编　　号：_____
审定成绩：_____

重庆邮电大学
毕业设计（论文）

设计（论文）题目：基于 SVM 分类器的动作识别系统

学　院　名　称：计算机科学与技术

学　生　姓　名：朱　轲

专　　业：计算机科学与技术卓越工程师班

班　　级：0491201

学　　号：2012211830

指　导　教　师：邓　欣

答辩组　负责人：陈　龙

填表时间：2016　年　5　月

重庆邮电大学教务处制

摘　要

动作识别领域近年来随着动作采集技术的成熟而高速发展，通过对三维动作数据进行处理挖掘，从而无需借助任何传统计算机输入设备就能够准确识别出用户意图。现已广泛运用到了计算机动画、游戏、新型人机交互和智能家居控制等领域。

支持向量机（Support Vector Machine, SVM）凭借其在小训练样本、非线性和高维模式识别中的优势而广受关注。本文对经典 SVM 二分类算法进行研究，在此基础上将 SVM 算法推广到了多分类中。此外通过获取智能手机中的加速度传感器、陀螺仪和方位传感器的数据，搭建了一个动作数据采集、传输和存储平台，支持多用户传输存储其动作数据。采用 SVM 多分类算法训练预处理后的动作数据，并采用粒子群优化算法(PSO)对 SVM 参数进行优化，建立动作分类模型，实验证明该模型能够 94.03%的准确率识别出用户的动作意图。

为了验证基于 SVM 分类器的动作识别系统的运用场景，本文将其运用到了智能家居家电控制领域，通过软件搭建了一个智能家居模拟模块，可以模拟实体智能家居的一系列状态信息（如打开电灯）。通过对用户动作数据的分类学习，可以达到通过动作信息控制家电开关等状态动作的功能目的，为基于 SVM 分类器的动作识别系统找到了一个应用场景。

【关键词】动作识别　支持向量机　多分类　粒子群优化　智能家居

ABSTRACT

With the motion capture technology matures and promotion, fast and efficient access to a large number of real-time and three-dimensional motion data has become a reality. Because has the ability to accurately identify the user's intention without the help of any conventional computer system input devices, the action of the three-dimensional data processing technology has been widely applied to computer animation, games, new human-computer interaction and intelligent home control field.

Support vector machine (SVM) has its unique advantages in solving the small sample, nonlinear and high dimensional pattern recognition. This paper through to studying classical SVM binary classification algorithm, on the basis promoted the binary classification to multi-classification field. In addition, by obtaining data on the smart phone accelerometer, gyroscope and compass sensor, built a movement of data collection, transmission and storage platform, to support multi-user transmission data and stored its action data. After data normalization, using SVM multi-classification algorithm, and using PSO algorithm for SVM parameters optimization, built their action data model. The experiment proved that the action model can predict the user's action intentions with 94.03% accuracy.

In order to verify the application scenarios of motion recognition system based on SVM classifier, this paper applied it to the fields of smart home control module, through built the smart home simulation module that can simulate a series of status information of appliances (such as turn on lights) by software. Based on user’s motion data learning, SVM classification model can control the appliance’s status. This find an application scenario based SVM classifier action recognition system.

【Key words】 Motion recognition SVM Multi-classification PSO Smart home

目　录

前　言

SVM 分类算法是一种基于统计学习理论的分类算法，其在小样本、高维和非线性的分类条件下拥有较高的分类正确率。动作识别是通过对采集到的动作数据进行数学建模和分析，建立相应的分类模型，对新的动作数据输入能够得到合理的分类预测值的一项技术。智能家居系统经过多年的发展，已日趋成熟。但人对智能家居的控制方式，依然停留在遥控器控制、APP 远程控制、定时控制等水平上。此类控制方式都需要用户进行手动设置，控制方式的不灵活造成了用户的体验较差。

本文搭建了一个基于 SVM 分类器的动作识别系统。系统通过软件模拟智能家居系统中的各种行为（如电灯的开关，电视机的开关、换台等），智能手机能在日常活动中实时监测用户动作，收集用户对家电进行特定操作的动作数据。通过对数据分析和学习，建立特定用户的动作模型，预测用户下一时刻动作所对应的家居操作，并向该智能网络下的所有家电广播预测结果。家电自动响应相应操作，实现了动作的自学习和自响应。

通过将本动作识别系统应用到智能家居的控制方式的领域，打破传统智能家居系统需要人工设置的半智能局面，为 SVM 算法和动作识别领域找到了一个全新的应用领域。

第一章 绪　论

本章将介绍课题研究背景、意义及国内外研究现状，同时也介绍了本文的主要行文结构以及课题的主要研究成果。

第一节 研究背景及意义

SVM 是基于统计学习的学习算法。有限样本空间即可得到较高准确度的模型，能够对训练样本进行非线性分类，且在高维空间中具有较低的算法复杂度的特性，使得 SVM 算法被广泛应用[1]。

SVM 算法目前已经在生物信息学、语音识别、人脸检测和数据挖掘等许多领域取得了优秀的成果。SVM 既优化模型复杂度又优化经验风险，且随着样本空间维数的增高不会增加计算复杂度，因此它经常被应用到高维模式中。

人体动作包括四肢、头或面部等的姿态或运动过程，这些动作具有丰富的含义，是一种人与环境的信息传递方式。计算机接收动作信号的输入，对动作信号进行检测分析，得出动作的特定意图的过程，称为计算机动作识别。其已经在教育、体育、游戏等领域得到了广泛的运用。动作识别领域主要通过数学方法来分析手势信号，其目的是让计算机理解人类的动作语言。

围绕人体动作所展开的工作可回溯至上世纪 70 年代，心理学家通过实验结果证明：人类视觉系统可以通过感知运动过程中的光点序列变化，从而推断出运动的类型比如站立、奔跑等。实验也表示人体动作往往代表了特定的动作序列代表了特定的用户意图。

随着传感器和计算机视觉技术的成熟和推广，动作识别领域渐渐发展成了两个方向：基于视觉传感器，和基于可穿戴传感器的人体动作识别。前者主要通过外部图像采集设备，如运动摄像头等，对人体的动作信息进行捕捉，通过动作关键帧的提取和分析，得出用户的动作意图。后者主要通过安装传感器，对运动中的动作信号（如加速度大小）进行收集，而后对数据进行数学分析，从而得出用户的具体意图。近年来，随着人工智能的发展以及其在动作识别领域的成功运用，也为动作识别的分析预测精确度带来了极大的提升。其中比较常见的是各种基于概率的统计识

别方法：决策树、K 临近、贝叶斯、SVM、神经网络和隐形马尔科夫（HMM）等。

本文通过对 SVM 算法的学习和研究，并将其应用在动作识别系统中，结合智能家居的概念，模拟实现了基于 SVM 分类的动作结果对家电状态的控制，验证了一种新型的智能家居控制方式的可行性，对 SVM 算法和动作识别系统的推广和应用有重要的意义。

第二节 国内外研究现状

SVM 方法被提出后，由于 SVM 算法能解决之前使用神经网络等算法无法解决的问题，从而引起了国内外研究学者的广泛兴趣。在国内外学者的广泛研究下，SVM 算法理论在短短几年就涌现了大量的研究成果。1998 年 Weston et al.和 Vapnik 在 SVM 在回归问题和多值分类扩展以及它的泛化性方面进行了研究；1999 年 Anthony et al.等人在 SVM 硬领域的学习误差上给出了严格的理论界定；2000 年 Shawe-Tylor 和 Cristinanini 软领域 SVM 也给出了类似的误差界定；随着学者们对 SVM 理论研究的深入，许多以 SVM 为基础的延伸算法得到了提出，如 1999 年 Smolaet al.提出的 SVM 应用在分类和回归问题上；其他一些学者还扩展了 SVM 算法的概念，如 1997 年 Mangasarian 等人对通用 SVM 的概念进行了详细的阐述[1]。

SVM 最初被应用到了手写识别领域。当时美国邮政服务局为了自动识别手写的邮政编码而引入了 SVM，实际证明，采用 SVM 方法识别效果优于其他神经网络算法。SVM 的出现同时为字符识别领域开创了一个新的研究方向。

近些年，国内也涌现出了大批 SVM 研究成果。比如 LS_SVMlab、OSU_SVM3.00、stprtool\SVM 和 SVM_Steve Gunn 四种 SVM 方法的 MATLAB 工具箱被海军工程大学的陆振波博士研究开发了出来。刘向东等提出了一种在提高识别速度情况下保持较高精度的快速 SVM 算法（FCSVM）。

基于机械和物理、基于图像和基于传感器的动作识别是该领域的三个主要研究方向。基于机械和物理的动作识别的研究起源于 1983 年，该时期也产生了被认为是动作识别领域最早的专利 Grimes 数据手套的专利。1987 年，通过光纤传感器测量手指的弯曲数据，VPL 公司制造了 Data Glove 进行手势识别。国内的高文等人将 Data Glove 数据手套与 ANN 和 HMM 相结合建立了中国手势识别系统，使一些较为简单的手势识别率达到了 90%左右。

由于数据手套对动作采集设备的依赖性太强等，研究人员开始将计算机视觉技

术应用到了动作识别领域。通过图像处理技术对动作图像进行分析，从而实现手势动作的识别。这其中最具代表性的产品为微软公司的产品 Kinect，其能够通过摄像头捕捉人体各个部分比较细微的动作，从而在人机交互方式中带给用户一种全新的舒适体验。

基于图像的动作识别系统受到对周围环境依赖大和识别过程中对硬件要求高的约束而发展受阻。而基于传感器的动作识别由于其特有的优势而广受关注。孔俊其等人的研究课题——基于三维加速度传感器的手势识别，对简单的数字和动作进行了分类识别，取得了较好的实验效果。

目前国内研究者们在将 SVM 方法应用于动作识别领域的过程中，取得了一些显著的成就，但是大部分研究者都是针对动作的图像信号，而针对传感器信号的研究和实验较少。为了验证这种方法的可行性，本文采用 SVM 方法对采集智能手机的加速度、角速度和方位信号作为 SVM 算法的输入信号，进行了实验和验证。

第三节 论文结构

本文内容共分为五个章节，章节介绍如下：

第一章，绪论。首先介绍了动作识别和 SVM 的基本现状和发展情况，为后续的研究和开发工作奠定了背景基础。

第二章，SVM 算法研究。主要对 SVM 进行了理论知识的推导，并介绍了多分类 SVM 的构造方法，和利用 PSO 对 SVM 参数进行优化的步骤。

第三章，动作采集模块的设计与实现。主要介绍了动作采集和传输的方案与实现。

第四章，动作识别模块的设计与实现。详细介绍了有关数据接收和处理， SVM 算法，智能家居模拟模块的实现方案。

第五章，结论。对 SVM 分类算法运用到动作识别系统中的分类效果进行了总结，并分析了比较了神经网络、K 临近算法等其他分类器对动作数据的分类效果。

第四节 本文主要任务与成果

本文主要任务与成果在以下六个方面：

(1) 在分析理解 SVM 算法的基础上，将传统的 SVM 二分类模型延伸到多分类，使其适应本文的动作识别系统的需要。并提出使用 PSO 对 SVM 参数进行优化；

(2) 搭建了一个针对动作数据的采集、传输和存储平台。

(3) 搭建智能家居模拟模块，能够模拟实体智能家居设备的各类属性和状态，包括：能够发出学习信号，对新的动作数据进行有效的学习；能够接收控制命令，并根据控制命令改变自身的属性和状态；

(4) 完成 SVM 核心算法模块，模块功能包含：对动作数据进行系列预处理，对新的动作数据进行 SVM 算法学习并构建模型，能够通过模型对用户的动作数据进行学习并预测得出的动作意图结果，向智能家居模拟模块广播控制命令，使其相应地改变属性状态；

(5) 为了验证 SVM 分类器动作识别系统应用到智能家居控制交互模块中的大规模应用的可能性，整合上述功能，并实现了多用户同时在线使用和分析的场景；

(6) 最后运用其他分类算法，如 KNN 算法、神经网络等对相同动作数据进行分类学习，简单比较了不同算法的不同特性。

动作采集方面，通过对智能手机中的加速度、陀螺仪和方位传感器的调用，采集到设备持有者的实时动作信号，通过对信号的预处理过程，采用 SVM 对信号数据进行训练得到分类模型。该模型能够对动作数据进行分类得到用户的特定动作意图。并结合智能家居的概念，用软件模拟了智能家居网络中的各种家电设备（智能电灯）的状态，搭建完成了智能家居模拟系统。

SVM 分类算法方面，根据提取到的动作数据特性，对动作数据进行归一化处理，减少算法的计算复杂度和增加分类的准确率。本文在对 SVM 算法进行理论推导和分析的基础上，将传统的二分类 SVM 算法推广到了多分类上，更好地满足了本文的动作识别系统的需要。利用 PSO 对 SVM 的参数进行了优化，较之传统参数寻优方法在预测准确度上有了更优秀的表现。

