 为英特尔核 i5，CPU 运行运行频率 3.3GHz，内存 8GB，操作系统为 Wondows7，单线程执行指令；协处理器板卡由一片协处理器芯片、一块内存条、PCI-e 接口及其外围电路构成。协处理器芯片是 Xilinx 的 Virtex 7 系列芯片 XC7VX690T、其速度等级-2，内存条是 DDR3 DRAM、存储容量 1GB、时钟频率 400MHz。软件端代码包括 PCI-e 在 Wondows7 系统下的驱动及用户的应用层代码，用户代码基于应用软件 VS2013 进行开发。硬件端代码主要是协处理器的内部逻辑功能代码：Xilinx 芯片内部逻辑代码的编写、仿真、综合及布局布线均基于 Xilinx 公司的自动化开发工具 vivado 完成。FPGA 与 DDR3 的控制逻辑通过调用 vivado 自带的存储器控制 IP 核加上层用户代码实现。FPGA 使用 PCI-e 与主机通信，PCI-e 使用第三代协议，其控制逻辑同样基于 vivado 自带 IP 核加上层用户功能代码实现。

4.2 测试数据集及测试预处理

4.2.1 测试数据集

为评估 SVM 在 LIBSVM、基于 SDPM 算法的软件实现(SDPM-S)及基于 SDPM 算法的软硬件协同实现(SDPM-H&S)三种不同方式下搜索 OTPC 时的速度性能，我们从 UCI 机器学习库[60]选取三组数据集(Pendigits，Iris，SPECTF)及我们在现实应用(调制信号识别)中采集到的两组数据集(TestD1，TestD2)总计五组数据集，用于测试及分析结果。这些数据集的类别、长度及特征向量均不同，具体信息如表 4-1：

表 4-1 测试数据集表

Dataset	Class	Length	Dimension
Pendigits	3	2145	16
Iris	3	150	4
SPECTF	2	80	44
TestD1	2	500	100
TestD2	2	400	100

其中，钢笔字体识别 Pendigits 属于多分类问题，总共包含十类数据，分别代表了阿拉伯数字 0~9，通过训练不同类别组合下的组合模型实现钢笔字体的自动识别，本文实际测试时，只选取了其中的 0~2 三类数据用于评估，测试时采用 *one-to-one* 方式将三类分类问题转化为二类分类问题处理；安德森鸢尾花卉 Iris 是最经典也是使用最广泛数据集之一，总共包含三类共 150 个数据，三类花卉分别是(Iris-setosa，Iris-versicolor，Iris-virginica)，四个特征维度分别表示花萼长度、花萼宽度、花瓣长度和花瓣宽度，由于其也是多类分类问题，处理时采取和数据集 Pendigits 一样的方式；心脏病检测 SPECTF 、应用数据 TestD1 及应用数据 TestD2 均只有两类，其中，SPECTF 分别就健康的心脏和有疾病的心脏统计其多个特征，其训练得到的模型可用于心脏疾病的诊断；TestD1 和 TestD2 是调制信号识别应用中采集到的数据，其训练得到的模型可用于调制信号的检测。Pendigits，Iris 及 SPECTF 在 UCI 机器学习库中的长度和表 4-1 所列的长度有一些差异，这是因为搜索过程中要保证两类训练数据的平衡，测试前需要对测试数据集进行必要的筛选，保证训练时两类数据集的长度是相等的。

4.2.2 测试预处理

A. 数据预处理

本文使用的数据集，比如从 UCI 机器学习库中下载的三组数据，其某些特征向量存在缺失，另外，不同的实现方式对数据的输入格式、表示方式均有要求。因此，对数据集的预处理包括三部分：首先，在正式测试前，对测试数据集进行必要的筛选，剔除掉包含缺失特征项的数据，保证两类数据集的长度是相等的；其次，

由于基于 SDPM 算法的软硬件协同实现在硬件部分使用定点量化数据，在定点格式下，有限位宽的数据所能表示的范围是有限的，为了保证训练数据的一致性，故不同实现方式下所有训练数据均作了归一化处理；最后，LIBSVM、基于 SDPM 算法的软件实现及基于 SDPM 算法的软硬件协同实现均对输入数据的格式有要求，因此，在真正开始搜索 SVM 的 OTPC 前，还需要先将数据修改为每种实现方式下指定的格式。

B. 训练参数选择

我们对不同实现方式进行性能评估的关键指标之一是 SVM 搜索 OTPC 的时长。进行具体测试时，对于要搜索的训练参数组(C，σ)，需先指定 C，σ 的搜索范围，对于本文的测试，C 的取值范围为$[2^{-16},2^{15}]$，σ 的取值范围为$[2^{-16},2^{15}]$。设定训练参数组的指数阶起始值为-16、终止值为 15 及步进值为 1，这样，C 的指数阶取值集合为[-16，-15，-14，-13，-12，-11，-10，-9，-8，-7，-6，-5，-4，-3，-2，-1，0，1，2，3，4，5，6，7，8，9，10，11，12，13，14，15]；σ 的指数阶取值集合为[-16，-15，-14，-13，-12，-11，-10，-9，-8，-7，-6，-5，-4，-3，-2，-1，0，1，2，3，4，5，6，7，8，9，10，11，12，13，14，15]。由于每个参数的指数阶数组的长度为 32，故参数两两组合下共有$32\times32=1024$种参数组合。

