学校代码：10286
分类号：
密　级：
U D C：
学　号：140740

东南大学
SOUTHEAST UNIVERSITY

硕士学位论文

基于智能手机传感器的用户隐私安全研究

研究生姓名：**董启宏**
导 师 姓 名：**宋宇波**

申请学位类别 工学硕士　　学位授予单位 **东 南 大 学**

一级学科名称 信息与通信工程　　论文答辩日期 **20　年　月　日**

二级学科名称 信息安全　　学位授予日期 **20　年　月　日**

答辩委员会主席　　评　阅　人

20　年　月　日

東南大學

硕士学位论文

基于智能手机传感器的用户隐私安全研究

专 业 名 称： 信息安全

研究生姓名： 董启宏

导 师 姓 名： 宋宇波

RESEARCH ON USER PRIVACY SECURITY BASED ON SMARTPHONE SENSORS

A Thesis Submitted to

Southeast University

For the Academic Degree of Master of Engineering

BY

DONG Qi-hong

Supervised by

Prof. SONG Yu-bo

School of Information Science and Engineering

Southeast University

2018/3/15

东南大学学位论文独创性声明

■

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，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，也不包含为获得东南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了谢意。

研究生签名：_____日期：_____

东南大学学位论文使用授权声明

东南大学、中国科学技术信息研究所、国家图书馆有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括以电子信息形式刊登）论文的全部内容或中、英文摘要等部分内容。论文的公布（包括以电子信息形式刊登）授权东南大学研究生院办理。

研究生签名：_____导师签名：_____日期：_____

摘要

随着移动互联网的普及，智能手机不仅仅是一个用来打电话沟通的工具，还承担着社交、电子金融和网络购物等任务。因此用户隐私信息泄漏问题一直是网络安全研究中的重要话题。目前 Android 和 iOS 两大主流手机操作系统均允许应用程序自由读取动作传感器数据而无需用户许可。而近年的研究表明，通过采集智能手机内置传感器数据，可以从中推测出用户的输入内容，从而对用户的隐私安全造成显著威胁。

本文主要研究基于智能手机内置的加速度传感器、陀螺仪传感器、方向传感器等动作传感器的用户隐私安全问题。针对现有研究基于传感器数据时域特征的 PIN 码推测准确率较低的问题，本文提出了一种新型的传感器数据频域特征提取算法—改进的 MFCC 算法。在此基础上构建了一种基于时域频域特征相结合的 PIN 码推测通用框架。该框架通过手机内置动作传感器采集按键数据并进行预处理，使用改进的 MFCC 算法提取频域特征，结合时域特征共同对神经网络算法中的多层感知机模型进行训练并推测用户 PIN 码。最后本文设计并实现了原型系统，对该框架的可行性进行了验证。本文主要工作内容如下：

（1） 研究 Android 和 iOS 操作系统的手机内置传感器安全机制，对动作传感器的权限控制机制进行了安全性分析。研究了机器学习中数据采集、预处理、特征提取、模型训练等各个阶段的理论和算法，并对本文将会使用的几种分类算法做了分析。

（2） 在传感器数据采集方面，本文提出了一种基于 Web 和 NodeJS 的跨平台多终端手机传感器数据采集方法。该方法可通过手机浏览器运行采集程序将传感器数据实时传输到后台服务器存储。为了解决不同平台下传感器采样率不同导致的数据差异，本文使用三次样条插值算法对传感器数据做平滑滤波预处理得到一致的数据样本。

（3） 在特征提取方面，本文提出一种改进的 MFCC 算法来提取动作传感器数据频域特征。该算法考虑动作传感器信号与语音信号均具有频域能量分布集中、信号特异性强的相似特性，针对动作传感器信号自身特点，去除原 MFCC 算法中预加重和汉明窗口的处理，推导了动作传感器信号 0-4kHz 频域范围内 Mel 弯折公式，提取出动作传感器信号的频域特征。

（4） 在模型训练方面，本文基于神经网络中的多层感知机算法建立分类模型，选择 ReLU 作为激活函数，Adam 为优化算法，通过反向传播算法进行训练。

（5） 设计并实现了基于时域频域特征相结合的 PIN 码推测原型系统。该系统由传感器数据采集系统、机器学习推测系统、PIN 码推测概念验证程序构成。可在 Android 和 iOS 平台下采集 PIN 码输入时的动作传感器数据样本，使用不同超参数对多层感知机模型进行迭代训练得到最佳结果，在 XSS 模拟攻击环境下实现 PIN 码推测。本文通过在校园里随机邀请的方式采集了 10k 个样本进行训练，经测试表明该系统在 iOS 和 Android 平台下 PIN 码 4 次推测准确率分别达到 94.34%和 91.28%，优于现有研究 4 次推测准确率 43%-71.6%的结果。

（6） 针对通过传感器推测 PIN 码可能导致隐私泄漏的安全风险，本文提出一种基于噪声注入的防御措施，通过在传感器数据中加入可编程控制的自适应噪声来抵御基于动作传感器的隐私窃取。经测试表明在引入该防御措施后 PIN 码识别的准确率降低到 19.3%。进一步，本文还从操作系统设计、应用程序开发、用户使用等方面的安全改进进行探讨。

关键词：智能手机，侧信道分析，动作传感器，机器学习，PIN 推测

Abstract

With the popularity of the Mobile Internet, smartphone is not only a tool for making calls, but also for tasks such as social networking, e-finance, and online shopping. Therefore, the leakage of user privacy information has always been an important topic in the study of network security. Currently, two major mobile operating systems, Android and iOS, allow applications to freely read motion sensor data without user permission. In recent years, research has shown that user's input content can be inferred by collecting smartphone built-in sensor data, thereby posing a significant threat to user's privacy security.

This thesis focuses on user's privacy security issues based on smartphone built-in motion sensors such as accelerometers, gyroscopes, and direction sensors. Aiming at the problem that the accuracy of the PIN inferring based on the time domain feature of sensor data is low in the existing research, this thesis proposes a novel algorithm for frequency domain feature extraction of sensor data—an improved MFCC algorithm. Based on this, a general framework for PIN inference based on time-domain and frequency-domain features is constructed. The framework collects keystroke data through the built-in motion sensor of the smartphone and preprocesses it, uses the improved MFCC algorithm to extract frequency domain features, and combines the time domain features together to train the multilayer perceptron model, and then infer the user PIN code. Finally, this thesis designs and implements a prototype system and verifies the feasibility of the framework. The main work of this article is as follows:

(1) Analyze the security mechanism of the smartphone's built-in sensor in the Android and iOS operating systems, and study the security control mechanism of the motion sensor. The theory and algorithm of data acquisition, preprocessing, feature extraction, model training and other stages in machine learning are studied, and several classification algorithms that will be used in this paper are analyzed.

(2) In the aspect of sensor data collection, this thesis proposes a cross-platform multi-terminal mobile sensor data acquisition method based on Web and NodeJS. This method can use the acquisition program that is running through the smartphone browser to transfer the sensor data in real time to the background server and storage. In order to solve the data discrepancy caused by the different sampling rates of sensors under different platforms, this thesis uses the cubic spline interpolation algorithm to do smooth filter preprocessing on the sensor data to obtain a consistent data sample.

(3) In feature extraction, this thesis proposes an improved MFCC algorithm to extract the frequency domain features of motion sensor data. The algorithm considers that both the motion sensor signal and the speech signal have the same characteristics of frequency domain energy distribution and strong signal specificity. According to the motion sensor signal characteristics, remove the pre emphasis and Hamming window in the original MFCC algorithm, and deduces the Mel formula of bending motion sensor signal in 0-4kHz frequency range, then extract the frequency characteristics from motion sensor signal.

(4) In terms of model training, this thesis builds a classification model based on multilayer perceptron algorithms in neural networks. We choose ReLU as activation function, Adam as optimization algorithm, and use backpropagation algorithm to train.

(5) Designed and implemented the prototype system of PIN inference based on the combination of time domain and frequency domain features. The system consists of a sensor data acquisition system, a machine learning inference system, and a PIN inference concept verification program. Under the platform of Android and iOS, we can collect data samples of motion sensors when PIN code is typed, and use different hyperparameters to training multilayer perceptron model iterately to get the best result, and to achieve PIN code inference in XSS simulation attack environment. In this thesis, 10K samples were collected through random invitation in the campus for training. The test results show that the accuracy of at most four-time PIN inference under the iOS and Android platform reached 94.34% and 91.28% respectively, which is better than the existing research's four-time accuracy of 43%-71.6%.

(6) For the security risk that the PIN is inferred through the motion sensor may lead to privacy leakage, this paper proposes a defensive measure based on noise injection, which is based on adding programmable adaptive noise in sensor data to resist privacy stealing based on motion sensors. Tests have shown that the accuracy of PIN identification after the introduction of this defensive measure is reduced to 19.3%. Further, this thesis also discusses the security improvement in the aspects of operating system design, application development, and user use.

Keywords: smartphone, side channel analysis, motion sensor, machine learning, PIN inference

目录

插图目录

第一章 绪论

本章首先介绍了使用智能手机传感器数据通过数据挖掘和机器学习来获取用户隐私信息的课题背景。描述了在移动互联网和智能手机爆发时代，用户的手机中存储着大量的隐私信息，而通过内置动作传感器所产生的数据，使用机器学习技术可以推测出用户的输入信息，从而导致隐私泄漏。其次详细阐述了国内外基于智能手机动作传感器的用户隐私安全研究现状，同时还分析和讨论了这些研究工作的重要意义以及存在的一些问题。接着从宏观层面介绍了本文基于智能手机浏览器来获取动作传感器数据并利用机器学习算法推测出用户输入 PIN 码的研究内容和创新点。最后一部分则介绍了本文的组织结构。

1.1 研究背景

如今的智能手机已经不仅仅是一个用来打电话沟通的工具了，在移动互联网时代它还承担着社交、电子金融和网络购物等任务。因此，用户的智能手机中存储着越来越多的敏感信息和个人隐私，比如支付密码、短信记录、通讯录和位置信息等。如何获取用户的隐私信息一直是网络安全研究中的重要话题，个人电脑时代就有利用电磁泄露、键盘钩子等方式获取用户隐私的方法。近几年，攻击者开始把目光投向智能手机，因为其频繁的使用频率和海量的隐私信息相较于传统 PC 平台更具有吸引力。然而现代智能手机的人机交互的模式发生了彻底性的革命，它使用触摸屏来代替物理键盘进行输入工作，因此传统的电磁泄漏和声音窃取等边信道分析模式都无法直接套用。

科技的发展和工艺的进步极大的降低了传感器的体积和功耗，让现代智能手机普遍配备了丰富的动作传感器来增强用户体验，如加速度传感器、陀螺仪、地磁方向传感器、光线传感器等。它们可以测量智能手机的状态，比如方向、加速度、角度、高度等。这些动作传感器可以支持创新的应用 UI 设计、环境感知、基于动作的指令（如摇一摇切歌）等，极大的扩展了应用程序的想象空间，让很多健康应用、社交应用、环境监测、移动支付变得可行，也使人们的生活更加便利。

然而有研究发现，通过动作传感器获得的手机加速度、角速度、方向等信息有可能推测出用户在手机触摸屏上的输入，从而构成旁路攻击窃取用户隐私。也许是考虑到这些动作传感器产生的数据对用户来说并不敏感，到目前为止主流手机操作系统 Andorid、iOS 均允许第三方应用程序无需用户许可就可以读取手机中的加速度传感器、陀螺仪、地磁方向传感器采集的数据；与之相比，摄像头、GPS 则需要用户明确的许可。最近，Marquardt et al[1]展示了一个可以通过检测 iPhone 的震动来推测用户在键盘上的输入的恶意程序。Cai 和 Chen[2]研究了通过手机应用程序采集动作传感器数据来推测用户在触摸屏上的按键动作。这类推测行为构成了旁路攻击，相较于传统的计算机平台，智能手机作为触屏的手持设备，用户需要触摸屏幕来输入，而这一行为会让智能手机产生微小的物理偏移，对于人来说这些位移很微小，有时候甚至是不可察觉的，但是对于机器来说，却足以测量和分辨。

使用机器学习技术，理论上可以通过大量的训练样本来建立模型，从用户在屏幕上敲击按键时动作传感器的数据中推测出用户输入的内容。机器学习经过多年的发展，现在已经进入大规模应用的阶段。由于机器学习需要强大的处理资源，一般需要通过 GPU 来进行模型的学习和训练，而手机上相对薄弱的处理器资源和有限的电池决定了不可能在手机上完成训练，需要有一台云端的服务器专门进行学习。如今随处可及的网络服务让恶意攻击将传感器数据传输到远程服务器变得可行，恶意程序通常在服务器执行具体的数据处理和模型训练工作，手机端仅负责捕获和传输传感器数据。

因为这些任务并不会耗费太多电池和内存资源，普通用户往往不能察觉到异常。

获取手机动作传感器的数据一般需要通过三种方法：① 编写原生应用程序，使用手机操作系统提供的 API；② 将 HTML5 代码重打包为原生应用程序从而使用操作系统提供的 API；③ 使用由 W3C 提供的标准 API，在浏览器中通过 JavaScript 直接读取传感器数据。其中第三种方法相较于其他方法有着跨平台、不需要安装应用、发布无需审核、可远程升级的优势，对于用户来说接受度更高。W3C 发布了一系列 API 标准来支持 Web 程序获取手机内置传感器的数据，它根据安全和隐私等级将手机上的传感器划分为不同的安全级别，其中地理位置和摄像头的使用需要用户许可，而加速度传感器、陀螺仪、地磁方向传感器等不需要用户许可即可使用。

很多流行的手机浏览器都已经实现了 W3C 的传感器 API 标准，比如 Safari、Chrome、Firefox、Opera、遨游浏览器等，有些允许 iframe 中的 JavaScript 执行访问传感器数据的操作。有些浏览器甚至允许非活动页中的代码继续执行，也就是说如果用户打开含有恶意访问传感器代码的网页后没有关闭，此时再打开其他标签页时该恶意代码仍然在运行。有些使用 Webkit 的应用程序（如微信）也支持在应用中访问外部网页，在其中打开含有恶意读取传感器数据代码的网页后锁屏，恶意代码甚至会继续运行，动作传感器数据会在用户不知情的情况下继续往恶意服务器传输，直到该应用程序进程退出为止。

如果攻击者编写恶意程序利用浏览器平台来获取用户动作传感器的数据，危害将会更大，因为使用 JavaScript 编写的恶意代码不需要安装在用户手机上，只要打开含有恶意代码的网页，程序就会自动执行。并且以往的恶意软件需要针对不同的手机操作系统进行编写适配，而网页应用程序有着天然的跨平台特性，恶意代码的攻击面更为广泛。另外使用 JavaScript 获取传感器数据不需要用户许可，在读取数据的过程中也不会给用户任何提示，攻击方可以在用户没有察觉的情况下将动作传感器数据传输到云端进行分析处理，并从中获取用户的隐私信息。

综上所述，在移动互联网时代，手机在人们的日常生活扮演越来越重要的角色，用户在使用手机时经常会输入一些诸如帐号、密码、PIN 码之类的隐私信息，这些信息一旦被泄漏势必会对人们的生活和财产造成重大损失。HTML5 应用的广泛使用、W3C 动作传感器标准薄弱的安全限制、越来越强大的机器学习技术，综合这些因素来看，基于手机浏览器平台通过动作传感器数据进行旁路攻击从而泄漏用户隐私信息的风险越来越大。因此必须认真研究通过机器学习技术从手机动作传感器数据获取用户隐私的潜在威胁。

1.2 国内外研究现状

随着智能手机数量的急剧增长，针对手机的恶意软件也越来越多。根据现有的攻击统计结果来看，移动设备已经成为新的信息安全薄弱点，由于手机上通常存储着大量的个人信息，包括保存的密码、登录信息、电子邮件、银行账号、联系人、地理位置等重要隐私信息，另外还有可能包含 ECG 心率等生理信息，这些信息对于黑客和不法分子有着巨大的吸引力。

Nahapetian[3]在其论文中提到，智能手机上通常装有大量的传感器，已经有证据表明，通过对这些传感器数据进行分析和提炼可以推测出相当数量的个人信息，无论本意是正当的还是恶意的。更糟糕的是，其中有很多传感器，尤其是动作传感器缺乏有效的访问权限控制，手机应用和浏览器都可以直接访问这些传感器的数据。官方的软件商店如 iTunes、Google Play 以及第三方应用市场为恶意软件获取手机的传感器数据提供了机会[4]。这些恶意应用看起来和正常软件没有什么差别，因为访问手机的传感器似乎是很稀松平常的一件事，有鉴于此想要检测出此类恶意软件非常困难。例如，一个健身应用可以正当声明自己需要读取动作传感器数据来实现健身功能，但随后是否会利用这些数据做些非法应用就不得而知了，即使应用市场会对上架的应用做详细检查，也并不清楚这些数据是否会被恶意利用。

Cai[5]首次提出使用智能手机上的地磁方向传感器来推测用户输入，他发现当用户在使用智能手

机上的软键盘输入的时候，尤其是手持时手机会产生震动，而这些震动跟用户输入的内容有关。图 1-1 展示了在触屏手机的数字键盘上按不同的键时，手机发生旋转的角度路径。当用户在手机上做一个敲击按键动作时，手机的俯仰角（pitch angle）和旋转角（roll angle）会从该模式的中心沿着一条路径到达顶部定点，然后再沿着另一条路径移动到底部定点，最后再返回到中心。Cai 设计了一个叫 TouchLogger 的安卓应用，通过使用智能手机上的地磁方向传感器数据来判断用户的手指触碰到了屏幕的哪个位置。实验结果在收集到的数据集中振奋人心，在受控的特定实验条件下，该系统可以以 70%的准确率判断出 10 个有效区域中用户点击的那个区域。Cai 认为影响预测准确率的因素有很多，比如按键时的力度，手持手机的力度大小，发生按键动作前手机的初始位置以及另一只手在手机上的摆放位置都会影响的预测结果。其中只有最后一项会明显干扰 TouchLogger 的判断，因为它会改变手机的中心点。Cai 的样本数据是通过收集同一个人在几天内使用手机时产生的，对于不同的用户该系统的表现并没有提及，但如论文中所说，不同的握持方法会对该系统的预测结果产生较大影响。

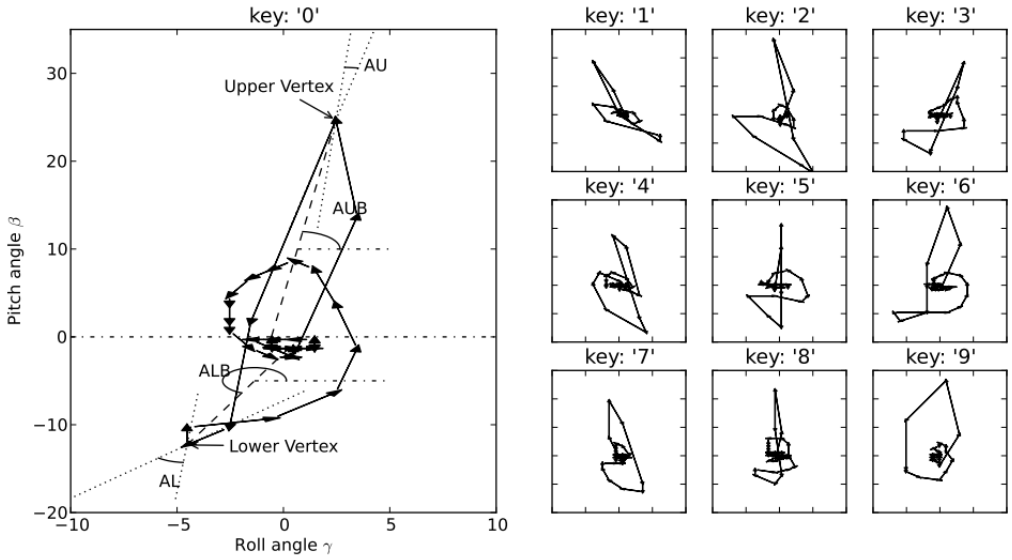

图 1-1 敲击不同的数字按键时手机的角度旋转路径

之后 Xu 等人在 Cai 的基础之上做了进一步研究[6]，他们设计了一个叫 TapLogger 的系统，可以推测用户在手机上输入的类似数字 PIN 码的内容，该系统包含一个伪装成游戏的恶意安卓软件，当用户使用该软件时手机上的动作传感器数据会被记录下来，同时还有与之对应的具体按键信息，将这些数据传输到云端服务器进行学习和训练，此后当用户执行敏感输入时，该恶意软件会利用之前的学习模型去推测用户输入的内容，比如盗取开机 PIN 码等，具体的流程如图 1-1 所示。Xu 在论文中提到该模型在完成三次推测步骤后可以检测出 PIN 码的所有数字，具体来说就是对连续无重叠的数字进行推测。然而 Xu 并没有详细阐述在所有可能排列组合中做出选择的过程。例如，在三次预测后，对于一个 4 位数字的 PIN 来说其中 3 个数字已经有了相应的推测结果，然而对于第 4 个数字，在最坏情况下却需要猜测 81 次。令人惊讶的是，Xu 等人并没有使用标准的序列预测技术，如隐含马尔科夫模型（HMM），来将各次独立的预测结果连接起来。

上述几篇研究有一个共同点，都是基于手机的地磁方向传感器数据来推测用户输入的 PIN 码，而没有利用加速度传感器和陀螺仪的数据，可能是因为作者考虑到当时的手机还未普遍配备此类传感器。另外他们用来训练和识别的样本库规模也比较小，有的甚至只有一个人，这对于验证模型的准确性和普适性也会有较大影响。

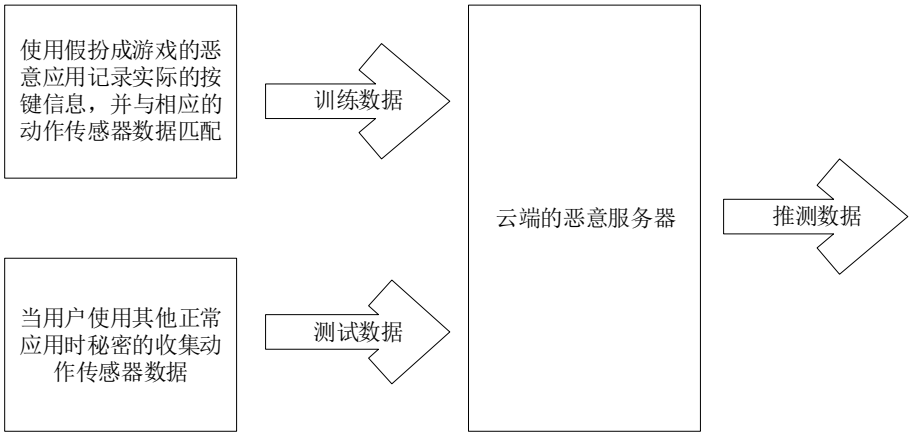

图 1-2 TapLogger 的工作流程

Milluzo 使用加速度传感器和方向传感器数据来推测用户的按键动作以及在手机屏幕上按键的位置[7]。Cai 等人又在之前的研究基础上使用加速度传感器和方向传感器来推测用户在触摸屏上的数字键盘上的输入[8]。Owusu 等人的论文中展示了加速度传感器可以作为一个基础的旁路攻击方法来获取用户在智能手机触摸屏上的软键盘中敲击时产生的数据[9]，通过标准的机器学习技术可以从中推测出密码。

Aviv 在其论文中展示了使用加速度传感器来学习用户在智能手机上的按键和滑动手势[10]，并用其来解锁使用基于手势或者 PIN 码锁定的安卓智能手机。通过收集由 24 人组成的两组实验者在静坐和行走两种状态下使用手机的数据，利用信号处理和多项式拟合技术实现了对于加速度传感器采样率不敏感的特性。在由静坐状态下采集的 50 个 PIN 码和 50 个手势组成的样本库中测试发现，该预测模型可以在 5 次预测尝试中以 43%的正确率推测出 PIN 码，73%的正确率推测出解锁手势图案。在行走状态下采集的样本库中测试发现，该模型对 PIN 和手势图案分别能达到 20%和 40%的正确率。Aviv 还利用隐藏马尔科夫模型（HMM）来预测变长的 PIN 码数字，在这个更加苛刻的条件下（随机猜对的概率大约为 0.01%），该模型在最多 20 尝试中以 40%的正确率预测出正确的 PIN 码，以及以 26%的概率预测出解锁图形。Aviv 提到基于加速度传感器的实验结果要比之前 Cai 的基于地磁方向传感器的结果略好一些。

上述的研究在前人的基础上又进一步利用加速度传感器数据来做用户输入推测，他们使用了规模更大的样本库，并完善了不同的环境和条件下对实验数据的影响。他们还利用隐藏马尔科夫模型来提高多位数字 PIN 码的预测准确率。同时 Aviv 还进一步拓展了攻击的应用场景，利用加速度传感器数据来推测安卓手机的锁屏图案。

1.3 论文研究内容

如前文 1.2 节所述，目前已有的研究均在 Android 平台上通过原生应用获取动作传感器数据，并仅使用时域特征来进行 PIN 码的推测，准确率较低。 本文研究使用智能手机传感器来获取用户隐私的安全风险，相较于已有的研究内容，本文提出了一种新型的传感器数据频域特征提取算法—改进的 MFCC 算法，在此基础上构建了一种基于时域频域特征相结合的 PIN 码推测通用框架。该框架通过手机内置动作传感器采集按键数据并进行预处理，使用改进的 MFCC 算法提取频域特征，结合时域特征共同对神经网络算法中的多层感知机模型进行训练并推测用户 PIN 码。最后本文设计并实现了原型系统，对该框架的可行性进行了验证，并给出应对这种安全威胁的防御措施。本文具体的研究内容如下所述：

（1） 研究 Android 和 iOS 操作系统的手机内置传感器安全机制，对动作传感器的权限控制机制进行安全性分析。研究机器学习中数据采集、预处理、特征提取、模型训练等各个阶段的理论和算法。

（2） 在传感器数据采集方面，本文提出了一种基于 Web 和 NodeJS 的跨平台多终端手机传感器数据采集方法。该方法可通过手机浏览器运行采集程序将传感器数据实时传输到后台服务器存储。为了解决不同平台下传感器采样率不同导致的数据差异，本文使用三次样条插值算法对传感器数据做平滑滤波预处理得到一致的数据样本。

（3） 在特征提取方面，本文提出一种改进的 MFCC 算法来提取动作传感器数据频域特征。该算法考虑动作传感器信号与语音信号均具有频域能量分布集中、信号特异性强的相似特性，针对动作传感器信号自身特点，去除原 MFCC 算法中预加重和汉明窗口的处理，推导了动作传感器信号 0-4kHz 频域范围内 Mel 弯折公式，提取出动作传感器信号的频域特征。

（4） 在模型训练方面，本文基于神经网络中的多层感知机算法建立分类模型，选择 ReLU 作为激活函数，Adam 为优化算法，通过反向传播算法进行训练。

（5） 设计并实现了基于时域频域特征相结合的 PIN 码推测原型系统。该系统由传感器数据采集系统、机器学习推测系统、PIN 码推测概念验证程序构成。可在 Android 和 iOS 平台下采集 PIN 码输入时的动作传感器数据样本，使用不同超参数对多层感知机模型进行迭代训练得到最佳结果，在 XSS 模拟攻击环境下实现 PIN 码推测。

（6） 针对通过传感器推测 PIN 码可能导致隐私泄漏的安全风险，本文提出一种基于噪声注入的防御措施，通过在传感器数据中加入可编程控制的自适应噪声来抵御基于动作传感器的隐私窃取。经测试表明在引入该防御措施后 PIN 码识别的准确率降低到 19.3%。进一步，本文还从操作系统设计、应用程序开发、用户使用等方面的安全改进进行探讨。

1.4 论文组织结构

本文共分六章，从第二章开始的各章结构内容安排入下：

第二章，相关技术研究。本章介绍了跟本文研究内容相关的一些知识和技术。首先介绍现代手机操作系统的权限控制，具体包括基本的安全机制和权限管理，接着对智能手机内置的动作传感器进行详细介绍，包括加速度传感器、陀螺仪传感器、方向传感器的工作原理以及传感器的数据模式。随后对机器学习的相关概念进行介绍，并对机器学习过程中的重要步骤：数据预处理、数据聚合和特征提取等内容进行重点说明，最后对本文用到的分类算法进行简要说明。

第三章，基于时域频域特征相结合的 PIN 码推测通用框架。本章提出一个基于时域频域特征相结合的 PIN 码推测通用框架，其中包含传感器数据的采集、特征向量的提取、机器学习模型的训练与 PIN 码推测整个流程。首先介绍整体方案设计并论证可行性，然后介绍基于 Web 的跨平台传感器数据采集以及相应的预处理方法。随后介绍特征向量的提取，包括时域特征和基于改进 MFCC 算法提取的频域特征，并对如何改进 MFCC 算法进行详细介绍。最后介绍本文采用的多层感知机模型，从网络结构的搭建、激活函数的选择、优化算法的使用以及超参数的配置和调整方面进行详细说明。

第四章，PIN 码推测框架原型系统的实现。本章实现了基于时域频域特征相结合的 PIN 码推测框架原型系统，首先介绍跨平台传感器数据采集系统，对系统功能进行详细的需求分析，接着从采集系统的客户端和服务端分别就设计和实现过程进行介绍，并对一些关键挑战进行重点说明。然后介绍本文实现的 PIN 码推测概念验证程序，展示了通过 XSS 攻击可以在微信等支持 WebView 的社交应用中运行嵌入恶意代码的 Web 程序，从而窃取传感器数据的可能性。本章第三部分介绍机器学习推测系统，利用 Scikit-learn 搭建多层感知机模型，通过 Python 自动化实现预处理、特征提取、模型训练等步骤。最后进行实验测试，从多个角度对实验结果进行分析，并从中得出一些结论。

第五章，防御措施与安全建议。本章对现有的一些防御措施进行了介绍和分析，并指出其缺陷。接着介绍本文提出的一种新的防御措施，即通过在传感器数据中加入可编程控制的自适应噪声来抵御基于动作传感器的边信道攻击，并通过三种不同场景下的实验来测试抵抗边信道攻击的效果。接着本章从操作系统、应用开发、用户使用等层面提出了一些安全建议。

第六章，对全文进行总结，结合现有工作的不足之处和实验过程中新的想法提出了对未来工作的展望。

第二章 相关技术研究

本章首先介绍现代手机操作系统的权限控制，具体包括基本的安全机制、Android 系统和 iOS 系统的权限管理以及它们对内置传感器的权限控制策略。接着对智能手机内置的动作传感器进行详细介绍，包括加速度传感器、陀螺仪传感器、方向传感器的工作原理以及传感器的数据模式。随后对机器学习的相关概念进行介绍，并对机器学习过程中的重要步骤：数据预处理、数据聚合和特征提取等内容进行重点说明，最后对本文用到的分类算法进行简要说明。

2.1 智能终端权限控制

Android 和 iOS 分别是基于 Linux 内核和 Unix 内核进行设计的现代操作系统，它们继承了类 Unix 操作系统的基础安全框架，又在此基础上各自发展出了一些新的安全措施和隐私保护机制。现代手机操作系统安全机制的一个重要功能就是权限管理。通过设定访问设备中各种软硬件资源的权限需求、权限组和权限等级，操作系统就能够将软硬件资源的访问牢牢地控制住。在多年的攻防实践中，系统安全工程师们不断地修补漏洞，构筑更完善的安全体系。在经过长足的发展后，现在的手机操作系统安全机制已经基本成熟，能够预防一定强度的网络攻击，保护用户的隐私信息。

因此，想要实现对用户输入内容的推测或窃取，就必须对手机操作系统的安全机制和防护体系有具体的认识。只有通过分析和理解手机操作系统的安全机制和权限控制原理，才能找到合适的信息源辅助数据窃取和分析工作。本节首先简要介绍手机操作系统的安全机制，然后分别对 Android 和 iOS 两个系统的权限控制机制进行详细介绍，最后说明这两个操作系统对内置动作传感器的权限控制。