第五节 本章小结

本章主要介绍了课题的研究背景和国内外研究成果，而后对本文的行文结构进行了阐述，并引出了本课题的主要任务和研究成果。

第二章 SVM 算法研究

本章将对 SVM 原理进行推导分析，随后提出构造多分类 SVM 的几种方法。最后介绍了利用 PSO 对 SVM 参数进行优化的方法。

第一节 SVM 方法介绍

一、最优分类面

SVM 的核心思想是从线性可分条件下，寻找最优边界距离的分类超平面问题得以提出。

1、最优分类面

在图 2.1 中，两类不同数据点表示两类样本，在线性可分条件下，H 能够对两类样本进行正确分隔。H_1，H_2两直线过两类样本中离 H 最近的点，且与 H 平行。H_1和H_2之间的线段长度叫做两类的分类间隔(margin)[2]。

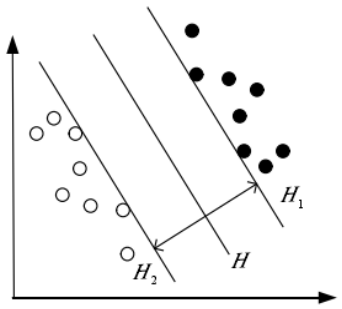


图 2.1 最优分类面示意图

下面对最优分类面做如下定义：能将图中的两类训练样本完全正确地分隔，且保证H_1，H_2之间的分类间隔最大的分类面，称为最优分类面。分类线 H 保证了训练样本的经验风险最小化，和测试样本的错误率最小。

设(x_i, y_i)为线性可分样本集，其中 i= 1, …, n, x∈ R^d, 其中 y ∈ $\{-1, 1\}$是分类的类别标识，则分类判别函数具有如下一般形式：

$$g(x) = \boldsymbol{w}\cdot \mathrm{x} + \mathrm{b}$$

则最优分类面的方程为：

$$\boldsymbol{w}\cdot \mathrm{x} + \mathrm{b} = 0 \qquad (2.1)$$

观察式(2-1)可得，w、b 的取值决定了最优分类面的形式，可将分类超平面简单表示为 $(\boldsymbol{w}, \mathrm{b})$。但是当 $\boldsymbol{w}$ 和 b 同时与非零常数 k 相乘时，$(\mathrm{k}\boldsymbol{w}, \mathrm{kb})$决定的分类超平面不变，两组不同的参数决定了同一个分类超平面[3]。

引入规范超平面的概念就是为了解决这种矛盾。在规范超平面中，所有的训练样本都满足$|g(x)| \geq 1$，H_1，H_2上的样本则满足$|g(x)| = 1$，H_1，H_2的平面间距离为

$\frac{2}{||w||}$，当$||w||$最小使，可得H_1，H_2间的分类间隔最大。

同时分类线必须满足：

$$y_i[(\mathrm{w}\cdot x_i)+\mathrm{b}]-1\geq 0,\mathrm{i}=1,2,\ldots,\mathrm{n} \tag{2.2}$$

如果一个分类面同时满足式(2-2)和$||w||$最小的条件，我们就称之为最优分类面。我们称满足$f[(\mathrm{w}\cdot x_i)+\mathrm{b}]-1=0,\mathrm{i}=1,2,\ldots,\mathrm{n}$要求的点为支持向量。支持向量支撑了最有分类面，既平行于最优分类面，又距离最优分类面最近。

下面将介绍如何对最优分类面进行求解。

$$\begin{cases}\min[\phi(\omega)=\frac{1}{2}(\omega\cdot\omega)]\\ s.t.\quad y_i\big((\omega\cdot x_i)+b\big)\geq 1,i=1,2,\ldots,l\end{cases} \tag{2.3}$$

将式(2-3)进行拉格朗日函数转换：

$$\mathrm{L}(\omega,\mathrm{b},\alpha)=\frac{1}{2}(\omega\cdot\omega)-\sum_{i=1}^{n}\alpha_i\{y_i[(\omega\cdot x_i)+b]-1\} \tag{2.4}$$

将原问题化为了 Lagrange multiplier 的极小值问题，即：

$$\frac{\partial L}{\partial\omega}=\omega-\sum_{i=1}^{l}\alpha_i y_i x_i=0 \tag{2.5}$$

$$\frac{\partial L}{\partial b}=\sum_{i=1}^{l}\alpha_i y_i=0 \tag{2.6}$$

由 KKT 定理可得，最优解满足：

$$\alpha_i\big(y_i\big((\omega\cdot x_i)+b\big)-1\big)=0 \tag{2.7}$$

由此可得只有支持向量的系数α_i不为 0，因此ω可以表示为：

$$\omega=\sum_{SV}\alpha_i y_i x_i \tag{2.8}$$

将式(2-5)和式(2-6)代入到式(2-4)中，原拉格朗日函数化简为：

$$\mathrm{L}(\omega,\mathrm{b},\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{l}y_i y_j\alpha_i\alpha_j(x_i\cdot x_j) \tag{2.9}$$

将原问题化为如下的 Lagrange multiplier 对偶问题：

$$\begin{cases}\max\quad\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{l}y_i y_j\alpha_i\alpha_j(x_i\cdot x_j)\\ s.t.\quad\sum_{i=1}^{n}\alpha_i y_i=0,\alpha_i\geq 0,i=1,2,3,\ldots,l\end{cases} \tag{2.10}$$

若α^*为问题(2-9)的一个解，则：

$$\omega^*=\sum_{i=1}^{n}\alpha^* y_i y_i \tag{2.11}$$

选择不为零的α_i代入式(2-7)中解出b^*后得到最优分类面是：

$$f(x)=sgn\{(\omega^*\cdot x)+b^*\}=sgn\{\sum_{j=1}^{n}\alpha_i^* y_i(x_i\cdot x)+b^*\} \tag{2.12}$$

其中 *sgn* 为函数符号。最优分类函数由训练样本中的支持向量作为支撑，非支

持向量对最优分类面的求解无意义，其拉格朗日系数为 0[4,5]。

至此，对于任意未知样本的输入，都可以将其代入最优分类函数 *f(x)*，得到其分类结果。

2、广义最优分类面

在本文 2.1.1 节中所讨论的最优分类面有一个前提条件：数据样本能够被线性函数没有差错地进行分类。这种假设是在最理想的情况下，实际是大部分数据集都不能使用线性函数进行分类。在求解 SVM 最优分类面的时，误差样本会对结果造成很大的影响。

因此，在解决实际问题的时候，往往在约束条件中添加一个松弛项ξ_i，$\xi_i \geq 0$，式(2-2)转化为：

$$y_i[(\mathrm{w}\cdot x_i)+\mathrm{b}]-1+\xi_i \geq 0, \mathrm{i}=1,2,\ldots,\mathrm{n} \tag{2.13}$$

目标函数为：

$$\phi(\omega,\xi)=\ \frac{1}{2}(\omega\cdot\omega)+C\sum_{i=1}^{l}\xi_i \tag{2.14}$$

其中参数 C 可调节，表示松弛项ξ_i在目标函数中所占比例。C 越大表示最优分类面对误差项的调整程度越大。C 的可调节是算法有了较高的鲁棒性。

3、高维空间中的最优分类面

如果训练样本在原特征空间维度是线性不可分的，那么在原特征维度下进行 SVM 计算寻找到的最优分类面分类效果往往不佳。为了解决这个问题，可通过非线性变换，将问题转换到高维数据空间中，进而求解分类问题。

如图 2.2 展示了原空间到高维空间的转换原理图。

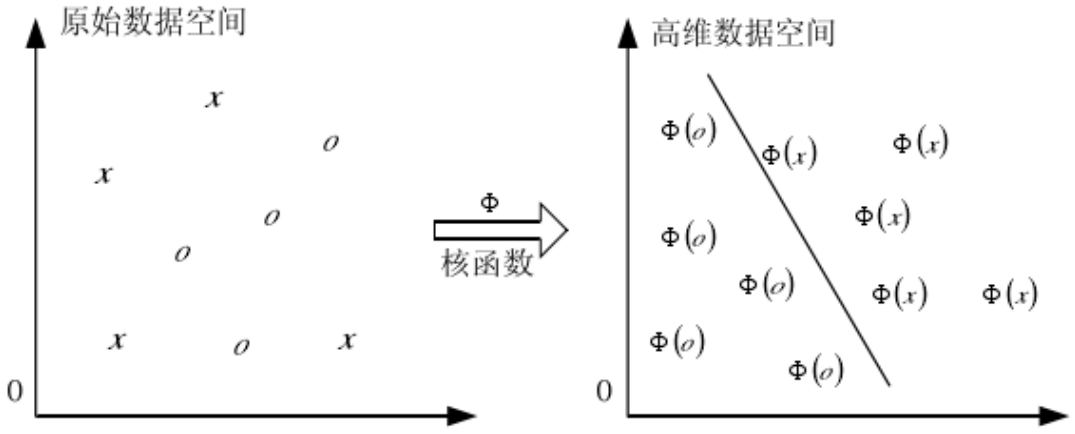


图 2.2 原空间到高维空间的映射

SVM 算法需要合理选择核函数类型。实际应用中常用的核函数类型有：

(1) 线性核函数：

$$\mathrm{K}(x_i, x_j) = \ x_i^T x_j \tag{2.15}$$

(2) 多项式核函数：

$$\mathrm{K}(x_i, x_j) = \ (\gamma x_i^T x_j + \gamma)^d, \gamma > 0 \tag{2.16}$$

(3) 径向基核函数(RBF)：

$$\mathrm{K}(x_i, x_j) = \exp\left(-\gamma \left|\left|x_i, x_j\right|\right|^2\right), \gamma > 0 \tag{2.17}$$

(4) S 型核函数(Sigmoid)：

$$\mathrm{K}(x_i, x_j) = tanh(\gamma x_i^T x_j + \gamma) \tag{2.18}$$

RBF 所对应的特征维数无穷大，可对任意数目的样本得到最有分类结果，因为得到了最广泛的应用。而 RBF 中最常用的又数高斯核函数，其核函数表达式为：

$$K\left(\left|\left|x - \ x_c\right|\right|\right) = \exp(-\left|\left|x - x_c\right|\right|^2/\sigma^2)$$

其中x_c为函数的中心点，σ为函数的宽带参数。

将该核函数应用到分类函数中，得到最优分类函数：

$$f(x) = sgn\{\sum_{j=1}^{n} \alpha_i^* \, y_i \exp\left\{-\frac{|x-x_i|^2}{\sigma^2}\right\} + b^*\} \tag{2.19}$$

二、支持向量机

通过引入核函数变换，将问题变换到了更高维数据空间的优化问题，在高维空间中求解最优的分类函数，我们称这种方法为支持向量机。

支持向量机具有如下特点：

(1) 引入核函数对原数据空间进行高维映射，使对原本低维空间中不可分的数据得到了较好的分类结果。

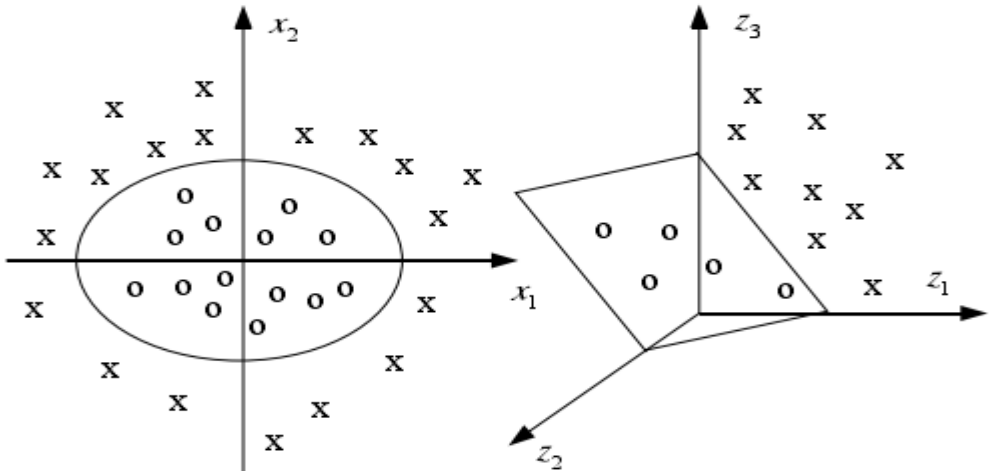


图 2.3 三维空间的分类结果

如图 2.3 的分类例子，在二维空间中线性不可分，但是将其扩展到三维空间时，能够很好地找到最优分类面。

(2) SVM 方法旨在划分最优超平面，该算法的核心为找到使分类边界之间的间隔最大的参数值。如图 2.4 所示，H 为最优超平面，SVM 算法是要将 H_1 和 H_2 之间的边界距离最大化。通过 Lagrange multiplier 变换得到最优分类决策函数为：

$$f(x) = sgn\{\sum_{i=1}^{l} y_i \alpha_i^* < \phi(z_i) \cdot \phi(z) > + b^*\} \qquad (2.20)$$