C. 折叠次数选择

对训练数据进行折叠是为了更充分的使用数据集。折叠次数太大会使得训练任务加重，折叠次数太小不能体现折叠的效果。综合考虑，交叉验证折叠次数为 5。

4.3 测试结果及比较

本系统设计的主要目标是要最大限度的缩短 SVM 搜索 OTPC 的时长，即在给定的训练数据集下，找到这组训练集用于训练应用模型的最好的训练参数组。为此，本文从算法及软硬件实现上作了各种改进。本节将就资源消耗、参数误差及运行时间三方面分别从单次交叉验证训练和搜索 OTPC 过程两个层面上对测试结果进行了详尽的分析和评估。

4.3.1 资源消耗

4.3.1.1 交叉验证训练资源评估

交叉验证训练是基于改进的 SMO 算法实现的，其内部的训练进程：主调度模块、搜索工作集 i 索引模块、搜索工作集 j 索引模块、更新拉格朗日系数 α 模块、更新梯度 G 模块均是使用逻辑电路实现算法，主要消耗的是 FPGA 内部的寄存器、查找表和乘法器硬核资源；而 BRAM_Ei、BRAM_Ej 分别存储的是 i 索引和 j 索引

对应的核函数向量，BRAM_α 存储的是拉格朗日系数向量，BRAM_addr 存储的是索引值向量，BRAM_G、BRAM_Gbuf 存储的是梯度向量，故这些模块主要消耗的是 FPGA 内部的 Block RAM。交叉验证训练各子模块详细的资源消耗如表 4-2：

表 4-2 交叉验证训练子模块资源消耗

模块	Register	LUT	DSP48	BRAM
训练主调度	80588	20969	0	0
搜索 i 索引	18264	13553	0	0
搜索 j 索引	211100	86908	256	0
更新系数 α	3649	2199	0	0
更新梯度 G	34749	22893	256	0
BRAM_Ei	0	0	0	28
BRAM_Ej	0	0	0	28
BRAM_α	0	0	0	57
BRAM_addr	0	0	0	18
BRAM_G	0	0	0	36
BRAM_Gbuf	0	0	0	36
总计	348350	146522	512	203

交叉验证训练是 SVM 搜索 OTPC 的重要组成部分，同时，也可独立的用于训练数据集在确定的 OTPC 下训练应用模型。从表 4-2 可知，训练进程主调度模块由于要调度大量中间数据在不同模块之间传输，而搜索 *i* 索引模块、搜索 *j* 索引模块和更新梯度 *G* 模块均是高并行电路结构，因此，四个模块均消耗了较多的逻辑资源。更新系数 α 模块没有并行结构，消耗的逻辑资源较少。总的来说，交叉验证训练模块需要耗费了一定的 FPGA 内部资源，然而，资源的消耗换取的训练速度提升效果是非常显著的。

4.3.1.2 搜索 OTPC 资源评估

基于 SDPM 算法的软件实现只是通过 PC 机编写应用层代码实现 SVM 搜索 OTPC 的速度提升，不消耗硬件资源，因此，对资源消耗的统计主要针对的是基于 SDPM 算法的软硬件协同实现。统计资源消耗时，数据接收&点积计算模块被分为数据接收模块和点积计算模块独立统计。基于 SDPM 算法的软硬件协同实现各模块资源消耗如表 4-3，分析表 4-3 可知，PCI-e Core 和 DDR3 Controller 由于需实现较为复杂的 PCI-e 通信协议和 DDR3 控制协议，故需消耗较多的逻辑资源；数据接收模块需完成训练数据的接收和存储，由于要在硬件里存储可能的最大长度的训练数据，故消耗了绝大部分的 BRAM。

表 4-3 软硬件协同实现硬件部分资源使用

模块	Register	LUT	DSP48	BRAM
PCI-e Core	13022	12201	0	57
DDR3 Controller	5814	3952	0	154
主调度进程	145	749	0	0
数据接收模块	84843	54858	0	892
点积计算模块	139152	115360	1280	0
核函数运算模块	116866	95773	384	24
交叉验证训练	348350	146522	512	203
总计	718192	429415	2176	1330

点积计算模块是 64 路并行结构，每一路一次计算 20 个维度的数据点积，需 $64\times20=1280$ 个硬核乘法器；核函数运算模块需实现较为复杂的指数运算，交叉验证训练模块需实现完整的改进的 SMO 算法，且两个模块均包含大规模并行结构，故需消耗较多的逻辑资源和部分硬核乘法器。总的来说，基于 SDPM 算法的软硬件协同实现消耗了 FPGA 较多的资源，这是我们选择 XC7VX690T 的原因。当然，这种对硬件资源的消耗使得搜索 OTPC 的速度性能得到极大提升，从而满足了 SVM 在更多场合的应用。

4.3.2 参数误差

4.3.2.1 交叉验证训练参数误差评估

对交叉验证训练过程参数误差的评估，主要是衡量 LIBSVM 训练得到的模型和 SDPM-H&S 得到的模型的差异。具体来说，就是统计 (α^*,b) 及迭代次数在两种训练方式下的差异，其中，α^* 是拉格朗日系数向量，b 是模型的阈值。交叉验证训练在数据集 Iris（Iris-setosa&Iris-versicolor），TestD1 和 TestD2 下的 α^* 参数误差分别在表 4-4、表 4-5、表 4-6 描述，数据集 Pendigits 和 SPECTF 由于得到的 α^* 数量过多，故不予表格形式统计：