2.1.1 操作系统安全机制

Android 操作系统采用一种名为 SandBox（沙盒）的机制来进行进程隔离和应用隔离。一般情况下，各个运行中的应用程序都以独立进程运行在各自的虚拟机中[11]。每个应用程序在首次安装时都会被分配一个 UserID，这个 ID 在全局中是唯一固定且保持不变的。并且，该 UserID 与 Linux 内核中的进程 UID 用户名一一对应。而手机中运行的不同进程无法访问其他进程的软硬件资源，也无法和其他程序共享数据。因此，除了相关进程的程序外，其他应用程序是不能直接获得用户在屏幕软键盘上的输入的，这就有效地防止了恶意程序对用户隐私的窥探。

iOS 系统的隔离机制则更为激进，它采用了被称为“岛式存储”的存储结构。Android 系统虽然运行中的程序无法共享资源，但还存在统一的资源管理器，由系统维护着基本的数据目录。而 iOS 系统中，各个应用程序独自维护自己的数据资源和储存文件，在不经用户许可的情况下无法和其他程序交换文件。在 iOS 中，SandBox 的本质是一个文件夹，其路径是固定的，文件夹名字使用 UUID（全球唯一标识符）随机分配。如果某个应用程序要求其他应用程序目录下的文件资源时，需要严格的权限检查和用户提醒，合格后才能将该文件以文件副本的方式拷贝到自己的目录下。

2.1.2 Android 权限控制

Android 是一个权限分隔的操作系统，其中每个应用都有其独特的系统标识（Linux 用户 ID 和组 ID）。系统各部分也分隔为不同的标识。Android 据此将不同的应用以及应用与系统分隔开来。在默认情况下任何应用都没有权限执行对其他应用、操作系统或用户有不利影响的任何操作。这包括读取或写入用户的私有数据（例如联系人或电子邮件）、读取或写入其他应用程序的文件、执行网络

访问、使设备保持唤醒状态等。由于每个 Android 应用都是在进程沙盒中运行，因此应用必须显式共享资源和数据。它们的方法是声明需要哪些权限来获取基本 SandBox 未提供的额外功能。应用以静态方式声明它们需要的权限，然后 Android 系统提示用户同意。

基本 Android 应用默认情况下不会关联权限，这意味着它无法执行对用户体验或设备上任何数据产生不利影响的任何操作。要利用受保护的设备功能，必须在应用清单中包含一个或多个<uses-permission>标记。在 Android 操作系统中，系统权限通常以其涉及到的数据性质或使用到的硬件的特性而分为安全或危险的。Android 操作系统会判定权限是否涉及到使用者的个人私密信息，或是否存在破坏设备的软硬件环境的可能性。对于操作系统认为安全的权限，一般会直接赋予应用程序而不会触发警告消息；而对于系统认为危险的权限，系统将触发警告或提示信息，明确告知风险和责任，用户同意后方可赋予该应用。

2.1.3 iOS 权限控制

iOS 的权限管理与 Android 大致类似。主要的不同点在于，iOS 系统中所有软件需要读取的数据都会在读取的瞬间触发系统的强制提示，这个时候用户会看到一个系统提示框询问是否授予软件此项权限。如果用户选否，那么软件将无法获取任何对应的内容。相关授权在进行过第一次询问之后，用户可以在隐私设置中调整软件的对应授权。值得一提的是，越狱之后由于软件可以获取整个机器内置储存的访问权限，从某种程度上来说，软件可以直接读取对应的数据库内容而无需通过 API 进行访问，这个时候的隐私选项和权限控制也就形同虚设了。与 Android 系统相比还有些不同在于，iOS 系统并不提供对于短信、通话记录等 Android 系统提供读取的接口，所以有些东西，除非用户越狱，是一定无法被应用读取的。

2.2 传感器权限控制

如今的智能手机通常会内置很多低功耗的传感器来增强手机的功能和提升用户体验，图 2-1 展示了 iPhon6s 内置的传感器。例如加速度传感器通常可以用来实现当手机旋转时自动旋转屏幕，并重新组织应用的内容来适应当前的屏幕比例。距离传感器和光线传感器可以让手机识别出一些场景，比如用户在贴耳打电话的时候就可以关闭屏幕节省电源，放入口袋时防止误触，在晚上光线较弱时自动调低亮度防止刺眼，在户外强光下增加亮度提升可视性。

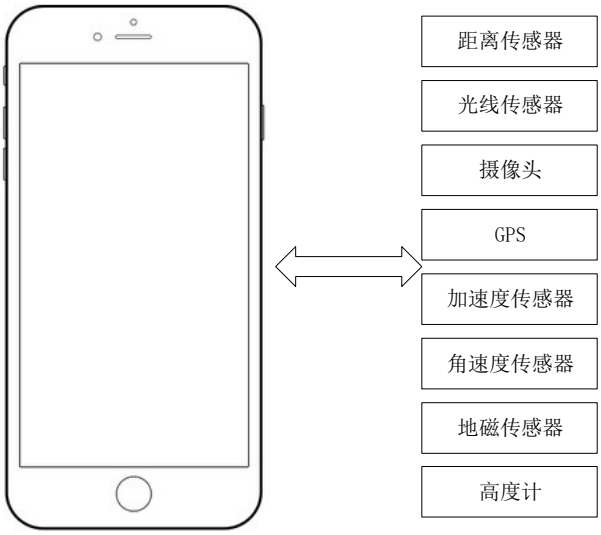

图 2-1 iPhone 6s 内置传感器

2.2.1 传感器简介

智能手机内置的传感器可分为动作传感器和非动作传感器两大类，其中动作传感器是指能够反映设备姿态、方向、加速度等属性的传感器，如加速度传感器、角速度传感器、方向传感器等。而非动作传感器就是除此之外的其他传感器，比如摄像头、光线传感器、压力传感器等。本文主要研究动作传感器数据，接下来对手机内置的动作传感器进行简要介绍。

2.2.1.1 传感器数据模式

动作传感器采集的数据主要反映手机的方向、运动速度和加速度以及姿态信息，这些数据在物理上都属于矢量。在手机操作系统中，传感器通常采用三轴笛卡尔坐标系对方向矢量进行分解。为了便于传感器提供商和手机设备生产商的统一，传感器的三轴设定一般采用相同的标准，如图 2-2 所示。

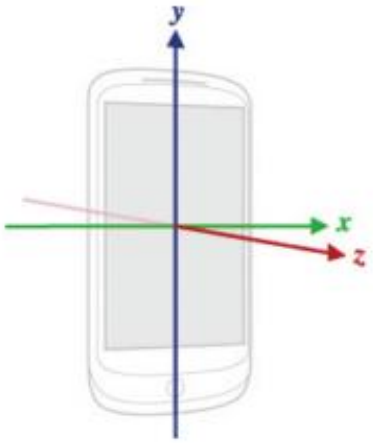

图 2-2 加速度传感器的坐标系统

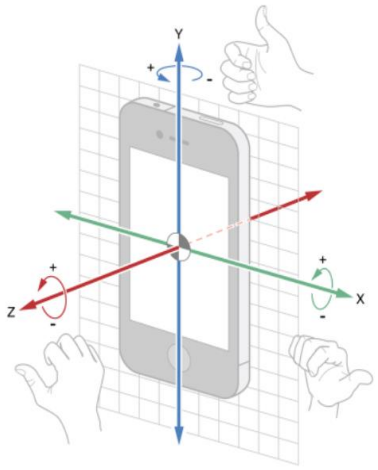

图 2-3 陀螺仪的坐标系统

动作传感器各轴的定义以手机形状为基准。一般操作系统定义手机屏幕的法线方向为 Z 轴，其正方向垂直屏幕由内向外；X 轴与 Z 轴垂直，正方向沿着手机横向自左向右；Y 轴同样垂直于 Z 轴，其正方向沿手机纵向自下向上，如图 2-2 所示。该坐标系设定中最重要的一点是，当手机的姿态或方向发生变化时，上述定义的三轴不会发生改变。动作传感器如加速度传感器、陀螺仪和地磁方向传感器都使用这样的数据格式。

2.2.1.2 加速度传感器

手机上的加速度传感器通常是指线性加速度传感器，它测量前后、左右、上下三个方向上的加速度大小和方向，坐标系统如 2.2.1.1 所述。加速度传感器通常由弹性元件、质量块、敏感元件、阻尼器和适调电路等部分组成，通过对质量块所受惯性力的测量，利用牛顿第二定律获得加速度值，每次读取该传感器时都会返回以m/s^2为单位的三个轴上的数据。加速度传感器有着非常丰富的应用场景，可用于游戏控制、电子指南针倾斜矫正、GPS 导航系统死角补偿、计步器等应用。

2.2.1.3 陀螺仪传感器

陀螺仪传感器（Gyroscope Sensor）又称角速度传感器，它可以测量手机转动时 X、Y、Z 三个轴上的转动角速度，其单位为rad/s。当手机相对于某轴逆时针转动时，陀螺仪上相应轴的角速度值为正，顺时针时则为负，如图 2-3 所示。智能手机上装备的陀螺仪通常采用的是 MEMS（微机电）传感器，内部结构非常精密，含有超微小的陀螺，具有体积小、成本低、功耗低、数字化和模块化的优

点。陀螺仪测量的参考标准是内部中间与地面垂直的方向上进行转动的陀螺，内部的微机械结构为振动件，通过测量旋转产生的科式加速度来获得角速度，强项在于测量手机自身的旋转运动，但不能确定设备方位。可以用来实现照相防抖、配合加速度传感器实现惯性导航等功能。

2.2.1.4 方向传感器

方向传感器（Orientation Sensor）用于测量手机的姿态信息。方向传感器的数据反映了手机相对地球参考系的位置和姿态变化。由于方向传感器监测的是设备与地球的相对信息，所以其坐标规范与数据模式都与 2.2.1.1 小节描述的规范不同。方向传感器同样记录 3 组数据，如图 2-4 所示。在 Android 操作系统中，物理方向传感器芯片已在 Android 2.2(API 8)版本时弃用，在 Android 4.4W(API 20)之后的系统版本中，该项数据由地磁方向传感器和加速度传感器的数据通过矩阵运算模拟得到。下面详细介绍方向传感器采集的数据：

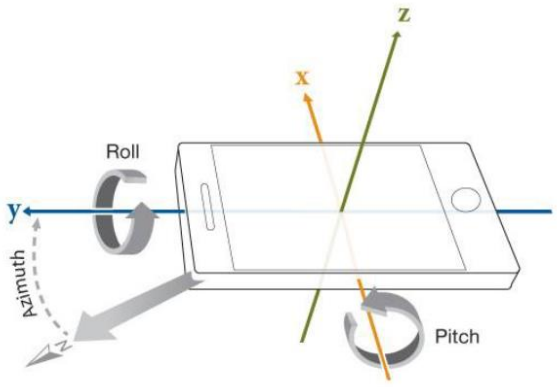

图 2-4 方向传感器坐标系统

方向角(Azimuth)：指设备围绕 Z 轴的旋转角。该参量的定义是设备的方向与磁北极的角。若设备的顶部边缘朝向地磁北极，则该参量为 0 度；相对地，若设备的顶部边缘朝向地磁南极，则该参量为 180 度。类似地，如果顶部边缘朝向东边，则方位角为 90 度，如果顶部边缘向西，方位角为 270 度。通常，手机内置的指南针功能就是基于该参量提供的信息设计的。

倾斜角(Pitch)：指设备围绕 X 轴的倾角。该参量的准确定义是设备屏幕所在平面与地平面的夹角。若设备平行于地平面并且底部边缘朝向使用者，在此时把设备的顶部边缘朝向地面倾斜时，该参量变为正值。相对地，将设备的顶部边缘抬起而逐渐远离地面时，该参量取负值。该参量的取值范围为-180 度到+180 度。

滚动角(Roll)：指设备围绕 Y 轴的滚动角度。该参量的准确定义为设备屏幕所在平面与地平面的垂面的夹角。若设备平行于地平面并且底部边缘朝向使用者，在此时把设备的左边缘朝向地面倾斜时，该参量变为正值。相对地，将设备的右边缘朝向地面倾斜时，该参量取负值。该参量的取值范围为-90 度到+90 度。

2.2.2 权限控制策略

目前市面上流行的绝大多数智能手机都配备了如加速度传感器、陀螺仪、方向传感器和地球磁场感应器等动作传感器。以 Android 操作系统为例，该系统支持的部分动作传感器如表 2-1 所示。在 Android 官方文档中提到，手机的加速度传感器和陀螺仪被定义为动作监测传感器，数据都由专门的硬件传感器芯片组提供；而方向传感器被定义为位置监测传感器。事实上，手机中并不存在专门的“方向传感器芯片组”，方向传感器的数据实际是由地球磁场传感器和重力传感器的数据通过矩阵运算得到的。

iOS 中的传感器软硬件各项参数和调用原理与 Android 基本相似，iOS 系统引入了更多的传感器调度算法和 CoreMotion.framework 框架，能够选择 Push 和 Pull 两种数据请求模式，实现更加智能的传感器调用，以达到在节约系统资源的同时采集到满足需求的数据。

表 2-1 Android 平台支持的部分动作传感器列表

传感器	类型	描述	常见用途
加速度传感器	硬件	测量设备 X、Y、Z 三轴的加速度，以m/s^2为单位。该加速度值中包含重力加速度。	运动检测（如摇晃，倾斜等）
陀螺仪	硬件	测量设备以 X、Y、Z 三轴为中心轴旋转的角速度，以rad/s为单位。	旋转检测（如旋转，转动等）
方向传感器	软件	测量设备在 X、Y、Z 三轴上的偏转角度，采用传统角度单位。	确定设备位置

无论是 Android 还是 iOS，它们都对动作传感器和非动作传感器的权限控制策略做了区分。在 Android 系统中，当应用程序想要获取摄像头、蓝牙等非动作传感器数据时，系统会在安装前明确提示用户，并只有用户允许后应用程序才能访问相应数据，而对于动作传感器，任何应用程序均可以直接访问而无需许可。对于 iOS 系统，当应用程序访问非动作传感器时系统会弹出询问框提示用户，并只有在得到许可后才允许应用程序访问传感器数据，而对于动作传感器，与 Android 系统类似，也是无需许可即可使用。

2.2.3 安全分析

Android 系统和 iOS 系统对于应用读取动作传感器数据的态度都是默认放行，也就是说不需要提示用户并获得许可，只需要在代码中声明就可以直接使用操作系统提供的 API 获取传感器数据。Android 操作系统可以通过 SensorManager 获取所有的内置传感器信息，iOS 操作系统提供的 CoreMotion.freamework 框架还集成多种算法，支持剥离重力加速度省去应用自行高通滤波的操作。除了原生应用外，在 Web 平台由于 W3C 颁布的 DeviceOrientation Event Specification 标准允许 Web 程序可以通过 JavaScript 直接访问手机上的动作传感器数据而无需经过用户许可，目前 Android 和 iOS 均支持这一标准。

无论是智能手机上安装的原生应用，还是运行在浏览器中的 Web 应用，都可以不经需用许可而自由读取智能手机内置的动作传感器数据。这原本是开发者为了方便用户使用，提高用户体验而设计的特性，由于近年来机器学习的飞速发展，使得通过动作传感器数据可以从中推测出用户的一些隐私信息，从而使这一便民特性有造成隐私泄露的风险。

2.3 机器学习

机器学习是计算机科学的一个分支，它的研究目的是让计算机具有无需特别编程而能够自身学习的能力，通俗来讲就是从过去的经验中总结归纳出一般化的结论或知识。作为机器学习前身的统计学已经在历史上存在了很长时间，从 1990 年开始机器学习开始成为单独的分支，它从计算机科学、统计科学、仿生学、心理学等很多学科汲取灵感，是一门涉及多领域的交叉学科。

2.3.1 机器学习简介

机器学习包含众多学习算法，它们分别适合解决不同类型的问题，根据学习过程中训练数据的

使用方式可以简单分为监督学习（Supervised Learning）、无监督学习（Unsupervised Learning）、半监督学习（Semi-supervised Learning）三种类型。无监督学习是指事先并不知道训练数据的标签，通过算法对数据的特征自动产生类别属性；监督学习所使用的样本数据会提前标记好标签，通过不断的对训练样本分析、学习、调整参数来构建模型，然后利用模型来判断标签未知的新数据的类别，该类型的方法在目前的机器学习中应用最为广泛。半监督学习则是监督学习和非监督学习二者的有机结合，主要区别在于学习过程中所使用的训练样本既包含标签样本也包含非标签样本，这种算法可以通过挖掘非标签样本中蕴含的信息来矫正由于标签样本自身代表性不好而导致的拟合分类器误差。

尽管机器学习已经有这么多的算法了，然而决定学习效果好坏更多的其实是数据。通常来说有标签的数据可以更好的提升分类器的性能，这类数据需要人工去标注，如果数据量很大，则会消耗相当数量的人力和物力，甚至是不可实现的。解决问题的办法主要有两种，一是利用众包的方式来协作标注数据；二是使用半监督式学习，通过标注少量数据和大量的未标注数据来构建模型。

2.3.2 机器学习流程

典型的机器学习流程如图 2-5 所示，其中数据的获取是整个项目的必要条件，在分析了问题的背景和需求后，通常需要将具体问题抽象成数学问题，然后获取数据样本来进行后续步骤，数据决定了机器学习结果的上限，算法只是尽可能的逼近这个上限。

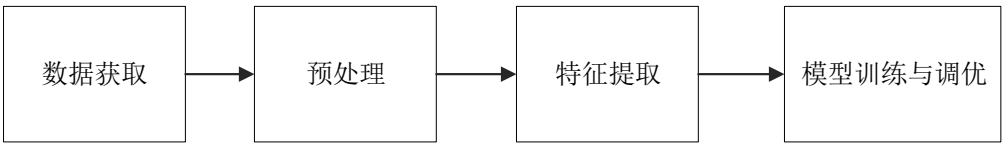

图 2-5 机器学习流程图

在工程实践中，得到的数据会存在有缺失值、重复值等，在使用之前需要进行数据预处理。数据预处理没有标准的流程，通常针对不同的任务和数据集属性的不同而不同。数据预处理的常用流程为：去除唯一属性、处理缺失值、属性编码、数据标准化正则化、特征选择、主成分分析，下面依次做出简要介绍。

（1）去除唯一属性：唯一属性通常是一些 id 属性，这些属性并不能刻画样本自身的分布规律，所以简单地删除这些属性即可。

（2）处理缺失值：原始数据中缺失值的处理主要有三种方法：①直接使用含有缺失值的特征；②删除含有缺失值的特征；③缺失值补全。常见的缺失值补全方法：均值插补、同类均值插补、建模预测、高维映射、多重插补、极大似然估计、压缩感知和矩阵补全。

（3）特征编码：常用的特征编码主要由两种形式，特征二元化和独热编码。特征二元化的过程是将数值型的属性转换为布尔值的属性，设定一个阈值作为划分属性值为 0 和 1 的分隔点。独热编码采用 N 位状态寄存器来对 N 个可能的取值进行编码，每个状态都由独立的寄存器来表示，并且在任意时刻只有其中一位有效。独热编码的优点是能够处理非数值属性，在一定程度上扩充了特征,编码后的属性是稀疏的，存在大量的零元分量。

（4）数据标准化：当某些算法要求样本具有零均值和单位方差，或者需要消除样本不同属性具有不同量级时的影响时就需要做数据标准化处理，数据标准化是将样本的属性缩放到某个指定的范围，常见的标准化有 min-max 标准化，z-score 标准化等方法。

（5）特征选择：从给定的特征集合中选出相关特征子集的过程称为特征选择，进行特征选择通常有两点原因，一是减轻维数灾难问题，二是降低学习任务的难度。进行特征选择必须确保不丢失重要特征，常见的特征选择方法分为三类：过滤式（filter）、包裹式（wrapper）、嵌入式（embedding）。

在得到特征集后，就要使用不同的算法来进行训练，目前有很多成熟的算法被封装成黑盒供人

使用，但真正影响结果的还有许多超参数，通过不断的调整和优化，可以使机器学习的结果变得更加优良。

2.3.3 分类算法

在后续章节中，本文使用多种分类算法进行实验，为了文章内容的完整性，在此对使用到的各种分类算法进行简要介绍。

2.3.3.1 KNN 算法

KNN(k-Nearest Neighbor)分类算法，是一个理论上比较成熟的机器学习算法。当 KNN 算法被用于分类问题时，它的处理逻辑为：计算某个对象或数据和靠近它的数个对象的距离(距离的计算方法视具体情况而各有不同)，确定其中距离带预测数据最近的 k 个数据；在这 k 个数据中，选取出现最多的数据分类，赋予被预测的数据作为它的分类。

KNN 算法属于一种即时学习的算法，与大部分的算法不同，即时学习方法不在训练时立即产生分类器，而只是保存训练数据的信息，在预测时才对预测数据即时判决[12]。也就是说，在训练时，该算法只会对训练数据的信息做分类保存，而在测试时才对测试数据进行判决。即时学习的优点在于可能能够更好地拟合样本的局部特性。在分类问题下使用 KNN 算法通常采用投票机制，即选择上述 k 个样本中出现的最多的类别标记作为分类的结果。在回归问题下使用 KNN 算法则通常采用平均方法，将 k 个样本的实际输出值的平均值作为回归结果。

2.3.3.2 SMO 算法

SMO 的全称为 Sequential Minimal Optimization，即顺次最小优化算法，该算法是基于 SVM 理论的一种实践算法，因此理解 SVM 算法是理解 SMO 算法的重要前提。

SVM（支持向量机）是一种非常重要的机器学习算法，常被用于监督学习问题中。SVM 模型将样本集表示为空间中的点集，其分类任务的目的是在空间中找到明确的边界将点集进行分类，且边界的宽度应该尽可能越宽越好。当判决新样本时，将新样本映射到相同的点集，并判断新样本与哪一边的距离较小，则新样本属于该分类中。简而言之，SVM 的思想是在空间中寻找到一种可能的判决边界，该边界能够将样本进行分类，且满足边界距离两边样本点的距离尽量最大化。在空间里，可以用式(2-1)来描述判决边界：

$$\omega^T x + b = 0 \tag{2-1}$$

其中$\omega = [\omega_1, \omega_2, \dots, \omega_n]$。空间中任一点 x 到判决边界的距离即为：

$$\gamma = \frac{|\omega^T x + b|}{\|\omega\|} \tag{2-2}$$

若上述的判决边界能够判决新的输入样本，也就是：

$$\begin{cases} \omega^T x + b > 0, y_i = +1 \\ \omega^T x + b < 0, y_i = -1 \end{cases} \tag{2-3}$$

可以有：

$$\begin{cases} \omega^T x + b \geq +1, y_i = +1 \\ \omega^T x + b \leq -1, y_i = -1 \end{cases} \tag{2-4}$$

如图 2-6 所示，判决边界附近的样本使得式(2-4)取得等号，这些样本点就是所谓的“支持向量”。使得两式取得等号的情况下，样本点到达判决边界的距离和为：

$$\gamma = \frac{2}{\|\omega\|} \tag{2-5}$$

该值被称作“间隔”。使用 SVM 算法的核心就是找到间隔的最大值，也就是找到使得式(2-4)成立的ω和 b,使得γ最大[13]。

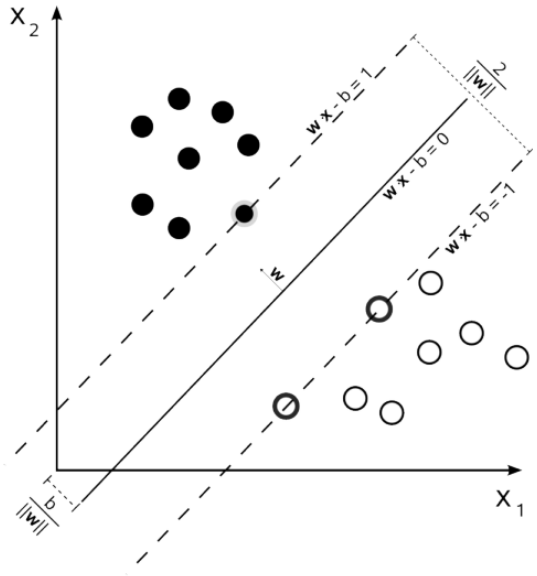

图 2-6 支持向量机原理图

SMO 算法主要通过把整体性的线性规划问题进行解构，从而利用分治思想解决各个简单的数学归类问题。SMO 方法能够将线性规划问题合理地分解为尽量小的结构，甚至能够使得程序在每次判断过程中只对两个样本进行比较。通过对原始问题的切割和最小化，可以顺次地解决每个“原子问题”并通过合理地安排和设计得到较为准确的的判断，而且能够规避迭代计算带来的程序开销和硬件负担。

2.3.3.3 C4.5 分类决策树算法

C4.5 是在 ID3 算法(Iterative Dichotomiser 3)的基础上发展而来的算法，该算法由 Ross Quinlan 提出[14]。ID3 算法也是一种决策树逻辑结构[15]，下面先简要介绍决策树算法的原理。

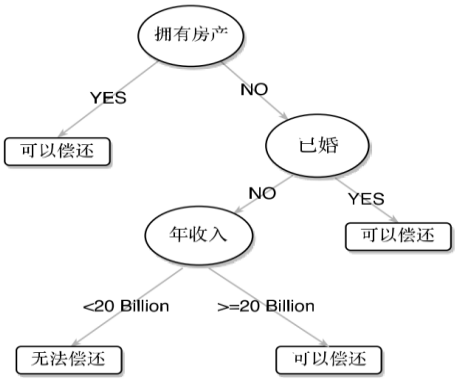

图 2-7 贷款还款能力的决策树实例

决策树学习算法使用一种树状模型来进行分支预测[16]。在决策树中，树的分枝表示每个分支选择项目，树的叶子节点表示每个选择项目的目标值。决策树模型是统计学，深度学习和机器学习中经常使用的预测数学模型之一。目标变量集是有限集的树结构常被称为“分类树”，在分类树中，叶子节点代表分类的标签，树的分枝代表与分类相关的属性值。在判决分析问题上，决策树可以直观

明确地表示判决过程和判决结果[17]。

决策树的中心思想是构造一个基于一系列输入变量来预测结果的模型。图 2-7 展示了一个例子，其中的每一个内部节点都表示一个输入属性或变量，而对于不同的输入造成的不同分类走向，则用分支和子节点来表示。每个叶子节点则代表按照输入变量进行判决过程后最终得到的目标值。

决策树被广泛运用于分类学习中。假设所有的输入特征值都属于一个有限离散域中，其目标特征即被称为“分类”。“分类”中的每个元素都被称为一个类。那么决策树或分类树可以被定义为每个内部节点（即非叶子节点）都被一个输入特征标记的树结构。来自用输入特征标记的节点的弧会用目标或输出特征的每个可能值标记，树的每个叶子节点都被标记为一个类或落入该类的概率分布值。根据属性值测试将源集合拆分为子集，可以进行树的学习。在学习过程中，算法递归地在每个节点上依照输入的不同情况生长出子树，基于训练数据构建出独一无二的层次结构。当节点上的子集具有所有相同的目标变量值时，或当分割不再向预测值添加值时，递归就宣告完成。决策树的自上而下归纳的这个过程是贪心算法的一个例子，它是迄今为止从数据中构建决策树的最常用的策略[18]。

ID3 算法基于朴素的奥卡姆剃刀(Occam’s razor)原理，认为小型的、简单的决策树结构优于复杂的、深度的决策树。ID3 算法的核心是信息论中的信息增益概念，用信息增益来调整属性的选择权重，采用贪心算法自上而下的遍历所有的判决可能性，直到生成最优的决策树。C4.5 在 ID3 的基础上又进行了几点改进，包括改用信息增益率代替信息增益值作为属性选择的判定标准，在决策树的生成过程中能够自动进行剪枝，实现将连续属性离散化。这样的改进提高了决策树构造过程中对属性的处理能力，产生的决策树模型的准确率较高，在数学上的表现也更为简单优雅。当然，C4.5 算法并不是完美无缺，它也有一些如训练时间过长、效率较低的缺点。

2.3.3.4 LMT 算法

LMT（Logistic Model Trees）是一种构建 Logistic 模型树的分类算法[19]，该算法同样也是基于决策树算法，特点是针对叶子节点使用 Logistic 回归函数进行分类判决。

Logistic 回归又称对数几率回归，是一种广泛运用于机器学习领域的基础算法。对于回归学习任务通常采用线性回归：

$$f(x) = w^T x + b \tag{2-6}$$

线性回归能够较好的处理回归问题，但是在分类问题上则会遇到问题。与回归问题不同，分类问题的目标结果并非连续的函数，而是一个间断的离散集合。因此分类问题的核心目的是找到合理的方法，将数据集映射到一个有限的标签集合上。因此，分类问题需要的是一个能在判决门线附近产生明显波动的判决器。以二分类任务为例，需要这样一个函数，它能将输入映射到 0,1 的离散集合中。Sigmoid 函数就是能够满足这样需求的函数：

$$y = \frac{1}{1+e^{-z}} \tag{2-7}$$

该函数在实数域上单调可微，取值为 0,1 间的连续值，在 0 附近变化陡峭，距离 0 稍远的情况下函数快速趋近于 0 或 1，如图 2-8 所示。

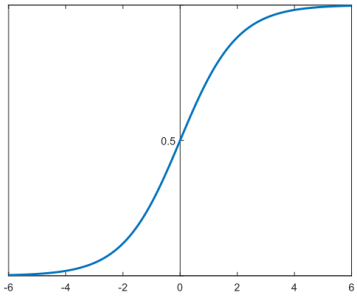

图 2-8 Sigmoid 函数

Logistic 回归非常适合分类问题。将式(2-7)代入广义线性模型中，可得到：

$$y = \frac{1}{1+e^{-(w^T x+b)}} \tag{2-8}$$

式(2-8)可化为：

$$ln\frac{y}{1-y} = w^T x + b \tag{2-9}$$

式(2-9)中$ln\frac{y}{1-y}$为样本正例的相对可能性，被称为“对数几率”。可见，Logistic 回归是用线性回归结果去逼近真实标记的对数几率。虽然被称作“Logistic 回归”，其实际上是一种分类的机器学习算法。

LMT 算法的优点是当多元分类的标签或源数值有缺漏时也能够稳健地进行分类处理。在大多数实例中，该算法的准确性和稳健性都优于只简单采用 SimpleLogistic 算法或决策树算法的情况。

2.3.3.5 Random Forest 算法

Random Forest 算法的中文名称为随机森林算法，是一种用于分类问题、回归问题和其他任务的综合学习方法。随机森林方法在训练时构建多个决策树，在分类问题中，该方法输出的是每个决策树中获得投票最多的分类，如图 2-9 所示；而在回归问题中，该方法输出的是每个决策树结果的数学期望。随机森林算法能够有效地克服由于训练集的数据的倾向性导致的模型过拟合问题。

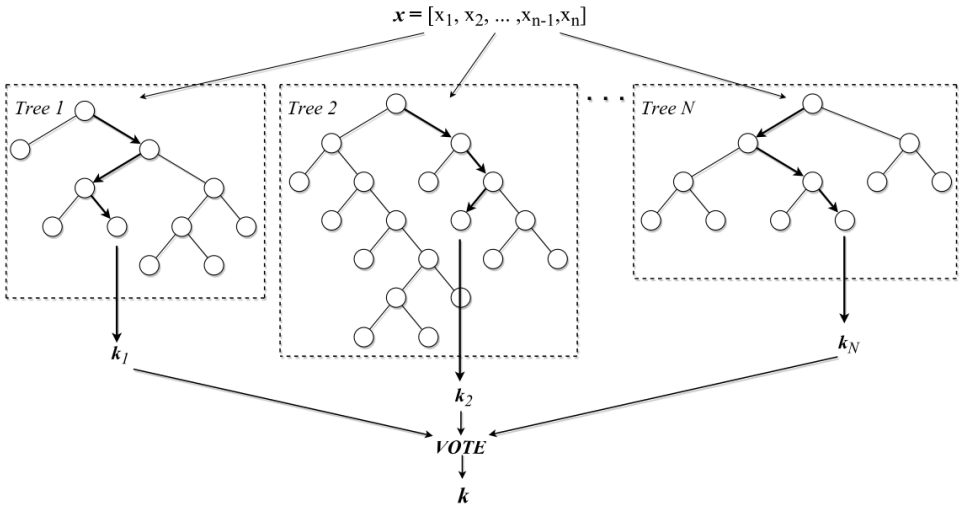

图 2-9 随机森林算法示意图

通常认为，深度过高的决策树倾向于构造出极度不规则的学习模式，在这种模式下会出现训练集的过拟合问题，主要反映在较低的偏差值和较高的方差值。随机森林是一种均衡考虑多个深层决策树的方法。通过采用同一训练集的不同子集分别训练决策树，可以有效地降低方差，但是这种设