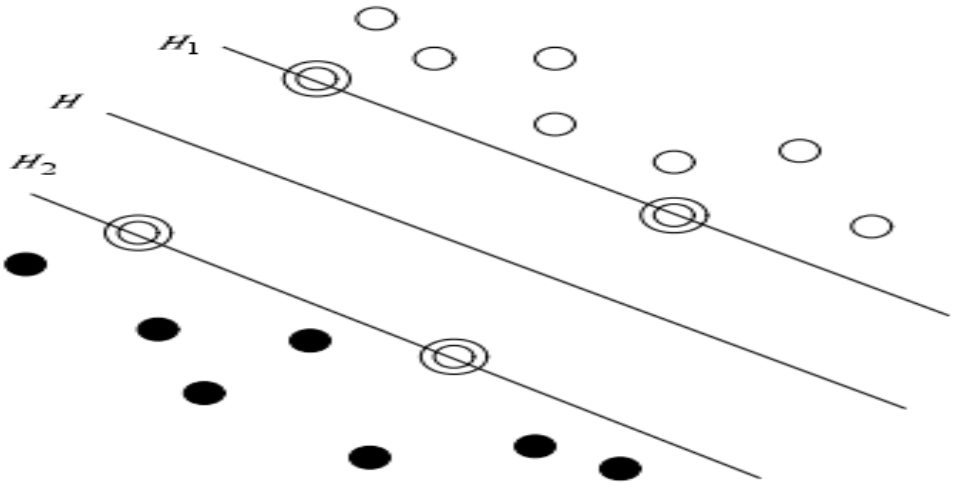


图 2.4 支持向量机与最优分类面

(3) SVM 决策函数的分类结果最终依赖于为输入值和支持向量的内积结果。通过核函数作为桥梁连接，将高维空间和低维空间的内积运算进行相互转化，优化了算法效率。

(4) 支持向量在计算 SVM 决策函数时起决定性作用。由图 2.4 可知，由两个决策平面 H_1 和 H_2 决定了最优分类平面 H，除了 H_1 和 H_2 上的点外，其他样本点都未参与计算，我们成 H_1 和 H_2 上的点位支持向量。

(5) SVM 算法具有较好的鲁棒性。其决策函数由支持向量进行构建，当样本空间中增加或删除非支持向量的数据时不会对决策函数产生影响。

第二节 多分类的支持向量机

本文构建的 SVM 分类器的动作识别系统，用户动作意图的多样性是传统二分类 SVM 所无法解决的，故此需要将 SVM 算法从二分类推广到多分类的应用中。下面将介绍几种常见的多分类 SVM 构造方法。

(1) 多分类问题可以由多个二分类问题组成，将多个二分类进行组合分解，从而形成一个多分类 SVM 模型。常用的技巧包括一对一、一对多、有向无环图方法。

(2) 一次求解多分类 SVM 问题的所有分类超平面。该方法逻辑简便易懂，但是具有计算时间复杂度高的缺点。

下面将对本文采用的一对一的 SVM 多分类器的构造方式进行研究介绍。

一对一方法是基于传统的二分类问题，对于 K 分类问题，构造所有可能存在的$\frac{K(K-1)}{2}$个二分类 SVM，构建二分类 SVM 模型，得到$\frac{k(k-1)}{2}$个分类判别函数。

在对未知样本输入进行分类时，将输入分别代入$\frac{k(k-1)}{2}$个分类判别函数中，分别得到不同的分类结果，对最终结果进行统计，得票最多的分类结果将成为最终分类结果。

第三节 本文 SVM 算法的实现

通过上面对 SVM 算法的介绍和分析推导，说明 SVM 算法能很好地满足本文构建的动作分类器的要求：有限样本空间、样本数据线性不可分。本文选取了台湾大

学林智仁教授等开发的 SVM 开源代码包 LIBSVM 作为本文的算法实现。

LIBSVM 的多分类实现方式为上文介绍的一对一方法，通过在在 k 个分类之间设置$\frac{k(k-1)}{2}$个 SVM 的方式，构建多分类模型。每个 SVM 各自得出其分类结果，选出得票最高的结果作为分类结果。

LIBSVM 代码包封装并实现了 SVM 算法中的大量复杂的推导和计算过程，开发了对 SVM 算法影响较大的各类参数可供调节，如惩罚系数 C，核函数系数σ等，能够很好地满足本文搭建基于 SVM 分类器的动作识别系统的需要。

第四节 粒子群优化 SVM 参数

通过上文对 SVM 的推导过程可知，参数的选取对其准确性有较大的影响，本节介绍 PSO 对 SVM 参数进行优化的方法步骤。

一、SVM 参数介绍

ε：不敏感系数，该参数表示算法对误差的敏感程度。支持向量的数量和ε的值成反比关系；

σ：核函数参数。该参数影响样本在高维特征空间中的复杂程度，核函数参数的改变隐含地改变了映射函数，从而决定了线性分类面的最小分类误差[7]。

C: 该参数影响置信范围和经验风险的比例，通过对该参数的调节，能够使 SVM 算法达到最佳的推广能力。经验误差的惩罚和 C 的值成正比关系。

目前常用的方法是通过不断尝试的方法来确定较好的算法参数，从而获得相对较为优秀的参数组合。

二、粒子群优化算法介绍

粒子群优化算法(Particle Swarm Optimization,简称 PSO)是一种新型进化计算方法，算法是对鸟群的群聚行为的模拟。

PSO 对问题进行优化的过程中，空间中有唯一的一个食物（即 PSO 要寻找的最优解），假设空间中存在一群在搜索食物的鸟（每只鸟为一个粒子），但是鸟不知道要寻找的食物的具体位置，只知道和食物的距离远近、自己曾经搜索到的和食物最

近的位置和鸟群搜索到的和食物最近的位置，每只鸟会根据当时所在位置和个体经验最近距离和鸟群经验最近距离比较，从而更新自身的速度和位置，通过不断迭代，找到食物的位置（或距离食物最近的位置）。这就是 PSO 的核心思想。

PSO 的数学描述为：在一个 D 维空间中，m 个粒子组成一个粒子群。向量$p_i = (p_1, p_2, \dots, p_D)$, i = 1,2, … , m 表示第 i 个粒子的位置。*f(x)*为待优化的目标函数，通过将 x_i 代入 *f(x)*得到的适应值与 pbest 和 gbest 进行比较，得到该粒子位置的优劣，从而更新期速度和位置。$v_i = (v_{i1}, v_{i2}, \dots, v_{iD})$, i = 1,2, … , m表示粒子 i 的飞行速度的矢量，$p_{pbesti} = (p_{i1}, p_{i2}, \dots, p_{iD})$, i = 1,2, … , m表示粒子 i 的经验最优位置，$p_{gbest} = (p_{g_1}, p_{g_2}, \dots, p_{g_D})$表示整个种群搜索到的经验最优位置。

粒子群搜索的每一次迭代更新:

$$v_i = w * v_i + c_1 * rand() * (p_{pbesti} - p_i) + c_2 * rand() * (p_{gbest} - p_i)$$

$$p_i = p_i + v_i$$

其中：i = 1,2,⋯ ,m，w 控制了历史速度对当前速度的影响，表示惯性权重；c_1和c_2为加速因子，分别表示个体经验值对速度的影响和群体经验值对速度的影响；rand()产生[0,1]的随机数。

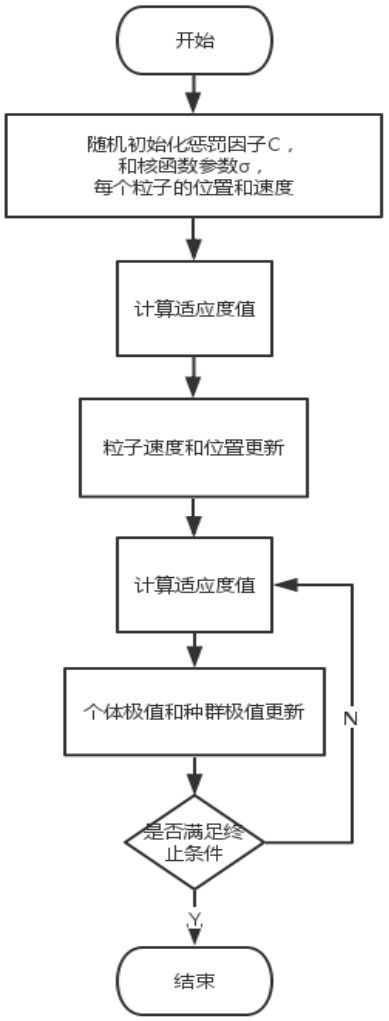


图 2.5 PSO 优化流程图

三、粒子群优化 SVM 参数

选取 SVM 的两个参数惩罚因子 C 和核函数参数σ作为 PSO 的优化目标。

训练样本和测试样本集通过本文第三章构建的动作采集模块进行采集，总共 100 组数据，随机选取 70 组数据作为训练集，其余作为测试样本集。

设每一个 PSO 粒子代表 SVM 算法的一组参数，将该组参数下的 SVM 分类准确度作为粒子的适应度函数，其形式如下：

$$f_{fitness} = \frac{1}{n}\sum_{i=1}^{n} f(x) \qquad (2.21)$$

$$f(x)=\begin{cases}1, \text{预测正确}\\ 0, \text{预测错误}\end{cases} \qquad (2.22)$$

其中，n 是训练样本的数量，y_i是实际值，$\overline{y_i}$是预测值，$f_{fitness}$是适应度值。

算法具体实现过程如下[13]：

1. 读入样本集，对数据进行预处理；
2. PSO 参数初始化，设 PSO 算法参数 w = 0.8，$c_1 = c_2 = 2$，种群规模为 50，最大迭代次数C_{max} =500，解空间为二维空间，分别为惩罚因子$C \in [0.01, 500]$和核函数参数$\sigma \in [0.01, 10]$；
3. 随机各粒子的初始速度和位置，将每个粒子的p_{pbesti}设为当前位置，p_{gbest}设为群体中位置最好的粒子位置；
4. 使用训练集训练 SVM 模型，并根据适应度函数$f_{fitness} = \frac{1}{n}\sum_{i=1}^{n} f(x)$更新粒子的适应度值；
5. 将粒子的适应度值和p_{pbesti}，p_{gbest}进行比较，并根据比较结果进行更新;
6. 按照 $v_i = w * v_i + c_1 * rand() * (p_{pbesti} - p_i) + c_2 * rand() * (p_{gbest} - p_i)$，$p_i = p_i + v_i$ 更新粒子的速度和位置；
7. 如未达到算法的停止条件，则返回至 3 继续循环。否则求出最优解，算法结束。

注：算法的终止条件为达到最大迭代次数C_{max}。经过粒子群优化算法得到的 SVM 参数为：**$\sigma = 0.000488$，C = 512.0**，得到的 SVM 预测准确度为 **94.03%**。

第五节 本章小结

本章学习和研究了 SVM 的原理方法，对影响 SVM 算法性能和准确度的基本参数进行了介绍，而后对 SVM 的各个特点进行了分析和总结。随后确定了选用 LIBSVM 作为本系统的 SVM 算法实现，最后介绍和说明了利用粒子群优化算法对 SVM 参数进行优化的方案。本章是基于 SVM 分类器的动作识别系统的算法理论基础。

第三章 动作采集模块设计与实现

本章主要对动作采集模块进行的相关设计和功能进行了详细介绍，对动作采集模块的三个动作采集特征进行了介绍，而后将对动作采集 APP 的设计和实现做详细描述。

第一节 动作采集模块概述

随着移动互联网的飞速发展，智能手机的硬件配置也不断攀升，如今智能手机中已经集成了大量的传感器模块。智能手机中的各类传感器模块可以很好地满足本文搭建动作识别系统的动作采集需要。本文搭建的动作识别系统的动作采集模块如图 3.1 所示。