表 4-4 交叉验证训练在 Iris 下的 α*参数误差

LIBSVM α 索引	LIBSVM α 值	SDPM-H&S α 索引	SDPM-H&S α 值	索引误差	α 值误差
15	-4.000000	15	-4.000000	0	0.0
46	4.000000	46	-4.000000	0	0.0
48	2.598229	48	2.599380	0	$4.4*10^{-4}$
82	1.401771	82	1.400620	0	$8.2*10^{-4}$
97	-4.000000	97	-4.000000	0	0.0

表 4-5 交叉验证训练在 TestD1 下的 α*参数误差

LIBSVM α 索引	LIBSVM α 值	SDPM-H&S α 索引	SDPM-H&S α 值	索引误差	α 值误差
353	-0.427900	353	-0.429764	0	$4.3*10^{-3}$
356	3.116950	356	3.118530	0	$5.0*10^{-4}$
451	-2.689050	451	-2.688766	0	$1.0*10^{-4}$

表 4-6 交叉验证训练在 TestD2 下的 α*参数误差

LIBSVM α 索引	LIBSVM α 值	SDPM-H&S α 索引	SDPM-H&S α 值	索引误差	α 值误差
57	-0.376618	57	-0.377767	0	$3.1*10^{-3}$
93	-0.271477	93	-0.272451	0	$3.6*10^{-3}$
148	0.514288	148	0.512930	0	$2.6*10^{-3}$
211	-0.366889	211	-0.364644	0	$6.1*10^{-3}$
384	0.500695	384	0.501932	0	$2.5*10^{-3}$

分析表 4-4，表 4-5，表 4-6 可知，不同数据集下，LIBSVM 和 SDPM-H&S 训练得到的模型中的α^*参数索引值完全一样，而α^*参数值的最大误差不超过 10^{-3}，其误差计算方式是：$|\alpha_{LIBSVM}-\alpha_{SDPM-H\&S}|/\alpha_{LIBSVM}$ ，显然，由于 SDPM-H&S 采用定点格式表示中间数据，且中间有防溢出处理，故其执行结果与采用全浮点运算的 LIBSVM 存在一定的误差，然而这个误差是完全可以接收的，不会影响最终的分类结果。对交叉验证训练在不同实现方式下阈值和迭代次数的统计如表 4-7 所示：

表 4-7 交叉验证训练的阈值和迭代次数统计

Datasheet	LIBSVM b 值	LIBSVM 迭代次数	SDPM-H&S b 值	SDPM-H&S 迭代次数	b 值误差	迭代次数差异
Pendigits	0.075811	568	0.075906	323	1.3*10-3	-245
Iris	-0.132300	92	-0.132282	8	1.4*10-4	-84
SPECTF	1.049607	324	1.044293	191	5.1*10-3	-133
TestD1	0.003356	37	0.003203	6	4.5*10-2	-30
TestD1	0.002298	11	0.002337	14	1.7*10-2	+3

分析表 4-7 可知，对于阈值 *b*，误差计算公式为：$|b_{LIBSVM}-b_{SDPM-H\&S}|/b_{LIBSVM}$，其最大误差不超过 10^{-2}，在定点格式下，SDPM-H&S 相比于浮点运算的 LIBSVM 存在这样误差范围，效果是非常显著的，完全不会影响最终的分类。同时，从表 4-7 还可看出，在可接受的误差范围内，SDPM-H&S 大大减少了迭代次数，迭代次数的减少意味着训练时间的缩短，由此，SDPM-H&S 对于缩短训练时间是很有优势的。对于训练时间的统计和评估在下一小节。

4.3.2.2 搜索 OTPC 参数误差评估

对 SVM 搜索 OTPC 的参数误差的评估将基于三种方式进行：LIBSVM、基于 SDPM 算法的软件实现（SDPM-S）、基于 SDPM 算法的软硬件协同实现（SDPM-H&S）；评估的参数是不同搜索方式下的 OTPC（C，σ）及该 OTPC 对应的交叉验证准确率；评估方式是统计这些参数在不同搜索方式下的差异。

对不同方式搜索 OTPC 的 C 参数误差的评估如表 4-8，第一列是数据集名称，第二列到第四列是不同方式下搜索出的 C 值，最后两列是 SDPM-S 和 SDPM-H&S 两种方式相比于 LIBSVM 的参数搜索误差，由于我们搜索参数的方式是按指数阶步进的，故最后两列的数值代表了指数阶的绝对值差异：

表 4-8 搜索 OTPC 的 C 参数的误差评估

Dataset	LIBSVM C value	SDPM-S C value	SDPM-H&S C value	SDPM-S Accuracy lost	SDPM-H&S Accuracy lost
Pendigits（1&2）	2^{-5}	2^{-5}	2^{-5}	0	0
Pendigits（1&3）	2^{-16}	2^{-16}	2^{-15}	0	1
Pendigits（2&3）	2^{2}	2^{2}	2^{2}	0	0
Iris（1&2）	2^{-16}	2^{-16}	2^{-15}	0	1
Iris（1&3）	2^{-16}	2^{-16}	2^{-15}	0	1
Iris（2&3）	2^{-1}	2^{-1}	2^{-2}	0	1
SPECTF	2^{9}	2^{9}	2^{8}	0	1
TestD1	2^{-7}	2^{-7}	2^{-7}	0	0
TestD2	2^{-7}	2^{-7}	2^{-7}	0	0

