Demo for Detection of Driver's Distraction

一 项目简介

1 功能

本项目使用支持向量机算法， 根据司机的 EEG 信号特征判断司机的注意力情况（集中/分散）。主要由信号发送器 (data sender) 和信号处理器 (data analyzer) 两个部分组成。

1.1 信号发送器 (data sender)

（1）训练：

运行信号发送器后，用户可以指定主体（1-8）和 SVM 种类（一类/二类）。在这之后，用户点击“训练”按键，程序将从指定文件夹中读取用户的 EEG 特征值（25 维）并使用指定种类的 SVM 训练模型，并将模型（txt 文件）保存在指定位置。

（2）发送数据：

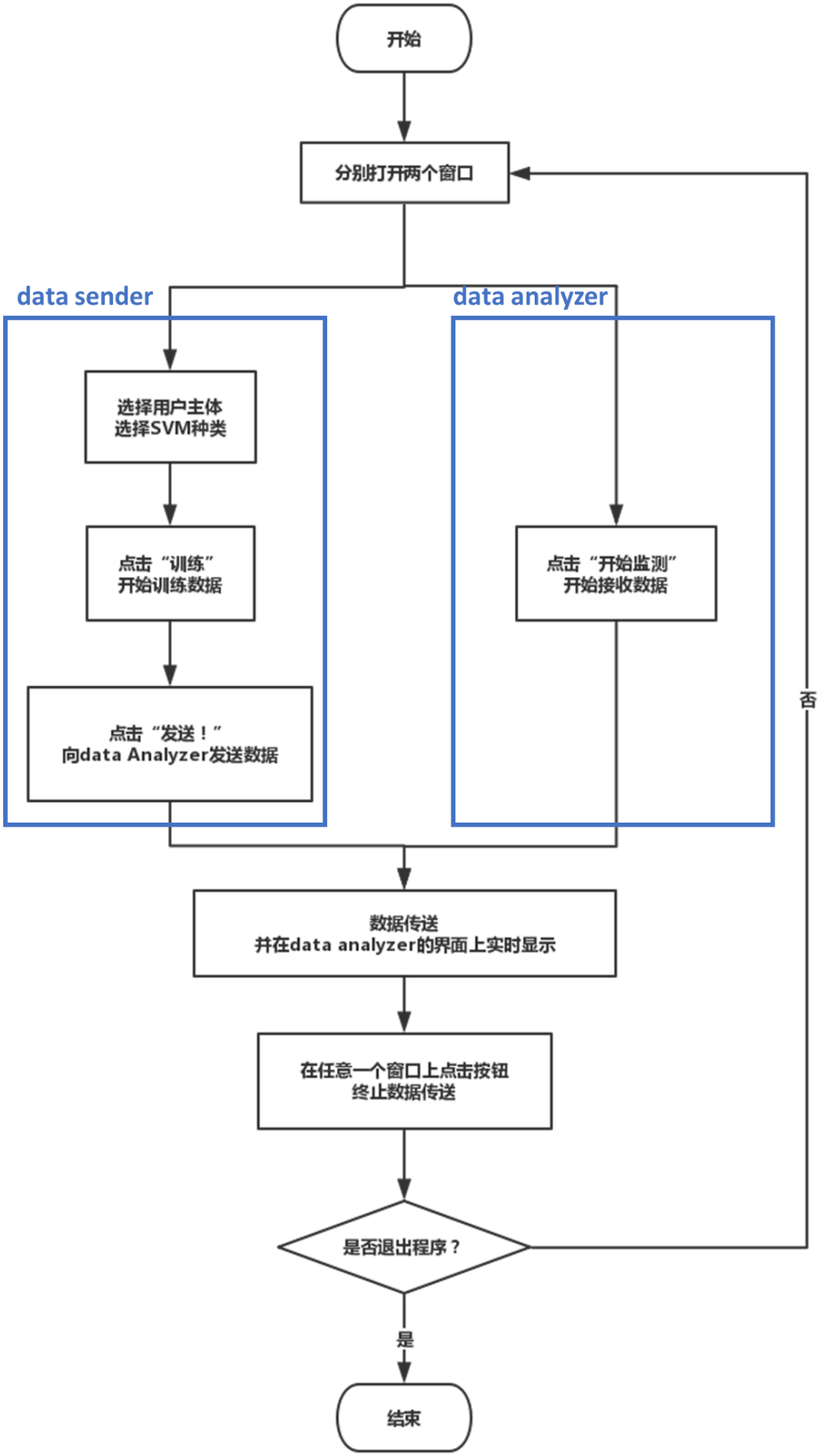
在用户点击“发送”键后，它可以通过 socket 将指定主体的特征值发送给信号处理器进行处理。不断重复上述过程，直至用户停止任务或者该主体的全部特征值均被发送。