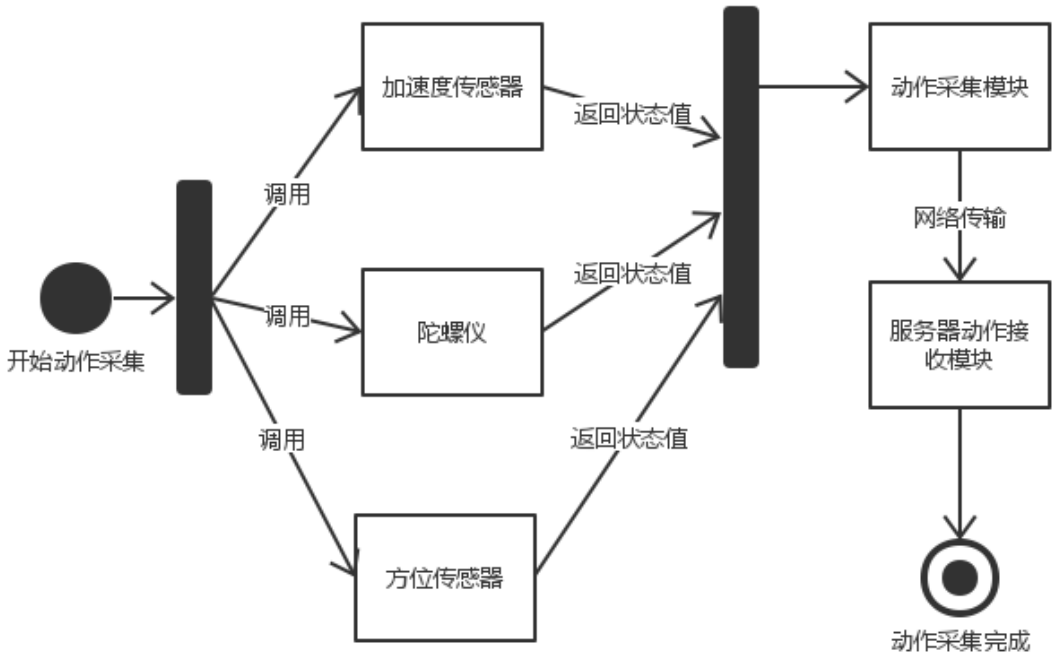


图 3.1 动作采集模块活动图

智能手机集成的各类传感器模块不断丰富，这些传感器模块可以通过 Android 编程很顺利地进行调用并获取到其传感器状态信息，这为我们实时高效地捕捉动作

数据带来了极大的便利性。本文通过 Android 编程，调用和获取传感器的状态信息，包括：加速度传感器、陀螺仪和方位传感器，加速度、角速度和方位这几个维度的数据可以很好地还原出设备持有者当时的三维轨迹信息，为后面的 SVM 算法分类器训练带来了可能。

第二节 采集动作特征说明

通过调用智能手机的加速度传感器、陀螺仪和方位传感器，每种传感器分别返回了状态信息，这些状态信息分别表示了不同的含义，下面将这几个特征量进行详细解释[6]。

一、加速度传感器

该装置被用来测量加速度值。通过测量外界加速度对内部机械结构的震动特性的影响，以此获得设备加速度值的大小。调用加速度传感器模块，可以获得该时刻下传感器的 x、y、z 轴的加速度数值，单位为m/s^2。

其数据具有如下特点：

向左倾斜运动，x 轴为正。

向右倾斜运动，x 轴为负。

向上倾斜运动，y 轴为正。

向下倾斜运动，y 轴为负。

垂直向上运动，z 轴为正。

垂直向下运动，z 轴为负。

二、陀螺仪

陀螺仪（英文：gyroscope）是感测与维持方向的仪器。陀螺仪的主要构成模块为一个可旋转的转子[8]。

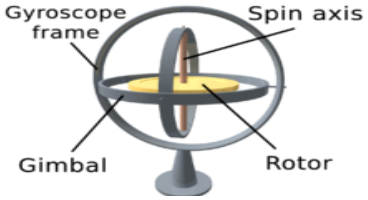


图 3.2 陀螺仪

可以通过智能手机中的陀螺仪传感器得到 x、y、z 轴的三轴角加速度的值，其单位为r/s。且其数据具有如下特点：

将手机水平正面朝上静止与桌面上，x 轴默认为 0，y 轴默认 0，z 轴默认 0。

逆时针旋转，z 轴为正。

顺时针旋转，z 轴为负。

向左旋转，y 轴为负。

向右旋转，y 轴为正。

向上旋转，x 轴为负。

向下旋转，x 轴为正。

三、方向传感器

方向传感器获取三轴角度（度）。如下：

方位：获得 Y 轴与地磁北极的偏移角度（水平时）。角度取值 0°到 360°。

x 轴和水平面的夹角：角度取值-180°到 180°。z 轴向 y 轴旋转取值为正。

y 轴和水平面的夹角：角度取值-90°到 90°。x 轴向 z 轴旋转取值为正。

第三节 动作采集 APP 设计与实现

本文为了调用智能手机的加速度传感器、陀螺仪和方位传感器的状态值，获得设备持有者的系列动作数据，编写完成了一个安卓应用，APP 提供登录、注册、动作特征量的获取显示和更新，并将动作数据发送至服务器的功能。

一、APP 设计概要

动作采集系统搭载运行在 Android 平台，通过编程实现了对智能手机传感器模块（包括：加速度传感器、陀螺仪和方位传感器模块）的调用，得到了某一时刻下三种传感器的状态信息[15]。

系统主要功能和流程图如图所示。

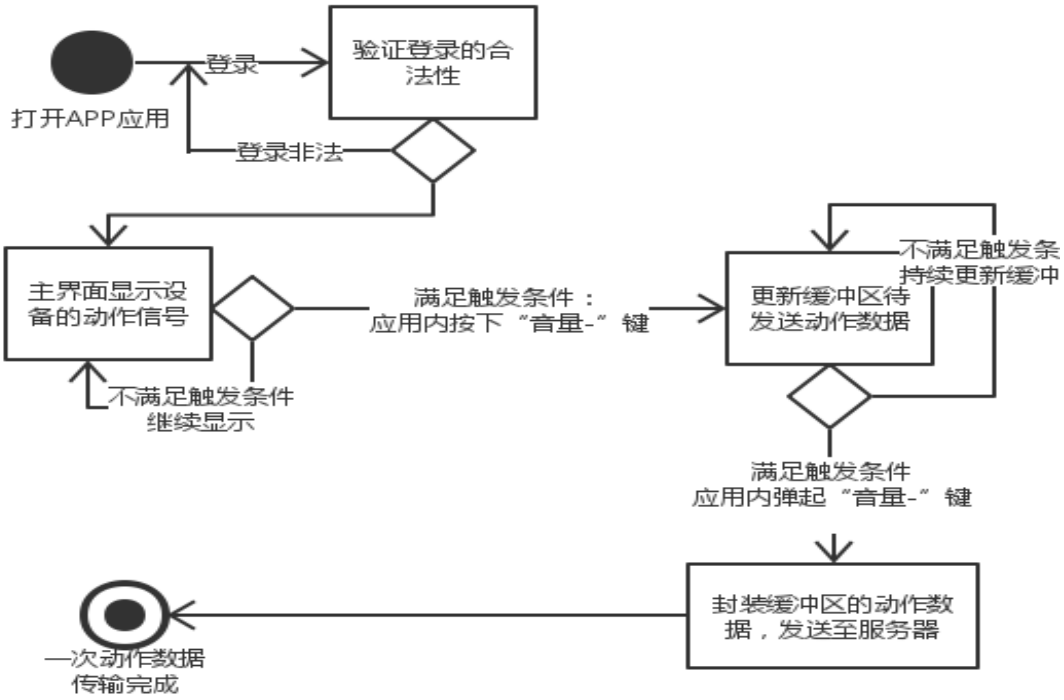


图 3.2 APP 系统活动图

其中登录注册模块主要提供了用户的登录和注册功能，服务器将会对不同用户建立不同的 SVM 分类器模型，响应不同用户的请求，并广播至不同用户的模拟智能家居网络，使智能家居的状态属性做相应的改变。

图 3.4 APP 系统功能界面样例图

数据采集模块是动作采集 APP 的核心模块，主要完成了对加速度传感器、角速度传感器和方位传感器模块调用，获取了设备某一时刻的状态信息，并通过网络请求的方式，设备的状态消息，对应着的是设备持有者的动作消息发送到了服务端，在服务端对数据进行了有效的存储和计算。

二、动作采集和传输方案

基于 SVM 分类器的动作识别系统，通过采集设备持有者的某一段时间的动作数据，处理和传输至服务器。本文搭建的动作识别系统的动作采集模块的设备采集 APP 有关动作采集和传输的说明如下。

首先每一次动作获取，采集到加速度传感器、陀螺仪、方向传感器三个模块的共 9 个动作数据，从中选取我们需要的 8 个数据，分别为：x 轴加速度（a_x）、y 轴加速度（a_y）、z 轴加速度（a_z）、x 轴和水平方向夹角（o_y）、y 轴和水平方向夹角（o_z）、x 轴角速度（g_x）、y 轴角速度（g_y）、z 轴加速度（g_z），通过这 8 个数据，可以完整还原出设备持有者某一时刻的三维动作模型[9]。

同时考虑到动作数据的时效性，只采集动作结束前的 2 秒时间内的动作数据，在这 2s 的时间内共采集 20 组数据，即数据采集频率 10 次/s。

在 APP 启动时，会初始化一个 20*8 的二维数组 ACTION_TEMP，用于缓存每次更新的动作数据，数据的排列和储存格式为：a_x, a_y, a_z, o_y, o_z, g_x, g_y, g_z。

通过按下智能手机的“音量-”按键触发，开始以 10 次/秒的频率更新动作数据，每次更新动作数据，都会在 ACTION_TEMP 中新产生一条动作数据，如果 ACTION_TEMP 数组的存储满后，会覆盖数组中存在时间最久的数据。当弹起智能手机的“音量-”按键时，会根据 ACTION_TEMP 的状态组装待发送的数据。

考虑到采集动作开始采集和结束采集这两个时间点的动作有可能会有较大的噪声，只采用弹起智能手机“音量-”按键前的 0.5s – 1.5s 的 1s 时间内的动作数据进行组装，组装格式成一个待发送的动作字符串，格式为：

a_x1, a_x2, … , a_x10,

a_y1, a_y2, … , a_y10,

a_z1, a_z2, … , a_z10,

o_y1, o_y2, … , o_y10,

o_z1, o_z2, … , o_z10,

g_x1, g_x2, … , g_x10,

g_y1, g_y2, … , g_y10,

g_z1, g_z2, …, g_z10

10 组数据共 80 个动作信号值，中间以逗号”,”分隔，将该组数据通过 HTTP 网络请求方式，发送至服务器。

第四节 本章小结

本章首先对动作采集模块的主体功能进行了概述，而后通过对动作采集的三个特征进行论述，引出了动作采集 APP 的设计和实现方案，主要介绍了动作采集 APP 的功能和动作采集和传输方案。

第四章 动作识别模块设计与实现

本章将对动作识别模块的设计与实现做讲解，详细介绍了动作识别模块的后台架构和系统运行流程，对 SVM 算法预测的过程也做了详细介绍，主要包括数据预处理流程和动作预测流程等。

第一节 动作识别模块概述

本文构建了一个基于 SVM 分类器的动作识别系统，通过对动作采集模块采集到的动作数据进行有效的接收和储存，而后对数据进行噪声处理、归一化等预处理过程，并利用改进的 SVM 多分类算法对动作数据进行分类器的训练。通过实践，本文搭建的 SVM 分类器的动作识别系统，动作识别准确率达到 94.03%。

下面将就动作识别系统构建过程中的动作识别模块的架构和功能方面做如下概述。

一、后台架构

动作识别系统后台需要为该系统的用户提供稳定可靠、高效便捷的后台服务，据此，系统技术选型如下：

编程语言：**JAVA**

对象生命周期管理：**Spring**

数据持久化框架：**Hibernate**

数据库：**MySQL**

WEB 容器：**Tomcat**

后台系统采用 JAVA 语言编写完成，使用 Spring 框架作为系统对象生命周期的管理，并采用 SpringMVC 框架编写具有 RESTful 风格的接口，使动作采集 APP 和智能家居模拟模块能够通过 HTTP 服务调用的方式管理后台资源。系统运行在 Tomcat 容器中，提供高可用性和并发访问的服务。