对不同方式搜索 OTPC 的 σ 参数误差的评估如表 4-9，表 4-9 数据格式同表 4-8：

表 4-9 搜索 OTPC 的 σ 参数的误差评估

Dataset	LIBSVM σ value	SDPM-S σ value	SDPM-H&S σ value	SDPM-S Accuracy lost	SDPM-H&S Accuracy lost
Pendigits（1&2）	2^{-2}	2^{-2}	2^{-1}	0	1
Pendigits（1&3）	2^{0}	2^{0}	2^{1}	0	1
Pendigits（2&3）	2^{2}	2^{2}	2^{2}	0	0
Iris（1&2）	2^{-16}	2^{-16}	2^{-15}	0	1
Iris（1&3）	2^{-16}	2^{-16}	2^{-15}	0	1
Iris（2&3）	2^{-6}	2^{-6}	2^{-6}	0	0
SPECTF	2^{2}	2^{2}	2^{2}	0	0
TestD1	2^{0}	2^{0}	2^{0}	0	0
TestD2	2^{2}	2^{2}	2^{2}	0	0

对不同方式搜索 OTPC 的 OTPC 对应的的交叉验证训练准确率如表 4-10，第一列是数据集名称，第二列到第四列不同方式下 OTPC 对应的交叉验证准确率，

最后两列是准确率的误差，SDPM-S 的误差计算方式是$|A_{LIBSVM}-A_{SDPM-S}|/A_{LIBSVM}$，SDPM-H&S 的误差计算方式是$|A_{LIBSVM}-A_{SDPM-H\&S}|/A_{LIBSVM}$，$A$ 表示准确率。

表 4-10 不同方式的 OTPC 对应的交叉验证训练准确率

Dataset	LIBSVM 准确率	SDPM-S 准确率	SDPM-H&S 准确率	SDPM-S Accuracy lost	SDPM-H&S Accuracy lost
Pendigits（1&2）	99.93%	99.93%	99.93%	0	0
Pendigits（1&3）	100%	100%	100%	0	0
Pendigits（2&3）	99.79%	99.79%	99.79%	0	0
Iris（1&2）	100%	100%	100%	0	0
Iris（1&3）	100%	100%	100%	0	0
Iris（2&3）	98.00%	98.00%	98.00%	0	0
SPECTF	83.75%	83.75%	83.75%	0	0
TestD1	100%	100%	100%	0	0
TestD2	100%	100%	100%	0	0

LIBSVM 使用的双精度浮点表示数据，基于 SDPM 算法的软件实现只是在搜索算法上做改进，并不改变数据的精度，故两者搜索 OTPC 的结果几乎完全一样。而基于 SDPM 算法的软硬件协同实现的数据以定点格式表示，其表示精度相比于浮点表示，存在一定的精度损失，另外，数据是以限位宽的比特表示，其表示的范围是有限的，在中间过程的计算中，为防止数据溢出或者防止计算结果过长，会对数据进行截位。这使得基于 SDPM 算法的软硬件协同实现的搜索结果相比于 LIBSVM 存在一定的误差，但是，由表 4-8、表 4-9 可看出，这种误差完全在可接受的范围内，使用硬件搜索出 OTPC 后，为找到浮点格式下的 OTPC，只需在硬件的 OTPC 的指数阶一个绝对值内做更细致的搜索即可，无需做更大范围的搜索，由表 4-10 可知，不同方式下 OTPC 对应的准确率是完全一样的，这是由于训练参数组对应的最高准确率可能不止一组，这种情况下确定 OTPC 时，取尽量小的 C 值和 σ 值。大量的测试结果表明，基于 SDPM 算法的软硬件协同实现的搜索误差是完全可以接受的，不会影响最终构建模型的分类结果。

4.3.3 运行时间

4.3.3.1 交叉验证训练运行时间评估

对交叉验证训练过程运行时间的评估，主要是对比 LIBSVM 训练模型和 SDPM-H&S 两种方式下生成应用模型所需的时长。表 4-11 列出了不同测试数据集下 LIBSVM 和 SDPM-H&S 的执行时长，同时列出了 SDPM-H&S 相比于 LIBSVM 的加速增益。由于测试数据集 Pendigits 和 Iris 包含三类数据，而 SVM 训练具体的

应用模型只能使用两类数据，故在评估运行时间时，均只从 Pendigits 和 Iris 数据集中选取第 1 类和第 2 类数据用于测试和评估。由于 Iris 在 LIBSVM 下的训练时长过短（不足 1ms），上位机无法统计，故表格第三行的第二列和第四列没有数据。

表 4-11 交叉验证训练的运行时间对比

Dataset	LIBSVM Times(ms)	SDPM-H&S Times(ms)	SDPM-H&S Speed-up
Pendigits（1&2）	304	7.4	41.1
Iris（1&2）	-	0.1	-
SPECTF	46	2.2	20.9
TestD1	124	3.2	38.8
TestD2	78	2.4	32.5