计会带来偏差的小幅增大和数据可解释性方面的损失，但总体上会大幅提升模型的最终性能[20]。该算法牺牲了部分学习效率，但实现了对分类精确度的大幅提升。通过对训练集进行多重分类，将各子集投放给多个决策树，并用投票机制选择出最优的决策树分类结果，该算法有效地克服了在机器学习领域经常出现的过拟合问题，有效地降低了泛化误差，并且能够有效地处理个别数据的大幅误差和数据集的整体噪声。该算法在大量的分类学习实践中都能够得到较优秀的准确率[21]。

2.4 本章小结

本章对智能手机操作系统的权限控制进行介绍，描述了现代手机操作系统的基本安全机制，并详细介绍了 Android 系统和 iOS 系统的权限控制原理，以及它们对内置传感器的权限管理策略，揭露了目前手机内置传感器在各个系统平台以及 Web 平台下均无需用户许可即可访问的安全缺陷。接着介绍了智能手机内置的加速度传感器、陀螺仪传感器、方向传感器的原理、特性以及它们的坐标系和数据格式。最后简要介绍了机器学习的发展历史和应用场景，从宏观层面对预处理和特征提取的整个流程进行了介绍，并对本文用到的机器学习算法原理进行了简要介绍。

第三章 基于时域频域特征相结合的 PIN 码推测通用框架

目前国内外对使用手机动作传感器数据推测用户输入信息已有了一定的研究。本文的研究内容是在现有研究的基础上，研究如何使用传感器数据推测用户输入的数字 PIN 码。当用户输入 PIN 码时，会在触摸屏的不同位置敲击从而导致手机产生不同程度的倾斜，通过建立机器学习模型，收集按键过程中的动作传感器数据并进行训练，最后可从中推测出用户输入的 PIN 码。

本文针对现有研究基于传感器数据时域特征的 PIN 码推测准确率较低的问题，提出了一种新型的传感器数据频域特征提取算法—改进的 MFCC 算法。在此基础上构建了一种基于时域频域特征相结合的 PIN 码推测通用框架。该框架通过手机内置动作传感器采集按键数据并进行预处理，使用改进的 MFCC 算法提取频域特征，结合时域特征共同对神经网络算法中的多层感知机模型进行训练并推测用户 PIN 码。

3.1 整体架构设计

现有的很多手机软件都通过数字 PIN 码来认证用户，只有输入正确的 PIN 码才能使用。PIN 码由 4 位由 0 至 9 的数字组成序列，其中数字可以重复。iOS 和 Android 都支持使用 PIN 码作为手机锁屏后的解锁认证机制，除此之外还有图形解锁，指纹解锁、人脸识别解锁等方式，当使用上述额外解锁方式时，系统通常仍会要求用户设置 PIN 码作为备用解锁方式。PIN 码共有 10000 种可能的组合，主流手机系统都会在用户进行有限次失败尝试后锁定手机，防止被恶意用户暴力破解解锁手机。除此之外 PIN 码还被广泛用于银行账户、运营商帐号、电子钱包等领域作为安全认证手段。

用户在触摸屏上敲击输入数字 PIN 码的过程中，手机会发生位移，位移的角度和大小会因手指在屏幕不同位置敲击而不同，如图 3-1 所示，在手机边缘（a）敲击会比在手机中间（b）敲击产生更大角度的偏移。尽管肉眼观察并不明显，但现代智能手机基本都配备了加速度传感器、陀螺仪传感器、方向传感器等动作传感器，当用户在触摸屏上敲击时，这些动作传感器数据会精确的显示出这些不同点，其中包含着丰富的模式信息。

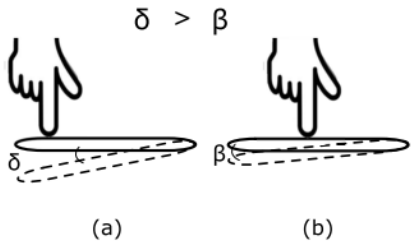

图 3-1 在屏幕不同位置敲击造成手机位移示意图

当用户在触摸屏上敲击输入数字按键时，一次按键动作可以分为按下和抬起这两个连续的过程，伴随着手指的按下与释放，手机也会产生物理位移的加速和减速，内置的动作传感器数值都会有明显的变化[5]，图 3-2 显示了在敲击过程中加速度传感器的三个轴上的数据变化。因此利用用户在屏幕不同位置敲击所产生的具有不同模式的动作传感器数据来推测用户输入的 PIN 码是完全可行的，尤其是在 Web 环境下，由于 W3C 标准的规定，Web 上所有 type 为 number 的 input 输入框所弹出的键盘为标准九宫格数字键盘，在 Android 和 iOS 上都是一致的。这意味着即使用户使用不同的操作系统，但在输入 PIN 码时手指的相对位移大致是一样的，因此通过机器学习训练出模型后，对于绝

大部分市面上流行的智能手机都可能通过动作传感器数据推测出 PIN 码。

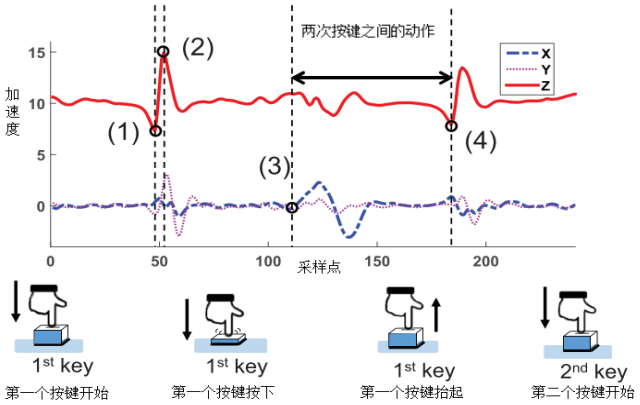

图 3-2 按键过程中加速度传感器三个轴上的数据变化[5]

另外 W3C 发布的 DeviceOrientation Event Specification 标准允许 Web 程序可以通过 JavaScript 直接访问手机上的动作传感器数据而无需经过用户许可[22]，当前的主流手机操作系统 Android 和 iOS 均支持这一标准，通过对 iPhone 6s 和小米 3 这两部手机进行真机测试，本文发现在读取传感器数据的过程中用户没有收到任何提示。根据 Maryam[23]对用户是否担心其手机内置的传感器会隐私泄漏的调查显示，只有 30%的用户知道动作传感器的作用，几乎 90%的用户表示并不觉得动作传感器会泄漏他们的隐私。因此利用 Web 程序通过手机浏览器来获取用户的传感器数据具有无需安装、跨平台、行为隐秘不易发现的特点，这也使这种攻击方式危害极大。

本文设计的利用动作传感器数据推测 PIN 码框架由三部分组成，分别是：①采集并存储手机内置动作传感器数据；②从传感器数据中提取特征向量；③建立并训练机器学习模型，从动作传感器数据中推测 PIN 码，整个框架的整体架构如图 3-3 所示。

整个框架的工作流程可以简述为：首先让一部分用户使用采集程序输入 PIN 码，存储这些传感器数据作为训练样本，从样本数据中提取特征向量，建立机器学习模型并使用特征向量进行训练，然后使用 XSS 攻击的方式在正常网页中嵌入恶意 JavaScript 代码，在用户输入 PIN 码时将相应的传感器数据传输至后台，通过先前训练好的机器学习模型推测出 PIN 码。

对于动作传感器数据的采集，本文提出一种新的方案，即通过 Web 客户端程序获取用户的传感器数据，通过网络实时传输至后端服务器。相较于已有研究，本文提出的方案有如下优点：现有研究均是在 Android 系统上编写原生应用来采集传感器数据，本文使用跨平台的 Web 程序，可在 Android、iOS 等主流手机操作系统中运行，在采集过程中无需用户许可，且由于 Hybrid 应用的流行，很多原生应用也开始支持运行外部 Web 程序，比如使用非常广泛的社交应用微信，本方案也可以在微信中运行，大大增加了受众面；由于移动网络的飞速发展，遍布各地的互联网接入已是现代城市的必备条件，本方案将传感器数据实时传输至后端服务器而不是存储在手机本地，可充分利用后端服务器强大的运算能力，避免手机本地运行机器学习模型耗费手机处理器以及电池资源，同时客户端、服务器模式的经典 CS 架构可以支持同时采集多台手机。

对于特征向量的提取，本文也提出一种新的方案，除了从传感器数据中提取时域特征外，本文改进语音信号处理中常用的 MFCC 算法，用其提取传感器信号的频域特征。相较于已有研究仅使用时域特征，本方案通过 MFCC 算法可以充分利用传感器信号中丰富的频域特征来提高后续机器学习的分类准确率。

对于机器学习模型的建立与训练，从传感器数据中推测出 PIN 码，其核心目标可以理解为对传

感器数据组成的时间序列进行分类，对应 0 至 9 这 10 个数字。本文提出一种新的方案，使用人工神经网络算法中一种叫做多层感知机（Multi Layer Perceptron）的分类算法建立模型，同时为了比较与其他分类算法的性能差异，本文也会使用 KNN、SMO、C4.5、LMT、Random Forest 算法建立分类模型并进行实验。为了提高效率，本文使用 Python 语言开发自动化模型训练与 PIN 码识别系统，相较于已有研究，本方案支持动态添加多种分类算法，可自动训练并生成丰富的图表结果，并可通过电子邮件远程查看结果。

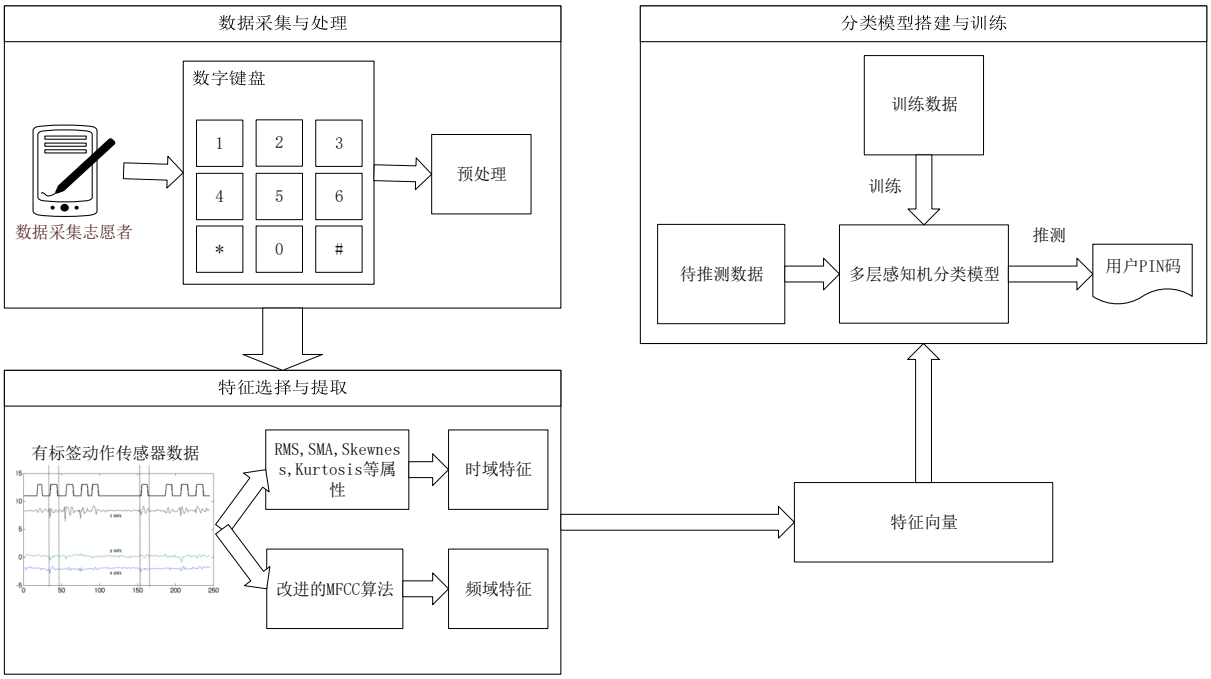

图 3-3 基于频域分析的按键推测通用框架整体架构图

3.2 基于跨平台 Web 程序的数据采集与处理

在进行手机传感器数据分析之前，要先获得相应的传感器数据，对于这类分析应用通常都有标准的数据集，如人脸识别有很多成熟的标注数据集，所有数据都已经按照说明处理并且打好标签，研究人员只需要关心算法在数据集上的表现即可。然而手机传感器数据目前还没有一个完善的公开数据集可供使用，因此需要自己采集。一般采集到的传感器数据都是原始数字信号，这些数据可能存在一些缺陷：如数据缺失（用户没有按照要求输入），数据异常（硬件故障、偶然情况）等，需要进一步的处理才能为模型使用，比如过滤异常值、对缺失的数据进行填充等操作，这类操作通常称为数据预处理。接下来从数据采集和预处理两个方面进行详细介绍。

3.2.1 数据采集

现有研究均是在 Android 系统上开发原生应用并调用系统提供的 API 来获取手机内置动作传感器的数据，使用这种方案需要先在手机上安装相应的软件，对于用户来说侵入性太强，使用体验并不好，当采集完成后还需要手动将传感器数据传输到电脑进行分析。除此之外，该应用只能在 Android 系统运行，并不支持跨平台使用，因此对于 iOS 手机或其他操作系统的手机来说，想要进行类似实验就得基于特定操作系统重新开发，费时费力且效率不高，截至目前，本文尚未发现针对 iOS 等其他手机操作系统的类似研究。

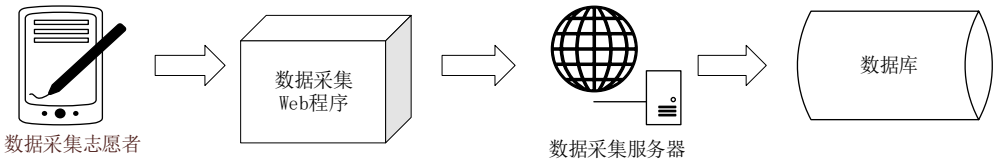

图 3-4 数据采集流程示意图

本文提出的新方案是使用 W3C 颁布的 Web 标准 API 来获取手机内置传感器数据，通过 Web 采集传感器数据，不仅有着跨平台的巨大优势，对于采集用户来说也无需安装特定应用，同时还可将采集到的传感器数据实时发送至远程服务器并在数据库中存储，支持同时采集多台手机。除此之外，采集程序可远程更新，无需用户手动操作。整个数据采集流程如图 3-4 所示。

采集训练数据时需要准备样本 PIN 码，为了各个数字在 PIN 码训练集中出现的频率保持一致，为此需要使用程序自动生成随机 PIN 码，以便保证统计特性的稳定。本文使用监督机器学习，需要保证实验中收集的数据均为正确输入 PIN 码后的动作传感器数据，如果用户输入错误，需要将这个 PIN 码对应的所有传感器数据删除，防止污染训练集。为了保证采集传感器数据在时序上的顺序，除了记录手机的动作传感器数据外还应记录相应的时间戳信息。

考虑到不同手机操作系统上传感器的表现可能具有差异性，为此本文选择在 Android 系统和 iOS 系统上采集传感器数据，它们是当今应用最为广泛的手机操作系统，在当前市场有相当庞大的用户基数，具有较好的代表性。

采集数据的志愿者对于实验结果也有影响，不能只选一个人，因为每个人的输入习惯不一样，有人偏好左手输入、有人偏好右手输入、还有人喜欢双手输入。如果只采集一个人的数据，那训练出的分类器很难有良好的泛化能力，甚至可能将实验人员个人的输入习惯特性误当作传感器数据的特性。为此需要多名不同性别的志愿者协助采集训练数据，同时应当在静止的状态下采集数据，以便排除外界因素的影响。为了尽可能模拟真实生活环境中的数据，在数据采集过程中，志愿者可以使用自己平常使用的方式握持手机，比如单手握持手机并输入；一只手握持，另一只手输入；双手输入等。

3.2.2 预处理

智能手机内置的传感器是一种微型的电子元器件，它能够感应手机物理姿态的变化，并将其转换为电信号，然后通过 AD 转换为数字信号输出。在采集数据的过程中或多或少都会有噪声干扰，噪声来源主要有两种，一是手机内置传感器自身的精度，二是用户在握持手机时手的抖动。因此需要对采集到的原始数据进行降噪处理，在保留重要信息的同时尽可能减少噪声对结果分析的影响。为此在预处理阶段会通过归一化方法，利用缩放和平移参数对数据集中属性进行规范化处理，从而达到降噪的目的。

为了有效的提取特征向量，需要保证每次按键敲击都采集到同等数量的数据记录，然而因为两个因素导致很难达到这个要求。第一，每次敲击可能持续不同的时间，因为每个人按键的习惯不同，有的按键比较轻，也就是说按键持续时间短，采集到的数据记录就少；有人按键比较重，持续时间长，采集到的数据记录就多。第二，由于读取传感器数据是经过手机操作系统提供的接口，因此当系统处于高负载状态时，可能导致采集到的数据记录变少。图 3-5 展示了实验采集的 PIN 码样本数量统计直方图，其中横轴表示输入一个 PIN 码时采集到的传感器数据点数，纵轴为有相应点数的 PIN 码数量，从中可以看出，大部分样本采集到的数据记录长度集中在 70 到 120 之间，但也有一些异常

值。

除此之外，不同的手机一般应用不同的传感器芯片，不同的操作系统也会使用不同的采样率，为了尽量消除这些差异性，本文使用三次样条插值法[24]来预处理采集到的动作传感器数据，让每次按键敲击都有同样数量的记录。

样条插值是使用一种名为样条的特殊分段多项式进行插值的形式，这一概念起源自上世纪初的工业设计行业，工程师和设计师用细木条将几个间断点连接成一条光滑的曲线，样条在手工业、造船业的图纸设计上都有广泛的应用。由于样条插值可以使用低阶多项式样条实现较小的插值误差，这样就可以避免使用高阶多项式所出现的龙格现象。

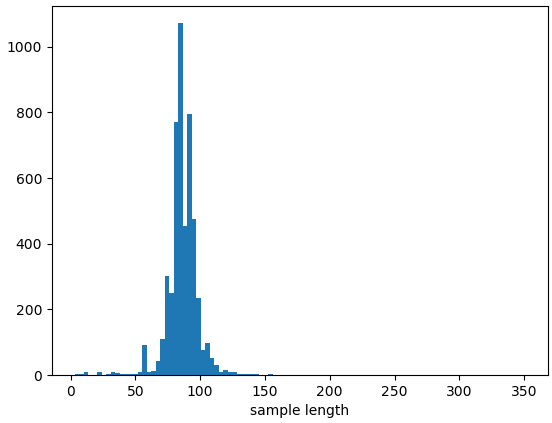

图 3-5 PIN 码样本长度统计直方图

样条函数的定义如下：设区间$[a,b]$上给定一个节点（knot）划分：

$$a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b \tag{3-1}$$

如果存在正整数k使得$[a,b]$上的分段函数$s(x)$满足：

(1) 在$[a,b]$上有直到$k-1$阶的连续导数;

(2) 在每个小区间$[x_i, x_{i+1}]$上都为次数不大于k的多项式。

则称分段函数$s(x)$是区间$[a,b]$上的k次样条函数。如果给定函数$f(x)$在式(3-1)中所示的节点$x_0, x_1, \ldots, x_{n-1}, x_n$处的函数值为：

$$f(x_j) = y_j, j = 0,1,\ldots,n \tag{3-2}$$

而样条函数$s(x)$满足：

$$s(x_j) = y_j, j = 0,1,\ldots,n \tag{3-3}$$

则称样条函数$s(x)$为函数$f(x)$在区间$[a,b]$上的k次样条插值函数。当k取 3 时，函数$s(x)$为三次样条插值函数。

图 3-6 展示了加速度传感器的 X,Y,Z 三个轴上的数据在三次样条插值前和插值后的图像，从中可以看出通过插值算法补齐了时域上的不连续点，并利用三次样条函数的一阶导数和二阶导数都连续的特性将采集到的传感器数据在时域做了一次平滑滤波，方便后续进行特征提取操作。

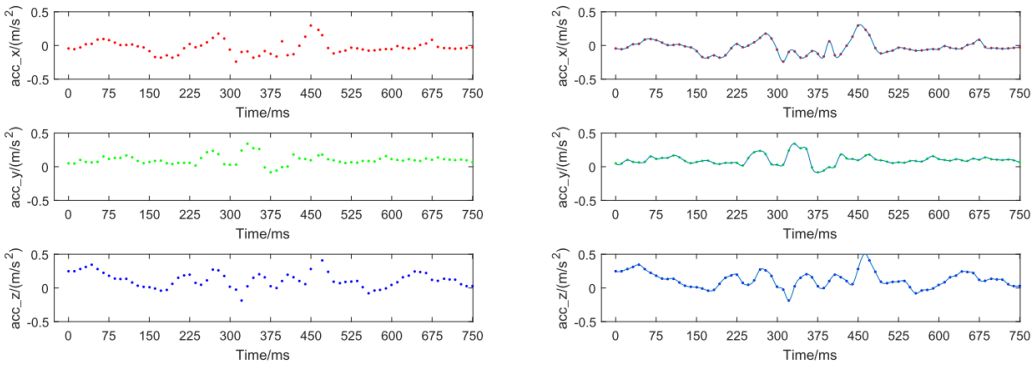

图 3-6 对加速度传感器数据进行三次样条插值的前后对比

3.3 基于时域频域相结合的特征选择与提取

本节详细讨论从手机动作传感器数据中提取的特征向量，它是后续机器学习分类器的输入向量。正如前文数据采集小节中所描述的，本文从手机的三种动作传感器中共采集到 acc（加速度）、ori（方向）、accG（带重力的加速度）、rot（角速度）等 4 种类型的数据，每一种都包含 3 个轴（x,y,z 或 α，β，γ），所以每一个样本都由 12 个向量组成。

为了后续描述方便，在此作如下定义：

$$Acc^d = \{acc_1^d, \dots, acc_n^d\}, \mathrm{d} \in [x, y, z] \tag{3-4}$$

$$GAcc^d = \{gacc_1^d, \dots, gacc_n^d\}, \mathrm{d} \in [x, y, z] \tag{3-5}$$

$$Ori^d = \{ori_1^d, \dots, ori_n^d\}, \mathrm{d} \in [\alpha, \beta, \gamma] \tag{3-6}$$

$$Rot^d = \{rot_1^d, \dots, rot_n^d\}, \mathrm{d} \in [\alpha, \beta, \gamma] \tag{3-7}$$

其中Acc^d表示 x, y, z三个方向上去除重力的加速度数据，单位为m/s^2；$GAcc^d$表示 x, y, z三个方向上带重力的加速度数据；Ori^d表示α, β, γ三个方向上的角度，单位为rad；Rot^d表示α, β, γ三个方向上的角速度，单位为rad/s。

定义采集输入一个 PIN 码时动作传感器的数据为：

$$\mathrm{Sensor} = \{Acc^d, GAcc^d, Ori^d, Rot^d\}, \mathrm{d} \in [(x, y, z)|(\alpha, \beta, \gamma)] \tag{3-8}$$

$$\mathrm{Sensor} = \left\{acc_i^x, acc_i^y, acc_i^z, gacc_i^x, gacc_i^y, gacc_i^z, ori_i^\alpha, ori_i^\beta, ori_i^\gamma, rot_i^\alpha, rot_i^\beta, rot_i^\gamma\right\}, \mathrm{i} \in [1, \mathrm{n}] \tag{3-9}$$

因此一个样本的原始数据可以看作一个矩阵：

$$D = N \times M \tag{3-10}$$

其中N代表采样的点数，与采样频率和采样时间有关；M代表 12 列传感器数据。

本文建立的特征向量共包含 294 个特征，分别提取自时域和频域。这些特征将尽可能多的从按键这一物理动作所引发的传感器数据变化中提取信息。其中有些特征跟按键动作的能量相关，其他一些特征则是计算传感器向量的变化率，比如一阶导数，接下来将从时域和频域两个维度详细描述本文使用的特征。

3.3.1 时域特征选择和提取

在屏幕不同位置按键会产生不同的传感器数据变化模式，因此选择的特征应尽可能多的获取传感器数据的各种特性。总体来说，本文选择两种类型的特征，分别是列特征和矩阵特征，如图 3-7 所

示。

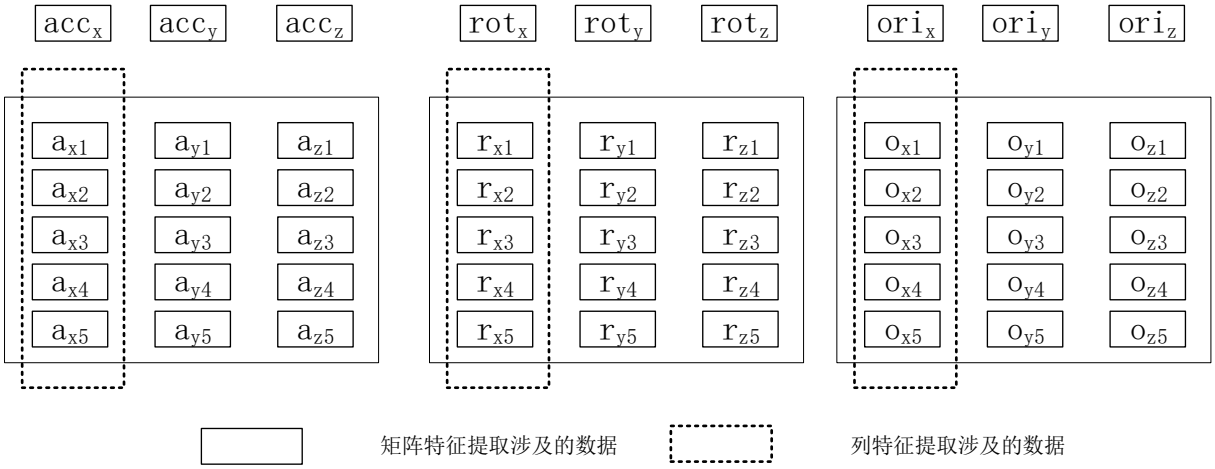

图 3-7 特征提取示意图

列特征是指那些从传感器每个轴的数据中提取到的特征，本文使用的列特征如表 3-1 所示，对 12 个向量分别求其最大值、最小值、均值、均方根值、均方根误差、波峰数、波谷数等 13 统计种特征，共计 156 个特征，下面简要介绍一下选取的这些特征的数学原理。

表 3-1 统计特征

特征	描述
Max	最大值
Min	最小值
RMS	均方根值
RMSE	均方根误差
Mean	均值
PeakNum	波峰数
TroughNum	波谷数
SMA	信号幅度区
Kurtosis	曲线的锐度
Skewness	统计分布中的不对称性
ATP	平均到达波峰时间
ATT	平均到达波谷时间
TotalEnergy	序列平方和

均方根误差（Root Mean Square Error），RMSE 是一种常用于度量模型估计值和实际观察值间差异的统计量。具体来说，均方根误差表示预测值和观察值的样本标准差的差值。均方根误差这一属性能够将在各个时间的计算误差聚合成单次预测误差。

对于参数θ的估计值$\hat{\theta}$的均方根误差被定义为其均方差的平方根：

$$\mathrm{RMSE}(\hat{\theta}) = \sqrt{MSE(\hat{\theta})} = \sqrt{E[(\hat{\theta} - \theta)^2]} \tag{3-11}$$

对于一个无偏估计，其 RMSE 为其方差的平方根，即其标准差。

对于一个以时间t为自变量的回归函数y_t，若其预测值为$\hat{y_t}$，则其均方根误差是n次不同预测的预测误差的方均根值：

$$\mathrm{RMSE}=\sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t-y_t)^2}{n}} \tag{3-12}$$

偏度（Skewness），在概率学中，偏度用于考察随机变量的概率分布偏差情况。而在统计学中，偏度主要用于衡量曲线的非对称性。直观上来说，偏度分为两种形态：负偏态/左偏态：曲线的左侧扫尾更长。局部峰的主体集中于区间右侧；正偏态/右偏态：曲线的右侧扫尾更长，局部峰的主体集中于区间左侧；图 3-8 可以直观地反映出这两种形态的差别。

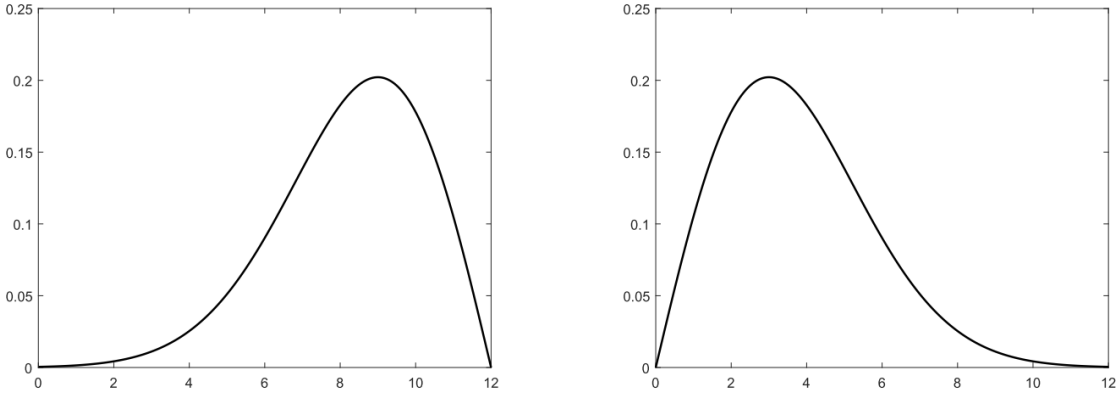

图 3-8 偏度示意图

随机变量χ的偏度即为三届标准矩γ_1，其定义为：

$$\mathrm{Skew}[\chi]=\mathrm{E}\left[\left(\frac{\chi-\mu}{\sigma}\right)^3\right]=\frac{\mu_3}{\sigma^3}=\frac{E[(\chi-\mu)^3]}{(E[\chi-\mu^2])^{\frac{3}{2}}}=\frac{\kappa_3}{\kappa_2^{\frac{3}{2}}} \tag{3-13}$$

上式中，μ_3表示χ的三阶中心距，σ表示χ的标准差，E是期望算子。等式中的κ_k表示χ的k阶累积量。

通过对式(3-13)的扩展，也可以用非中心矩$\mathrm{E}[\chi^3]$来表示偏度，见下式：

$$\mathrm{Skew}[\chi]=\mathrm{E}\left[\left(\frac{\chi-\mu}{\sigma}\right)^3\right]=\frac{E[\chi^3]-3\mu\sigma^2-\mu^3}{\sigma^3} \tag{3-14}$$

峰度（Kurtosis），在概率统计学中，峰度是一种描述随机变量分布聚集或分散程度的属性。与偏度一样，峰度也是一种高度概括曲线形状的参数，如图 3-9 所示。峰度的具体度量方法有很多，本文主要采用的是常见的一种算法。在这种算法中，本文用变量的四阶标准矩来代表变量的峰度，其定义如下：

$$Kurt[\chi]=\frac{\mu_4}{\sigma^4}=\frac{E[(\chi-\mu)^4]}{(E[(\chi-\mu)^2])^2} \tag{3-15}$$

其中，μ_4是随机变量χ的四阶中心距，σ是χ的标准差。

如果仿照偏度的定义方法，采用累积量来定义峰度，也可以有如下的定义式：

$$Kurt[\chi]=\frac{\mu_4}{k_2^2}=\frac{E[(\chi-\mu)^4]}{(E[(\chi-\mu)^2])^2}-3 \tag{3-16}$$

按照式(3-15)的计算方法，标准正态分布的峰度为 3，为了校正这样的偏差，式(3-16)中将其减 3，使得标准正态分布的峰度系数为 0。峰度的计算通常有校准前和校准后两套计算公式，对于独立同分布的随即序列x_i，其均值为$\bar{x}$，校准前的公式如下：

$$\mathrm{k}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^4}{\left[\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2\right]^2} \tag{3-17}$$