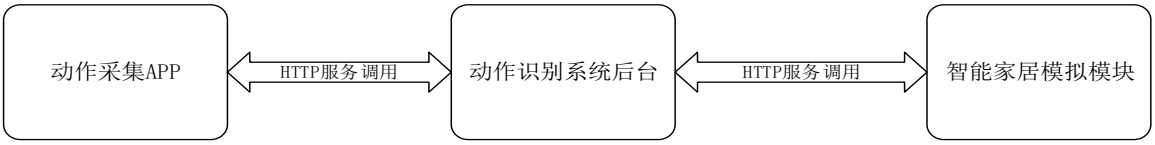


图 4.1 后台服务调用示意图

二、系统运行流程

动作识别的动作学习过程功能流程图如下：

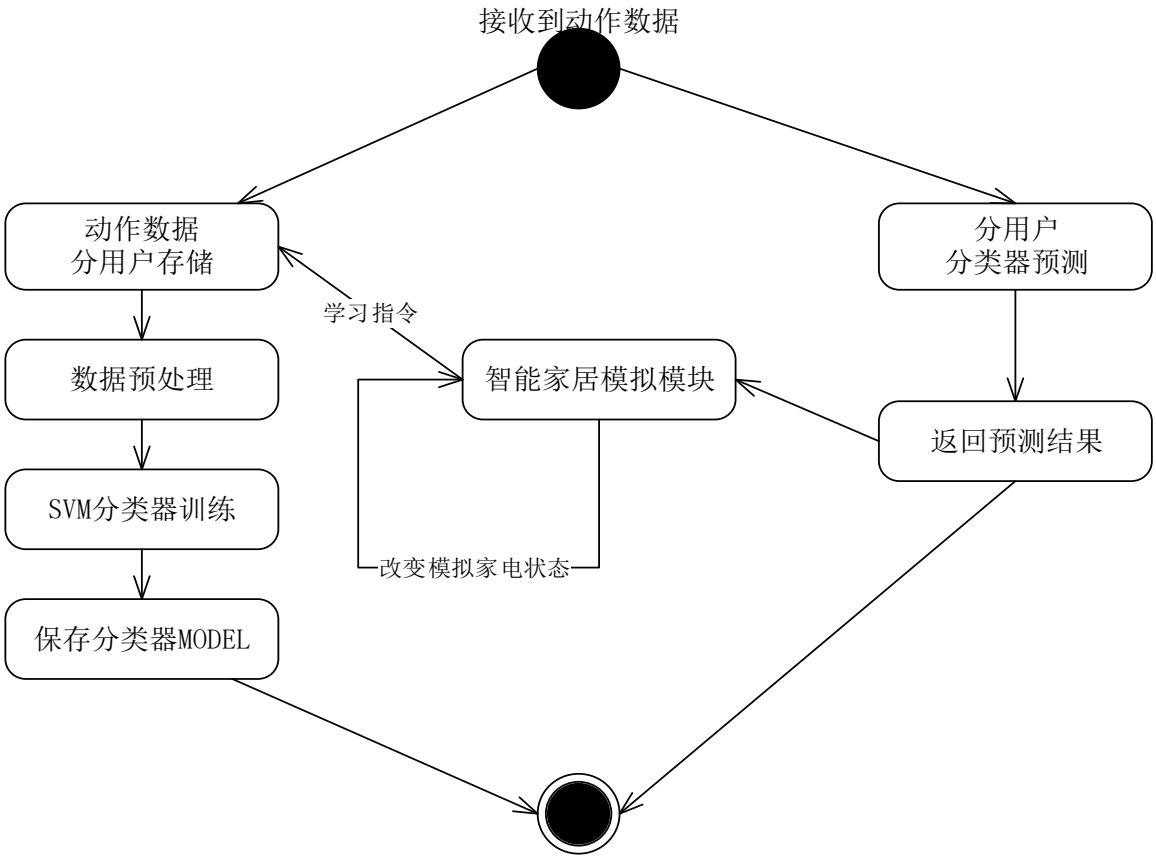


图 4.2 动作识别系统活动图

如图 4.2 所示，在接收到通过网络发送过来的动作数据后，会根据目前登录的用户，找到该用户对应的已经训练好的 SVM 分类器模型，通过将接受到的动作数据输入到分类器模型中，得到预测值，即表示了该动作对应的用户特定意图。

如接受到模拟智能家居模块发来的学习指令，会激活 SVM 分类器模型的更新学习流程，对该用户的 SVM 分类器模型进行再学习和更新。

整个 SVM 动作分类器的动作识别模块包含：动作学习模块、动作预测模块、智能家居模拟模块。

第二节 数据预处理

算法效果的好坏程度很大程度上受限于数据的预处理的效果，常用的数据预处理方法有归一化处理。

在使用传感器采集动作信息时，由于人体动作的幅度不可能每次都相同，因此采集到的动作信号值范围相差很大,这样采集到的信号在进行手势识别时难以区分。而且由第二章的分析可知，输入数值太大，在进行 SVM 分类器计算时会影响识别速度。因此我们将数据进行归一化处理，把需要处理的数据经过处理后限制在需要的一定范围内，使它们的数值更加规范，具有更明显的可比性，以此来降低识别难度，提高识别速度。 本系统中为了便于数据的运算将数据归一化到[0-1]的范围内。通过分析可知本文采集的动作特征具有一定的取值范围，可针对数据的不同取值范围做数据的归一化处理[10,11]。

针对加速度传感器的数据，在应用在智能家居控制的场景下，用户正常使用，通过对大量动作数据的观测发现，用户的 ax 动作值得大小分布如图所示：

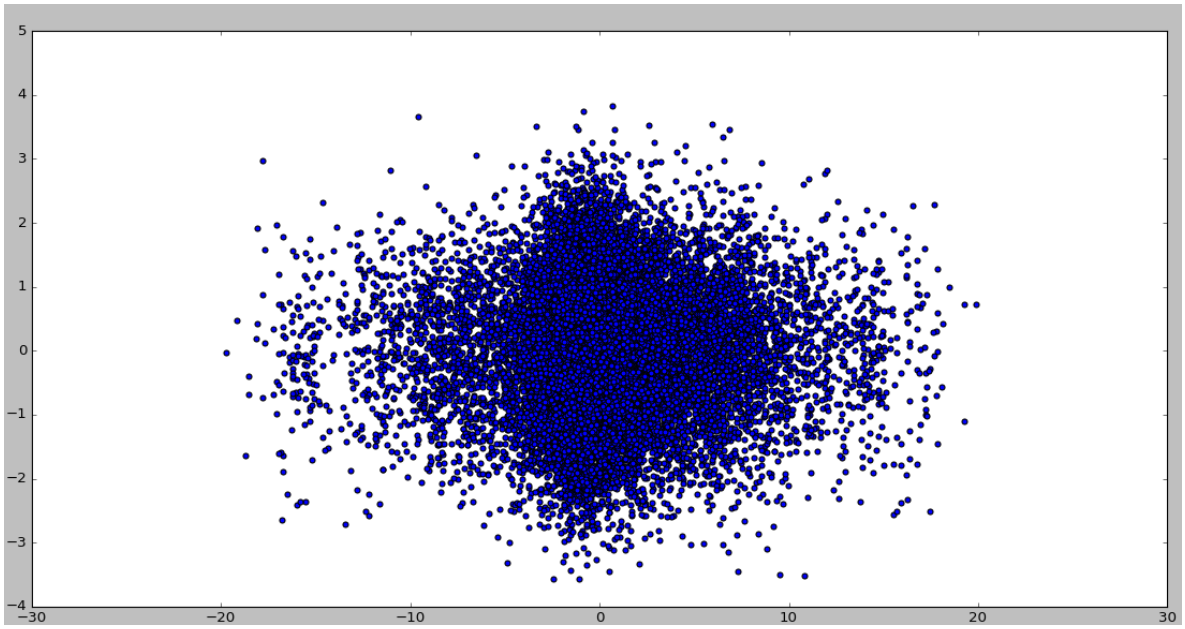


图 4.3 ax 数据分布散点图

由上图可得ax ∈ [−20, 20]，同理可得ay ∈ [−25, 25]，az ∈ [−30, 30]。

因此选取动作数据的取值范围为：

a ∈ $[a_{min}, a_{max}]$ ，$a_{min} = 10.0, a_{max} = 10.0$，其中 a 表示加速度传感器的值的

大小，即 ax, ay, az；

g ∈ $[g_{min}, g_{max}]$ ， $g_{min} = 10.0, g_{max} = 10.0$，其中 g 表示陀螺仪的值的大小，即 gx, gy, gz；

oy ∈ $[oy_{min}, oy_{max}]$ ， $oy_{min} = -180^{\circ}, oy_{max} = 180^{\circ}$；

oz ∈ $[oz_{min}, oz_{max}]$ ， $oz_{min} = -90^{\circ}, oz_{max} = 90^{\circ}$；

按照如下公式对数据进行归一化：

$$data_{normalization} = (data - data_{min}) / (data_{max} - data_{\min})$$

动作接收模块

动作接收模块，主要提供接收动作数据的功能。

服务器会监听是否有新的动作数据从动作采集 APP 发送至服务器。该过程通过服务器后台提供一个动作接收的 HTTP 服务调用接口的方式实现。当接收到从动作采集 APP 发送的数据后，服务器会执行如下流程：

1. 将动作数据进行预处理；
2. 更新该用户的待预测动作属性 TO_PREDICT, 待学习动作属性 TO_LEARN 的值（动作预测模块将会获取 TO_PREDICT 的值作为 SVM 动作分类器模型的输入，动作学习模块将会在接受到动作学习指令后，获取 TO_LEARN 的值作为新模型训练的训练集）；

第三节 动作预测模块

当后台系统监听到有新的动作数据从动作采集 APP 发送达到的时候，会启动动作预测的流程：

1. 查找该用户的 SVM 分类器模型 Model；
2. 将待预测数据 TO_PREDICT 作为输入，根据 Model 得出预测值；
3. 将预测值添加到动作预测缓存队列中，等待智能家居模拟模块的获取。

通过这种方式，使动作预测模块能够对每一个接受到的新动作数据进行预测，并得出具体的预测值，并将预测值的信息传递到智能家居模拟模块中，改变智能家居模块的状态属性。

第四节 动作学习模块

动作学习模块是基于 SVM 分类器的动作识别系统的核心模块，其提供的主要功能包括：对动作数据进行预处理、根据动作数据利用 SVM 分类器算法得出分类模型。

动作学习模块会监听智能家居模拟模块，当接收到学习指令后，会执行如下的动作学习流程：

1. 接收到的学习指令的含义为某家电的某状态，拼接动作学习数据和动作意图（例如电灯 1 的开启状态对应的标签值为 1，动作学习模块接收到学习指令 1，表示此时该用户对应的最新动作数据 TO_LEARN 对应的动作意图为电灯 1 开启状态）；
2. 将 TO_LEARN 数据持久化到数据库中；
3. 从数据库中获取该用户的所有学习动作数据，将其作为训练集，按照第二章得到的 SVM 算法，对该用户的 SVM 分类器模型进行训练；
4. 将新的分类器模型保存为文件，替换旧的模型。

通过以上步骤，实现了对每一次学习指令的响应，更新学习得到的最新的动作数据，并替换旧的分类器模型。实现了旧模型的替换，使新的预测动作作用在新的分类模型上。

第五节 本章小结

本章主要论述了动作识别模块的相关功能设计和实现，首先介绍了模块的后台架构方式和系统运行流程，而后对 SVM 分类算法运行过程的步骤进行了详细介绍，主要包括：动作预处理过程、动作预测过程和动作学习过程。本章是本文搭建的基于 SVM 分类器的动作识别系统的算法核心模块。

第五章 智能家居模拟模块的设计与实现

本章将介绍 SVM 分类器的动作识别系统的智能家居模拟模块的设计与实现，分别介绍了模拟学习模块和模拟现实模块的相应功能组件。

第一节 模拟学习模块

本文构建的基于 SVM 分类器的动作识别系统，是一个从动作采集，到动作数据的传输和存储系统，并通过改进的 SVM 分类算法，训练得到 SVM 分类器，对新的动作数据输入能够有较好的分类效果。

并结合时下智能家居概念，将该动作识别系统应用到了智能家居的控制模块中，实现了一种新型的控制方式，改变了传统的基于遥控器或者手机 APP 点按的控制方式，对用户的手势动作进行动作识别，得到其特定的动作意图，以此来控制智能家居做出相应的状态和属性的改变。本节将构建一个智能家居模拟模块，通过软件模拟了实体智能家居的各种属性和状态，可以发送学习指令到动作识别模块，接收动作识别模块发送的控制命令并改变其状态和属性。