分析表 4-11 可知，对于单次的交叉验证训练，即对于确定的训练数据集和训练参数组，SDPM-H&S 相比于 LIBSVM 在生成 SVM 的应用模型时，有至少 20 倍的速度提升，30 倍的平均速度提升。另外，由于数据集长度：$N_{\text{SPECTF}} < N_{\text{TestD2}} < N_{\text{TestD1}} < N_{\text{Pendigits}}$，而训练速度提升倍数：$M_{\text{SPECTF}} < M_{\text{TestD2}} < M_{\text{TestD1}} < M_{\text{Pendigits}}$，因此，交叉验证训练速度倍数的提升是随着数据集长度渐进增长的。硬件实现时，SDPM-H&S 较之于 LIBSVM 速度的提升表现在两个层面：在单次迭代层面，由于硬件采用的并行结构加流水线，该设计方式使得硬件可在单时钟周期内完成大数据量的更新。例如更新梯度模块，由于采用 64 并行结构加流水，而且单路电路结构一次更新两个梯度值，因此，当梯度长度 N 大于 128 时，在单时钟周期内即可完成 128 个梯度值的更新，这对于串行执行的软件线程，优势是极大的。另外，LIBSVM 在数据集长度很大的时候，频繁的对内存进行读取和存储一定程度上限制了其执行的速度，而 SDPM-H&S 里存在专用的 RAM 存取中间数据，从而进一步加速了训练过程；在不同迭代层面，由表 4-7 可知，SDPM-H&S 减少了迭代次数，从而有效的缩短了交叉验证训练时间。

4.3.3.2 搜索 OTPC 运行时间评估

本文设计基于 SDPM 算法的软硬件协同实现的目的是最大限度的提升 SVM 搜索 OTPC 的速度性能。因此搜索 OTPC 的时长是我们关注的关键指标。表 4-12 列出了不同训练数据集在 LIBSVM、基于 SDPM 算法的软件实现（SDPM-S）、基于 SDPM 算法的软硬件协同实现（SDPM-H&S）下搜索 OTPC 的时长及 SDPM-S 和 SDPM-H&S 相比于 LIBSVM 在训练速度上的增益，其中，数据集 Pendigits 和 Iris 均包含三类数据，以 one-to-one 方式搜索时，每个数据集的类别组合方式是六种，因此，数据集 Pendigits 和 Iris 的运行时间是六种类别组合下搜索 OTPC 的总

时间，而数据集 SPECTF、TestD1 和 TestD2 均只包含两种数据，故其运行时间就是该类别组合下的搜索 OTPC 的时间。

表 4-12 不同算法下搜索 OTPC 的运行时长对比

Dataset	LIBSVM Times(ms)	SDPM-S Times(ms)	SDPM-H&S Times(ms)	SDPM-S Speed-up	SDPM-H&S Speed-up
Pendigits	7521492	3130930	45806	2.4	164.2
Iris	42374	20934	3214	2.0	13.2
SPECTF	9188	7599	2375	1.2	3.8
TestD1	281526	138803	8721	2.0	32.2
TestD2	170820	85478	6973	2.0	24.4

从表 4-12 可看出，基于 SDPM 算法的软件实现在搜索 OTPC 时，其搜索速度相比于 LIBSVM 平均提升了 2 倍，这是由于 SDPM 算法减少了搜索过程中总的点积的计算量而实现的，这个结果与第三章节第二部分关于 SDPM 算法性能评估的理论推导结果吻合。同时，基于 SDPM 算法的软硬件协同实现在搜索 OTPC 时，其搜索速度相比于 LIBSVM 可提升 30 倍。搜索 OTPC 过程是由大量的交叉验证训练过程构成，而交叉验证训练又是由大量的迭代构成，由表 4-11 可知，在训练数据长度达到一定值时，SDPM-H&S 能显著缩短单次迭代的时间，同时，减少单次训练的迭代次数，且由表 4-12 可知，SDPM-H&S 在算法上减少了点积的计算量，在算法上就有一定的速度提升，这三者综合的结果，使得 SDPM-H&S 相比于 LIBSVM 实现了搜索 OTPC 的最大速度提升，且训练数据长度越长，这种速度提升的效果越显著。

4.4 本章小结

本章首先介绍了基于 SDPM 算法的软件实现及基于 SDPM 算法的软硬件协同系统的开发和测试平台；然后介绍了测试的数据集及测试前的预处理；最后从单次交叉验证训练和搜索 OTPC 过程两个角度，分别就资源消耗、参数误差及运行时间三个方面进行了详尽了对比和分析。结果表明，在可允许的精度损失内，硬件实现交叉验证训练过程比软件实现交叉验证训练过程有约 40 倍速度提升；在搜索 SVM 的 OTPC 时，基于 SDPM 算法的软件实现比 LIBSVM 有约 2 倍的速度提升；基于 SDPM 算法的软硬件协同系统实现比 LIBSVM 有约 30 倍的速度提升。这种训练或者搜索速度的提升会耗费一定的硬件资源，但却扩展了 SVM 在更多场合的应用，本文设计的软硬件协同系统具有很大的实际意义。

第五章 全文总结

5.1 论文总结

本文提出了 SDPM 算法，能显著减少搜索 OTPC 过程点积的计算量；完成了基于 SDPM 算法的搜索 OTPC 的软件实现，其搜索速度相比于 LIBSVM 平均提升了 2 倍；提出了基于 SDPM 算法的搜索 OTPC 的软硬件协同架构并完成其实现，其搜索速度相比于 LIBSVM 可提升 30 倍；本文还提出了幂指数分解法，并基于此方法在 FPGA 上实现了限定范围的 e 指数求解。

本文第二章首先对统计学习及 SVM 相关的理论、算法进行了深入分析；然后介绍了以训练数据集是否线性可分为划分依据的不同类型的支持向量分类机；最后介绍了 SVM 模型，SVM 模型是 SVM 在实际应用时解决实际问题的思路和方法，重点阐述了 SVM 改进 SMO 实现算法。本章内容是全文的理论基础。