校准后的公式如下：

$$\mathrm{k}_0 = \frac{n-1}{(n-2)(n-3)}[(n+1)\mathrm{k}_1 - 3(n-1)] + 3 \tag{3-18}$$

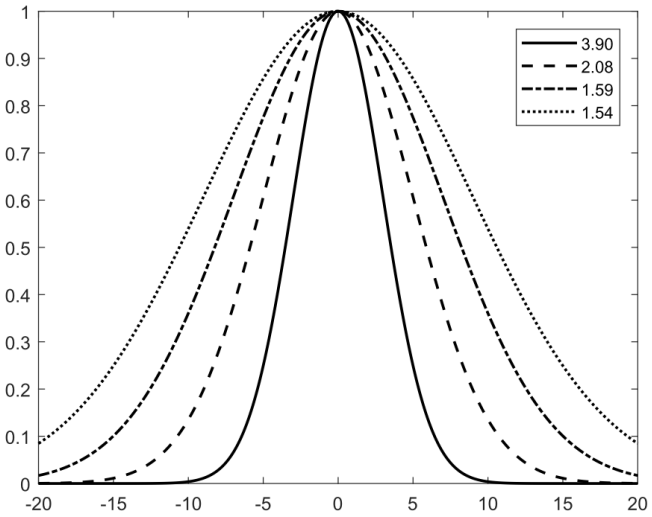

图 3-9 峰度示意图

矩阵特征负责提取传感器三个轴上数据的互相关关系，由一系列矩阵范数组成，主要由：1-范数（矩阵中每列元素的绝对值求和后的最大值），Infinity 范数（矩阵每行元素绝对值之和的最大值），Frobenius 范数（矩阵中所有元素的平方和的平方根）。除此之外考虑到不同的传感器的各个轴之间数据的相互关系，本文使用相关系数计算了(ori，acc)、（ori，accG）、（ori、rot）、（acc，accG）、（acc，rot）、（accG,rot）6 种组合在 3 个轴（$\mathrm{x,y,z}$ 或 α,β,γ）的相关值，共计 18 个特征。

相关系数经常用做比较两个信号形状的相似度[25]，给定两个序列 A 和 B，$\mathrm{Cov(A,B)}$表示 A 和 B 之间的协方差，相关系数可以用式(3-19)式计算。

$$R_{AB} = \frac{Cov(A,B)}{\sqrt{Cov(A,A)\cdot Cov(B,B)}} \tag{3-19}$$

3.3.2 改进的 MFCC 算法

为了从动作传感器数据提取更多信息，除了时域特征外，通常还会从频域提取特征。然而针对在屏幕上敲击按键后产生的动作传感器信号的研究并不多，语音信号作为同样是人体物理运动产生的信号，已经有相当丰富的研究成果了[26]。尤其是语音识别中的特征提取在最近几年取得了重要的进展，其中最常用的几种特征提取方法有 Mel-Frequency 倒谱系数（MFCC）、线性预测编码（LPC）、线性预测倒谱系数（LPCC）、感知线性预测倒谱系数（PLPCC）等[27]。MFCC 是语音信号处理中最流行也是最常见的一种算法，经常用于语音识别系统中[28]。由于语音信号和动作传感器信号有着很多相似性，因此可以改进 MFCC 算法来帮助提取动作传感器的频域特征。接下来从语音信号和动作传感器信号之间的相似性、MFCC 简介、MFCC 改进等方面做出详细介绍。

MFCC（Mel-Frequency Cepstrum Coefficients）的中文全称是 Mel-Frequency 倒谱系数，它相较于普通的频率倒谱分析最大的不同是：MFCC 主要针对人耳的听觉特性做出优化，聚焦于低频高分辨率分析。人耳的生理构造特性决定了人们听到的声音高低和声音频率并不成线性关系，这就要求

需要定义新的频率单位，这个单位应该满足在低频上有较高的分辨率，在高频上有较低的分辨率，从而符合临界带宽的特性。为此语义学引入了音调（Pitch）的概念，简单来说，人们通常对频率低的声音感觉音调低，对频率高的声音感觉音调高。由于音调和声音频率并不成线性关系，因此引入了Mel标度，规定音调的单位为Mel，通常将频率为1000Hz 60dB的纯音所产生的音调定为1000Mel。

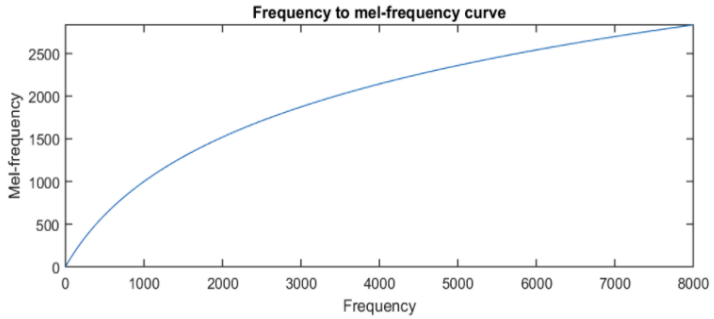

图 3-10 Mel 频率与实际频率的映射图

MFCC 算法利用人耳对高于 1000Hz 以上频率的信号并不敏感这个音调特性，使用 Mel 频率刻度对实际频率进行弯折，即对于 0 到 1000Hz 的信号进行线性映射，对高于 1000Hz 的信号进行对数映射，如图 3-10 所示，通过这种弯折来模拟人的耳朵所听到的声音的高低和声音频率之间的非线性关系。求一般的倒谱系数时，会得到在频率轴上等间隔分布的 FFT 谱线，与此不同的是，MFCC 使用一组符合临界频带分布的带通滤波器序列来对 FFT 的谱线进行滤波，通过该滤波器序列来模拟人耳听觉的非线性特性，这些滤波器的中心频率在实际频率轴上是不等间隔分布的，而在以音调单位 Mel 的频率轴上等间隔分布。由于 MFCC 模拟了人耳的听觉特性，因此在有噪声和频谱变形的语音信号处理中，通过提取 MFCC 特征参数可以获得更好的训练效果，在语音识别和说话人识别中，使用 MFCC 作为特征参数通常会比 LPCC 得到更高的正确率[29]。

语音信号和动作传感器信号都由人体物理动作产生，它们在几个方面有着重要的相似性，接下来从三点详细说明这些相似特性。

第一，语音信号和敲击屏幕产生的动作传感器信号都在频域包含相关信息[30]。如今绝大多数的语音处理系统都从频域获取特征向量[31]，而敲击屏幕产生的惯性信号也在频域包含相关信息，这一点已经在人体活动识别的相关实验中证实[32]，通过包含频域特征显著的提升了识别结果的准确性。

第二，语音信号和动作传感器信号的能量都集中在低频分量中[33]。尽管这两种信号的频率范围不同，但信号的能量却都集中在低频部分。在语音信号处理中，通常会对低频部分进行多种频率弯折处理以便在低频部分获得更高的分辨率，对于动作传感器信号，也有研究宣称具有这种特性[34]。

第三，语音信号和动作传感器信号都有较低的重现性[35]。尽管让同一个人多次说同一句话，产生的语音信号也会各不相同，与之类似的，让同一个人在触摸屏上多次输入同一个 PIN 码，产生的动作传感器信号也不相同。因此，分析和处理这类信号的系统必须对这种变化有较高的鲁棒性[36]。具体来说，使用实验人员收集的传感器数据来训练系统模型后，对于陌生人产生的传感器数据，系统也要能够给出正确结果。因此，特征提取模块必须包含可变频率模式以便正确处理训练场景与测试场景下信号的差异。

由于语音信号和动作传感器信号之间存在的这些重要相似特性，因此可以尝试使用 MFCC 算法来对动作传感器信号进行频域特征的提取，然而标准的 MFCC 算法是针对语音信号提出的，语音信号的频率与动作传感器的频率范围并不相同，频谱特征也有差异，为了更好的提取频域特征，需要对 MFCC 算法流程做出改进，以便适配动作传感器信号。

图 3-11 展示了标准 MFCC 算法的主要步骤，接下来详细介绍本文为了将 MFCC 算法用于传感

器信号所作出的重要改进。

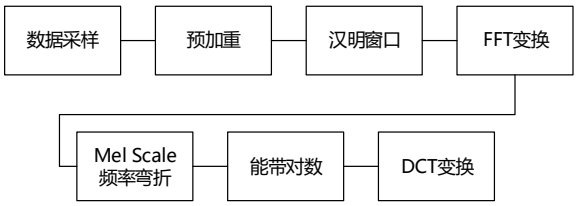

图 3-11 标准 MFCC 算法的主要流程

（1） 去除预加重

预加重的目的是放大信号中的高频部分用以补偿由于对高频部分的抑制，从而让频谱更加平坦。预加重事实上是一个高通滤波器，它的转移方程是：

$$\mathrm{H(z)} = 1 - \mathrm{k}z^{-1} \tag{3-20}$$

式(3-20)中k的范围在 0.9 到 1.0 之间，通常设为 0.97。可以使用如下方程在时域完成这种变换：

$$S_n' = S_n - k \times S_{n-1} \tag{3-21}$$

预加重本来是用于补偿语音信号频谱中的快速衰退，在动作传感器信号中，频率的变化范围相较于语音信号要小得多，且根据下文 4.5.4 的实验测试发现，去除预加重步骤可提高训练的准确率，因此在改进的 MFCC 算法中去除了预加重步骤。

（2） 去除汉明窗口

语音信号处理中 MFCC 算法需要对原始数据的每一帧都经过汉明窗口处理。由于原始数据的边缘并不光滑，可能会影响频域特性的提取，通过对信号中的每一帧都乘一个光滑窗口函数，可以将两个边缘都平滑到 0。同时汉明窗口也可以减少傅立叶变换后旁瓣的强度。这一步骤可以通过式(3-22)实现：

$$S_n' = [0.54 - 0.46\cos(\frac{2\pi(n-1)}{N-1})] \times S_n \tag{3-22}$$

由于本文研究的传感器信号在采集过程中已经由各个按键天然分割，不需要再进行加窗，所以在改进的 MFCC 算法可以省去这一步骤。

（3） Mel 频率弯折适配

在这一步骤中将线性标尺转为 Mel 标尺，即基于人类听觉的非线性频率标尺。由于 MFCC 原本是为声音识别设计的，所以将频域范围设置为 0 到 4kHz，根据奈奎斯特采样定理，将采样率设为 8kHz。原始的 Mel 标尺到线性频率标尺的转换可以由式(3-23)表示：

$$\mathrm{Mel}(f) = 2595 \times log_{10}(1 + \frac{f}{700}) \tag{3-23}$$

其中 2595 是通过式(3-24)得到：

$$\frac{1000}{log_{10}(1+\frac{1000}{700})} \tag{3-24}$$

当$f = 1000Hz$时，$\mathrm{Mel}(f) = 1000$，因此式(3-23)可以重写为：

$$\mathrm{Mel}(f) = 1000 \times \frac{log_{10}(1+\frac{f}{700})}{log_{10}(1+\frac{1000}{700})} \tag{3-25}$$

为了在动作传感器数据分析中使用这个方法，需要修改 MFCC 公式以便适配新的频率范围，考虑到实际传感器采样频率，可以将公式修改为如下形式：

$$\mathrm{Mel}_{adp}(f) = 1000 \times \frac{log_{10}\left(1+\frac{f}{8.75}\right)}{log_{10}\left(1+\frac{12.5}{8.75}\right)} \tag{3-26}$$

进一步可化简为如下形式：

$$\mathrm{Mel}_{adp}(f) = 32.438 \times log_{10}(1+\frac{f}{8.75}) \tag{3-27}$$

3.3.3 频域特征选择和提取

频域的特征有助于描述敲击按键后手机产生的振动中蕴含的信息。本文使用改进的 MFCC 算法提取动作传感器信号的 MFCC 系数作为频域特征，具体来说就是对原始动作传感器信号的 12 个向量通过改进 MFCC 算法提取 10 个系数，得到共计 120 个频域特征。此处提取系数的个数也会影响分类准确率，10 这个数是由后文 4.5 节中实验得出。图 3-12 展示了改进后的 MFCC 算法流程，下面对提取流程做详细说明。

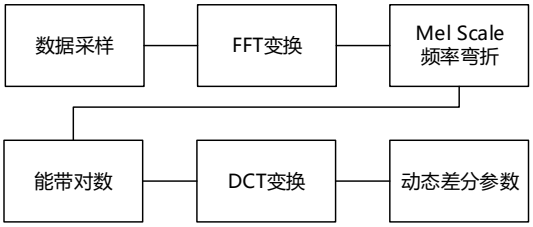

图 3-12 改进 MFCC 算法的主要流程

（1） FFT 变换

对原始信号的每一帧都进行快速傅立叶变换来获得频谱。一般只关心幅度谱和功率谱，因为帧间隔是固定的所以忽略相位谱。幅度谱可以用如下公式计算：

$$S_i(\mathrm{k}) = \sum_{n=1}^{N} S_i(n) e^{-\frac{j2\pi kn}{N}} \tag{3-28}$$

功率谱可以通过如下公式计算：

$$P_i(k) = \frac{1}{N}|S_i(k)^2| = \frac{1}{N}\left|\left[\sum_{n=1}^{N} S_i(n) e^{-\frac{j2\pi kn}{N}}\right]^2\right| \tag{3-29}$$

（2） Mel 频率弯折

在这一步骤中将线性标尺转为 Mel 标尺，为了将 MFCC 用于动作传感器信号，本文已将 Mel 标尺到线性频率标尺的转换做了适配，可以由式(3-30)表示：

$$\mathrm{Mel}_{adp}(f) = 32.438 \times log_{10}(1+\frac{f}{8.75}) \tag{3-30}$$

当信号经过标尺转换后将通过一组三角形滤波器组，它们在低频部分分布稠密，在高频部分稀疏，这是因为动作传感器的信号能量和有用的信息都集中在低频部分。

（3） 能带对数

在这一步骤中将计算频谱的对数，由于高频部分的信号能量相对于低频部分要低，所以取对数可以放大能量上的差异。式(3-31)中的 M 表示滤波器的个数，H_m代表三角滤波器的频率响应。

$$\mathrm{S(m)} = \ln(\sum_{k=0}^{N-1}|X_a(k)|^2 H_m(k)), 0 \le m \le M \tag{3-31}$$

（4） 离散余弦变换

最后一步是计算离散余弦变换，在这步之前，使用傅立叶变换来进行倒谱计算。MFCC 使用 DCT

（Discrete Cosine Transform）是因为计算结果只有实部而没有虚部，除此之外，对于动作传感器信号，DCT 可以进一步压缩特征集并以较少的系数获得有用的数据。DCT 系数可以通过式(3-32)计算：

$$C_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} Mel_j \cos\left[\frac{\pi i}{N}(j-0.5)\right] \tag{3-32}$$

（5） 动态差分参数提取

标准的 MFCC 只能提取信号中的静态属性，由于动作传感器信号在时域是连续的，如果只用标准的 MFCC 算法则只能反映每一帧信号的属性。因此需要通过计算动态差分系数来反映原始信号时域中的连续特征，差分系数可以通过一下式(3-33)计算：

$$d_i = \frac{\sum_{n=1}^{N} n(c_{i+n} - c_{i-n})}{2\sum_{n=1}^{N} n^2} \tag{3-33}$$

3.4 基于多层感知机的分类模型搭建与训练

在 3.3 小节中本文从时域和频域两个维度对传感器信号进行分析，从中提取出特征向量。在采集传感器数据的时候就已经对每个样本进行了 PIN 码标注，因此有了标注好的训练集，其中每一个样本都对应一个 294 维度的特征向量和相应的 PIN 码。此时从传感器数据中推测 PIN 码的问题就变为对一个 294 维的向量进行分类的问题，随着机器学习的发展，业界已有很多成熟的分类算法，本文使用与现有研究不同的人工神经网络算法来解决这个分类问题，具体来说是其中一种叫做多层感知机（Multi Layer Perceptron）的算法。在这一节中将详细描述本文所使用的模型的搭建以及训练过程。

3.4.1 多层感知机

多层感知机是人工神经网络算法中的一种，为此有必要先介绍人工神经网络算法。人工神经网络算法简称 ANN 算法，该算法受生物学中的神经网络所启发，由一系列称为神经元的节点组成，节点之间的连接用来传递信号，每个节点都单独处理信号。神经网络中计算的基本单元是神经元，一般称作节点（node）或者单元（unit）。节点从其他节点接收输入，或者从外部源接收输入，然后计算输出。每个输入都有权重（weight），权重取决于其他输入的相对重要性。节点将函数f应用到加权后的输入总和，如图 3-13 所示。其中函数f是非线性的，叫做激活函数。激活函数的作用是将非线性引入神经元的输出。因为大多数现实世界的数据都是非线性的，本文希望神经元能够学习非线性的函数表示，所以这种应用非常重要。

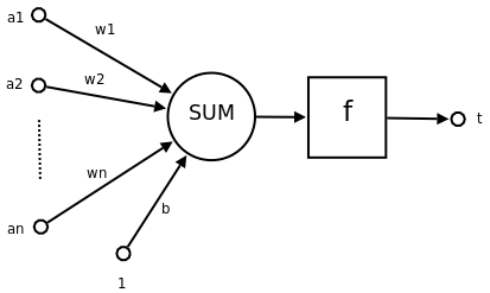

图 3-13 单个神经元结构

本文使用前馈神经网络，信息只单向移动，从输入层开始前向移动，然后通过隐藏层，再到输出层，在网络中没有循环或回路。在一个前馈神经网络中包含三种类型的节点，下面依次做出介绍：

（1） 输入节点（Input Nodes）：输入节点从外部世界提供信息，总称为输入层。在输入节点中，不进行任何的计算，仅向隐藏节点传递信息。

（2） 隐藏节点（Hidden Nodes）：隐藏节点和外部世界没有直接联系。这些节点进行计算，并将信息从输入节点传递到输出节点。隐藏节点总称为。尽管一个前馈神经网络只有一个输入层和一个输出层，但网络里可以没有也可以有多个隐藏层。

（3） 输出节点（Output Nodes）：输出节点总称为输出层，负责计算，并从网络向外部世界传递信息。

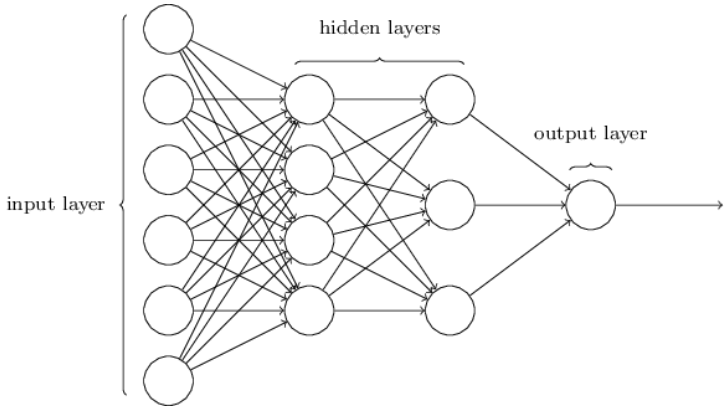

图 3-14 多层感知机结构

多层感知机是前馈神经网络中的一种，包括至少一个隐藏层（除了一个输入层和一个输出层以外），其结构如图 3-14 所示。单层感知器只能学习线性函数，而多层感知器可以学习非线性函数。

3.4.2 网络训练

本文使用反向传播算法（BackPropagation）来训练多层感知机网络，它是目前最常用且最有效的训练算法，其主要思想是：

（1）将训练集数据输入到网络的输入层，经过隐藏层，最后达到输出层并输出结果，这是多层感知机的前向传播过程；

（2）由于多层感知机网络的输出结果与实际结果有误差，则计算估计值与实际值之间的误差，并将该误差从输出层向隐藏层反向传播，直至传播到输入层；

（3）在反向传播的过程中，根据误差调整各种参数的值；不断迭代上述过程，直至收敛。

标准反向传播算法的流程如图 3-15 所示，在输入数据固定的情况下，反向传播算法利用神经网络的输出敏感度来快速计算出神经网络中的各种超参数。尤其重要的是，它计算输出 f 对所有的参数 w 的偏微分，即如下所示：$\frac{\partial \mathrm{f}}{\partial w_i}$，f 代表神经元的输出，$w_i$是函数 f 的第 i 个参数。参数$w_i$代表网络的中边的权重或者神经元的阈值，神经元的激活函数具体细节并不重要，它可以是非线性函数 Sigmoid 或 ReLU。这样就可以得到 f 相对于网络参数的梯度$\nabla \mathrm{f}$，有了这个梯度，就可以使用梯度下降法对网络进行训练，即每次沿着梯度的负方向$-\nabla \mathrm{f}$ 移动一小步，不断重复，直到网络输出误差最小。

3.4.3 激活函数的选择

神经网络中的激活函数是用来加入非线性因素，因为线性模型的表达能力不够，大多数现实世界的数据都是非线性的。每个非线性激活函数都接收一个数字，并进行特定、固定的数学计算。实践中常用的几种激活函数如下所示：

Sigmoid（S 型激活函数）：输入一个实值，输出一个 0 至 1 间的值。它可以将一个实数映射到(0,1)的区间，可以用来做二分类，在特征相差比较复杂或是相差不是特别大时效果比较好。但是在反向传播时，很容易就会出现梯度消失的情况，从而无法完成深层网络的训练。

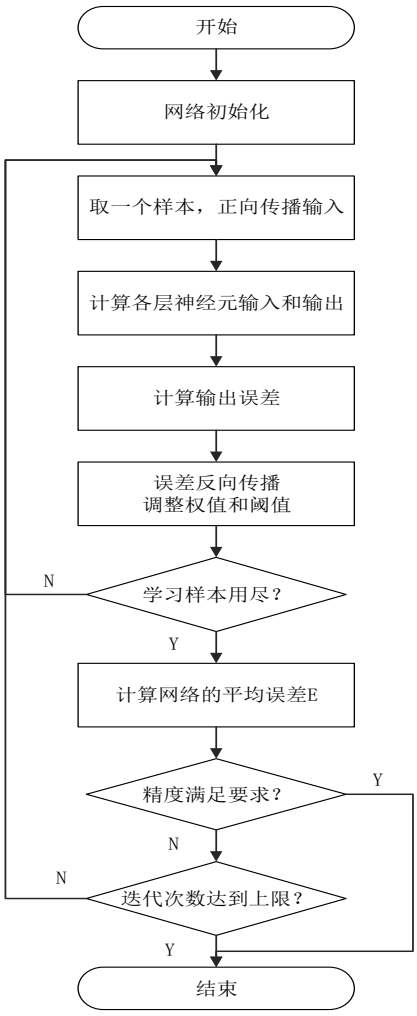

图 3-15 反向传播算法流程图

$$f(z)=\frac{1}{1+e^{-z}} \tag{3-34}$$

tanh（双曲正切函数）：输入一个实值，输出一个[-1,1]间的值。它在特征相差明显时的效果会很好，在循环过程中会不断扩大特征效果。

$$f(z)=\tan(h)=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \tag{3-35}$$

ReLU（The Rectified Linear Unit）：代表修正线性单元，输出一个实值，并设定 0 的阈值（函数会将负值变为零）。

$$\phi(x)=\max(0,x) \tag{3-36}$$

因为神经网络的数学基础是处处可微的，所以选取的激活函数要能保证数据输入与输出也是可微的。神经网络中，运算特征是不断进行循环计算，所以在每代循环过程中，每个神经元的值也是在不断变化的。这就导致了 tanh 特征相差明显时的效果会很好，在循环过程中会不断扩大特征效果显示出来。但有时候，特征相差比较复杂或是相差不是特别大时，需要更细微的分类判断的时候，sigmoid 效果会更好，因此 sigmoid 相比 tanh 用得更多。但近年发现数据有一个很有意思的特征，也就是稀疏性，数据有很多的冗余，而近似程度的最大保留数据特征，可以用大多数元素为 0 的稀疏矩阵来实现。Relu 就是取的 max(0,x)，因为神经网络是不断反复计算，实际上变成了不断试探如何用一个大多数为 0 的矩阵来尝试表达数据特征，因为稀疏特性的存在，这种方法运算得又快效果又好。

综上所述，本文使用 ReLU 作为激活函数，相较于 Sigmoid，ReLU 可以有效减少梯度消失的可能性，并且稀疏性也可以防止梯度爆炸。另外，使用 ReLU 还可极大的加快收敛速度，相比 tanh 可快 6 倍左右。

3.4.4 优化算法

在调整模型更新权重和偏差参数的方式时，通常会采用一些优化算法来使模型产生更好更快的效果，优化算法的功能，是通过改善训练方式，来最小化损失函数 E(x)。模型内部有些参数，是用来计算测试集中目标值 Y 的真实值和预测值的偏差程度的，基于这些参数，就形成了损失函数 E(x)。比如说，权重(W)和偏差(b)就是这样的内部参数，一般用于计算输出值，在训练神经网络模型时起到主要作用。在有效地训练模型并产生准确结果时，模型的内部参数起到了非常重要的作用。这也是为什么应该用各种优化策略和算法，来更新和计算影响模型训练和模型输出的网络参数，使其逼近或达到最优值。目前常用的一些优化算法有随机梯度下降（SGD）、动量（Momentum）、自适应时刻估计方法（Adam）等，下面做简要介绍。

随机梯度下降(SDG)：对每个训练样本进行参数更新，每次执行都进行一次更新，且执行速度更快。频繁的更新使得参数间具有高方差，损失函数会以不同的强度波动。这实际上是一件好事，因为它有助于发现新的和可能更优的局部最小值，而标准梯度下降将只会收敛到某个局部最优值。但 SGD 的问题是，由于频繁的更新和波动，最终将收敛到最小限度，并会因波动频繁存在超调量。

动量（Momentum）：SGD 方法中的高方差振荡使得网络很难稳定收敛，所以有研究者提出了一种称为动量的技术，通过优化相关方向的训练和弱化无关方向的振荡，来加速 SGD 训练。换句话说，这种新方法将上个步骤中更新向量的分量 γ 添加到当前更新向量。动量项 γ 通常设定为 0.9，或相近的某个值。这里的动量与经典物理学中的动量是一致的，就像从山上投出一个球，在下落过程中收集动量，小球的速度不断增加。当其梯度指向实际移动方向时，动量项 γ 增大；当梯度与实际移动方向相反时，γ 减小。这种方式意味着动量项只对相关样本进行参数更新，减少了不必要的参数更新，从而得到更快且稳定的收敛，也减少了振荡过程。

自适应时刻估计方法（Adaptive Moment Estimation，Adam）：Adam 优化算法是随机梯度下降算法的扩展式，近来其广泛用于深度学习应用中，尤其是计算机视觉和自然语言处理等任务。Adam 算法和传统的随机梯度下降不同。随机梯度下降保持单一的学习率更新所有的权重，学习率在训练过程中并不会改变。而 Adam 通过计算梯度的一阶矩估计和二阶矩估计而为不同的参数设计独立的自适应性学习率。Adam 算法同时获得了 AdaGrad 和 RMSProp 算法的优点。Adam 不仅如 RMSProp 算法那样基于一阶矩均值计算适应性参数学习率，它同时还充分利用了梯度的二阶矩均值。具体来说，算法计算了梯度的指数移动均值（exponential moving average），超参数 beta1 和 beta2 控制了这些移动均值的衰减率。移动均值的初始值和 beta1、beta2 值接近于 1（推荐值），因此矩估计的偏差接近于 0，该偏差通过首先计算带偏差的估计而后计算偏差修正后的估计而得到提升。

Adam 在深度学习领域内是十分流行的算法，因为它能很快地实现优良的结果。经验性结果证明 Adam 算法在实践中性能优异，相对于其他种类的随机优化算法具有很大的优势。因此本文搭建的多层感知机模型中也采用 Adam 算法作为优化算法。

3.4.5 超参数选择策略

搭建多层感知机模型时，涉及众多的超参数调整和配置，下面对这些超参数的调整做详细说明。

（1） 隐层以及每层神经元数量的选择

正如 3.4.1 小节所描述的，本文搭建的多层感知机由三部分组成，分别是输入层、隐藏层、输出层。其中输入层节点数量为从动作传感器数据中提取的特征向量维度，也就是 294；隐层数量的选择目前没有很好的理论依据，通常来说层数越多网络误差越低，精度越高，但也使网络结构越复杂，

这意味着训练时间会变长，同时有出现过拟合的倾向。本文使用一个隐藏层，因为通过增加隐层节点数也可以获得更低的误差，其训练效果要比增加隐层数量更容易提高。隐层节点数的选择非常重要，它不仅对建立的神经网络模型的性能影响很大，而且是训练时出现过拟合的直接原因，但是目前理论上还没有一种科学的和普遍的确定方法。若隐层节点数太少，网络可能根本不能训练或网络性能很差；若隐层节点数太多，虽然可使网络的系统误差减小，但一方面使网络训练时间延长，另一方面，训练容易陷入局部极小点而得不到最优点，也是训练时出现过拟合的内在原因。为尽可能避免训练时出现过拟合现象，保证足够高的网络性能和泛化能力，确定隐层节点数的最基本原则是：在满足精度要求的前提下取尽可能紧凑的结构，即取尽可能少的隐层节点数。通过多次实验，本文选择隐层节点数量为 200，可达到较好的准确率和泛化能力，同时也能满足训练时间的要求。

（2） 网络初始权值的选择

在搭建好神经网络模型后，通常需要对权值和偏置进行初始化，权值初始化方法主要有：常量初始化、高斯分布初始化、均匀分布初始化等方法，本文采用均值为 0，方差为 1 的标准正态分布来初始化权值。

（3） 训练 Batch Size 的选择

Batch Size（批尺寸）是机器学习中一个重要参数，批表示了全样本的部分抽样实现，相当于人为引入修正梯度上的采样噪声，批训练的引入最大好处是针对非凸损失函数来做的，可以避免陷入局部最优，一路不通找别路更有可能搜索最优值。一般来说应该在合理的范围内选择较大的 Batch Size，因为这样做可以有效提高内存利用率，在 GPU 上进行大矩阵乘法的并行化效率提高，且 Batch Size 越大，其确定的下降方向越准，引起训练的震荡越小。然而如果盲目增大 Batch Size 也会带来坏处，除了会急剧增加内存占用外，还有可能陷入局部最优，对参数的修正也会变得更加缓慢。经过多次尝试后，本文选择 256 作为 Batch Size，可达到训练速度与精度较好的平衡。

（4） 学习率的选择

运用梯度下降算法进行优化时，权重的更新规则中，在梯度项前会乘以一个系数，这个系数就叫学习速率。如果学习速率太小，则会使收敛过慢，如果学习速率太大，则会导致代价函数振荡。本文选择的学习速率为 0.01，实验发现效果良好。

（5） 最大迭代次数选择

由于神经网络计算并不能保证在各种参数配置下迭代结果收敛，当迭代结果不收敛时，允许最大的迭代次数就称为最大迭代次数，如果选择太小就有可能导致训练不充分，还未完成就已结束，需要根据不同的应用场景做出选择，本文选择最大迭代次数为 1000 次。

3.5 本章小结

本章首先对使用动作传感器推测 PIN 码进行可行性论证，随后提出了一个基于时域频域特征相结合的 PIN 码推测通用框架，其中包含传感器数据采集、特征向量的提取、机器学习模型的训练与 PIN 码推测。相较于已有研究，本框架有诸多创新：使用跨平台的 Web 程序采集 iOS 和 Android 平台下的多种动作传感器数据；结合动作传感器本身的信号特性，对语音信号处理中非常流行的 MFCC 方法进行了改进，并用其提取动作传感器信号的频域特征；使用多层感知机算法搭建分类模型，对整个模型的原理、结构、超参数选择策略、训练算法进行详细介绍。

第四章 PIN 码推测框架原型系统的实现

在第三章中，本文提出了一个基于时域频域特征相结合的 PIN 码推测通用框架，该框架中包含从手机内置传感器数据的获取，特征向量的提取，以及模型的搭建与训练这一整套流程，本章将设计并实现一个原型系统来验证并测试之前提出的 PIN 码推测通用框架。