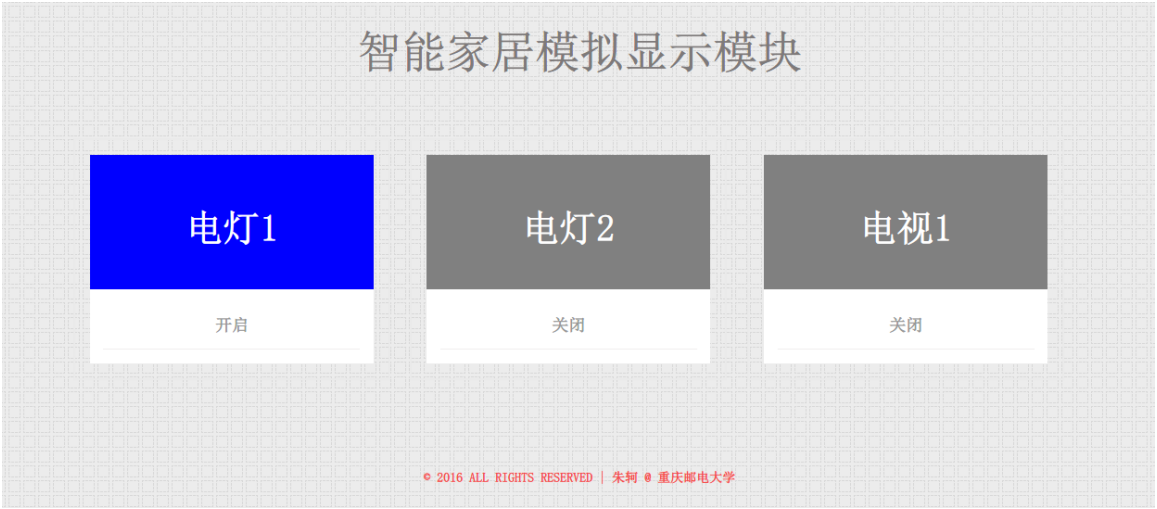
模拟学习模块可以向动作识别模块发送学习指令。其主界面如下图所示。

主界面包含一系列的控制选项，开始预测和停止预测作为一个开关选项，控制后台服务是否对新动作数据进行实时预测并返回。

学习列表包含电灯 1、电灯 2、电视 1 等模拟智能家电，并以按钮的形式表名了该家电的状态，点击学习列表的某一按钮，向服务器发送一条学习指令，服务器的动作学习模块，会将最新的动作数据 TO_PREDICT 和该学习指令对应起来，作为一条新的 SVM 分类器训练数据，服务器会将新训练得到的 SVM 分类器模型替换旧模型。

第二节 模拟显示模块

模拟显示模块能接收动作识别模块对于新动作的预测结果，并更新该家电的状态显示。其主界面如图所示。



模拟显示模块通过轮询的方式和服务器进行通信，每 0.5 秒向服务器发起请求获取是否有新的控制消息，如果得到有新的控制命令，则更新显示模块中智能家电的状态属性。

第三节 本章小结

本章主要介绍了智能家居模拟模块的设计与实现，该模拟模块主要包括：模拟学习模块和模拟显示模块，详细阐述了通过软件模拟智能家居的各种行为的设计和技术细节。

第六章 结　论

通过本文的一至五章的论述后，本章将对基于 SVM 分类器的动作识别系统做下一个结论性的论述，主要包括：对 SVM 于其他分类算法在次场景下的分类情况做比较和对课题做总结与展望。

第一节 其他分类算法效果比较

本文搭建的动作识别系统，采用 SVM 算法进行分类，得到了较好的//TODO 效果。本节将使用相同的动作数据集，对 SVM 和 K 临近算法、神经网络进行对比和分析。

一、K 临近算法

K 临近算法(K-Nearest Neighbor, KNN)是最简单的机器学习算法之一。其分类依据是，　如果一个样本在特征空间中的 K 个最相似(即特征空间中的最近邻)的样本中的大多数属于某一个类别，则该样本也属于这个类别。

使用相同的训练数据集和测试数据集训练 KNN 分类器，本节通过 sklearn 机器学习代码库编写测试代码，其 Python 源码详见附录。

实验得到最终的分类准确度为 **85.58%**；

二、神经网络算法

神经网络算法(Neural Network Algorithm)是机器模拟人类思维方式的一种学习方式。算法信息储存在各个神经元中，信息的分布式存储带来的好处是，各个神经元可以并行地计算处理，从而使算法的运行过程高效执行[16]。

通过实验可得，神经网络算法的分类准确度为 **89.78%**。

三、结论

本文搭建的动作识别系统，用户需要在只经过极少的几次训练之后就得到准确分类模型（小样本）；用户每次的数据往往都和上次数据具有一定差别，但是又表示

的是同一意图，动作数据在特征空间上表现出线性不可分，和低维下分类效果不佳的情况，通过上节对 KNN 和神经网络的分类效果比较可得知，SVM 算法在动作识别上具有较好的效果。

第二节 总结与展望

一、总结

本文在综合介绍和了解分析动作识别领域和 SVM 算法的背景下，提出并实现了通过调用智能手机各动作传感器来采集用户的动作信号，并使用 SVM 算法分析动作信号，最终搭建完成了一个基于 SVM 分类器的动作识别系统。

为了验证该基于 SVM 分类器的动作识别系统的应用前景，结合智能家居概念，将该动作识别系统应用到智能家居的控制模块，通过识别用户特定动作的意图，从而控制智能家电状态。实验证明，该方案提供了一种新颖的家电控制方式，并取得了较好的实验结果。

最后比较了 SVM 算法和其他常见分类算法，如 KNN、神经网络的分类效果，证明了 SVM 算法在小样本情况下的分类效果较为优秀。

二、展望

本文提出的动作识别系统虽然具有较高的识别准确率 94.03%，但是仍然有一定的优化空间。

用户动作数据的不精确，和动作采集过程中的误差，会影响分类效果的准确性。为了提高分类准确性，可以在数据预处理环节对误差进行进一步的处理。

本文通过将动作识别系统和智能家居概念结合，找到了一个动作识别系统的一种应用途径。但是本文提出的结合方案还是比较初级的，仅仅是将其应用到了核心控制方式的改变上，可以在后期通过对应用场景的深入挖掘，从而对动作识别系统的应用场景有更加广阔的扩展。

第三节 本章小结

本章主要比较了几种分类算法和 SVM 算法在本场景下的分类效果，并对本文

所研究课题进行了总结，并对未来进行了一定的展望。

致　谢

四年的大学生活即将结束，在学校学习和生活的时间里，我得到了各位的极大帮助与支持，在论文完成之际，我想借此机会表达我对你们的感谢。

首先感谢我的指导老师邓欣副教授，本文是在老师的精心指导下完成的，从选题到论文框架和行文思路的确定，再到后期的细节修改，老师都给我提出了宝贵且极具指导意义的意见，使我受益良多。是老师的细致指导，才使本文得以顺利完成。除此之外，老师对待生活幽默乐观和聪明睿智的态度，对待学术的大胆创新和孜孜不倦的作风，都对我的成长产生了深远的影响。老师始终以他平易近人的样子，用渊博的知识，对我进行学术和人生态度上的指导。

除此之外，这篇论文是从我们参加“挑战杯”课外学术科技竞赛的项目中分离一个部分，用来作为本文的主要研究方向，感谢前期在参加“挑战杯”的过程中基于支持和鼓励的参赛同伴们：张竞成、陈果、李东奇、邹亚琳等等。是大家的通力协作，才使“挑战杯”项目得以顺利完成，也才造就了我这篇论文的原型，谢谢曾经一起奋斗的伙伴们。

感谢所有给予我关心、帮助和支持的老师、同学和朋友们，谢谢你们。

最后，感谢各位老师在百忙之中抽身评阅本文，谢谢你们。

参考文献

[1] 杨石焕. 基于支持向量机的手势识别研究[D]. 河北省:燕山大学信息科学与工程学院, 2014.

[2] 张倩；杨耀权. 基于支持向量机核函数的研究[J]. 电力科学与工程, 2012, 28(5): 43-45.

[3] 谌璐. PSO-SVM 学习算法及其在空间数据分析中的应用[D].西安工程大学,2012.

[4] 张妤. 支持向量机集成学习方法研究[D].山西大学,2008.

[5] 蔡婕. 支持向量机训练算法的研究[D].西安理工大学,2009.

[6] 王昌喜, 杨先军等. 基于三维加速度传感器的上肢动作识别系统[J]. 传感器技术学报, 2010, 23(6): 816-819.

[7] 王见, 陈义, 邓帅. 基于改进 SVM 分类器的动作识别方法[J]. 重庆大学学报, 2016, 39(1): 13-17.

[8] 蔡美玲. 三维人体运动分析与动作识别方法[D]. 湖南省:中南大学信息科学与工程学院, 2013.

[9] Sushmita M, Tinku A. Gesture Recognition[J]. A Survey IEEE Transactions on System, Man And Cybemetics, 2007, 37(3): 311-324.

[10] 刘为, 高尚. 一种新条件下的三次样条插值[J]. 信息技术, 2011,87(8): 23-24.

[11] 潘泉, 孟晋丽. 小波滤波方法及应用[J]. 电子与信息学报, 2007, 29(1): 236-239.

[12] 白鹏, 张喜斌, 张斌.支持向量机理论及工程应用实例[M]. 西安: 西安电子科技大学出版社, 2008: 5-46.

[13] 任洪娥, 霍满冬. 基于 PSO 优化的 SVM 预测应用研究[J]. 计算机应用研究, 2009, 26(3): 867-869.

[14] Bingquan, Shen, Jinfu, Li, Fengjun, Bai, and, Chee-Meng, Chew, “Motion Intent Recognition for Control of a Lower Extremity Assistive Device (LEAD)”, IEEE International Conference on Mechatronics ans Automation, Tapan, 2013, pp. 926-931;

[15] 聂云. 基于 Android 的 WoT 物联网商城终端应用的设计与开发[D].北京邮电大学,2014.

[16] 王东. 人工神经网络理论及其在泥沙科学中的应用研究[D].四川大学,2003.

附 录

一、 英文原文

Support vector machine for classification based on fuzzy training data

Ai-bing Ji, Jia-hong Pang, Hong-jie Qiu

ABSTRACT

Support vector machines (SVM) have been very successful in pattern recognition and function estimation problems, but in the support vector machines for classification, the training example is non-fuzzy input and output is y = ±1; In this paper, we introduce the support vector machine which the training examples are fuzzy input, and give some solving procedure of the Support vector machine with fuzzy training data.

Introduction

The support vector machine (SVM) is a training algorithm for learning classification and regression rules from data, SVMS were first introduced by Vapnik. SVM is based on the structural risk minimization principle, this principle incorporates capacity control to prevent over-fitting and thus is a partial solution to the bias-variance trade-off dilemma.

In the support vector machine for classification, the training example is non-fuzzy input and the output is y = ±1. Considering the noisy in the training example set, fuzzy membership was introduced in classification by Chen and Chen, 2002, Tsujinishi and Abe, 2003 and Kikuchi and Abe, 2005 and Lin and Wang (2004) introduced the fuzzy support vector machine, it used the membership function to express the membership grade of an example to positive class or negative class. But in nature, it is still a common support vector machine of Vapnik.

In fact, because the noisy and error of measurement, the training examples are usually uncertain (Jeng, Chuang, & Su, 2003) or fuzzy. Then the study of support vector machine with fuzzy training data is very significant.

In this paper, we first give some preliminary knowledge, then for fuzzy training data, introduce the concept of fuzzy linear separable and approximately fuzzy linear separable. At last, we systematically study the support vector machine for two-class classification with fuzzy training data.

Preliminary

Here we focus on SVM for two-class classification, for the training sample (x1,y1),(x2,y2),…,(xk,yk) ∈ Rn × {±1}, yi = +1, −1 represent positive class and negative class respectively. The geometrical interpretation of support vector classification (SVC) is that the algorithm searches for the optimal separating hyperplane, SVM is outlined first for the linearly separable case.

The training data are linearly separable, if there exists a pair (w,b) such that

$$\mathbf{w}^T\mathbf{x}_i + b \geqslant 1, \text{ for all } y_i = +1$$
$$\mathbf{w}^T\mathbf{x}_i + b \leqslant -1, \text{ for all } y_i = -1 \quad (1)$$

with the decision rule given by

f**w**,b(**x**)=sgn(**w**T**x**+b) (2)

w is termed the weight vector and *b* is the bias (or −*b* is termed the threshold). The inequality constraints (1) can be combined to give

yi(**w**T**x**i+b)⩾1 (3)

The learning problem is hence reformulated as the convex quadratic programming (QP) problem

$$\underset{\mathbf{w},b}{\text{Minimize}}\ \Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1, \quad i = 1, \ldots, l. \quad (4)$$

This problem has a global optimum, and its dual problem is to maximize the objective function

equation(5)

$$Q(\lambda) = \sum_{i=1}^{l}\lambda_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\lambda_i\lambda_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \quad (5)$$

subject to the constraints

$$\begin{cases} \sum_{i=1}^{l}\lambda_i y_i = 0 \\ \lambda_i \geqslant 0, \end{cases} \quad (6)$$

The decision function is then given by

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i\lambda_i^*\mathbf{x}^T\mathbf{x}_i + b^*\right). \quad (7)$$

The SVM can be used to learn non-linear decision functions by first mapping the data to some higher dimensional *feature space* and constructing a separating hyperplane in this space. Denoting the mapping to feature space by

$$X \to H$$

$$\mathbf{x} \mapsto \phi(\mathbf{x})$$

$K(\mathbf{x},\mathbf{z}) \equiv \phi(\mathbf{x})T\phi(\mathbf{z})$ is kernel function. To take account of the fact that some data points may be misclassified, we introduce a vector of slack variables $\xi = (\xi 1,\ldots,\xi l)T$ that measure the amount of violation of the constraints (3). The problem can then be written

$$\underset{\mathbf{w},b,\Xi}{\text{Minimize}}\ \Phi(\mathbf{w},b,\Xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i$$

$$\text{s.t. } y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \geqslant 1-\xi_i, \xi_i \geqslant 0, \quad i=1,\ldots,l \tag{8}$$

where C is specified beforehand. C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the training error term. If C is too small, then insufficient stress will be placed on fitting the training data. If C is too large, then the algorithm will over-fit the training data.