为了提升 SVM 搜索 OTPC 的速度性能，在第三章本文首先阐述了传统实现搜索 OTPC 时面临的问题；然后介绍了 SDPM 算法及基于 SDPM 算法的软件实现，主要阐述其执行流程和思想，基于 SDPM 算法的软件实现是基于 SDPM 算法的软硬件协同实现的基础；最后详细介绍了基于 SDPM 算法的软硬件协同实现，包括系统框架及各模块硬件实现的电路级结构，重点阐述了交叉验证训练模块。

为了对比 LIBSVM、基于 SDPM 算法的软件实现和基于 SDPM 算法的软硬件协同实现三种不同方式在搜索 OTPC 的速度性能，本文第四章进行了大量的测试和评估：分别从搜索 OTPC 和单次交叉验证训练的两个层面，在资源消耗、参数误差和运行时间三个维度上进行了详尽的测试和分析。大量的测试表明：本文设计的方法能极大的提升 SVM 搜索 OTPC 的速度性能，满足 SVM 在更多场合的应用。

5.2 下一步的研究工作

本文对 SVM 搜索 OTPC 时，基于 SDPM 算法的软件实现和基于 SDPM 算法的软硬件协同实现，仍有不足之处，在未来的工作中可继续在以下方面展开进一步研究：

1) SDPM 算法实质是以空间换时间，在搜索 OTPC 时，对训练数据集的长度有一定的限制要求，可进一步改进 SDPM 算法，使得 SDPM 算法可应用于任意长度的 SVM 模型；

2) 本文提出的基于 SDPM 算法的软硬件协同实现需占用较多的硬件逻辑资源，可考虑在可接受的搜索时长和搜索误差下进一步优化电路的并行结构及电路的具体结构，尽量减少硬件资源的消耗。
3) 本文提出的基于 SDPM 算法的软硬件协同实现在搜索 OTPC 时，其搜索结果在精度上仍有不足，可设计更高精度和更快收敛的加速器，使得 SVM 搜索 OTPC 能更快的收敛；
4) 在 FPGA 上准确完整的实现功能验证后，可考虑将此设计扩展为 ASIC，以应用于特定的场合；

致 谢

时间就像流水，它是如此欢快而又悄无声息。回首三年前，自己对于未来，一半是惶恐、一半是期待。如今，两年多的时光悄然流逝，一路走来，一直感恩能得到师长的关怀和指导、同学朋友的帮忙和照顾，让即将走出校门的自己，也能比三年前多了一份自信和从容。对此，我的内心一直充满了感恩。

首先要感谢我的导师马上老师，是马老师给了我读研的机会，使我能在科研的道路上继续前行。三年来，马老师严谨的科研态度时刻提醒我要认真细致，沉下心来做好每一件事；马老师丰富的项目经验激励着我要踏实勤奋，不断学习新的知识。马老师在科研和项目上的亲切指导，使我掌握了一定的科研能力，具备了较多的项目技能，再次感谢马老师。

同时要感谢胡剑浩老师、陈杰男老师和吴廷勇老师。胡老师深厚的学识和谦逊的风格深刻的感染着我，使我明白增强能力的同时要保持谦虚；陈老师对科研的热情和项目的执着使我感到钦佩，同时也在项目上给予我指导；吴老师诙谐幽默而又平易近人给我留下了深刻的印象，使我明白与人为善的道理。

还要感谢我的师兄师姐和一起同窗三年的教研室伙伴，特别是我的师兄刘剑峰、黄雕鹏和我的同门杨泽国，是你们在项目和生活上给予我的帮助，使我能更快的成长并感受着生活的美好和欢乐。

感谢父母对我的培养和关心，是你们一直的支持和理解，使我在读完大学后能继续读研并顺利完成学业。

最后，感谢评审和答辩老师在百忙之中抽出时间评审我的论文并且参加我的答辩，真的非常感谢！

参考文献

[1] J. Martinez, K. A. Rodriguez and J. V. Martinez. NOMOHiOS: Fundamental Aspects of Artificial Intelligence and Software Engineering in the Design of Management Systems[C]. 2012 Third World Congress on Software Engineering, Wuhan, 2012, 115-118

[2] A. Nayak and K. Dutta. Impacts of machine learning and artificial intelligence on mankind[C]. 2017 International Conference on Intelligent Computing and Control (I2C2), Coimbatore, India, 2017, 1-3

[3] A. Revathi and C. Jeyalakshmi. Robust speech recognition in noisy environment using perceptual features and adaptive filters[C]. 2017 2nd International Conference on Communication and Electronics Systems (ICCES), coimbatore, India, 2017, 692-696

[4] P. Chriskos, J. Munro, V. Mygdalis, et al. Face detection hindering[C]. 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, QC, Canada, 2017, 403-407

[5] N. Juneja and M. R. Kumar. Generating analytic insights on human behavior using image processing[C]. 2017 International Conference on Intelligent Computing and Control (I2C2), Coimbatore, India, 2017, 1-5

[6] V. N. Vapnik. An overview of statistical learning theory [J]. In IEEE Transactions on Neural Networks, vol. 10, no. 5, 988-999, Sep 1999

[7] V. N. Vapnik. Statistical Learning Theory [M]. New York: Wiley, 1998

[8] V. Koltchinskii, C. T. Abdallah, and M. Ariola. Statistical learning control of delay systems: theory and algorithms[C]. Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304), Phoenix, AZ, 1999, 4696-4699 vol.5

[9] C. You and D. Hong. Neural convolutional decoders in the satellite channel[C]. Neural Networks, 1995. Proceedings, IEEE International Conference on, Perth, WA, 1995, 443-448 vol.1

[10] S. Basu, J. Chakraborty, and M. Aftabuddin. Emotion recognition from speech using convolutional neural network with recurrent neural network architecture[C]. 2017 2nd International Conference on Communication and Electronics Systems (ICCES), coimbatore, India, 2017, 333-336.