4.1 系统设计

在 3.1 小节中详细描述了基于时域频域特征相结合的 PIN 码推测通用框架的方案设计，本文将按照该方案实现 PIN 码推测原型系统，该系统由三部分组成，分别是传感器数据采集系统、PIN 码推测概念验证程序、机器学习推测系统，如图 4-1 所示。

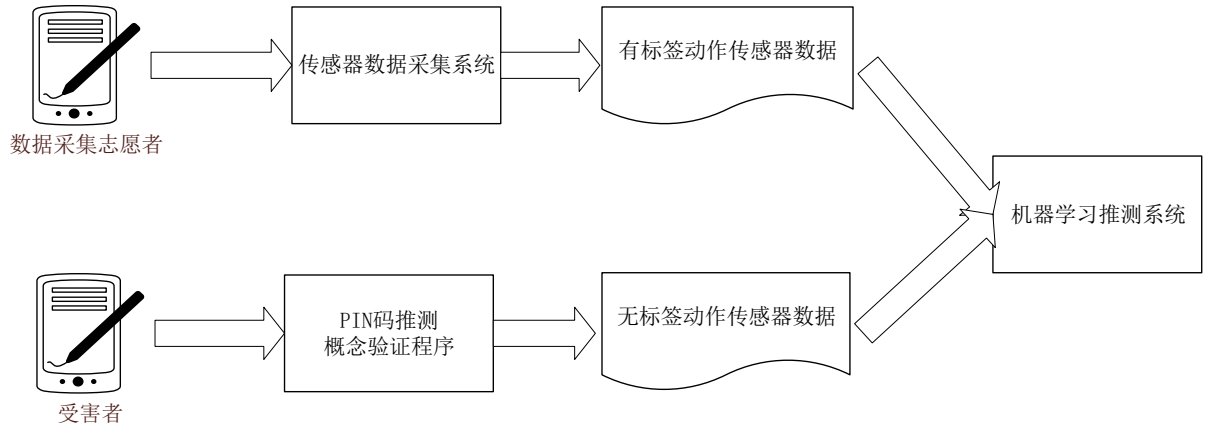

图 4-1 PIN 码推测原型系统架构图

首先需要通过传感器数据采集系统采集有标签的传感器数据，以便在机器学习推测系统中训练分类模型。在前面的章节中本文描述了使用机器学习的方法从智能手机内置的传感器数据中推测出 PIN 码。机器学习最不可或缺的就是数据了，可以毫不夸张的说，数据的获取、清洗、整理工作占据了整个机器学习流程中相当大一部分的时间，有良好的数据支持是进行后续机器学习工作的必要条件。为此本文设计并实现了跨平台传感器数据采集系统，旨在利用 Web 程序天然的跨平台能力，支持从各种搭载不同操作体系的手机中方便的收集动作传感器数据，并实时传输到后台服务器处理、存储，方便进一步的研究和使用。

PIN 码推测的核心在于机器学习推测系统，通过传感器数据采集系统拿到标注好的传感器数据后，还需要经过一系列的处理，包括预处理、特征提取、模型训练等步骤，本文使用 Python 语言实现自动化数据清洗、处理、提取特征，并利用 Scikit-learn 搭建多层感知机模型，为了与其他分类算法进行比较，本文设计了分类器的动态加载功能，可同时对多种分类模型进行训练。训练通常需要花费大量时间，为了提高效率，本文设计的机器学习推测系统可以支持批量处理训练任务，并将结果通过电子邮件远程发送。

为了演示在 Web 环境下程序无需用户许可即可获取传感器数据并发送到远程服务器导致 PIN 码泄漏的安全风险，本文设计并实现了 PIN 码推测概念验证程序。该客户端包含一个含有漏洞的模拟 Web 应用，该应用中需要输入数字 PIN 码，本文通过 XSS 攻击将恶意 JavaScript 代码嵌入到其中，当用户在微信等支持 WebView 的社交应用中打开后，会将传感器数据在用户不知情的情况下传输至

后端已经训练好的机器学习推测系统，从而推测出用户 PIN 码。

4.2 传感器数据采集系统

本节将从系统需求分析、系统架构和工作流程、手机客户端采集模块、服务端存储模块对整个传感器数据采集系统进行详细介绍。

4.2.1 系统需求与解决方案

传感器数据采集系统的设计目标是尽可能支持多种手机操作系统（如 iOS、Android、Windows Phone），并在用户无感知的前提下，采集手机内置加速度传感器、陀螺仪、地磁方向传感器的数据，并传输到远程服务器存储。下面依次列出整个系统的需求以及相应的解决方案：

（1） 跨平台支持

本文绪论部分提到，国外相关的研究均是通过编写原生 App，并安装到手机来采集数据。由于不同的操作系统开发语言和系统接口均不相同，为了方便通常只在 Android 平台采集数据，这对研究内容的完整性和丰富性有了限制。因此本系统使用 JavaScript 和 HTML5 编写基于 Web 的采集程序，有着优异的跨平台特性，在主流操作系统如 Android、iOS 上均可运行，甚至在支持 WebView 的 App（如微信）中也可以运行，最大程度的提高了传感器数据来源的范围，除此之外还有无需安装、实时更新、用户接受度高的特点。

（2） 用户无感知传感器读取

目前绝大多数智能手机都内置了动作传感器，而在主流操作系统中读取加速度传感器、陀螺仪、地磁方向传感器时需要向系统申请资源读取权限，本系统使用 W3C 颁布的 DeviceOrientation Event Specification 允许 Web 程序可以通过 JavaScript 直接访问手机上的动作传感器数据而无需经过用户许可。

（3） 传感器数据实时上传

为了证实恶意程序可以利用 XSS 攻击等方式获取传感器数据来推测用户隐私信息，本系统实现了传感器数据实时上传服务器来演示这种安全隐患。除此之外，有可能浏览器开启了隐私模式，导致不能在手机本地存储数据，也要求本系统支持数据实时上传到服务器，防止数据丢失。本系统使用 WebScoket 与服务端程序通信，节省了 HTTP 通信的包头开销，并实现了 Ajax 无法实现的高效双向通信，使得服务端可以向手机端传感器采集程序发送命令，让整个系统变得更加灵活强大。

（4） 多客户端数据采集

采集到的原始数据越多，对于后续机器学习就越有利，为了提升数据采集的速度，要求支持多个客户端并行采集数据。为此本系统实现了经典的 CS 架构，传感器数据采集程序可以运行在多台手机上，服务端程序运行在一台服务器上，利用 NodeJS 非阻塞 I/O 特性实现多客户端并发连接，通过非顺序执行 I/O 请求并采用回调来完成大量的数据写入请求。本系统支持不同操作系统的手机同时采集传感器数据，并且由于所有的数据都是通过网络传输，用户可在不同的地点采集数据，比如在室外采集运动时的传感器数据，极大的扩展了数据采集的应用场景。

4.2.2 系统架构与工作流程

本文设计的传感器数据采集系统采用经典的客户端、服务端架构，使用 Web 技术构建以便支持跨平台使用，兼容 Android、iOS 等主流手机操作系统。其中客户端是以 JavaScript 和 HTML5 编写的 Web 程序，它运行在智能手机浏览器上，不需要用户的安装，只需打开网页即可采集智能手机上的传感器数据。服务端主要使用 NodeJS 技术构建，负责向客户端下发具体的采集命令和相应配置，控制客户端的采集行为，以及实时接收客户端上传的传感器数据并进行格式化处理，然后持久化存储到数据库。整个传感器数据采集系统的架构如图 4-2 所示。

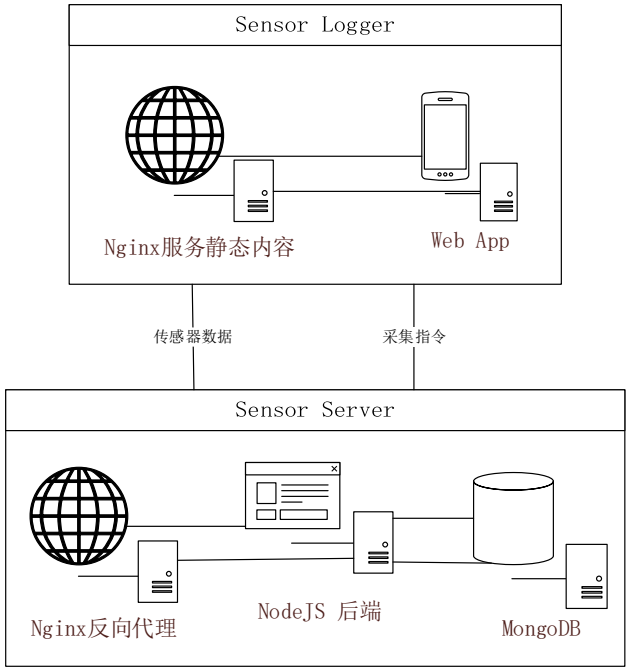

图 4-2 传感器数据采集系统架构图

Sensor Logger 部署在一台云服务器上，操作系统系统采用 Ubuntu 16.04 X64 Server 版本，由于此服务器只需要向用户提供采集程序，不需要其他数据传输，所以带宽选为 1M。整个系统由 Nginx 静态网页服务器和数据采集 WebApp 这两部分组成。静态网页服务由 Nginx 提供，通过设置 gzip 压缩和 HTTP 1.1 长连接支持，节省手机下载网页花费的流量。采集程序由 JavaScript 和 HTML5 编写，由于用到大量异步编程，所以使用了 babel 来支持 ES6 语法，并使用 Webpack 集中管理编译、优化、压缩、部署等任务流程。

Sensor Server 部署在一台云服务器上，操作系统系统采用 Ubuntu 16.04 X64 Server 版本，由于传感器数据采集程序需要将数据传输到此服务器，带宽要求较高，所以网络带宽选择 4M。整个系统由 Nginx 反向代理、NodeJS 后端服务、MongoDB 数据库这三部分组成。后端服务主要由 NodeJS 技术实现，利用非阻塞 I/O 特性实现多客户端并发连接，通过非顺序执行 I/O 请求并采用回调来完成大量的数据写入请求。当请求服务的客户端过多时，单个进程无法满足服务请求，此时可以通过多台服务器组成集群，利用 Nginx 进行反向代理来实现负载均衡。由于涉及大量的并发数据写入请求，服务端使用 MongoDB 来进行持久化存储，当单台服务器无法满足性能要求时，还可以通过多台 MongoDB 服务组成集群来线性扩展性能。

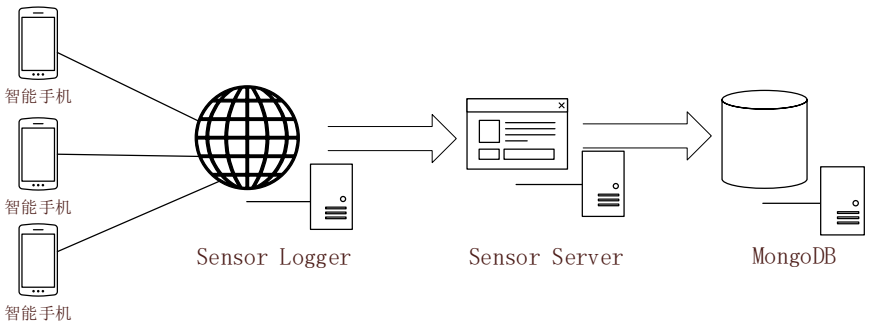

图 4-3 传感器数据采集系统工作流程图

整个采集系统的工作流程如图 4-3 所示，本文设计的传感器数据采集系统支持同时从多部智能

手机采集数据。首先由在智能手机浏览器上访问数据采集 Web 程序的地址，等到程序载入后会自动连接运行在远程服务器上的 Sensor Server 后端服务程序，向其请求采集命令；Sensor Server 在启动后会监听设置的端口，当有客户端向其发起连接时，由 Nginx 反向代理将请求转发到 NodeJS 服务，此时后端服务程序根据配置文件中的数据采集任务，向客户端发送具体的采集命令；Sensor Logger 接收到来自服务端的采集命令后，在浏览器上监听动作传感器事件，采集数据后实时上传到 Sensor Server，然后持久化存储到 MongoDB 中，至此整个数据采集流程结束。

4.2.3 Sensor Logger 实现

在上节中已经简单介绍了手机端传感器采集程序 Sensor Logger 的架构和功能，即通过手机浏览器获取动作传感器数据，并实时传输到云端服务器存储，程序在手机浏览器上运行后的显示效果如图 4-4 所示。接下来从工作流程、关键问题等方面作详细介绍。

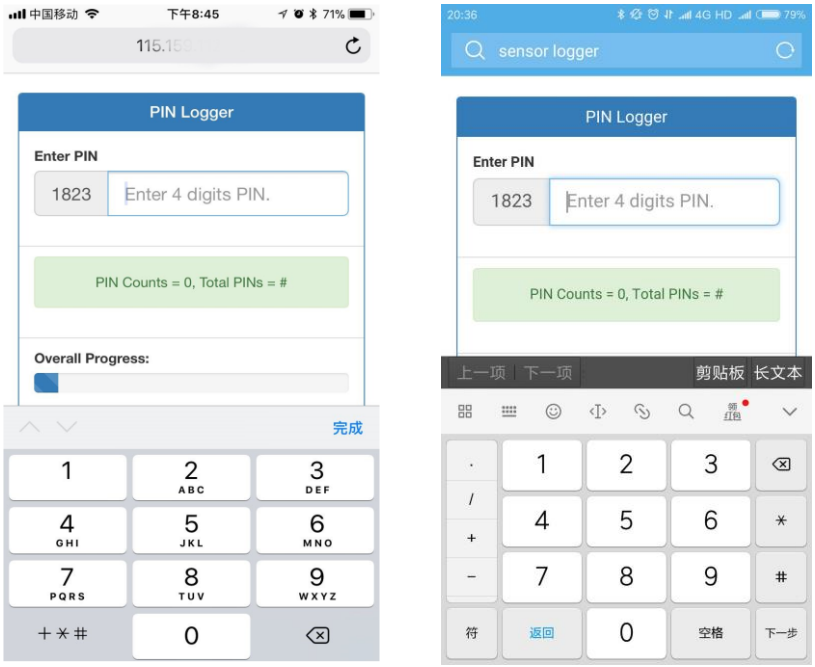

图 4-4 Web 采集程序在手机上的运行效果

4.2.3.1 工作流程

当用户使用智能手机浏览器访问 Sensor Logger 服务器网址后，传感器采集程序就自动开始运行，整个工作流程如图 4-5 中（a）所示。首先会通过 Ajax 请求向云端服务器请求采集指令，具体包括需要采集的 PIN 码，采集的次数、方式等信息。如果服务器没有回应，则自动进行三次重试连接，且每次连接之间的间隔时间按幂次增加，以减轻网络压力，若在三次重试后服务器仍没有回应，则对用户做出相应提示。

在成功连接云端服务器并取得采集指令后，Sensor Logger 会与服务器建立 WebSocket 连接，同时弹出用户信息登记界面，以便将用户与采集信息绑定。当用户点击开始采集后，屏幕上会依次显示需要用户输入的 PIN 码。这些 PIN 码是通过 Ajax 向云端服务请求得到的，因此顺序是固定的，用户在多轮数据采集过程中可能会因为熟悉了 PIN 码而产生肌肉记忆，导致采集到的数据会受到用户输入习惯的影响。会防止这种情况的发生，Sensor Logger 在每次采集前会对 PIN 码打乱顺序。

在用户使用键盘输入 PIN 码的过程中，Sensor Logger 会采集手机的动作传感器数据，通过 WebSocket 实时传输到远程服务器，当所有 PIN 码都采集完成后，会向云端服务器发送完成采集的信息，此信息中包含手机的操作系统版本、手机型号、浏览器型号等，方便后续机器学习中对不同

手机、操作系统、浏览器之间进行对比。整个采集过程中具体传输的数据如下：

- 加速度传感器和陀螺仪以及对应的时间戳
- 敲击开始和结束的时间戳
- 用户此次输入的 PIN 码
- 该用户的信息、手机系统的信息

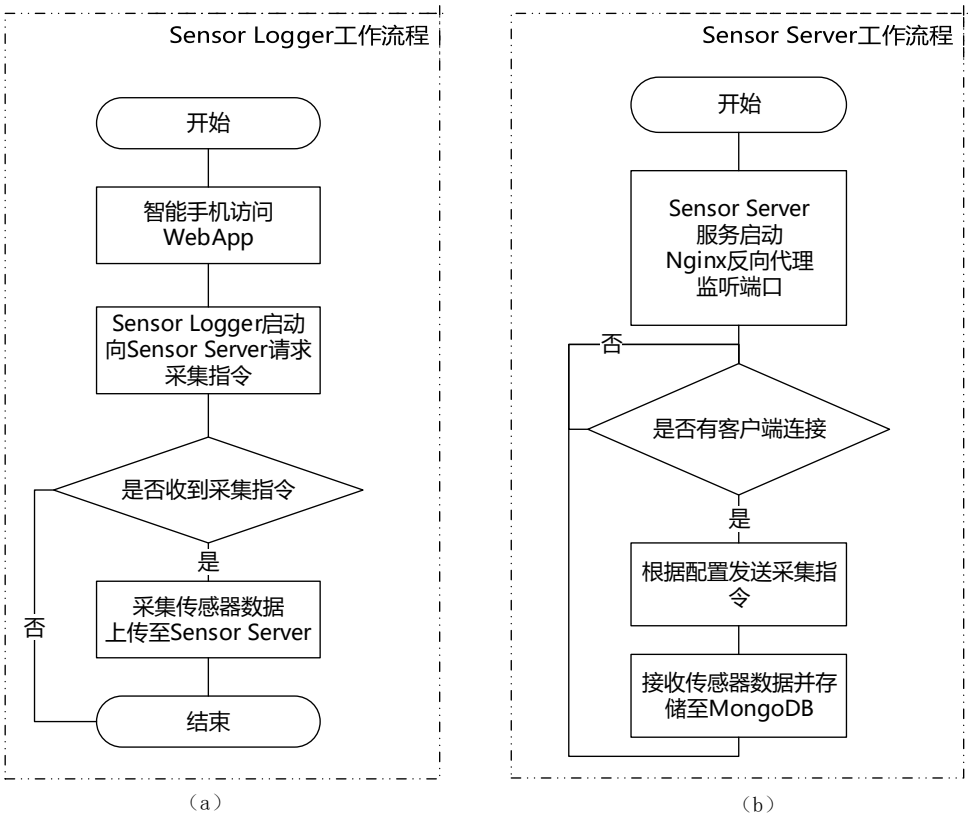

图 4-5 Sensor Logger 与 Sensor Server 工作流程图

4.2.3.2 关键问题与解决方法

要完整实现上述工作流程，有几个关键的问题需要解决，比如如何读取手机传感器数据、如何解决同源策略导致 Ajax 不能跨域请求、如何向服务器实时传输传感器数据、当用户输入错误的 PIN 码时该如何处理等。接下来详细说明这些挑战以及相应的解决方法。

（1） 如何读取手机传感器数据？

W3C 颁布了 DeviceOrientation Event Specification，该标准包括两个新的传感器事件，及 DeviceOrientationEvent 和 DeviceMotionEvent，这两个事件分别在手机方向和加速度发生变化时被触发，通过在浏览器的 window 对象监听这两个事件即可读取手机内置的加速度、角速度、地磁方向传感器数据，下面是读取方向传感器的核心代码。

```
function addSensorListener() {
  window.addEventListener('devicemotion', motionHandler, false);
  window.addEventListener('deviceorientation', orientationHandler, false);
}
function orientationHandler(event) {
  let data = {
    'ox-gamma': event.gamma,
    'oy-beta': event.beta,
    'oz-alpha': event.alpha
  };
  sendDataToServer(data);
}
```

（2） 如何解决同源策略导致 Ajax 不能跨域请求？

当客户端数据采集程序启动后需要向云端服务器请求采集指令，一般都会使用 Ajax 来完成这种数据请求，Ajax 的全称为 Asynchronous JavaScript and XML，即用 JavaScript 执行异步网络请求，通过创建 XMLHttpRequest 对象，并设置相应的回调函数，即可完成异步网络请求[37]。

然而客户端数据采集程序的 JavaScript 代码和云端的 Sensor Server 不在同一台服务器上，由于 HTML5 同源策略的安全限制，默认情况下 JavaScript 发送 Ajax 请求时 URL 的域名必须和当前页面完全一致，导致无法完成网络请求。

常见的解决方法有三种。第一，通过 Flash 插件来发送 HTTP 请求来绕过同源策略，然而这样做就要求浏览器必须安装 Flash 插件，这对于手机浏览器来说代价太大。第二，在同源域名下架设一个代理服务器来转发，代价是需要额外的服务器支持，占用资源。第三，使用 JSONP，即 JSON wih Padding，它允许在服务器端集成 Script tags 返回至客户端，通过 JavaScript callback 的形式实现跨域访问，总结来说，就是<script>标签中的 src 属性并不受同源策略所约束，所以可以获取服务器上的脚本并执行。

本文使用第三种方法，以 JSONP 方式的 Ajax 请求从服务端获取采集指令，并以 Promise 来处理这个异步请求，核心代码如下。

```
function getPinsFromServer(url) {
  return new Promise((resolve, reject) => {
    $.ajax({
      method: 'GET',
      url: url,
      dataType: 'jsonp'
    })
      .done((data) => {
        let shuffledPins = shuffle(data.pins);
        resolve(shuffledPins);
      });
  });
}
```

（3） 如何向服务器实时传输传感器数据？

由于采集程序运行在浏览器中，能够使用的数据存储方式主要由 Cookie 和 LocalStroage 这两种，它们的容量都很有限，除此之外，有可能开启了隐私模式，导致不能在手机本地存储数据。因此采集程序需要实时将采集到的数据实时传输到云端服务器。数据传输的方式可以采用 HTTP 方式的 PUT 方法传输，然而这种方式在每次传输数据时都需要携带 HTTP 头部，并且只能由客户端向服务

端发起请求，不能由服务端主动向客户端发起请求，当然也可以使用轮询的方式来模拟，然而效率低下并且效果也不理想。

本文使用 WebSocket 技术在客户端与服务端之间实时传输数据，WebSocket 是一个在 TCP 上提供双工传输的通信协议，并且 W3C 标准化了浏览器上相应的 API。当需要建立 WebSocket 连接时，客户端会发送一个握手请求，服务器会回应一个握手响应，此时就已经建立好连接了，后续数据传输将遵循 WebSocket 标准，握手与响应的请求内容如下所示。

客户端请求

```
GET /chat HTTP/1.1
Host: sensor-server.com
Upgrade: Websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://sensor-logger.com
```

服务端回应

```
HTTP/1.1 101 Switching Protocols
Upgrade: Websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sM1YUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

（4） 当用户输入错误的 PIN 码时该如何处理？

在传感器数据采集过程中，用户需要按照屏幕上提示的 PIN 码使用数字键盘输入同样的 PIN 码，以便给这些传感器数据打上相应的 PIN 码标签。然而在输入过程中很有可能输错数字，此时采集的数据就没有任何意义了，需要将其从数据库中删除。这个输入过程可以看作是一个事务，当最终输入的 PIN 码与提示的 PIN 码完全相同时，一次输入就算完成，事务也就结束，否则进行事务回滚，这在传统的关系型数据库，如 MySQL 中有着良好的支持。然而 MongoDB 作为非关系型数据库，并不支持事务，虽然有折中的两阶段提交方法，但对于本文的需要求来说太过复杂。

本文采用的方法是为每个当前采集的 PIN 码选择时间戳作为 sampleID，向服务器传输传感器数据时携带这个 ID 信息。当用户输入错误的 PIN 码后，向云端服务器发送一条含有这个 sampleID 以及对应 PIN 码和用户 ID 的回撤请求，同时停止监听多种传感器并提示用户输入了错误的 PIN 码。服务器在接收到回撤请求后会从数据库中删除这个 sampleID 的所有传感器信息，并发送回撤成功的信息给客户端，客户端收到后重新监听动作传感器数据，并提示用户重新输入这个 PIN 码，效果如下所示。这样就可以保证每次采集到的数据都是输入正确的 PIN 码后得到的传感器数据。