The dual problem of (8) is to maximize the objective function

$$Q(\lambda) = \sum_{i=1}^{l}\lambda_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\lambda_i\lambda_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) \tag{9}$$

subject to the constraints

$$\begin{cases} \sum_{i=1}^{l}\lambda_i y_i = 0 \\ 0 \leqslant \lambda_i \leqslant C, \end{cases}$$

where C is a user-specified positive parameter. The decision functions become

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{l} y_i\lambda_i^* K(\mathbf{x},\mathbf{x}_i) + b^*\right)$$

where the bias is given by

$$b^* = y_i - \mathbf{w}^{*T}\phi(\mathbf{x}_i) = y_i - \sum_{j=1}^{l} y_j\lambda_i^* K(\mathbf{x}_j,\mathbf{x}_i) \tag{10}$$

for any support vector $\mathbf{x}i$.

Possibility measure and fuzzy chance constrained programming

Definition 3.1 Dubois and Prade, 1988 **and** Klir, 1999**.**

Let X be a nonempty set, $P(X)$ be the class of all subsets of X, a mapping Pos: $P(X) \to [0,1]$ is called a possibility measure if it satisfies:

(1) $\text{Pos}(\phi) = 0$

(2) Pos(X) = 1

(3) Pos(⋃t∈TAt)=Supt∈TPos(At)

Definition 3.2.

Let a˜ be a fuzzy number, and its membership function is

Theorem 3.1 Liu and Liu, 2003**.**

Let a˜=(r1,r2,r3),b˜=(t1,t2,t3)*be two triangular fuzzy numbers, ρ be a real number, then*

(1)

a˜+b˜=(r1+t1,r2+t2,r3+t3);

(2)

ρa˜={(ρr1,ρr2,ρr3),ρ⩾0(ρr3,ρr2,ρr1),ρ<0

Theorem 3.2.

Let a˜=(r1,r2,r3)*be a triangular fuzzy number, then*

$$\mu_{\tilde{a}}(x) - \begin{cases} \frac{x-r_1}{r_2-r_1}, & r_1 \leqslant x < r_2 \\ 1 & x - r_2 \\ \frac{x-r_3}{r_2-r_3}, & r_2 < x \leqslant r_3 \end{cases}$$

Example

In the following, we shall apply the support vector machine for two-class classification with fuzzy training data in the diagnosis of Coronary. The data in Table 1 are the diastolic pressure xi1 and plasma cholesterol xi2 of 24 persons, where half of them are healthy (yi = 1), the others are Coronary patients(yi = -1), ~xi1 and ~xi2 are triangular fuzzy numbers. Based on the fuzzy training data given in Table 1, when parameter C=0.1, k = 0.65, solving the programming (16), we can obtain w0 = (0.415444,0.4792959),b = -0.6962587, then the decision rule is: for a given confidence level k = 0.65, If Pos{(w0X˜+b)>=0}>=0.65, then X˜=(x1˜, x2˜) is a positive example; If Pos{(w0X˜+b)<=0}>=0.65, then X˜=(x1˜, x2˜) is a negative example. Using this decision rule to fit the data in Table1, only three fuzzy examples are misclassification.

table 1:

i	$\tilde{x}_{i1}$ (KPa)	$\tilde{x}_{i2}$ (mmol/L)	y_i	i	$\tilde{x}_{i1}$	$\tilde{x}_{i2}$	y_i
1	(9.84,9.86,9.88)	(5.17,5.18,5.19)	1	13	(10.62,10.66,10.70)	(2.06,2.07,2.08)	−1
2	(13.31,13.33,13.35)	(3.72,3.73,3.74)	1	14	(12.51,12.53,12.55)	(4.44,4.45,4.46)	−1
3	(14.63,14.66,14.69)	(3.87,3.89,3.91)	1	15	(13.30,13.33,13.36)	(3.04,3.06,3.08)	−1
4	(9.32,9.33,9.34)	(7.08,7.10,7.12)	1	16	(9.32,9.33,9.34)	(3.90,3.94,3.98)	−1
5	(12.87,12.80,12.83)	(5.47,5.49,5.51)	1	17	(10.64,10.66,10.68)	(4.43,4.45,4.47)	−1
6	(10.64,10.66,10.68)	(4.06,4.09,4.12)	1	18	(10.64,10.66,10.68)	(4.89,4.92,4.95)	−1
7	(10.65,10.66,10.67)	(4.43,4.45,4.47)	1	19	(9.31,9.33,9.35)	(3.66,3.68,3.70)	−1
8	(13.31,13.33,13.35)	(3.60,3.63,3.66)	1	20	(10.64,10.66,10.68)	(3.20,3.21,3.22)	−1
9	(13.32,13.33,13.34)	(5.68,5.70,5.72)	1	21	(10.37,10.40,10.43)	(3.92,3.94,3.96)	−1
10	(11.97,12.00,12.03)	(6.17,6.19,6.21)	1	22	(9.31,9.33,9.35)	(4.90,4.92,4.94)	−1
11	(14.64,14.66,14.68)	(4.00,4.01,4.02)	1	23	(11.19,11.20,11.21)	(3.40,3.42,3.44)	−1
12	(13.31,13.33,13.35)	(3.99,4.01,4.03)	1	24	(9.31,9.33,9.35)	(3.62,3.63,3.64)	−1

Conclusions

This paper discusses the support vector machine which the training example is fuzzy input, and give some solution procedure of the Support vector machine with fuzzy training data, it is a generalization of support vector machine of V.N. Vapnik. In the further study, we are to discuss the support vector regression machine based on the fuzzy input and fuzzy output.

二、 英文翻译

基于模糊训练集的支持向量机分类算法

Ai-bing Ji, Jia-hong Pang, Hong-jie Qiu

摘要

支持向量机(SVM)已经在模式识别领域取得了巨大的成功，在将其用于分类时，如果训练样本的输入数据是确定的，会得到确定的分类结果(y＝±1)。本文介绍了当训练样本输入是模糊的情况，给出了几种该情况下构建 SVM 分类模型的解决办法。

简介

支持向量机是一种根据对输入数据进行训练学习得到分类和回归模型的算法，SVM 在 1995 年被 Vapnik 首次提出。SVM 基于结构风险最小化原则建立，可以有效防止过度拟合问题。

在支持向量机分类问题中，确定的训练样本输入可以得到确定的分类结果。考虑到训练样本集的噪声问题，Chen 在 2002 年建议对训练样本集中的模糊数据进行

考虑和研究。Tsujinishi 和 Abe 提出模糊支持向量机，运用隶属函数表示成员正负分类的等级。但是实际上，还是普通的支持向量机。

事实上，因为测量过程中的噪声或错误数据，训练数据集往往是不确定的或者是模糊的，所以基于模糊训练集的支持向量机才具有十分重大的意义。

本文首先介绍了一些基本知识，然后介绍了模糊线性分割和近似模糊线性分割的概念。最终系统学习模糊训练数据情况下的二分类支持向量机算法。

准备

这里我们讨论二分类支持向量机算法，(x1,y1),(x2,y2),…,(xk,yk) ∈ Rn × {±1} 表示训练数据集，yi = +1, −1 各自表示正负分类结果。对支持向量机的集合解释为：在搜索空间内寻找现行最优超平面。

假设训练数据线性可分，存在(w, b)满足：

$$\mathbf{w}^T\mathbf{x}_i + b \geqslant 1, \text{ for all } y_i = +1$$

$$\mathbf{w}^T\mathbf{x}_i + b \leqslant -1, \text{ for all } y_i = -1 \quad (1)$$

得到判别函数的形式如下：

$$f_{\mathbf{w},b}(\mathbf{x}) = \operatorname{sgn}(\mathbf{w}^T\mathbf{x} + b) \quad (2)$$

可以对(1)添加如下约束：

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 \quad (3)$$

所以 SVM 学习问题重新规划为：

$$\underset{\mathbf{w},b}{\text{Minimize}}\ \Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1, \quad i = 1, \ldots, l. \quad (4)$$

上面的全局优化问题可以转化为：

$$Q(\lambda) = \sum_{i=1}^{l}\lambda_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\lambda_i\lambda_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \quad (5)$$

且满足条件：

$$\begin{cases} \sum_{i=1}^{l}\lambda_i y_i = 0 \\ \lambda_i \geqslant 0, \end{cases} \quad (6)$$

所以给出判别函数如下：

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i\lambda_i^*\mathbf{x}^T\mathbf{x}_i + b^*\right). \quad (7)$$

SVM 可以学习非线性可分的数据集，首先通过核函数将数据映射到高维空间中，在高位空间中寻找最优超平面。.

$$X \to H$$

$$\mathbf{x} \mapsto \phi(\mathbf{x})$$

$K(\mathbf{x},\mathbf{z}) \equiv \phi(\mathbf{x})T\phi(\mathbf{z})$作为 SVM 的核函数. 某些数据集有可能会被错误分类，我们引入松弛变量 $\xi = (\xi 1,\ldots,\xi l)$来表示算法对误差的接受程度。上面的分类问题可以转化为：

$$\underset{\mathbf{w},b,\Xi}{\text{Minimize}}\ \Phi(\mathbf{w},b,\Xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i$$

$$\text{s.t. } y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \geqslant 1-\xi_i, \xi_i \geqslant 0, \quad i=1,\ldots,l \tag{8}$$

C 表示惩罚系数，控制算法对误差的接受程度。如果 C 过大，算法将会存在过度拟合的现象。

得到(8)的问题就是要求得下式的最大值：

$$Q(\lambda) = \sum_{i=1}^{l}\lambda_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\lambda_i\lambda_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) \tag{9}$$

且满足：

$$\begin{cases}\sum_{i=1}^{l}\lambda_i y_i = 0\\ 0 \leqslant \lambda_i \leqslant C,\end{cases}$$

得到判别函数的形式如下：

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i\lambda_i^* K(\mathbf{x},\mathbf{x}_i) + b^*\right)$$

$$b^* = y_i - \mathbf{w}^{*T}\phi(\mathbf{x}_i) = y_i - \sum_{j=1}^{l} y_j\lambda_i^* K(\mathbf{x}_j,\mathbf{x}_i) \tag{10}$$

可能性测量和模糊机会约束

定义 3.1

X 是一个非空集合，P(X)是 X 的子集，如果如下条件 Pos：P(X) → [0,1]的映射关系，就称 Pos 为可能性测量：

(1) Pos(ϕ) = 0

(2) Pos(X) = 1

(3) Pos(∪t∈TAt)=Supt∈TPos(At)

定义 3.2

假设 a˜是一个随即量，其隶属函数为：

$$\mu_{\tilde{a}}(x) - \begin{cases} \frac{x-r_1}{r_2-r_1}, & r_1 \leqslant x < r_2 \\ 1 & x - r_2 \\ \frac{x-r_3}{r_2-r_3}, & r_2 < x \leqslant r_3 \end{cases}$$

定理 3.1

假设 a˜=(r1,r2,r3),b˜=(t1,t2,t3)是两个三元随即量, ρ 为一个实数,

a˜+b˜=(r1+t1,r2+t2,r3+t3);

ρa˜={(ρr1,ρr2,ρr3),ρ⩾0(ρr3,ρr2,ρr1),ρ<0

实验

在本节中，我们将对模糊数据训练集进行 SVM 二分类训练。表 1 中的数据是 24 个实验个体的血压舒张压 xi1 和收缩压 xi2，其中半数为健康(yi=1)，另一半为冠心病患者(yi = -1)，xi1 和 xi2 是模糊量。 根据表 1 给出的模糊训练集数据，确定 SVM 算法参数为：C=0.1，k=0.65，通过上节的推导可得出判别函数的参数值为：w0 = (0.415444,0.4792959),b = -0.6962587，分类判定规则为：给定 k = 0.65，如果{(w0X˜+b)>=0}>=0.65 ， 得 到 X˜=(x1˜, x2˜) 是 一 个 负 分 类 (y=-1) ；{(w0X˜+b)<=0}>=0.65，得到 X˜=(x1˜, x2˜)是一个正分类(y=1)。