[11] 李彦冬,郝宗波,雷航.卷积神经网络研究综述[J].计算机应用,2016,36(09):2508-2515+2565

[12] V. Cherkassky, The Nature Of Statistical Learning Theory[C]. In IEEE Transactions on Neural Networks, vol. 8, no. 6, 1564-1564, Nov 1997

[13] Chen Junli and Jiao Licheng. Classification mechanism of support vector machines[C]. WCC 2000 - ICSP 2000. 2000 5th International Conference on Signal Processing Proceedings. 16th World Computer Congress 2000, Beijing, 2000, 1556-1559 vol.3

[14] Abhaya and K. Kumar. An efficient network intrusion detection system based on fuzzy C-means and support vector machine[C]. 2016 International Conference on Computer, Electrical & Communication Engineering (ICCECE), Kolkata, 2016, 1-6

[15] G. Gautam, K. Choudhary, S. Chatterjee, et al. Facial expression recognition using krawtchouk moments and support vector machine classifier[C]. 2017 Fourth International Conference on Image Information Processing (ICIIP), Shimla, India, 2017, 1-6

[16] A. Soni, J. Haupt and F. Porikli. Recycled linear classifiers for multiclass classification[C]. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, 2014, 2957-2961

[17] B. B. Damodaran and R. R. Nidamanuri. Dynamic Linear Classifier System for Hyperspectral Image Classification for Land Cover Mapping [J]. In IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 6, 2080-2093, June 2014

[18] E. Osuna, R. Freund and F. Girosi. An improved training algorithm for support vector machines [J]. Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop, Amelia Island, FL, 1997, pp. 276-285.

[19] 冯登国,蒋建春.网络环境下的信息对抗理论与技术[J].世界科技研究与发展,2000(02):27-30

[20] 王晓海.空间信息对抗技术的现状及发展[J].中国航天,2007(09):17-21

[21] 曾辉,戴强.空间信息对抗综述[J].舰船电子对抗,2011,34(05):30-34+59

[22] R. Armitano, R. Hasson. Digital Demodulation Verifies Accuracy of Advanced Microwave Signal Simulators: Motivation and Theory [J]. Microwave Journal, 183-194, June 1990

[23] K C Ho, W Prokopiw, and Y T Chan. Modulation identification of digital signals by the wavelet transform [J]. IEEE Proc Radar Sonar Naving, 2000, 147(4), 169-176

[24] R. J. Backscheider, M. A. Temple, D. E. Wruck, et al. Characterization of radar detection performance in the presence of modern signal interference[C]. 2004 International Waveform Diversity & Design Conference, Edinburgh, Scotland, 2004, 1-5

[25] Xueyao Li, Bing Han, Dongmei Fan, et al. A digital modulation signals recognition method under the lower SNR[C]. 2008 7th World Congress on Intelligent Control and Automation, Chongqing, 2008, 3958-3962

[26] Pan-Feng Sun, Zi-Wei Zheng and Man Li. Recognition of digital modulation signals based on statistical parameters[C]. Proceedings 2011 International Conference on Transportation,

Mechanical, and Electrical Engineering (TMEE), Changchun, 2011, 2467-2470

[27] Y. Zhao, Y. t. Xu, H. Jiang. Recognition of digital modulation signals based on high-order cumulants[C]. 2015 International Conference on Wireless Communications & Signal Processing (WCSP), Nanjing, 2015, 1-5

[28] H. Zhang and R. Zhou. The analysis and optimization of decision tree based on ID3 algorithm[C]. 2017 9th International Conference on Modelling, Identification and Control (ICMIC), Kunming, China, 2017, 924-928.

[29] L. X. Wang and Y. J. Ren. Recognition of digital modulation signals based on high order cumulants and support vector machines[C]. 2009 ISECS International Colloquium on Computing, Communication, Control, and Management, Sanya, 2009, 271-274

[30] S. p. Li, F. c. Chen and L. Wang. Modulation recognition algorithm of digital signal based on support vector machine[C]. 2012 24th Chinese Control and Decision Conference (CCDC), Taiyuan, 2012, 3326-3330

[31] J. Li, G. Zhang, Y. Sun, et al. Automatic intra-pulse modulation recognition using support vector machines and genetic algorithm[C]. 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, 2017, 309-312

[32] 范永东. 模型选择中的交叉验证方法综述[D].山西大学,2013

[33] 王健峰,张磊,陈国兴,何学文.基于改进的网格搜索法的 SVM 参数优化[J].应用科技,2012,39(03):28-31

[34] J. C. Platt. Fast training of support vector machines using sequential minimal optimization [J]. In Advances in Kernel Methods—Support Vector Learning, B. Scholkopf, C. J. C. Burges, and A. J. Smola, Eds.Cambridge, MA, USA: MIT Press, 1998

[35] D. Anguita, A. Boni and S. Ridella. A digital architecture for support vector machines: theory, algorithm, and FPGA implementation [J]. In IEEE Transactions on Neural Networks, vol. 14, no. 5, 993-1009, Sept. 2003