```
[2017-12-30 10:34:22.707] [INFO] socket.io - rollback PIN: 6328
[2017-12-30 10:34:22.713] [INFO] socket.io - rollback complete, deleted counts: 215
[2017-12-30 10:34:31.778] [INFO] socket.io - rollback PIN: 5544
[2017-12-30 10:34:31.788] [INFO] socket.io - rollback complete, deleted counts: 287
[2017-12-30 10:34:58.900] [INFO] socket.io - Log completed! [User: aaron, sampleID: 1514601298932, UAParser: [object Object]]
```

4.2.4 Sensor Server 实现

Sensor Server 是整个数据采集系统中的后端服务，它主要负责两件事：第一，向客户端程序发送

采集指令，控制客户端数据采集行为，随机生成 4 位数字 PIN 码，并在 Sensor Logger 向其发起 Ajax 请求时回应 JSON 格式的 PIN 码，以保证所有参与采集训练数据的客户端都得到同样的一组 PIN 码。第二，接收客户端数据采集程序向其传输的动作传感器数据，通过前后端分离设计，支持多台智能手机同时采集传感器数据，使用 MongoDB 持久化存储传感器数据。接下来将从工作流程、关键问题和解决方法两个方面详细介绍。

4.2.4.1 工作流程

Sensor Server 的工作流程如图 4-5 中（b）所示。在启动 Sensor Server 服务程序之前，需要配置数据采集任务并指定一些参数，如采集的 PIN 码位数、PIN 码数量、操作系统版本等。当服务启动后，会读取配置文件，并按照要求生成 PIN 码并存储到 MongoDB 中，然后在指定的端口开始监听来自客户端的连接。

当客户端启动数据采集程序后，会自动连接云端的 Sensor Server 服务，这些连接请求通过 Nginx 反向代理后转发到 Sensor Server 的 NodeJS 服务程序，然后服务程序会按照配置的采集任务向客户端发送采集指令以及 PIN 码，并在客户端之间建立 WebSocket 连接以便后续进行传感器数据传输。

客户端开始采集数据后，服务端开始接收 JSON 格式的传感器数据，将其按照设立的数据表格式化后持久化存储到 MongoDB。当客户端因为输入错误的 PIN 码而发起数据回滚请求后，按照要求从 MongoDB 中删除相应的数据并向客户端发送回滚成功的消息。

为了方便实验人员实时查看数据采集的进度以及样本中不同手机、浏览器占的比重等信息，Sensor Logger 还提供了 Restful 风格的信息获取 API，以及一个 HTML5 Web App 来可视化数据采集统计结果。

4.2.4.2 关键问题与解决方法

Sensor Server 在设计与开发过程中遇到了一些关键的问题，比如如何应对大量并发连接、如何实时存储大量传感器数据、如何保证存储的传感器数据顺序等，下面就遇到的这些问题以及如何解决进行详细介绍。

（1） 如何应对大量并发连接？

由于 Sensor Logger 不在手机本地存储传感器数据，而是采用实时传输的方式，因此 Sensor Server 需要满足大规模数据实时传输的要求，为此本文使用 NodeJS 技术实现，并采用 WebSocket 来代替传统的 HTTP 请求来传输数据，避免无谓的包头开销，提升传输效率。 NodeJS 本身基于 V8 JavaScript 解析引擎实现，该引擎是 Google 用于 Chrome 浏览器的底层 JavaScript 引擎，由于使用 C++编写，具有超快的解释性能。传统的并发 Web 程序设计瓶颈在于服务器能够处理的并发连接最大数量，NodeJS 通过更改连接到服务器的方式来解决这个问题[38]。在每个连接发射一个 NodeJS 引擎中运行的事件，而不是为每个连接生成一个新的 OS 线程。除此之外 NodeJS 不会发生死锁，因为它根本不允许使用锁，且不会直接调用阻塞 I/O 调用。NodeJS 官方宣称可以支持数万个并发连接，本文选择使用该技术作为技术栈可以很好的满足需求，同时可在后期面临大量 PIN Logger 客户端连接的场景时应对大规模数据传输的弹性需求。

（2） 如何实时存储大量传感器数据？

目前智能手机允许浏览器读取传感器数据的采样频率大约在 60Hz 到 100Hz 之间，由于客户端将读取到的传感器数据实时传输到云端服务器，也就是说每个客户端每秒至少有 60 到 100 次写入请求，当客户端数据变多时，数据的持久化存储将会成为瓶颈。

为了存储大量的动作传感器数据，本文采用 MongoDB 作为持久化数据存储数据库，之所以采用 MongoDB 这种 NoSQL 数据库而不是常见的 MySQL 等传统 SQL 数据库有几点原因：第一，传统关系数据库通常是基于行的表格型存储，而 NoSQL 类型的数据库则使用列式存储（cassandra）、

Key/Value 存储、文档型存储以及图结构式存储。相比前者的强关系型，对于本文中的存储需求并不需要，并且 NoSQL 类型的数据库所具有的无模式（Schemaless）特性可以更加方便的扩展属性，实时应对数据字段的变更需求；第二，MongoDB 作为文档型数据库，在存储数据时将所有相关的数据放到一个 document 中，而不像关系型数据库那样需要 Join 多张表格，在读取相关数据时可以一次性加载所需内容，无需计算，且读硬盘次数减少，所以有更高的性能。MongoDB 自带内存缓存，写入性能非常高，且内建 MapReduce 支持，可以在写入数据的过程中对相关数据做一些处理，比如在 Sensor Server 中对数据做均值滤波的预处理。

（3） 如何保证存储的传感器数据顺序？

虽然客户端在采集传感器数据并将其发送到服务端的过程中肯定是按照时间顺序进行的，但是在服务端处理这些依次到来的请求时，由于 NodeJS 核心是采用事件循环机制，该机制基于函数回调，所以并不会按照代码的先后顺序执行 I/O 请求，而是无阻塞的事件循环调用，因此存储到 MongoDB 中的传感器序列可能会产生顺序错位。

本文的解决方法是在每个采集到的传感器数据中都加入时间戳，在 MongoDB 的表字段中也加入时间戳字段，并通过 ensureIndex()方法来对时间戳建立索引。索引是特殊的数据结构，存储在一个易于遍历读取的数据集合中，是对数据库表中一列或多列的值进行排序的一种结构。索引通常能够极大的提高查询的效率，如果没有索引，MongoDB 在读取数据时必须扫描集合中的每个文件并选取那些符合查询条件的记录。这种扫描全集合的查询效率是非常低的，特别在处理大量的数据时，查询可以要花费几十秒甚至几分钟。后期需要提取传感器数据时通过时间戳字段进行排序即可，由于有了索引，所以排序查询速度非常快。

4.3 PIN 码推测概念验证程序

移动互联网时代，人们经常使用智能手机来访问互联网。除了手机上的浏览器外，原生应用也可以通过 WebView 支持加载外部网页，因此传统 Web 环境下的一些攻击方法依然能对智能手机用户产生安全威胁，如 XSS 攻击、中间人攻击等。为了演示在 Web 环境下程序无需用户许可即可获取传感器数据并发送到远程服务器导致 PIN 码泄漏的安全风险，本文设计并实现了一个 PIN 码推测概念验证程序。该程序是一个可在微信等支持 WebView 的社交应用中运行的 Web 程序，其中包含安全漏洞，本文通过 XSS 攻击将恶意 JavaScript 代码嵌入到其中，当用户在微信中打开后，会将传感器数据在用户不知情的情况下传输至后端已经训练好的机器学习推测系统，从而推测出用户 PIN 码。下面首先对关键技术 WebView 和 XSS 进行介绍，然后再介绍恶意客户端的实现。

4.3.1 WebView

本文主要研究使用 JavaScript 获取手机动作传感器数据，主流手机浏览器如 Safari、Chrome、Firefox 等内置了 JavaScript 解析引擎，只要访问包含有读取传感器代码的网页就可以获得传感器数据。除此之外还有许多手机应用如微信小程序、微信公众号等，也可以执行 JavaScript 代码，它们主要使用 WebView 这一操作系统组件来实现。

大部分手机操作系统，比如 Android，iOS，Windows Phone 都不原生支持 JavaScript 和 HTML，因此为了执行 JavaScript 代码和显示基于 HTML5 的界面，应用程序通常需要内嵌一个 Web 浏览器。在 Android 里叫做 WebView，在 iOS 里叫做 UIWebView，在 Windows Phone 中叫做 WebBrowser，它们都是指一个东西，为了不产生歧义，本文统称为 WebView。

WebView 是手机操作系统的一个重要组件，它可以让手机显示和运行来自外部的网页，为了安全考虑，通常会在 WebView 内部实现沙箱，它用来隔离 JavaScript 代码和手机内部资源，默认情况下 JavaScript 代码是无法访问存储于手机中的文件、相机、传感器等。这对于移动应用来说限制太

多，因此 WebView 提供了一个叫 addJavascriptInterface()的 API，允许在 JavaScript 代码与外部的 Java 代码之间架起一座桥梁，通过此接口，处于沙箱中的 JavaScript 可以顺利的访问手机内部资源。目前已有成熟的第三方中间件实现了常用资源的映射代码，应用开发者可以直接使用，如 PhoneGap、RhoMobile、AppMobi、Mosync、Appcelerator 等，图 4-6 展示了此类应用程序的典型架构。

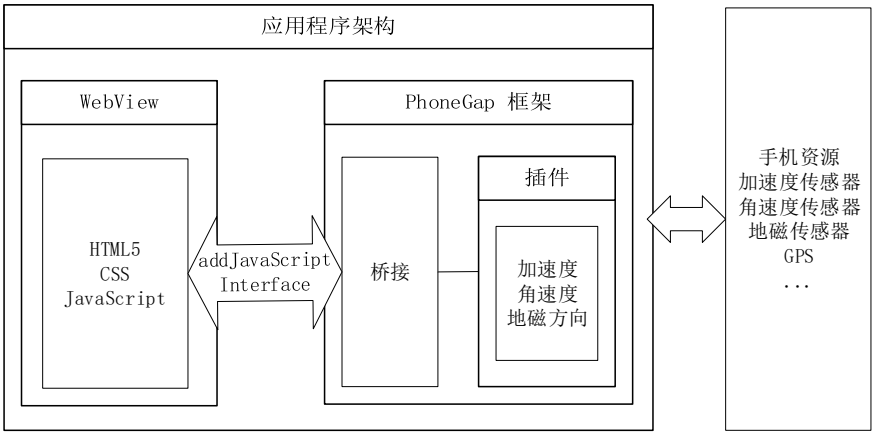

图 4-6 典型 Hybrid 应用程序架构

4.3.2 XSS 攻击

XSS 攻击又称为 Cross-Site Scripting（跨站脚本）攻击，它是 Web 中常见的攻击形式。在 Web 技术标准中，HTML 可以自由的嵌入 JavaScript 代码，这种代码跟数据混合在一起的特性成为安全漏洞的源头，如果开发者不够有经验，混入提交数据的恶意 JavaScript 代码可被后续访问用户的浏览器解析引擎执行，这就是 XSS 攻击的工作原理，如图 4-7 所示。

在典型的 XSS 攻击中，攻击者将恶意 JavaScript 代码插入到提交表单中，如果服务器没有对提交的数据进行过滤处理而是直接返回给正常用户的话，那么当用户使用浏览器访问该网页时浏览器会自动执行恶意 JavaScript 代码，根据 OWASP 的报告，XSS 是目前最为广泛的 Web 攻击手段。基于 Web 技术构建的 HTML5 应用自然也会受到 XSS 攻击的影响，包括微信小程序、微信公众号等，由于浏览器有沙盒保护导致 JavaScript 代码基本不能访问本机资源，对于 HTML5 构建的 Hybrid 应用来说，注入的恶意代码可以利用中间件（PhoneGap）提供的接口访问手机资源从而实现更大的破坏。

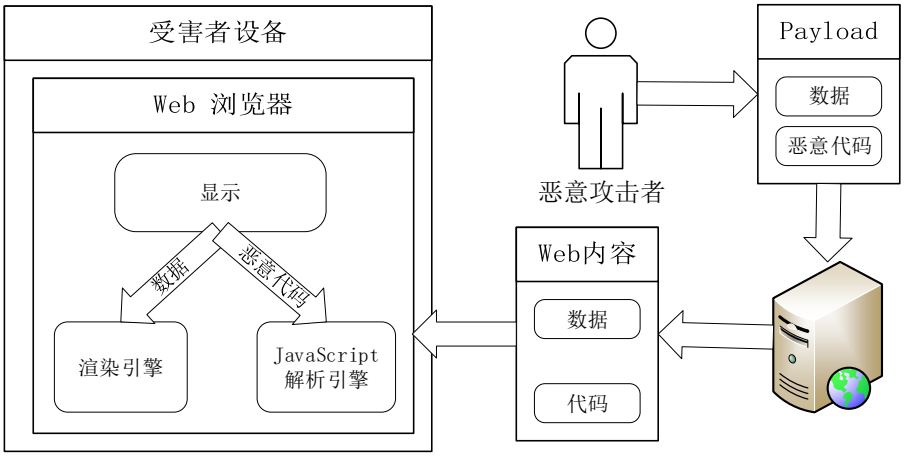

图 4-7 典型 XSS 攻击模式

4.3.3 程序实现

本文使用 Bootstrap 和 NodeJS 技术实现 PIN 码推测概念验证程序，如图 4-8 所示，该程序需要用户输入用户名、手机号、银行账户、PIN 码等信息，其中存在 XSS 漏洞，本文使用存储型 XSS 攻击，将恶意 JavaScript 代码嵌入其中，由于微信使用了 WebView 组件，所以可以在其中运行此 Web 程序。当用户在微信中打开 PIN 码推测概念验证程序后，恶意代码就开始运行，将手机内置传感器数据传输至远程服务器，由于是演示应用，此客户端中嵌入的 JavaScript 代码与传感器数据采集系统中的核心代码一致，在此就不再赘述。本文在实验过程中还发现一个重要现象，iOS 版本的微信在退出后默认不会被系统立刻停止运行，如果用户在微信中运行这个恶意应用，然后直接锁屏，此时程序还在继续执行，也就是说锁屏后手机的传感器数据依然被传输至远程服务器，如果用户又通过 PIN 码解锁手机，那么远程恶意用户就有可能获取用户的手机解锁 PIN 码。

图 4-8 模拟恶意应用在微信中运行效果

4.4 机器学习推测系统

在 3.4 节中介绍了本文将要使用的机器学习算法，仅有算法是不行的，还需要使用编程语言实现这些算法并对模型进行训练。

4.4.1 技术选型

随着机器学习在工业界的广泛应用，已经出现很多机器学习平台可供使用，比如 Mathworks 公司开发的 Matlab[39]，Google 公司开发的开源平台 TensorFlow、微软开发的开源机器学习框架 CNTK[40]、怀卡托大学开发的基于 Java 的可视化数据挖掘环境 Weka 等[41]。

本文使用 Scikit-Learn 开源机器学习库。Scikit-Learn 在 2007 年被 David Cournapeau 作为 Google summer of code project 最初设计、创建和开发。它使用简化版 BSD 许可证进行许可，并且在 Linux 多个版本下分发，鼓励学术和商业用途。Scikit-Learn 基于 NumPy、SciPy、Matplotlib 等强大的数据挖掘工具链，提供了数据建模、降维、特征提取与筛选、交叉验证等功能函数，以及分类、回归、聚类等常见类型的机器学习算法功能，如图 4-9 所示。

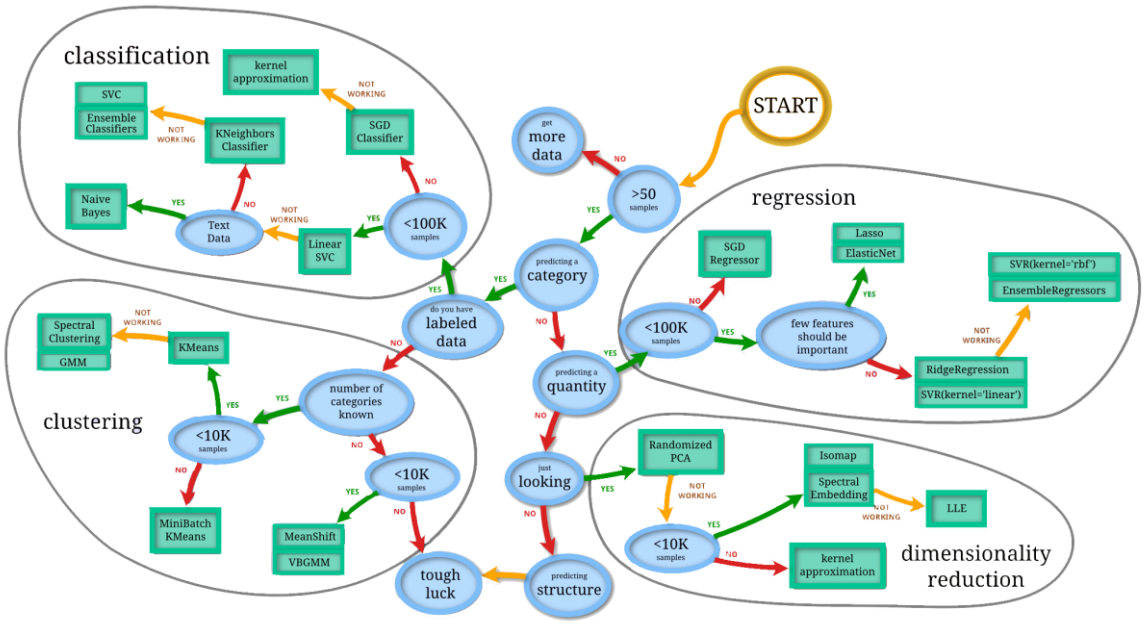

图 4-9 Scikit-Learn 算法地图[42]

本文使用 Python 编程语言来实现预处理、特征提取、模型训练等过程。Python 语言是在 1991 年创造的解释型语言，目前已经是最受欢迎的动态编程语言之一，并以由强大而活跃的科学计算社区著称。本文选择 Python 语言进行开发有以下几点原因：

（1） Python 是解释性语言

相较于 C++等编译型语言虽然运行速度慢一些，但无需担心内存资源管理问题，并且 Python 无需编译即可运行，可以快速搭建模型算法并进行验证，节省科研时间。

（2） Python 的开发生态成熟

有很多强大的科学计算库，如矩阵运算库 NumPy，科学计算库 SciPy，图像处理库 Matplotlib，格式化数据处理库 Pandas 等。对于数据挖掘应用，在 Pandas 支持下已经可以和商业工具 Matlab 相媲美。

（3） 原型开发与工业应用可以无痛转换

很多科研机构在前期使用 Matlab 来快速搭建模型验证想法，后期再使用 C++实现来提高性能，这样在两种语言之间不断转换是一种资源浪费。Python 可以让原型验证和构建生产系统使用同一种语言，降低成本提高效率。

4.4.2 需求分析

机器学习推测系统是本文设计的原型系统的核心部分，它负责对采集到的传感器数据进行预处理、特征提取、模型训练、PIN 码推测，下面依次列出整个系统的需求与相应的解决方案。

（1） 原始数据输入和清洗

机器学习的核心是数据，本文设计的传感器数据采集系统将采集到的数据存储到 MongoDB 数据库中，由于 NodeJS 核心是采用事件循环机制，存储到 MongoDB 中的传感器序列可能会产生顺序错位，因此在准备从 MongoDB 中读取原始数据时要按照时间戳字段进行排序，由于已经建立了索引，所以排序速度非常快。由于采集过程中可能会有异常值，因此也需要按照 3.2.2 小节中描述的预处理操作，对原始数据进行滤波、插值，以得到符合要求的数据。

（2） 特征提取

经过处理的原始数据由一系列传感器采样点组成的时间序列，可以看作一个二维矩阵，需要从中提取特征向量，本文 3.3 节中详细描述了时域、频域特征的提取方法，本文使用 NumPy 和 Pandas

工具来实现特征提取。

（3） 多种模型训练

本文主要使用多层感知机算法搭建分类模型，在 3.4 节中详细介绍了该模型的结构以及各种超参数的调整策略，在本系统中使用 Scikit-Learn 实现模型的搭建以及训练，为了比较多层感知机模型与其他分类算法性能的差异，本系统设计了动态分类模型加载功能，可以按照配置同时对多种分类模型进行训练。

（4） 自动生成图表

机器学习的分类结果包含很多文本信息，对其进行可视化处理可以更好的观察结果，并从中得出一些有用的结论，为此本系统使用 Matplotlib 实现多维度多种类的自动图表生成功能。

（5） 训练过程自动化

机器学习模型的训练通常会耗费大量的时间，不同参数的选择也会影响最终分类结果的准确性，因此经常需要调整参数进行实验，为此本文设计了训练自动化监视模块，可按照配置进行模型训练任务，同时可将训练结果通过电子邮件发送以便远程查看，提高效率。

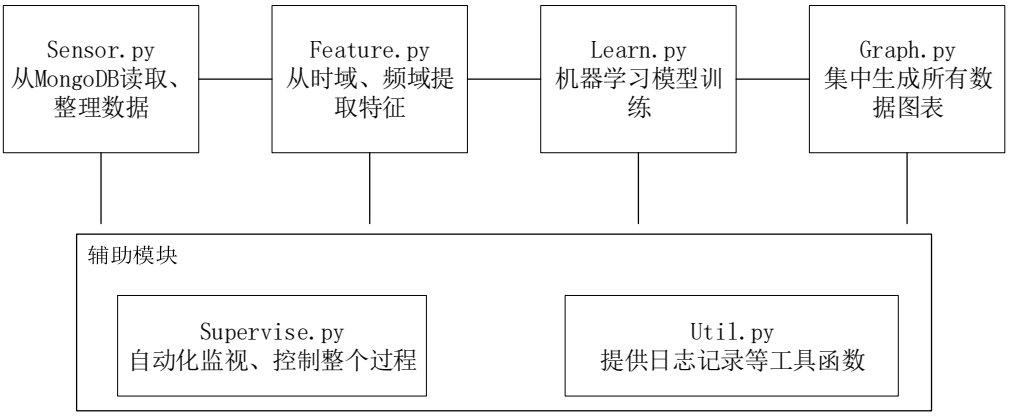

图 4-10 代码模块图示

4.4.3 系统实现

本文使用 Python 语言实现推测系统。为了让代码结构更加清晰易读，方便后期进行扩展，本文使用模块化组织所有代码，结构如图 4-10 所示，下面就各个模块的功能详细介绍。

Sensor.py 模块：正如前文第 3.3 小节所描述的，本文设计的传感器数据采集系统会将采集到的数据存储到 MongoDB 数据库中，Sensor.py 模块负责从数据库中读取数据并按照要求进行格式化操作，为后续特征提取 Feature.py 模块提供数据源。PyMongo 是 MongoDB 官方提供的 Python 驱动，通过它可以使用 MongoDB 语法方便的查询数据，在前面的章节中已经提到，由于 NodeJS 的事件循环处理特性，插入数据库的采样数据顺序可能会发生错乱，为此本文加入了时间戳字段并建立索引，在取出数据时按时间戳进行排序保证采样数据的时序。为了后续处理方便，在此模块中会将传感器数据组装成 DataFrame 格式，DataFrame 是 Python 中著名的数据分析库 Pandas 的二维数据格式，核心代码如下：

```
def get_samples_by_sampleid(sampleid):
    cursor_acc = db.sensor.find(
        {'sampleID': sampleid, 'data.acc-x': {'$exists': True}},
        {'data': 1, 'time': 1}
    ).sort([
        ('time', pymongo.ASCENDING)
    ])
    cursor_ori = db.sensor.find(
        {'sampleID': sampleid, 'data.ox-gamma': {'$exists': True}},
        {'data': 1, 'time': 1}
    ).sort([
        ('time', pymongo.ASCENDING)
    ])
    acc_data = (sample for sample in cursor_acc)
    ori_data = (sample for sample in cursor_ori)
    dates = []
    samples = []
    columns = ['acc-x', 'acc-y', 'acc-z', 'gacc-x', 'gacc-y', 'gacc-z', 'rot-
alpha', 'rot-beta', 'rot-gamma', 'ox-gamma', 'oy-beta', 'oz-alpha']
    for acc, ori in zip(acc_data, ori_data):
        raw_data = dict(acc['data'], **ori['data'])
        samples.append([raw_data[field] for field in columns])
        dates.append(datetime.strptime(acc['time'], TIME_FMT))
    data = np.array(samples)
    df = pd.DataFrame(data, index=dates, columns=columns)
    return df
```

Feature.py 模块：负责从样本中提取前文 3.3 小节中提到的时域特征和频域特征，由于输入是$N_{samples} \times M_{sensor}$的矩阵，其中$N_{samples}$是指输入一个 PIN 码的过程中采集到的传感器样本数量，M_{sensor}指不同传感器不同轴上的数据，具体来说就是加速度传感器、带重力加速度传感器、陀螺仪、方向传感器等四种传感器在x,y,z三个轴上的数据，所以共有$3 \times 4 = 12$个不同的数据流。本文使用 NumPy 来进行各种矩阵运算，NumPy 是 Python 科学计算中重要的矩阵运算库，它提供高效的 N 维数组数据结构 ndarray，底层使用 C++和 Fortran 实现。如下是本文提取矩阵特征的核心代码：

```
def extract_matrix_feature_vector(samples):
    acc = samples.loc[:, ['acc-x', 'acc-y', 'acc-z']]
    gacc = samples.loc[:, ['gacc-x', 'gacc-y', 'gacc-z']]
    rot = samples.loc[:, ['rot-alpha', 'rot-beta', 'rot-gamma']]
    ori = samples.loc[:, ['ox-gamma', 'oy-beta', 'oz-alpha']]
    feature_1_norm = pd.Series([
        acc.abs().sum().max(),
        gacc.abs().sum().max(),
        rot.abs().sum().max(),
        ori.abs().sum().max()
    ])
    feature_inifinity_norm = pd.Series([
        acc.abs().sum(axis=1).max(),
        gacc.abs().sum(axis=1).max(),
        rot.abs().sum(axis=1).max(),
        ori.abs().sum(axis=1).max()
    ])
    feature_frobenius_norm = pd.Series([
        np.sqrt(np.square(acc).values.sum()),
        np.sqrt(np.square(gacc).values.sum()),
        np.sqrt(np.square(rot).values.sum()),
        np.sqrt(np.square(ori).values.sum())
    ])
    return pd.concat([feature_1_norm, feature_inifinity_norm, feature_frobenius_norm])
```

Learn.py 模块：负责具体的机器学习模型训练。本文实现多种分类算法的动态加载，可根据配置同时对多种分类模型进行训练。本文主要使用人工神经网络算法中的多层感知机模型，为了与其他分类算法进行比较，文本也实现了 KNN、SMO、C4.5、LMT、Random Forest 等分类算法并进行训练。为了更好地反映出各个算法对 PIN 码的分类性能，本文没有简单地在原集上验证或直接分割原始数据集，而是采用了十折交叉验证的方式，即将原始数据分为 10 等分，在测试循环中，每个数据子集都有机会成为测试集，相应的，此时另外 9 个子集就成为原始训练集。在验证过程中，当每一个子集都被轮转为测试集时，验证过程结束。使用交叉验证的主要原因是，该方法能够更有效地利用有限的原始数据，而不会使得其中一部分数据丢失重要的训练或测试能力。此外，交叉验证还能够有效地限制过拟合的问题。在大量的机器学习和数据分析实践中，经常能看到研究者采用将数据集分为 10 份的交叉验证方法。这主要是因为，根据大量的实践统计数据来看，采用 10 折交叉验证往往能够达到测试准确率和实验效率的最优平衡。如下是本文对多个算法进行训练的核心代码。

```
Classifer = namedtuple('Classifer', ['name', 'clf'])
classifiers = [
    Classifer("Nearest Neighbors", KNeighborsClassifier()),
    Classifer("RBF SVM", SVC(gamma=2, C=1)),
    Classifer("Gaussian Process", GaussianProcessClassifier(1.0 * RBF(1.0))),
    Classifer("Decision Tree", DecisionTreeClassifier(max_depth=5)),
    Classifer("Random Forest", RandomForestClassifier(n_jobs=-1)),
    Classifer("AdaBoost", AdaBoostClassifier(n_estimators=100)),
    Classifer("Neural Net", MLPClassifier(activation='relu', max_iter=1000,
solver='adam', verbose=True))]
summary = []
for name, clf in classifiers:
    t = time.time()
    cv = CrossValidation(n_splits=10, random_state=0)
    score = train(clf, cv, X, y)
    elapsed_time = time.time() - t
    summary.append([name, score, elapsed_time])
```

Graph.py 模块：负责整个系统中所有图表的集中生成，本文使用 Matplotlib 库，它是一个开源的 Python 图像绘制库，可以很方便的和 IPython 与 jupyter notebook 集成，绘制语法与 Matlab 很相似，可以生成出版物级别的图像。如下是本文对绘制原始传感器信号的核心代码。