表 1

i	$\tilde{x}_{i1}$ (KPa)	$\tilde{x}_{i2}$ (mmol/L)	y_i	i	$\tilde{x}_{i1}$	$\tilde{x}_{i2}$	y_i
1	(9.84,9.86,9.88)	(5.17,5.18,5.19)	1	13	(10.62,10.66,10.70)	(2.06,2.07,2.08)	−1
2	(13.31,13.33,13.35)	(3.72,3.73,3.74)	1	14	(12.51,12.53,12.55)	(4.44,4.45,4.46)	−1
3	(14.63,14.66,14.69)	(3.87,3.89,3.91)	1	15	(13.30,13.33,13.36)	(3.04,3.06,3.08)	−1
4	(9.32,9.33,9.34)	(7.08,7.10,7.12)	1	16	(9.32,9.33,9.34)	(3.90,3.94,3.98)	−1
5	(12.87,12.80,12.83)	(5.47,5.49,5.51)	1	17	(10.64,10.66,10.68)	(4.43,4.45,4.47)	−1
6	(10.64,10.66,10.68)	(4.06,4.09,4.12)	1	18	(10.64,10.66,10.68)	(4.89,4.92,4.95)	−1
7	(10.65,10.66,10.67)	(4.43,4.45,4.47)	1	19	(9.31,9.33,9.35)	(3.66,3.68,3.70)	−1
8	(13.31,13.33,13.35)	(3.60,3.63,3.66)	1	20	(10.64,10.66,10.68)	(3.20,3.21,3.22)	−1
9	(13.32,13.33,13.34)	(5.68,5.70,5.72)	1	21	(10.37,10.40,10.43)	(3.92,3.94,3.96)	−1
10	(11.97,12.00,12.03)	(6.17,6.19,6.21)	1	22	(9.31,9.33,9.35)	(4.90,4.92,4.94)	−1
11	(14.64,14.66,14.68)	(4.00,4.01,4.02)	1	23	(11.19,11.20,11.21)	(3.40,3.42,3.44)	−1
12	(13.31,13.33,13.35)	(3.99,4.01,4.03)	1	24	(9.31,9.33,9.35)	(3.62,3.63,3.64)	−1

结论

本文讨论了模糊输入样本的支持向量机算法，给出了一种在模糊输入的情况下求解支持向量的办法，作为一种对经典 SVM 算法的泛化，我们讨论了在模糊输入的情况下，拟合得出模糊输出的 SVM 算法。

三、 部分核心源程序代码

1、PSO 优化 SVM 参数

```
import com.zhuke.svmclassifier.entity.SVMParam;
import com.zhuke.svmclassifier.util.ArrayUtil;
import libsvm.svm;
import libsvm.svm_model;
import libsvm.svm_problem;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.io.IOException;

/**
 * 粒子群优化算法
 * Created by ZHUKE on 2016/5/16.
 */
@Service
public class PSOService {

    @Autowired
    private DataSource2SvmProblemService dataSource2SvmProblemService;

    private svm_model model;

    private svm_problem probTrain;
    private svm_problem probTest;

    private int n = 2;//粒子维数
    private double[] gLimitation = {0.000001, 1000};//g 参数的位置限制
    private double[] cLimitation = {0.000001, 1000};//c 参数的位置限制

    private double weight = 0.8;//惯性权重
    private double c1 = 2;//粒子的个体学习因子
    private double c2 = 2;//粒子的社会学习因子
    private int maxgen = 50;//最大迭代次数
    private int populationSize = 50;//种群规模

    private double[] vLimitation = {-2, 2};//速度限制
    private double[][] position = new double[populationSize][n];//粒子的当前位置
```

```
    private double[][] v = new double[populationSize][n];//粒子的当前速度
    private double[] fitness = new double[populationSize];//粒子当前位置的适应值

    private double[][] pbestPositon = new double[populationSize][n];//粒子的个体最优解位置
    private double[] pbestFitness = new double[populationSize];//例子的个体最优解适应值
    private double[] gbestPosition = new double[n];//种群最优解位置
    private double gbestFitness = 0;//种群最优解适应值


    /**
     * 初始化种群离子位置和速度
     */
    private void init() {
        for (int i = 0; i < populationSize; i++) {
            double[] temp_p = new double[n];
            double[] temp_v = new double[n];
            for (int j = 0; j < n; j++) {
                temp_p[j] = Math.random();
                temp_v[j] = Math.random();
            }
            position[i] = temp_p;
            v[i] = temp_v;
        }
    }


    /**
     * 设置更新最优位置
     */
    private void updateBest() throws IOException {
        for (int i = 0; i < populationSize; i++) {
            double adoptedValue = getAdoptedValue(probTrain, probTest, position[i][0], position[i][1]);
            fitness[i] = adoptedValue;

            if (adoptedValue > pbestFitness[i]) {
                pbestFitness[i] = adoptedValue;
                pbestPositon[i] = position[i];
                System.out.println("+++++++++++++++Update pbest" + i + ", pbestFitness=" + adoptedValue + ", pbestPosition=" + position[i][0] + "," + position[i][1]);
            }

            if (adoptedValue > gbestFitness) {
```

```
                    gbestPosition = position[i];
                    gbestFitness = adoptedValue;
                    System.out.println("*****************Update gbest" + i + ", gbestFitness=" + adoptedValue + ", pbestPosition=" + position[i][0] + "," + position[i][1]);
                }
            }
        }

        /**
         * 开始 PSO 算法
         */
        public void pso() throws Exception {
            this.probTrain = dataSource2SvmProblemService.readFromDB(1L, 0, 111);
            this.probTest = dataSource2SvmProblemService.readFromDB(1L, 111, 20);
            init();
            for (int i = 0; i < maxgen; i++) {
                updateBest();
                //更新粒子速度
                for (int j = 0; j < populationSize; j++) {
                    v[i] = ArrayUtil.plusArray(ArrayUtil.plusArray(ArrayUtil.muiltiArray(weight, v[i]), ArrayUtil.muiltiArray(c1 * Math.random(), ArrayUtil.subArray(pbestPositon[i], position[i]))), ArrayUtil.muiltiArray(c2 * Math.random(), ArrayUtil.subArray(gbestPosition, position[i])));
                    position[i] = ArrayUtil.plusArray(position[i], v[i]);
                    for (int k = 0; k < n; k++) {
                        if (v[i][k] > vLimitation[1])
                            v[i][k] = vLimitation[1];
                        if (v[i][k] < vLimitation[0])
                            v[i][k] = vLimitation[0];
                        if (position[i][k] > cLimitation[1])
                            position[i][k] = cLimitation[1];
                        if (position[i][k] < cLimitation[0])
                            position[i][k] = cLimitation[0];
                    }
                }
            }
            System.out.println("PSO completed , the gbestFitness = " + gbestFitness + ", get the best parameter : c=" + gbestPosition[0] + ", g=" + gbestPosition[1]);

        }


        public double getAdoptedValue(svm_problem probTrain, svm_problem probTest, double c, double g) throws IOException {
```

```
            model = svm.svm_train(probTrain, SVMParam.customize(c, g));
            int errorCount = 0;
            for (int i = 0; i < probTest.l; i++) {
                  double v = svm.svm_predict(model, probTest.x[i]);
                  if (v != probTest.y[i]) {
                        errorCount++;
                  }
            }

            return new Double((probTrain.l - errorCount)) / probTrain.l;
      }

      public static void main(String[] args) throws IOException {
      }

}
```

2、动作采集模块

```
package svmclassifier.zhuke.com.action_record;

import java.io.IOException;
import java.io.PrintStream;
import java.math.BigDecimal;
import java.net.HttpURLConnection;
import java.net.Socket;
import java.net.URL;
import java.net.URLEncoder;
import java.util.StringTokenizer;

/**
 * 动作数据发送线程
 * Created by ZHUKE on 2016/4/17.
 */
public class ActionSender implements Runnable {

      private static Socket socket;
      private static PrintStream printStream;

      @Override
      public void run() {
            try {
                  //发送数据
```

```
                ActionSender.updateToSendArray();
                ActionSender.sendAction(ActionSender.actionStrBuiler());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        /**
         * 采用 http 请求的方式发送数据，时效性保证的精确性不高，服务器收到请求的时间不一
         *
         * @throws IOException
         */
        public static void sendAction(String message) throws IOException {
            HttpURLConnection conn = (HttpURLConnection) new URL(SVMConfig.serverActionURL + "?userId=" + SVMConfig.loginUserId + "&action=" + URLEncoder.encode(message, "UTF-8")).openConnection();
            conn.setRequestMethod("GET");
            conn.setReadTimeout(200);
            conn.setConnectTimeout(200);
            conn.setUseCaches(false);

            if (conn.getResponseCode() == 200) {
                System.out.println("Success send action data = " + message);
            }
            conn.disconnect();
        }

        /**
         * 更新缓冲区数组
         */
        public static void updateBuffer() throws InterruptedException {
            // 将新接收到的数据存入到 temp 数组的第 temp_state 行
            String s = ActionRecorder.getCurrentAction();
            double[] d = actionNormalize(s);
            System.out.println("得到状态值" + s);
            if (d != null) {
                System.arraycopy(d, 0, SVMConfig.ACTION_TEMP[SVMConfig.TEMP_STATE], 0, SVMConfig.FEATURE_NUM);
                SVMConfig.TEMP_STATE = (SVMConfig.TEMP_STATE + 1) % SVMConfig.ACTION_TO_RECORD;
            }
        }

        private static double[] actionNormalize(String action) {
            try {
```

```
                StringTokenizer st = new StringTokenizer(action, ",~");
                int count = st.countTokens();
                double[] d = new double[count];
                for (int i = 0; i < count; i++) {
                    d[i] = Double.parseDouble(st.nextToken());
                    //对方向传感器的值进行归一化处理
                    if (i == 3) {
                        d[i] = new BigDecimal(d[i] / 360).setScale(2,
BigDecimal.ROUND_HALF_UP).doubleValue();
                    } else if (i == 4) {
                        d[i] = new BigDecimal(d[i] / 180).setScale(2,
BigDecimal.ROUND_HALF_UP).doubleValue();
                    }
                }
                return d;
            } catch (NumberFormatException e) {
                e.printStackTrace();
            }
            return null;
        }


        /**
         * 更新待发送数组
         */
        public static void updateToSendArray() {
            if (SVMConfig.TEMP_STATE < SVMConfig.R - SVMConfig.L +
SVMConfig.NOISE) {
                //此时需要取历史数据
                int count = 0;
                for (int i = 0, j = SVMConfig.TEMP_STATE; j >= SVMConfig.NOISE; i++,
j--) {
                    SVMConfig.TO_SEND[i] = SVMConfig.ACTION_TEMP[j -
SVMConfig.NOISE];
                    count++;
                }
                for (int i = SVMConfig.ACTION_TO_RECORD, j = count; count <
SVMConfig.R - SVMConfig.L; i--, j++) {
                    SVMConfig.TO_SEND[count] = SVMConfig.ACTION_TEMP[i - 1];
                    count++;
                }
            } else {
                //直接存取
                for (int i = 0, j = SVMConfig.TEMP_STATE; i < SVMConfig.R -
SVMConfig.L; i++, j--) {
                    SVMConfig.TO_SEND[i] = SVMConfig.ACTION_TEMP[j -
```

```
SVMConfig.NOISE];
                }
            }
        }

        public static String actionStrBuiler() {
            // 将待预测数组进行格式化处理，将各属性值进行调整
            double[] t = new double[(SVMConfig.R - SVMConfig.L) *
SVMConfig.FEATURE_NUM];
            for (int i = 0; i < SVMConfig.FEATURE_NUM; i++) {
                for (int j = 0; j < SVMConfig.R - SVMConfig.L; j++) {
                    t[i * (SVMConfig.R - SVMConfig.L) + j] =
SVMConfig.TO_SEND[j][i];
                }
            }
            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < t.length; i++) {
                sb.append(t[i] + ",");
            }
            return sb.toString().substring(0, sb.toString().length() - 1);
        }
}
```

3、KNN 分类算法源码

```
__author__ = 'ZHUKE'
# coding=utf-8
from sklearn import neighbors
from sklearn.cross_validation import train_test_split
import numpy as np

data = []
lables = []

with open("action.txt") as ifile:
    for line in ifile:
        tokens = line.strip().split(" ")
        data.append([float(tk) for tk in tokens[1:]])
        lables.append(tokens[0])
x = np.array(data)
y = np.array(lables)
'''拆分训练数据与测试数据'''
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
'''训练 KNN 分类器'''
```

```
clf = neighbors.KNeighborsClassifier(algorithm="kd_tree")
clf.fit(x_train, y_train)
'''测试结果打印'''
answer = clf.predict(x)
print x
print answer
print y
print np.mean(answer == y)
```