[36] M. Papadonikolakis and C. S. Bouganis. A scalable FPGA architecture for non-linear SVM training[C]. 2008 International Conference on Field-Programmable Technology, Taipei, 2008, 337-340

[37] M. Papadonikolakis, C. S. Bouganis and G. Constantinides. Performance comparison of GPU and FPGA architectures for the SVM training problem[C]. 2009 International Conference on Field-Programmable Technology, Sydney, NSW, 2009, 388-391

[38] S. Cadambi et al. A Massively Parallel FPGA-Based Coprocessor for Support Vector Machines[C]. 2009 17th IEEE Symposium on Field Programmable Custom Computing

Machines, Napa, CA, 2009, 115-122

[39] J. F. Wang, J. S. Peng, J. C. Wang, et al. Hardware/software co-design for fast-trainable speaker identification system based on SMO[C]. 2011 IEEE International Conference on Systems, Man, and Cybernetics, Anchorage, AK, 2011, 1621-1625

[40] A. Hussain, S. Shahbudin, H. Husain, et al. Reduced set support vector machines: Application for 2-dimensional datasets[C]. 2008 2nd International Conference on Signal Processing and Communication Systems, Gold Coast, QLD, 2008, 1-4

[41] Chang C C, Lin C J. LIBSVM: A library for support vector machines [M]. ACM, 2011

[42] Art. ID 27.S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, et al. Improvements to Platt's SMO Algorithm for SVM Classifier Design [J]. In Neural Computation, vol. 13, no. 3, 637-649, March 1 2001

[43] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines [J]. J. Mach. Learn. Res., vol. 6, 1889–1918, Dec. 2005

[44] S. Afra and U. Braga-Neto. Studying the possibility of peaking phenomenon in linear support vector machines with non-separable data[C]. 2011 IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS), San Antonio, TX, 2011, 218-221

[45] 祝曙光,钱丽艳,樊卫兵,等.非线性支持向量机若干关键问题研究[J].计算机工程与科学,2010,32(05):41-44.

[46] 邵臻. 基于特征分析和数据降维的复杂数据预测与分类方法研究[D].合肥工业大学,2015

[47] M. Budhiraja. Multi label text classification for un-trained data through supervised learning[C]. 2017 International Conference on Intelligent Computing and Control (I2C2), Coimbatore, India, 2017, 1-3

[48] G. C. Sobabe, Y. Song, X. Bai and B. Guo. A cooperative spectrum sensing algorithm based on unsupervised learning[C]. 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, 2017, 1-6

[49] K. Yao and J. Gao. Law of Large Numbers for Uncertain Random Variables [J]. In IEEE Transactions on Fuzzy Systems, vol. 24, no. 3, 615-621, June 2016

[50] S. Whiteson, B. Tanner, M. E. Taylor, et al. Protecting against evaluation overfitting in empirical reinforcement learning[C]. 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), Paris, 2011, 120-127

[51] Y. Guerbai, Y. Chibani and B. Hadjadji. Handwriting gender recognition system based on the one-class support vector machines[C]. 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA), Montreal, QC, Canada, 2017, 1-5

[52] A. Hassaine, Z. Safi, J. Otaibi, et al. Text Categorization Using Weighted Hyper Rectangular Keyword Extraction[C]. 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 2017, 959-965

[53] Y. Kimura, A. Watanabe, T. Katoh, et al. Improvement of Referrer SPAM blocking system[C]. 2008 International Symposium on Information Theory and Its Applications, Auckland, 2008, 1-6

[54] A. U. Surwade, M. P. Patil and S. R. Kolhe. Effective and adaptive technological solution to block spam E-mails[C]. 2016 International Conference on Advances in Human Machine Interaction (HMI), Doddaballapur, 2016, 1-10

[55] A. Kozma, J. V. Frasch and M. Diehl. A distributed method for convex quadratic programming problems arising in optimal control of distributed systems[C]. 52nd IEEE Conference on Decision and Control, Firenze, 2013, 1526-1531

[56] 陈哲.向量优化中广义增广拉格朗日对偶理论及应用[J].数学物理学报,2008(03):570-577

[57] Y. Deng, T. Xu, Y. Li, et al. Icing thickness prediction of overhead transmission lines base on combined kernel function SVM[C]. 2017 IEEE Conference on Energy Internet and Energy System Integration (EI2), Beijing, 2017, 1-4

[58] 周晓剑,马义中,朱嘉钢.SMO 算法的简化及其在非正定核条件下的应用[J].计算机研究与发展,2010,47(11):1962-1969

[59] B. Scholkopf and A. J. Smola. Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond [J]. Cambridge, MA,USA: MIT Press, 2001

[60] A. Frank and A. Asuncion. (2010). UCI Machine Learning Repository. [Online]. Available: http://archive.ics.uci.edu/ml

攻读硕士期间获得成果

发表论文与专利

[1] Shang Ma, **Shengqiang Jiang**, Jienan Chen et al. Design and Implementation of Search SVM Parameters Based on Shared Dot Product Matrix [J]. Submitted to IEEE CIRCUITS AND SYSTEMS MAGAZINE，2017

[2] 马上，**蒋生强**，陈杰男等. 一种支持向量机的最佳参数组合（C，σ）快速搜索方法. 专利号：CN201710724607.8

参与项目

[1] 基于概率计算的机器学习算法设计与实现，57 所合作项目，主研