```
def plot_raw_data(pin, sampleid=None, sensor_list=None, sensor_axis=None):
    if sampleid is None:
        sampleid = random.choice(sensor.get_sampleid_by_pin(pin))
    samples = sensor.get_samples_by_sampleid(sampleid)
    key_times = sensor.get_key_time_by_sampleid(sampleid)
    x_time = samples.index
    row, col = 1, len(sensor_list)
    fig = plt.figure(figsize=(12, 6))
    fig.suptitle("Sensor Data for PIN %s" % pin)
    for index, sensor_name in enumerate(sensor_list):
        plt.subplot(row, col, col * index + 1)
        for i, label in enumerate(make_label(sensor_name, sensor_axis)):
            plt.plot(x_time, samples[label])
            plt.xticks(key_times, ['down', 'up'] * 4)
    plt.tight_layout()
    plt.show()
```

Util.py 模块：主要包含一些工具函数，尤其是日志记录功能，该模块支持使用 json 格式的配置

文件指定日志记录模式，通过分级日志记录，将正常运行过程中的输出和出错的信息分开存储，方便后期查错，另外使用滚动存储功能，当日志文件超过阈值后自动删除过旧内容，不用担心丢失重要日志。

Supervise.py 模块：负责整合上述四个模块，根据配置文件中设置的任务运行训练任务，由于机器学习算法涉及很多参数的调整，如果每次都人为去调整会大大降低效率，除此之外有些训练会花费大量时间，因此通过设计本模块实现自动按照配置调整参数，并且将模型结果使用电子邮件发送到指定邮箱，方便不在电脑前的时候查看结果，节省时间，效果如图 4-11 所示。

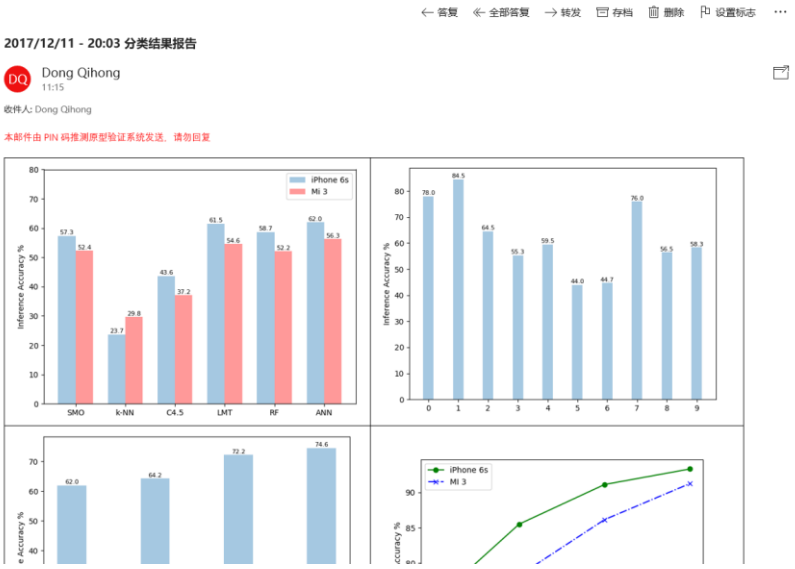

图 4-11 分类结果邮件图示

4.5 实验结果与分析

为了测试引入改进 MFCC 算法提取频域特征对 PIN 码识别率提升的效果，本文进行了两次实验。首先仅使用时域特征来训练 2.3.3 小节中提到的各种机器学习分类算法，然后再使用时域特征和 3.3.2 小节中介绍的改进 MFCC 算法提取的频域特征一起来训练各种机器学习分类算法。接下来从各个分类算法的性能，MFCC 的预加重处理、动态差分参数的提取、提取系数的个数对推测准确率的影响，各个按键的预测准确率差异等方面进行详细的分析和讨论。

4.5.1 实验环境说明

考虑到不同手机操作系统上传感器的表现可能具有差异性，为此本文准备了两部智能手机，分别是 iPhone 6s 和小米 3，它们分别是基于 iOS 和 Android 这两大当今应用最为广泛的手机操作系统，且在当前市场有较多的用户基数，具有较好的代表性。有关它们的详细信息在表 4-1 中已经列出。

表 4-1 实验手机环境说明

设备型号	操作系统	浏览器	浏览器引擎
iPhone 6s	iOS 11.1	Safari	Webkit
Xiaomi 3	Android 6.0	Chrome	Blink

本次实验共使用三台服务器，两台数据采集服务器是从腾讯云购买的虚拟服务器，其中分别运行着 Sensor Logger 手机端 Web 程序和 Sensor Server 数据采集后端服务，一台机器学习服务器是实验室的工作站，主要运行预处理、特征提取、模型训练等需要大规模计算的任务，有关他们的具体

配置详见表 4-2。

表 4-2 服务器硬件说明

参数	数据采集服务器	机器学习服务器
操作系统	Ubuntu 16.04 Server Edition	Ubuntu 16.04 Server Edition
内核版本	4.14.8	4.14.8
CPU	Intel Xeon E3 2650	Intel Xeon E5 2650
显卡	无	Nvidia GTX1060
内存	2G	16G

4.5.2 传感器数据采集

	acc-x	acc-y	acc-z	gacc-x	gacc-y	gacc-z	rot-alpha	rot-beta	rot-gamma	ox-gamma	oy-beta	oz-alpha
2017-12-25 00:57:34.616	-0.129546	-0.044509	-0.070055	-0.788141	-3.026720	-9.389011	-5.613454	18.783573	0.302083	-4.042520	17.704079	3.985912
2017-12-25 00:57:34.627	-0.158851	-0.122561	-0.290728	-0.790386	-3.096451	-9.614216	-4.961751	16.036473	0.852056	-3.875063	17.653055	3.990410
2017-12-25 00:57:34.644	-0.177075	-0.203819	-0.463536	-0.788889	-3.164835	-9.792435	-5.576711	4.762059	1.512471	-3.752240	17.574140	4.006653
2017-12-25 00:57:34.661	-0.205394	-0.202307	-0.108686	-0.821660	-3.146131	-9.442731	-4.519948	-1.826503	2.122396	-3.777381	17.468803	4.036670
2017-12-25 00:57:34.677	-0.206689	-0.105046	0.111239	-0.822708	-3.042881	-9.224709	-1.145314	2.789555	2.567281	-3.775102	17.432125	4.072821
2017-12-25 00:57:34.694	-0.137006	0.119357	0.217060	-0.741454	-2.821267	-9.118766	2.355296	6.495503	1.608541	-3.704446	17.449207	4.105685
2017-12-25 00:57:34.711	-0.131399	-0.051255	0.279283	-0.710330	-3.002778	-9.054721	2.986613	9.603413	0.881230	-3.549157	17.515970	4.131619

图 4-12 采集的动作传感器原始数据

本文从校园随机邀请了 10 名同学作为志愿者来协助采集输入 PIN 码时的动作传感器数据，志愿者只需要用实验手机在浏览器上打开 Sensor Logger 并按照屏幕提示输入 PIN 码，在输入过程中动作传感器数据会被自动实时上传到 Sensor Server 后台服务并存储到数据库中。训练数据中的 PIN 码均为数字，为了各个数字在 PIN 码训练集中出现的频率保持一致，本文使用程序自动生成随机 PIN 码，保证统计特性的稳定。实验中收集的数据均为正确输入 PIN 码后的动作传感器数据，如果用户输入错误，Sensor Logger 会自动删除之前存储到数据库中的错误数据，并提示用户重新输入正确的 PIN 码。在输入过程中，除了读取手机的动作传感器数据，还会记录相应的时间戳信息，方便后期对数据进行进一步的分析处理，具体采集的数据如图 4-12 所示。

为了尽可能模拟真实生活环境中的数据，在数据采集过程中，本文要求志愿者坐在椅子上，使用自己平常使用的方式握持手机，有单手握持手机并输入；也有一只手握持，另一只手输入；还有双手输入的方式。每人需要在 Android 和 iOS 手机上各完成 10 次，每次包含 50 个 PIN 码，总共获得 10000 个样本。

4.5.3 仅用时域特征下各分类算法比较

使用多层感知机算法和前文 2.3.3 小节中描述的各种机器学习分类算法对仅包含时域特征的数据进行训练，并使用 10 折交叉验证，其推测准确率的详细对比如图 4-13 所示。

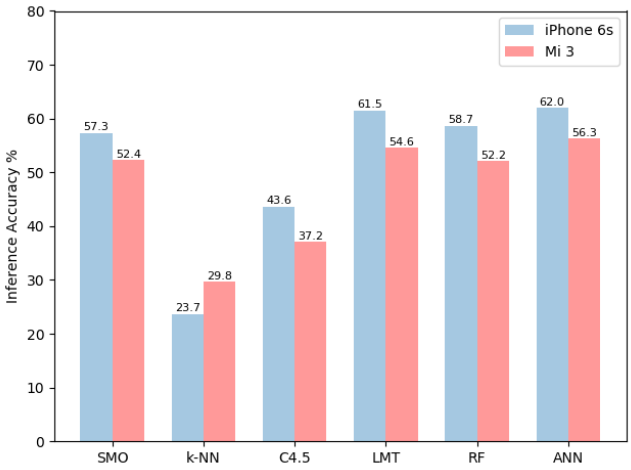

图 4-13 各分类算法的推测准确率对比

上图中的 ANN 符号代表本文使用的多层感知机算法。从图中可以看出，大部分情况下 iPhone 6s 的推测准确率要高于 MI 3，具体原因将在下文 4.5.6 小节给出。SMO 算法、LMT 算法、RandomForest（随机森林）算法以及多层感知机算法都取得了不错的表现，而 KNN 算法、C4.5 算法的表现较差。在测试的所有算法中多层感知机算法的准确率最高，而 LMT 算法表现的几乎与多层感知机算法一样好。

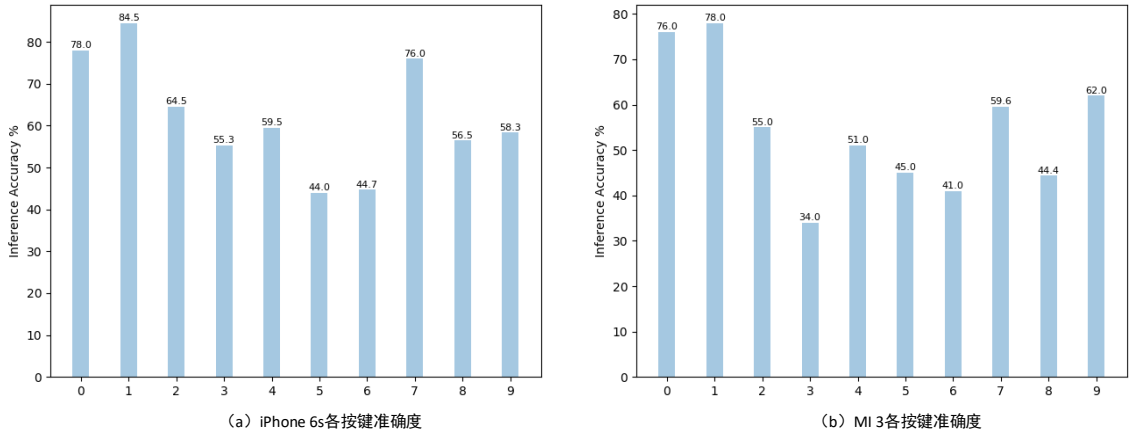

图 4-14 多层感知机算法在不同手机下各按键的推测准确率

图 4-14 展示了多层感知机算法下各按键的推测准确率，从中可以看出从 iPhone 6s 和 MI 3 两部手机中采集到的传感器数据，各按键的准确率都不相同。将二者的结果进行对比，可以看到不同环境下按键准确率的高低分布也表现出不同的范式，有些按键在两种环境下的准确率都明显高于其他按键，然而也有一些按键在不同环境下的准确率表现高低不定。关于出现这样的结果的原因，也将在下文 4.5.5 小节进行详细的分析。

接下来看使用多层感知机算法后的测试结果混淆矩阵，两个环境下的混淆矩阵分别如表 4-3 和表 4-4 所示。其中，竖排表示的是真实的按键值，横排表示的是被推测归类的按键值。矩阵中每行数据表示该真实值被分类器归类到各个不同预测值的概率分布情况。从这两张表中可以发现，每个真实值的预测结果总是分布在距离该真实按键较为邻近的按键中，比如：真实值为 0 的一次点按比较容易被误判为 7，8 或 9，因为这三个键在软键盘上距离 0 最为接近，数据上来看也确实如此。这说明本文的分类结果能很好地体现出真实的情况，并能印证理论上的预测。

表 4-3 iPhone 6s 环境下测试数据的混淆矩阵

真实值\预测值	0	1	2	3	4	5	6	7	8	9
0	78.00%	0.50%	0.00%	0.00%	0.00%	2.00%	0.50%	2.50%	12.00%	4.50%
1	0.00%	84.50%	2.00%	0.00%	11.50%	0.50%	0.00%	1.50%	0.00%	0.00%
2	0.50%	2.00%	64.50%	8.00%	7.00%	15.00%	0.00%	0.50%	2.00%	0.50%
3	0.00%	1.51%	5.53%	55.28%	0.00%	7.03%	16.08%	1.00%	6.53%	7.04%
4	0.50%	13.00%	9.50%	1.50%	59.50%	1.50%	0.00%	14.00%	0.00%	0.50%
5	2.00%	0.50%	18.50%	11.50%	1.50%	44.00%	3.50%	0.50%	17.00%	1.00%
6	1.01%	0.00%	0.50%	18.09%	0.50%	5.03%	44.72%	0.50%	3.52%	26.13%
7	3.00%	0.50%	1.00%	1.50%	13.50%	0.00%	0.00%	76.00%	4.50%	0.00%
8	9.50%	0.50%	1.00%	3.50%	1.00%	13.00%	4.50%	2.00%	56.50%	8.50%
9	3.02%	0.00%	0.50%	8.04%	0.00%	2.51%	22.11%	0.00%	5.53%	58.29%

表 4-4 MI 3 环境下测试数据的混淆矩阵

真实值\预测值	0	1	2	3	4	5	6	7	8	9
0	76.00%	0.00%	0.00%	0.00%	0.00%	2.00%	0.00%	14.00%	6.00%	2.00%
1	0.00%	78.00%	9.00%	1.00%	11.00%	0.00%	0.00%	1.00%	0.00%	0.00%
2	1.00%	17.00%	55.00%	7.00%	11.00%	8.00%	1.00%	0.00%	0.00%	0.00%
3	0.00%	2.00%	8.00%	34.00%	7.00%	12.00%	18.00%	4.00%	9.00%	6.00%
4	0.00%	11.00%	9.00%	9.00%	51.00%	10.00%	1.00%	4.00%	5.00%	0.00%
5	0.00%	1.00%	11.00%	16.00%	9.00%	45.00%	5.00%	4.00%	6.00%	3.00%
6	1.00%	2.00%	3.00%	22.00%	1.00%	9.00%	41.00%	0.00%	9.00%	12.00%
7	16.16%	2.02%	2.02%	2.02%	6.06%	1.01%	3.03%	59.60%	5.05%	3.03%
8	6.06%	1.01%	0.00%	2.02%	2.02%	10.10%	12.12%	6.06%	44.45%	16.16%
9	3.00%	0.00%	0.00%	1.00%	0.00%	1.00%	13.00%	3.00%	17.00%	62.00%

4.5.4 使用改进 MFCC 提取频域特征

本小节使用时域特征以及改进 MFCC 提取的频域特征来共同对前文 0 小节中描述的多层感知机分类算法进行训练，使用 10 折交叉进行验证。为了研究改进 MFCC 对分类性能的提升效果，本文将仅使用时域特征、使用 FFT 提取频域特征、使用 MFCC 提取频域特征后的分类推测准确率进行对比，如图 4-15 所示。

从中可以看出，加入线性频域特征后可以提升推测准确率，但效果并不明显，仅仅从 62%提升到 63.5%。与之相比使用改进的 MFCC 算法提取频域特征后，推测准确率提升非常明显，从 62%提升到 74.6%。在使用动作传感器数据进行 PIN 码识别的场景中，MFCC 算法提取频域特征比普通线性频域特征表现更好符合本文先前的理论分析，因为输入 PIN 码时动作传感器信号的能量主要集中在低频部分，而 Mel 尺度在低频部分稠密，高频部分稀疏，因此使用 MFCC 算法可以更好的从低频信号中挖掘特征。

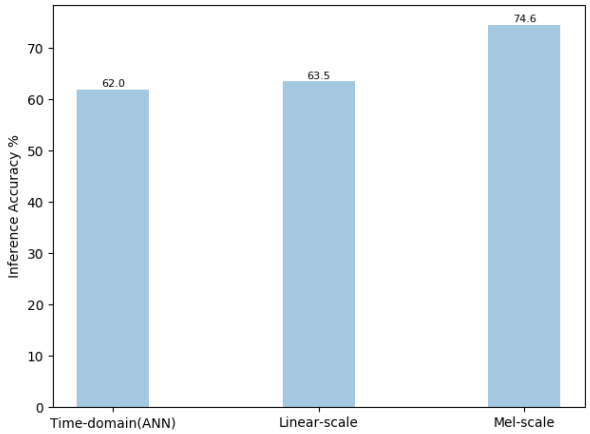

图 4-15 使用不同标尺下提取的频域特征的准确率

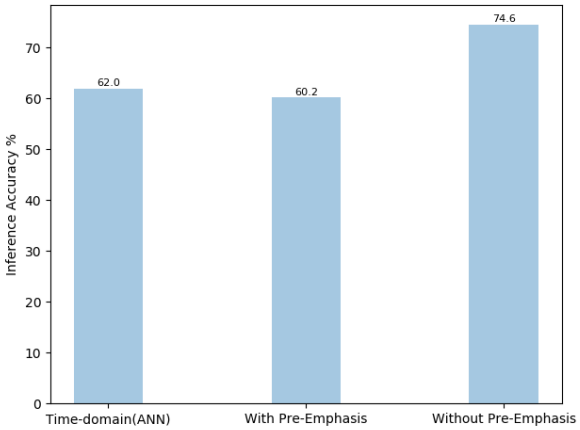

图 4-16 预加重对准确率的影响

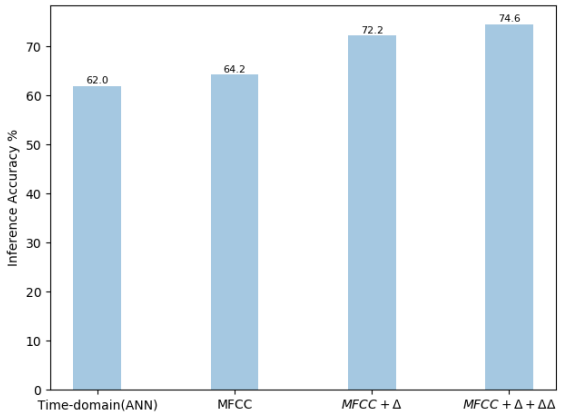

图 4-17 动态差分参数对推测准确率的影响

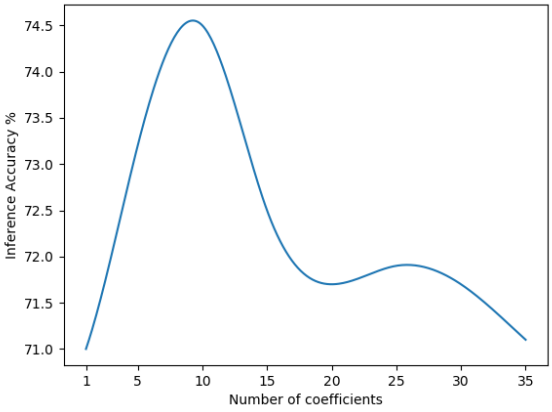

图 4-18 提取系数对推测准确率的影响

图 4-16 展示了在使用 MFCC 提取频域特征的过程中预加重处理过程的加入与否对准确率的影响，结果显示在傅立叶变换之前引入预加重处理过程后，判决准确性从 74.6%下降到了 60.2%，也印证了前文 3.3.2 小节中的分析，因为预加重处理本质上就是对原始信号使用一个设计的加强高频分量的高通滤波器过滤，然而动作传感器的采样率相对语音信号来说很低，且大部分信号能量都集中在低频部分。所以在改进的 MFCC 算法中，去除了预加重处理这一步骤。

图 4-17 展示了引入 MFCC 频域特征的动态差分参数作为新的特征后对推测准确率的影响，可以看出一阶导数和二阶导数的加入有效的提升了推测准确率，将其从原来的 64.2%提升到了 74.6%。如 3.3.2 小节描述的，MFCC 对时间的导数代表了帧与帧之间的相关性，反映了相邻按键在时域的动态特性，因此将其加入特征向量可以更加完整的描述信号的特征。

图 4-18 展示了从一帧信号中提取不同数量的 MFCC 系数对推测准确率的影响。提取的系数个数影响着最终特征向量的大小，从而间接影响机器学习分类器的性能和效率。从图中可以看出，推测准确率开始的时候随着提取的系数增多而逐渐增长，在提取 10 个系数的时候达到最高准确率，随后又开始下降。

4.5.5 按键位置对推测准确率的影响

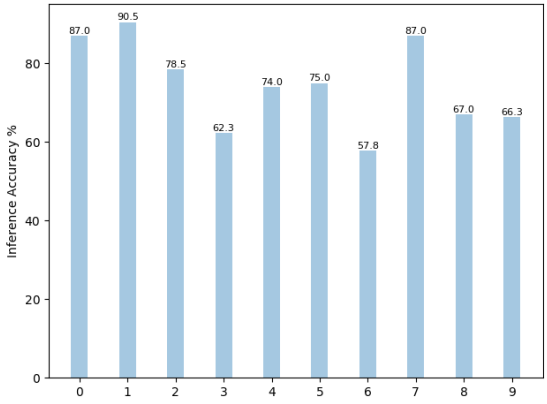

图 4-19 iPhone 6s 上各个按键的推测准确率

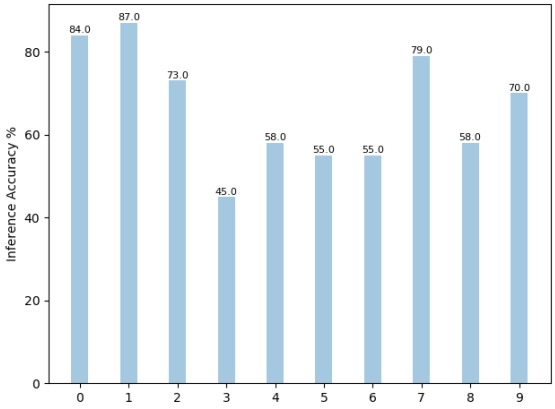

图 4-20 MI 3 上各个按键的推测准确率

本次实验还发现了一些有趣且重要的信息。图 4-19 和图 4-20。展示了 0 到 9 这 10 个数字所对应的推测准确率，从中可以看出不同数字的推测准确率并不相同，而且还呈现出与数字在屏幕上显示位置相关的分布规律，为了更直观的反映这种规律，本文将测试数据的混淆矩阵制作成如图 4-21 所示的灰度图。在该图中，每个子图的标签对应真实的分类值，图中的灰度色块表示对应该真实值的预测值概率分布，颜色愈深则概率越大。此外，在每个子图中将概率最大的 4 个按键预测值的概率特别标注以便于分析。

从图 4-21 中可以更为直观地看出，各按键的高概率预测值都围绕在真实值周围。并且对于每个按键标签，高概率预测值的个数一般都在 3 至 4 个左右。也就是说，每个按键的预测分布都呈现出高度集中的状态。除此之外，本文有如下发现：

（1） 边缘的按键预测准确率和预测按键集中度都高于中间的按键。

从图中可以发现 1、7、0 等按键的灰度色块更加集中，而靠内部的 5、8 等色块则相对分散，这说明外部的预测准确率高于内部。本文认为之所以会有这样的分布特点，是因为采集数据的志愿者在使用同样的力点按屏幕时，边缘更容易出现位移和摇晃，这样的运动更容易被传感器捕捉到，从而提升预测的准确率。而中间的按键在受点按操作时移动幅度则相对较小，不易采集和分辨。

（2） 屏幕偏右侧的按键的预测正确率要普遍低于屏幕左侧的按键。

从图中可以发现，屏幕中偏左侧的几个按键的灰度色块深度明显较右边更深，这说明左侧按键的预测准确率相对较高。在实验过程中本文注意到，采集数据的志愿者倾向于采取常用手，即右手握持手机进行单手点按操作。在这样的情况下，志愿者倾向于将手机的右下部与手掌牢牢贴合，以便更加稳固的握持手机。因为手机右下部与手掌紧密贴合，所以该方位的点按较难引起手机的震动或位移，也就较难在传感器数据中包含更多的点击信息。

如上所述，各按键的预测值总是围绕在真实值周围，各按键的分布都呈现出高度集中的状态。为了量化这一分布，本文定义按键距离为两个任意按键间的邻近程度：

$$\mathrm{Dist}_{A,B} = \mathrm{d}(\mathrm{A},\mathrm{B}) \tag{4-1}$$

具体表述为：若某按键 B 直接与某按键 A 相邻，则称 A、B 之间为 1 键距离，即$\mathrm{Dist}_{A,B} = 1$；倘若按键 C 与按键 B 相邻但与 A 不相邻，则称 A、C 之间为 2 键距离，即$\mathrm{Dist}_{A,C} = 2$，以此类推。

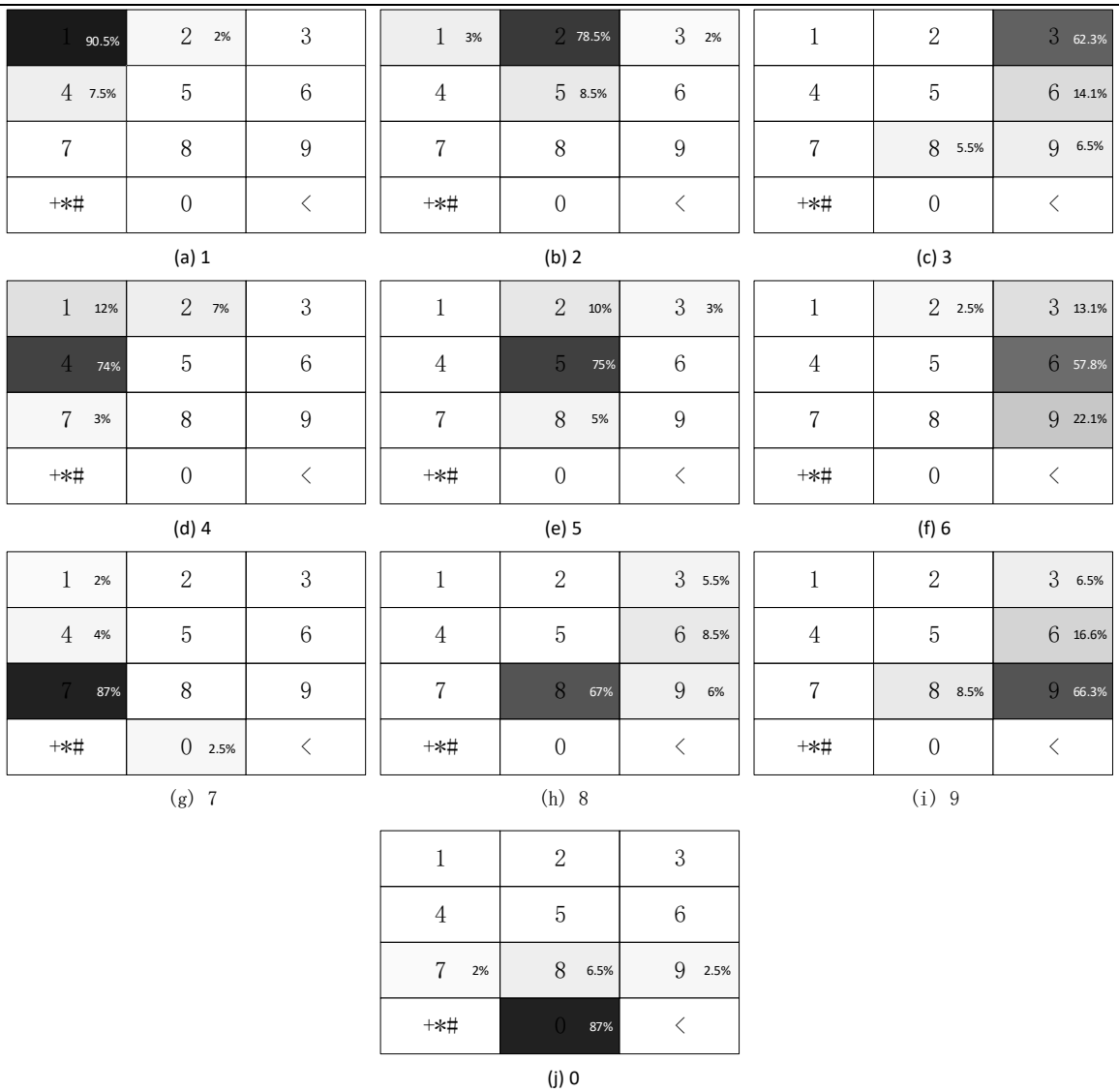

图 4-21 iPhone6s 各按键推测准确率分布示意图

以 iPhone 6s 的测试结果为例，本文绘制出按照按键距离分类的预测分布图，如图 4-22 所示。从中可以发现，预测值与真实值之间距离为 0 或 1 的情况占据了绝大多数的推测场景，其概率和约为 89.03%。

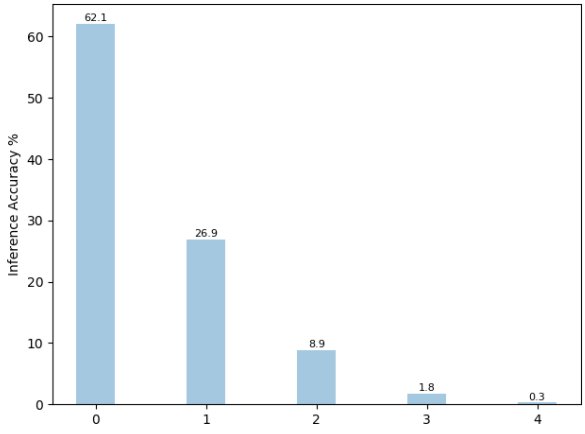

图 4-22 以按键距离归类的推测概率分布

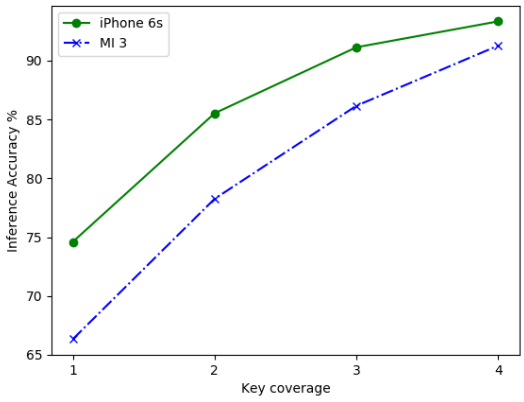

图 4-23 多次推测尝试时正确率的变化

根据这样的数据，可以得出结论，本文建立的模型的绝大多数错误预测都围绕在正确值的周围，

且其分布存在较为固定的分布概率。因此可以通过测试数据的混淆矩阵建立一个每个按键最有可能被误分类的表格。表 4 展示了根据 MFCC 频域特征和多层感知机分类算法的分类结果制作的误判表，其中每一列数据分别代表该数字被正确推测的概率，以及最有可能被误判成的其他数字及其概率。在后续的推测中可以根据表格来缩小搜索空间，图 4-23 展示了对传感器样本数据进行多次推测尝试时正确率的变化，尤其是进行至多 4 次尝试时 iPhone6s 和 MI 3 的正确率可以分别达到 94.34%和 91.28%，优于现有研究 4 次推测准确率 43%-71.6%的结果[10]。

表 4-5 每个按键最容易误判的数字及其概率

按键	首次就判决正确	第二高概率误判	第三高概率误判	第四高概率误判
1	90.5%	4：7.5%	2：2.0%	-
2	78.5%	5：8.5%	1：3.0%	3：2.0%
3	62.3%	6：14.1%	9：6.5%	8：5.5%
4	74%	1：12.0%	2：7.0%	7：3.0%
5	75%	2：10.0%	8：5.0%	3：3.0%
6	57.8%	9：22.1%	3：13.1%	2：2.5%
7	87%	4：4.0%	0：2.5%	1：2.0%
8	67%	6：8.5%	9：6.0%	3：5.5%
9	66.3%	6：16.6%	8：8.5%	3：6.5%
0	87%	8：6.5%	9：2.5%	7：2.0%

4.5.6 不同设备环境下准确率的比较

本文在 4.5.3 中提到，iPhone 6s 硬件环境下的预测准确率要比 MI 3 环境下稍高一点。结合实验环境的具体情况，本文认为出现这样的结果原因有以下几点：

（1） iPhone 6s 环境下的传感器数据采样率略高于 MI 3 环境：

经过调查发现，iPhone 6s 在进行实验时采样率保持在 120Hz 至 130Hz 之间，而 MI 3 的采样率则只有 80Hz 至 90Hz。传感器工作在较高的采样频率时，其记录下的数据能保留更多的有效信息，因此采样率是影响推测准确率的关键因素之一。

（2） 两款手机的用户界面和软键盘规格存在差距：

在实验过程中本文发现，两款手机弹出的输入数字键盘面板的大小和制式均有不同。iPhone 6s 的软键盘面板较 MI 3 稍大，如图 4-4 所示。当面板较大时，屏幕点按各点的相对空间距离就会更远，也就使得分类器能够更精确的辨认出不同的点按位置。

（3） Android 系统的震动反馈：

本文注意到，当使用 MI 3 进行输入实验时，每进行一次有效点击，手机就会发出一次震动反馈以示点按成功。这种震动也会被运动传感器捕捉到，从而成为运动传感器数据中的噪声，影响判决的准确性。相对地，iOS 系统中没有类似的震动反馈，所以这样的震动反馈也是影响推测准确率的因素之一。

4.6 本章小结

本章实现了一个 PIN 码推测原型系统，其中包含传感器数据采集系统、PIN 码推测概念验证程序、机器学习推测系统。本文实现的跨平台传感器数据采集系统可远程配置采集任务，支持多用户、多操作系统采集手机内置动作传感器数据，并实时传输到后台服务器持久化存储。本文对该系统的架构、数据采集流程进行了详细介绍，同时对 Sensor Logger 和 Sensor Server 在设计和实现过程中遇到的关键问题和挑战进行了详细介绍。为了演示在 Web 环境下程序无需用户许可即可获取传感器数据并发送到远程服务器导致 PIN 码泄漏的安全风险，本文设计并实现了一个需要用户输入 PIN 码的 PIN 码推测概念验证程序。PIN 码推测的核心在于机器学习推测系统，本文利用 Scikit-learn 搭建多层感知机模型，通过 Python 自动化实现预处理、特征提取、模型训练等步骤，最后进行实验测试。

本章从多个角度对实验结果进行了分析。首先仅使用时域特征对 SMO 算法、LMT 算法、KNN 算法、C4.5 算法、RandomForest（随机森林）算法以及多层感知机算法进行训练，其中多层感知机算法效果最好，以 62%的推测准确率从传感器数据中推测出用户输入的 PIN 码。接着，使用 MFCC 算法提取频域特征，以及之前提取的时域特征，对多层感知机分类器训练后达到 74.6%的推测准确率，当进行至多 4 次推测尝试时准确率可达 94.34%，优于现有研究 4 次推测准确率 43%-71.6%的结果。然后从 MFCC 算法流程中预加重处理、提取的系数个数对推测准确率的影响进行实验和分析，并从实验结果得出按键位置、不同手机环境对推测准确率的影响。最终得出结论，通过智能手机浏览器获取动作传感器数据后，可以通过机器学习模型以较高的成功率推测出用户输入的 PIN 码。这种安全隐患应该引起广大用户和开发者的重视，防止恶意用户利用传感器信息推测用户的隐私信息。

第五章 防御措施与安全建议

本文 4.5 节中的实验证实了在用户不知情的情况下通过智能手机传感器利用机器学习技术的确可以推测出 PIN 码，随着传感器技术的广泛应用和智能手机的快速普及，本文研究中揭示的这一漏洞将很有可能被恶意开发者利用并进行非法的网络活动，窃取用户的隐私信息。该领域的大部分研究都将采集程序设计成手机客户端应用程序，或将恶意代码嵌入正常的应用中以诱骗用户下载。这样设计的缺点显而易见，因为客户端应用需要用户主动地下载和安装，会引起用户的警觉，成功率较低。本文揭示了一种新的可能性，那就是直接通过 Web 应用来窃取用户的传感器数据。Web 程序具备强大的泛用性，攻击者只需要在设计程序时添加对多设备和多浏览器的支持，该 Web 程序就能够运行在横跨多个平台的绝大多数移动设备上。在这样的技术架构下，攻击者根本无需用户安装恶意程序，用户只要点击了含有恶意代码的 Web 页面，攻击就会立时被触发。此外，通过结合跨站脚本攻击(XSS)等网络攻击技术，攻击者可以实现针对特定用户群体的主动攻击。

本文的目的在于让更多的人关注手机隐私安全，尤其是目前动作传感器数据可以被应用自由使用的情况下，很难保证用户的传感器数据不被恶意收集和使用。最好能够让人们意识到传感器数据也属于隐私数据，并重新审视和制定传感器数据获取的规则，然而这在短时间内很难实现。为此本文提出一种基于噪声注入的防御措施，并进行实验验证其有效性，随后又针对操作系统设计、应用开发、用户使用提出一些安全建议，下面依次做详细介绍。

5.1 威胁模型分析

首先对基于手机内置动作传感器的边信道攻击进行简单建模。如前文所说，这种形式的边信道攻击通常包含两个步骤：模型训练和隐私推测，在训练阶段，需要使用测试传感器数据对分类模型进行训练，完成后在隐私推测阶段对传感器数据进行按键内容推测。

在模型训练阶段，恶意应用需要获取用户在屏幕上敲击按键时手机内置的动作传感器数据，以及此时的时间戳和具体的按键内容等信息，以便训练分类模型。在本模型中假设用户安装或使用假冒成正常应用实则窃取传感器数据的恶意应用，比如伪装成打地鼠游戏，在屏幕原本放置数字按键的位置上放置地鼠，然后按照需要的模式构造地鼠在不同位置出现的顺序，当用户点击时相当于敲击了数字按键，恶意应用从而获取训练数据，并可用其训练分类模型。

在隐私推测阶段，恶意应用需要在用户输入隐私信息时将传感器数据传输到远程服务器，以便利用之前训练好的模型推测出用户的隐私信息。在本模型中假设恶意应用运行在手机后台，或者通过 XSS 等方式嵌入用户浏览的网页中，在现有操作系统对传感器的权限控制策略中，应用无需用户许可即可读取传感器数据，因此恶意应用可在用户无感知的情况下将传感器数据传输到远程服务器。

5.2 现有防御措施分析

目前已有针对基于手机内置动作传感器边信道攻击的一些防御研究，本节对现有研究中提出的一些防御措施进行简要介绍，并分析它们存在的一些缺陷。

5.2.1 手机震动噪声

基于传感器的边信道攻击主要依靠用户在屏幕敲击时手机产生的物理震动和位移，如果在用户开始输入的时候让手机开始自动震动，恶意应用采集到的传感器信号中就包含了一些干扰信号，理论上就可以抵御此类边信道攻击[43]。然而该文献的作者并没有对这个防御措施进行实验论证，在前

文 4.5.6 小节中，本文提到基于 Android 系统的 MI 3 手机下推测准确率要低于基于 iOS 的 iPhone 6s，其中一项重要原因就是在 MI 3 手机中在屏幕上敲击按键式手机会产生震动来反馈按键动作，从实验中可以看出，在用户输入时让手机震动的确可以降低推测准确率，然而根据本文实验得出的结论，即使存在震动的情况下，依然可以得到较高的推测准确率。

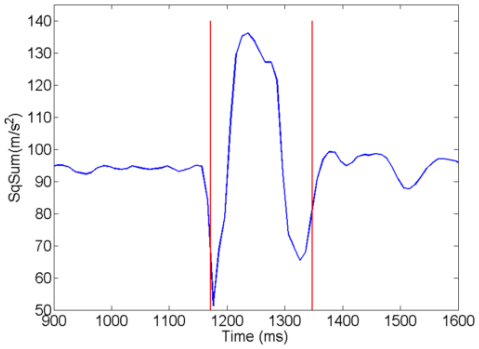

图 5-1 没有震动时加速度传感器数据图示

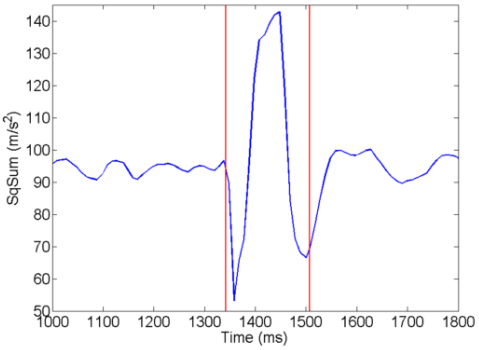

图 5-2 有震动时加速度传感器数据图示

为了进一步分析手机内置电机产生震动时对动作传感器数据的影响，本文绘制了在基于 Android 系统的 MI 3 上按键时加速度传感器的数据，图 5-1 展示了在取消震动反馈后加速度传感器的数据图示，图 5-2 展示了在有震动反馈后加速度传感器的数据图示，其中 SqSum 是 x,y,z 三轴数据的平方和。从上面两幅图中可以看出，尽管手机内置电机产生的震动确实会对动作传感器数据产生影响，然而这种影响并不足以掩盖按键的关键信息，从图中依然可以清晰的分辨出按键动作的开始与结束，甚至连数据变化的模式都非常相似，这也印证了本文在 4.5 节中得出的实验结果。因此可以得出结论，在用户输入时让手机产生震动来抵御基于传感器的边信道攻击收效甚微。

5.2.2 降低采样率

降低传感器的采样率可以有效降低基于传感器的边信道攻击推测准确率[44]。通常来说，采样率越高，按键推测的准确率就越高，因为这意味着有更多的传感器数据样本，分类模型可以从中获得更多细节和特征，从而更好的区分各个按键。因此降低传感器的采样率就可以有效降低按键推测的准确率，甚至让其不可用，然而降低传感器的采样率同样会影响系统中正常应用的使用，比如计步器。除此之外，随着机器学习技术的发展，此类按键推测模型对采样率的要求越来越低，甚至在 20Hz 采样率下仍可以推测[45]。因此降低采样率并不能有效抵御基于传感器的边信道攻击。

5.2.3 减小手机震动

通过传感器进行边信道攻击主要利用用户在输入过程中敲击屏幕时手机产生的物理震动和微小位移，如果能够尽可能消除这种位移，或者让震动的幅度变得很小，那么攻击方采集到的传感器数据就可能由于传感器本身的精度而变得模糊，推测精度也可能因此而降低。目前已有学者尝试通过给手机套上厚厚的橡胶手机壳来吸收因输入操作而导致的手机震动和位移[45]，理论上确实可以减小信息泄漏。然而这种方式要求用户必须套上厚厚的手机壳，且手机壳本身的材质也会影响最终的防御效果。除此之外，随着传感器技术的不断发展，手机中内置的动作传感器精度越来越高，可以想象在未来配备高精度传感器后，即使用户套上厚厚的手机壳进行输入操作，仍然有可能被传感器边信道攻击推测出隐私信息。另外还有一个重要因素，很少有用户愿意使用这种特殊的手机壳，因此这种防御措施恐怕在现实生活中很难有效应对传感器边信道攻击。

5.2.4 加强传感器权限控制

很多研究都建议将手机内置的动作传感器：加速度传感器、陀螺仪传感器、方向传感器看作包含用户隐私信息的敏感传感器，并据此加强对动作传感器的权限控制，任何应用都必须得到许可才

能读取数据。然而这种防御措施的可行性依赖于用户对传感器安全有一定了解，并且在安装应用之前会仔细阅读应用要求获取的权限列表。本文在 3.1 小节中提到，根据现有调查，只有 30%的用户知道动作传感器的作用，几乎 90%的用户表示并不觉得动作传感器会泄漏他们的隐私。加强传感器的权限控制在理论上肯定对抵御基于传感器的边信道攻击有一定作用，然而考虑到现实情况，短时间内这种防御措施并不能如预期那样有效。

5.3 基于噪声注入的防御措施

本文提出一个新的基于传感器噪声注入的防御措施，通过在传感器数据中加入可编程控制的自适应噪声来抵御基于动作传感器的边信道攻击。与现有防御措施相比，本方案无需改变用户的操作习惯，也无需降低传感器的采样率，同时只在键盘弹起时注入自动生成的随机噪声，对于用户来说完全透明。同时由于加入的噪声是根据用户在键盘上敲击时的信号特性生成的，因此恶意应用想要从中剔除干扰信号变得非常困难，因此可以有效抵御基于传感器的边信道攻击。

5.3.1 架构设计

正常应用在用户通过屏幕键盘输入数据时通常是不需要高精度的传感器数据的，因此如果在用户输入时操作系统对此时采集到的传感器数据加入一些噪声，将极大程度的降低此类 PIN 码推测恶意应用的成功率。然而如本文所描述的 PIN 码推测框架引入了频域分析，因此注入的噪声信号不能是普通的白噪声，否则很容易被过滤，最好是加入与原始信号相似的噪声信号。本文在实验过程中收集了大量的传感器数据，可以据此构造一些具有类似频域特征的噪声信号。本文设计的基于噪声注入的防御措施架构如图 5-3 所示，主要由两部分组成，分别是后台检测服务和噪声生成服务。

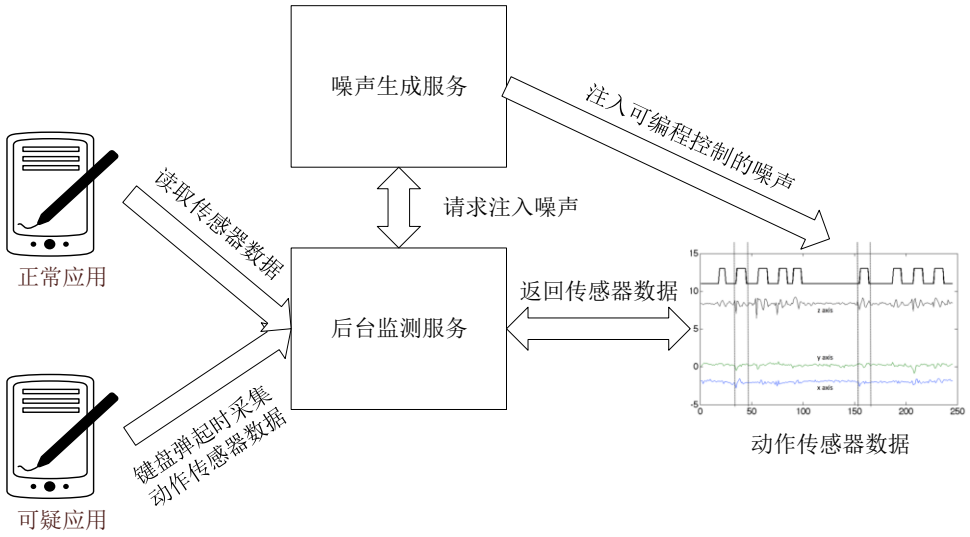

图 5-3 基于噪声注入的防御架构

噪声生成服务负责生成自适应噪声并注入到操作系统采集的动作传感器信号中，在整个系统初次运行时，会要求用户进行初始化操作，具体来说就是要求用户进行一些在各种握持姿势下的输入操作，同时记录此时手机内置的各种动作传感器数据。然后分析采集到的信号，利用其特征来生成自适应的随机噪声，比如各个轴上的最大值、最小值、信号能量等。通过这种方式生成的随机噪声具有与用户在屏幕上进行按键操作类似的特征，当将这些随机噪声注入到系统采集到的动作传感器数据后，对于恶意攻击者来说就无法从中有效推测隐私信息，就算恶意攻击者知道目标手机采用了这种防御措施，想要从采集到的信号中剔除随机噪声也非常困难。

后台检测服务常驻在系统中，它负责检测当前哪些应用需要请求读取动作传感器数据，同时判

断此时用户是否在进行输入操作。如果应用在请求传感器数据时用户并没有输入操作，后台检测服务不做任何操作，直接让应用获取原始传感器数据；如果应用在请求动作传感器数据时键盘是弹起的状态，就会向噪声生成服务发起请求，在系统采集到的动作传感器数据中注入特定随机噪声来抵御边信道攻击。由于注入的噪声与用户在键盘上敲击时产生的信号具有相似的特性，因此对于正常应用来说功能并不会受到影响，因为对于普通应用来说此时加入的噪声就好比用户在键盘上进行无意义的敲击，而这种使用场景对于其他需要使用动作传感器的应用来说并不会有所影响。而对于恶意应用来说，从注入了特定随机噪声的动作传感器数据中推测隐私信息会变得极其困难，其推测成功率会大幅降低。

5.3.2 实验分析

本节将通过实验来测试本文提出的防御措施的有效性。为此本文设计了三个实验场景，分别是：没有防御措施，有防御措施但攻击者使用无噪声的数据训练，有防御措施且攻击者使用含噪声的数据训练。通过这三种场景下攻击者推测 PIN 码的准确率来判断本文提出的防御措施的有效性。

无防御措施：在这种场景下，系统没有任何防御措施，攻击者可以直接获取原始的传感器数据进行训练和 PIN 码推测。这也是本文 4.5 节中进行的实验，我们选择在这种场景下攻击者推测出 PIN 码的成功率作为基准，以此来比较本文提出的防御措施对这种传感器侧通道攻击的防御效果。

有防御措施但攻击者使用无噪声的数据训练：在这种场景下，系统中已经采用本文提出的传感器噪声注入框架，然而攻击者在此前已经通过某种途径使用未注入噪声的原始传感器数据训练好了分类模型。当攻击者想要获取此系统中动作传感器的数据时，传感器噪声注入框架会在动作传感器原始信号中注入特定的随机噪声，攻击者随后使用注入了噪声的传感器数据进行推测，也就是使用无噪声的数据训练好的模型来从含噪声的数据中推测 PIN 码。

有防御措施且攻击者使用含噪声的数据训练：在这种场景下，系统已采用本文提出的传感器噪声注入框架，攻击者使用已注入特定随机噪声的数据进行模型训练，并随后使用含有噪声的传感器数据进行 PIN 码推测。

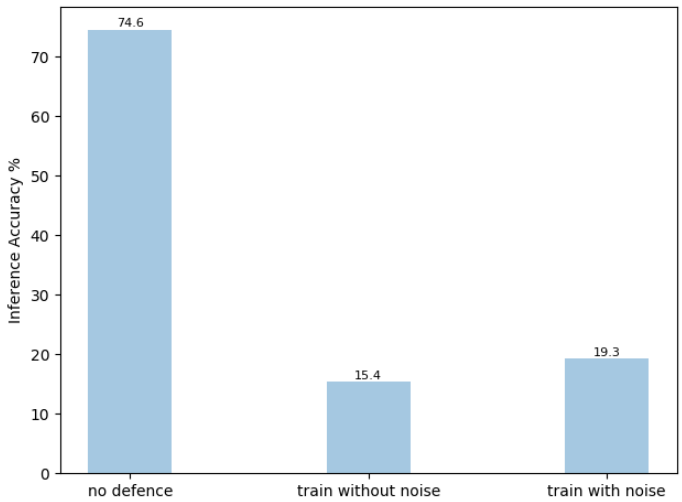

图 5-4 基于噪声注入防御措施的效果比较

由于整个实验环境与 4.5.1 小节中提出的一样，就不在此赘述。本文依然使用 4.4 中的机器学习推测系统进行 PIN 码的推测，使用多层感知机分类算法进行推测，实验使用的传感器数据也是 4.5.2 小节中通过本文设计的跨平台传感器数据采集系统从基于 iOS 的 iPhone 6s 采集的数据。

图 5-4 展示了在三种不同的实验场景下，同样的机器学习推测系统从传感器数据中推测出 PIN

码的准确率。其中 no defence 表示第一个场景，即在没有采用防御措施下的 PIN 码推测准确率，也就是本文 4.5 节中得到结果，其准确率达到了 74.6%；train with noise 表示第二个场景，即使用无噪声的传感器数据进行模型训练，使用含噪声的传感器数据进行推测，其准确率只有 15.4%；train with noise 表示第三个场景，即使用含噪声的传感器数据进行训练和推测，其准确率比第二个场景稍高一些，达到 19.3%。从图中可以看出，在引入传感器噪声注入框架后，无论是使用含噪声还是不含噪声的数据进行训练，攻击者的推测概率都急剧降低。

从实验结果中可以得出结论，本文设计的基于传感器噪声注入的防御措施，可以极大降低使用机器学习方法通过动作传感器推测 PIN 码的准确率，从而有效抵御此类边信道攻击。

5.4 安全建议

尽管本文提出的基于传感器噪声注入的防御措施可以极大程度降低推测准确率从而抵御侧通道攻击，然而仅仅靠这个还无法实现真正的安全。用户使用智能手机输入敏感信息，这个简单的操作涉及操作系统、应用和用户这三个环节，其中任一环节出现漏洞都可能导致隐私泄露，本节从操作系统设计、应用开发、用户使用等层面提出了一些安全建议供各方参考。

5.4.1 操作系统设计

5.4.1.1 传感器权限控制

在前期调研和系统设计过程中，本文发现现在的主流手机操作系统对于运动传感器调用的限制很少，在大部分情况下，第三方应用调用运动传感器时根本不会触发权限控制和用户提醒，这就给了相关恶意程序以可乘之机。基于这样的现状，本文建议 OEM 厂商加强运动传感器调用的权限管理，在应用调用动作传感器时能够给出明确提醒。结合本文研发的传感器数据采集系统和一些相关领域的研究，本文发现除了动作传感器的调用外，一些特定的权限组合可能包含潜在的安全风险，比如本文设计的传感器数据采集系统中需要的网络访问权限和动作传感器数据读取权限。本文认为操作系统设计者应该重点关注这样的权限组合，并及时告知用户潜在的安全风险。

5.4.1.2 恶意应用识别

本文多次提到，传感器的采样频率对最后的推测结果准确率有直接的相关性，因此降低传感器的采样率能够有效地减少数据被窃取的风险。然而强行降低传感器的采样率同样也会劣化合法应用的传感器调用体验，影响到智能手机的正常使用。如果能够识别出疑似恶意应用并对其降低采样率，就可以在功能需求和安全需求之间达成平衡。

正常应用在读取传感器数据时通常会有所选择，不会像恶意应用一样同时读取多种传感器数据。除此之外，正常应用只会在某些时刻读取传感器数据，而恶意应用通常会持续不断的读取传感器。大多数情况下，正常应用读取传感器数据后只会在本地进行分析和运算，以便实现设备姿态检测和人机交互。而恶意应用在读取传感器数据的同时还经常伴随着网络数据传输。因此，可以在操作系统中实现一个传感器监测程序，每当有应用发起读取传感器数据的请求时，记录应用程序的 PID、传感器类型的组合、读取传感器持续的时间以及此时网络通信的使用量。有了传感器监测程序记录的数据，可以从中提取相应的特征向量，然后使用常见的二分类算法如 SVM 识别恶意应用。操作系统可根据这一结果对疑似恶意应用采取降低传感器采样率或禁止读取传感器等措施。

5.4.2 应用开发

一些应用开发者可能认为本文研究中揭示的风险与开发者无关，然而这样的想法是错误的。大部分的恶意程序通过伪装合法程序，或直接在合法应用中嵌入恶意代码，然后将应用重新发布于应用市场并诱导用户下载。恶意代码的嵌入可能会对应用开发者的声誉和软件盈利带来沉重的打击，

因为消费者根本无法区分合法应用和加入恶意代码的应用，进而将隐私数据的丢失归咎于原本合法应用的开发者。本文认为，开发者应该全力配合操作系统厂商，使用操作系统厂商建议的应用安全开发最佳实践，以防止应用仿冒等恶意行为。

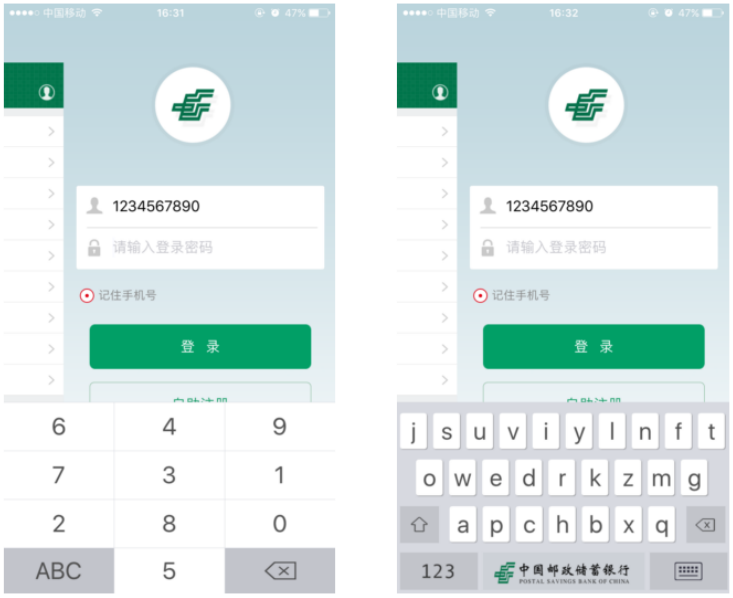

图 5-5 某银行手机应用的乱序软键盘

在一些有敏感数据流通的应用程序，如电子金融或网络支付应用中，数据的丢失将更为致命，开发者必须严密防护传感器数据的输入推测。基于本文系统设计的原理和实验的结果，本文建议应用的开发者能够自行设计数字或字母软键盘，并且在每次弹出时都能够更换各按键的排列顺序。这样的设计虽然不能够完全解决问题，但能够降低这种传感器侧通道攻击的成功率。目前已有一些应用开发者采用了类似的方法，如图 5-5 所示的某银行手机应用的软键盘。本文倡议有数据安全需求的应用开发者采用类似设计来加强应用的安全性，防止用户隐私信息泄漏。

5.4.3 用户使用

Android 系统和 iOS 系统的应用市场生态有所不同。在非越狱条件下，iOS 用户只能从官方商店 AppStore 上下载应用，而该商店内的应用都经过了苹果公司官方的严格审查，一般不会出现植入恶意代码的程序。然而 Android 系统由于谷歌的官方应用市场 Google Play 不能在国内使用，而目前市面上的很多第三方应用市场的应用审查能力参差不齐，这就给了攻击者可乘之机。因此，本文建议普通 iOS 用户不要进行越狱操作，建议 Android 用户从较为权威正规的应用商店下载应用程序。

本文揭示了一种不同于客户端应用程序攻击的 Web 应用攻击方法。对普通用户来说，这种攻击更难发现和防护。从本文的前期调研和实验结果来看，通过在输入时频繁变换姿态和手机握持方法可以有效地降低输入信息被窃取的可能性。然而这样的操作建议对于用户来说很难实现，并不具备实际的可行性。为了能够有效地降低信息泄漏风险，本文认为用户应该对于 Web 异常弹出页面和非正常的账号或密码的输入请求保持高度警惕，不要连接公共 Wifi 进行敏感数据的输入操作，尽量选择使用 HTTPS 加密认证的 Web 服务，使用可信的手机进行敏感信息的输入操作。

5.5 本章小节

本文 4.5 节实验证实了恶意应用可以无需用户许可获取传感器数据并从中推测用户 PIN 码，面对这一安全风险，本章首先对这种安全威胁进行建模分析，随后对现有的一些防御措施进行了介绍和分析，并指出其缺陷。接着介绍本文提出的一种新的防御措施，即通过在传感器数据中加入可编程控制的自适应噪声来抵御基于动作传感器的边信道攻击，并通过三种不同场景下的实验来测试抵抗边信道攻击的效果，结果显示 PIN 码推测的准确率在引入噪声注入后急剧降低，从而证明本文提出的方法对基于动作传感器的边信道攻击具有良好的抵御效果。接着本章从操作系统、应用开发、用户使用等层面提出了一些安全建议。在操作系统层可以通过加强传感器的权限控制、使用模式识别和机器学习检测恶意应用等方式来抵御此类攻击；在应用开发层面，可以设计乱序键盘；在用户使用层面，在涉及隐私信息输入的场景中，不要连接公共 Wifi，仔细检查域名和证书，防止上钓鱼网站。然而最好的方式是让人们意识到传感器数据也属于隐私数据，希望在未来能够重新审视和制定传感器数据获取的规则。

第六章 总结与展望

6.1 总结

在移动互联网时代，智能手机已经不仅仅是一个用来打电话沟通的工具了，它还承担着社交、电子金融和网络购物等任务。用户的智能手机中存储着越来越多的敏感信息和个人隐私，比如支付密码、短信记录、通讯录和位置信息等。如何获取用户的隐私信息一直是网络攻防中的重要话题，近几年，攻击者开始把目光投向智能手机内置的动作传感器，目前已有研究发现，通过建立机器学习模型可以从动作传感器数据中推测用户的输入内容，从而构成旁路攻击窃取用户隐私。针对现有研究基于传感器数据时域特征的 PIN 码推测准确率较低的问题，本文提出了一种新型的传感器数据频域特征提取算法—改进的 MFCC 算法。在此基础上构建了一种基于时域频域特征相结合的 PIN 码推测通用框架。该框架通过手机内置动作传感器采集按键数据并进行预处理，使用改进的 MFCC 算法提取频域特征，结合时域特征共同对神经网络算法中的多层感知机模型进行训练并推测用户 PIN 码。最后本文设计并实现了原型系统，对该框架的可行性进行了验证，同时还提出了一种基于噪声注入的防御措施。本文主要工作内容可以总结为如下几点：

（1） 对 Android 和 iOS 操作系统的手机内置传感器安全机制、对动作传感器的权限控制机制进行研究。对机器学习中数据采集、预处理、特征提取、模型训练等各个阶段的理论和算法进行介绍，并对本文将会使用的几种分类算法做了分析。

（2） 提出一种基于时域频域特征相结合的 PIN 码推测通用框架。在传感器数据采集方面，本文提出了一种基于 Web 和 NodeJS 的跨平台多终端手机传感器数据采集方法，可通过手机浏览器运行采集程序将传感器数据实时传输到后台服务器存储。为了解决不同平台下传感器采样率不同导致的数据差异，本文使用三次样条插值算法对传感器数据做平滑滤波预处理得到一致的数据样本。在特征提取方面，本文提出一种改进的 MFCC 算法来提取动作传感器数据频域特征。该算法考虑动作传感器信号与语音信号均具有频域能量分布集中、信号特异性强的相似特性，针对动作传感器信号自身特点，去除原 MFCC 算法中预加重和汉明窗口的处理，推导了动作传感器信号 0-4kHz 频域范围内 Mel 弯折公式，提取出动作传感器信号的频域特征。在模型训练方面，本文基于神经网络中的多层感知机算法建立分类模型，选择 ReLU 作为激活函数，Adam 为优化算法，通过反向传播算法进行训练。

（3） 设计并实现了基于时域频域特征相结合的 PIN 码推测原型系统。该系统由传感器数据采集系统、机器学习推测系统、PIN 码推测概念验证程序构成。可在 Android 和 iOS 平台下采集 PIN 码输入时的动作传感器数据样本，使用不同超参数对多层感知机模型进行迭代训练得到最佳结果，在 XSS 模拟攻击环境下实现 PIN 码推测。本文通过在校园里随机邀请的方式采集了 10k 个样本进行训练，经测试表明该系统在 iOS 和 Android 平台下 PIN 码 4 次推测准确率分别达到 94.34%和 91.28%，优于现有研究 4 次推测准确率 40%-71.6%的结果。

（4） 针对通过传感器推测 PIN 码可能导致隐私泄漏的安全风险，本文提出一种基于噪声注入的防御措施，通过在传感器数据中加入可编程控制的自适应噪声来抵御基于动作传感器的隐私窃取。经测试表明在引入该防御措施后 PIN 码识别的准确率降低到 19.3%。进一步，本文还从操作系统设计、应用程序开发、用户使用等方面的安全改进进行探讨。

6.2 展望

本文研究使用智能手机内置的动作传感器来获取用户的隐私信息，并通过实验验证了可以利用传感器数据和机器学习以较高准确率推测出用户输入的 PIN 码。回顾本文实验的设计结构和各个流程，仍然存在改进的空间，现将一些反思和改进思路列出，以为后续的拓展研究提供参考。

（1） 在实验过程中，本文注意到传感器的采样频率会对最后预测结果的准确率造成显著的影响。目前的实验中并没有对采样率对准确率的影响作出量化，考虑到不同型号的手机硬件不同，操作系统版本不同，采样率也各不相同。为了更好地反映日常生活的使用场景，可以考虑加入对传感器采样率的控制，在数据采集系统添加采样率控制模块，以不同的采样频率收集传感器数据，针对这一组数据分别训练分类器并测试，得出采样率与预测准确率的具体关系。

（2） 在进行特征提取工作时，本文综合已有的研究，从时域和频域提取了 294 维特征，根据机器学习的相关理论，过大的特征集不仅会带来学习效率的降低，还会有模型过拟合的风险。因此，本文也尝试使用程序进行特征权重比对及特征筛选，然而通过实验发现收效甚微。在今后的研究中，可以使用深度学习来规避人为的特征选择工作，尽可能多的提取样本信息。

（3） 本文将主要的精力放在数字 PIN 码的推测和窃取上，然而在日常生活中除了数字以外，用户还要进行很多文本输入。本文的一些研究方法和思路可以直接应用于文本的推测。由于字母键盘的特性与数字键盘不尽相同，用户在进行文本输入时的输入习惯也与输入数字时不同，所以想要进行文字的推测，还需要对研究路线和具体技术进行修改，然而文本的字符之间存在内在的逻辑，比如英文单词或汉语词组之间的组合是较为固定的，因此可以引入 NLP（自然语言处理）的相关技术来帮助进行文本的推测。

参考文献

[1] Marquardt P, Verma A, Carter H, et al. (sp) iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers[C]. Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011: 551-562.
[2] Cai L, Chen H. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion[J]. HotSec, 2011, 11: 9-9.
[3] Nahapetian A. Side-channel attacks on mobile and wearable systems[C]. Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual. IEEE, 2016: 243-247.
[4] Vanja Svajcer. Malicious cloned games attack Google Android Market[EB/OL]. Naked Security:http://nakedsecurity.sophos.com/2011/12/12/ malicious-cloned-games-attack-google-android-market/, December 2011.
[5] Cai L, Chen H. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion[J]. HotSec, 2011, 11: 9-9.
[6] Xu Z, Bai K, Zhu S. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors[C]. Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM, 2012: 113-124.
[7] Miluzzo E, Varshavsky A, Balakrishnan S, et al. Tapprints: your finger taps have fingerprints[C]. Proceedings of the 10th international conference on Mobile systems, applications, and services. ACm, 2012: 323-336.
[8] Cai L, Chen H. On the practicality of motion based keystroke inference attack[C]. International Conference on Trust and Trustworthy Computing. Springer, Berlin, Heidelberg, 2012: 273-290.
[9] Owusu E, Han J, Das S, et al. Accessory: Keystroke inference using accelerometers on smartphones[J]. Proc. of HotMobile, 2012.
[10] Aviv A J, Sapp B, Blaze M, et al. Practicality of accelerometer side channels on smartphones[C]. Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012: 41-50.
[11] Spreitzenbarth M, Freiling F, Echtler F, et al. Mobile-sandbox: having a deeper look into android applications[C]. Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, 2013: 1808-1815.
[12] Peterson L E. K-nearest neighbor[J]. Scholarpedia, 2009, 4(2):1883.
[13] 周志华. 机器学习[M]. 北京: 清华大学出版社，2016.
[14] Michie D, Nuñez M, Podgorelec V, et al. Quinlan, JR C4. 5: Programs for Machine Learning[J]. 1993.
[15] Umanol M, Okamoto H, Hatono I, et al. Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems[C]. Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on. IEEE, 1994: 2113-2118.
[16] Sitanggang I S, Yaakob R, Mustapha N, et al. An extended ID3 decision tree algorithm for spatial data[C]. Spatial Data Mining and Geographical Knowledge Services (ICSDM), 2011 IEEE International Conference on. IEEE, 2011: 48-53.
[17] Quinlan J R. Induction of decision trees[J]. Machine Learning, 1986, 1:81–106.
[18] Safavian S R, Landgrebe D. Asurvey of decision tree classifier methodology[J]. IEEE transactions on systems, man, and cybernetics, 1991, 21(3):660–674.
[19] Landwehr N, Frank E, Hall M. Logistic model trees[J]. Machine Learning, 2005, 59(1-2):161–205.
[20] Liaw A,Wiener M. Classification and regression by randomForest[J]. R news, 2002, 2(3):18–22.
[21] Breiman L. Random forests[J]. Machine learning, 2001, 45(1):5–32.
[22] Jin X, Hu X, Ying K, et al. Code injection attacks on html5-based mobile apps: Characterization, detection and mitigation[C]. In: Proceedings of the 2014ACMSIGSAC Conference on Computer and Communications Security. 2014. 66–77.
[23] Mehrnezhad M, Toreini E, Shahandashti S F, et al. Stealing PINs via mobile sensors: actual risk versus user perception[J]. International Journal of Information Security, 2017: 1-23.
[24] McKinley S, Levine M. Cubic spline interpolation[J]. College of the Redwoods, 1998, 45(1): 1049-1060.
[25] Bichler D, Stromberg G, Huemer M, et al. Key generation based on acceleration data of shaking processes[C]. International Conference on Ubiquitous Computing. Springer, Berlin, Heidelberg, 2007: 304-

317.
[26] Davis S B, Mermelstein P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences[M]. Readings in speech recognition. 1990: 65-74.
[27] O'Shaughnessy D. Linear predictive coding[J]. IEEE potentials, 1988, 7(1): 29-32.
[28] BenAbdelkader C, Cutler R, Davis L. Stride and cadence as a biometric in automatic person identification and verification[C]. Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on. IEEE, 2002: 372-377.
[29] Furui S. Speaker-independent isolated word recognition based on emphasized spectral dynamics[C]. Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86. IEEE, 1986, 11: 1991-1994.
[30] Allen F R, Ambikairajah E, Lovell N H, et al. Classification of a known sequence of motions and postures from accelerometry data using adapted Gaussian mixture models[J]. Physiological measurement, 2006, 27(10): 935.
[31] Mogran N, Bourlard H, Hermansky H. Automatic speech recognition: An auditory perspective[M]. Speech processing in the auditory system. Springer, New York, NY, 2004: 309-338.
[32] Khan A M, Lee Y K, Lee S Y, et al. Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis[C]. Future Information Technology (FutureTech), 2010 5th International Conference on. IEEE, 2010: 1-6.
[33] Poppe R. A survey on vision-based human action recognition[J]. Image and vision computing, 2010, 28(6): 976-990.
[34] Karantonis D M, Narayanan M R, Mathie M, et al. Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring[J]. IEEE transactions on information technology in biomedicine, 2006, 10(1): 156-167.
[35] Bao L, Intille S S. Activity recognition from user-annotated acceleration data[C]. International Conference on Pervasive Computing. Springer, Berlin, Heidelberg, 2004: 1-17.
[36] Lukowicz P, Ward J A, Junker H, et al. Recognizing workshop activity using body worn microphones and accelerometers[C]. International conference on pervasive computing. Springer, Berlin, Heidelberg, 2004: 18-32.
[37] Garrett J J. Ajax: A new approach to web applications[J]. 2005.
[38] Tilkov S,Vinoski S. Node.js: Using JavaScript to build high-performance network programs[J]. IEEE Internet Computing, 2010, 14(6):80–83.
[39] Paluszek M, Thomas S. MATLAB Machine Learning[M]. Apress, 2016.
[40] Yu D, Eversole A, Seltzer M, et al. An introduction to computational networks and the computational network toolkit[J]. Microsoft Technical Report, 2014, MSR-TR-2014:112.
[41] Hall M, Frank E, Holmes G, et al. TheWEKAdata mining software: an update[J]. ACM SIGKDD explorations newsletter, 2009, 11(1):10–18.
[42] Scikit-Learn. Choosing the right estimator[EB/OL]. http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html. 2017.
[43] Owusu E, Han J, Das S, et al. ACCessory: password inference using accelerometers on smartphones[C]. Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications. ACM, 2012: 9.
[44] Miluzzo E, Varshavsky A, Balakrishnan S, et al. Tapprints: your finger taps have fingerprints[C]. Proceedings of the 10th international conference on Mobile systems, applications, and services. ACm, 2012: 323-336.
[45] Aviv A J, Sapp B, Blaze M, et al. Practicality of accelerometer side channels on smartphones[C]. Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012: 41-50.

致谢

光阴似箭，时光如梭。在东南大学读研的三年生活就要随着毕设论文的交稿而告终了，虽然在漫长的一生中三年的时光很短暂，但却是我人生中最纯真的岁月，能够在美丽的东南大学完成硕士生涯让我倍感荣幸！

本次研究以及论文是在我的导师宋宇波副教授的亲切关怀和悉心指导下完成的。宋宇波老师知识渊博、谦逊温和，为人诚信宽厚，工作上认真负责，科研上更是精益求精，他严谨的治学精神、严肃的科学态度深深的感染和激励着我。宋宇波老师在毕设的选题、理论的分析、模型的验证、论文的写作等过程中对我悉心指导、循循善诱，让我能够顺利完成本次研究和学位论文。宋老师不仅学术水平高超，在体育运动上更是出类拔萃，每年都会参加马拉松比赛并获得优异成绩，他对生活乐观积极的态度激励着我不断努力，让我不再惧怕未知的挑战。

在此，还要感谢实验室的各位同学对本论文的帮助。信号专业的蒋程同学不辞辛劳，花费大量时间帮助我调整算法，刘袁同学在深度学习模型的调参方面给我了很多重要的建议，宋睿师弟的前期工作帮助我快速起步，本次研究的顺利完成离不开他们的帮助和指导，在此表示感谢！还有同门杨慧文、张克落、罗平，师弟黄强、魏一鸣、武威、宋睿、石伟、李轩，感谢你们一直以来的陪伴和支持，能够与你们一起度过这三年时光我感到十分快乐！还有好友兼舍友尹浩浩同学，这三年来你一直是我的榜样！以及健身房的老李在健身方面的悉心指导，让我拥有健康的体魄。

最后要感谢含辛茹苦养育我的父母，你们对我的爱和支持，让我能够勇敢的在这个世界自由飞翔，在未来的日子里我会好好报答你们的养育之恩！还要感谢我的女友徐娇娇，谢谢你一直以来的陪伴，谢谢你在我遇到挫折时的鼓励和支持，希望我能够在未来陪你走完一生。

东南大学，我的母校，在这里我度过了本科四年时光、研究生三年时光。能够在如此美丽的校园度过人生最美好的年华，让我感到非常幸福！祝愿母校能够越来越好，早日实现世界一流大学的目标！

作者简介

董启宏（1993—），男，藏族，青海西宁人，现为东南大学信息安全实验室硕士研究生，研究方向为基于移动终端的信息安全。

- **攻读硕士学位期间发表的论文**

[1] Song R, Song Y, Dong Q, et al. WebLogger: Stealing your personal PINs via mobile web application[C]. Wireless Communications and Signal Processing (WCSP), 2017 9th International Conference on. IEEE, 2017: 1-6.

[2] 董启宏. 基于心音的身份识别算法研究[C]. 2016 年东南大学校庆学术报告会. 2016.12