1.2 信号处理器 (data analyzer)

实时监测：

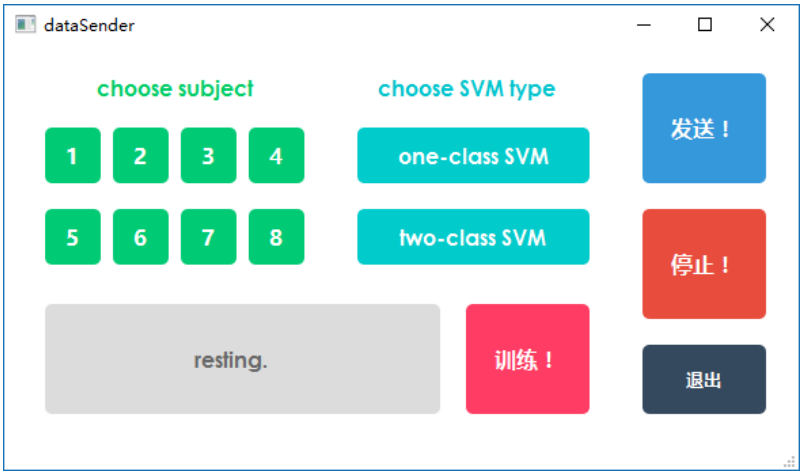
在用户点击“开始监测”后，程序通过 socket 接收来自信号发送器的信号，并结合已保存的模型，判断司机的注意力情况（focused/distracted），将接收到的数据和判断结果在 GUI 界面上实时显示。

2. 用户操作方案

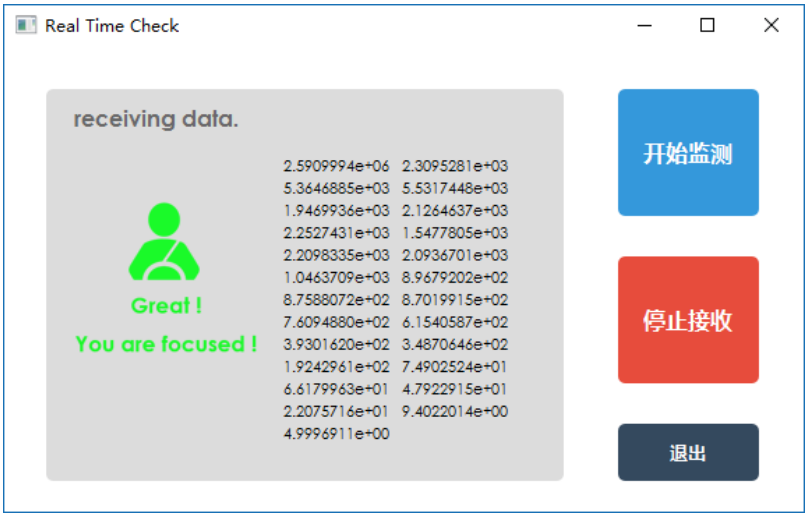

开始
分别打开两个窗口
data sender
data analyzer
选择用户主体
选择SVM种类
点击“训练”
开始训练数据
点击“发送！”
向data Analyzer发送数据
点击“开始监测”
开始接收数据
否
数据传送
并在data analyzer的界面上实时显示
在任意一个窗口上点击按钮
终止数据传送
是否退出程序？
是
结束

3. 项目 GUI 截图

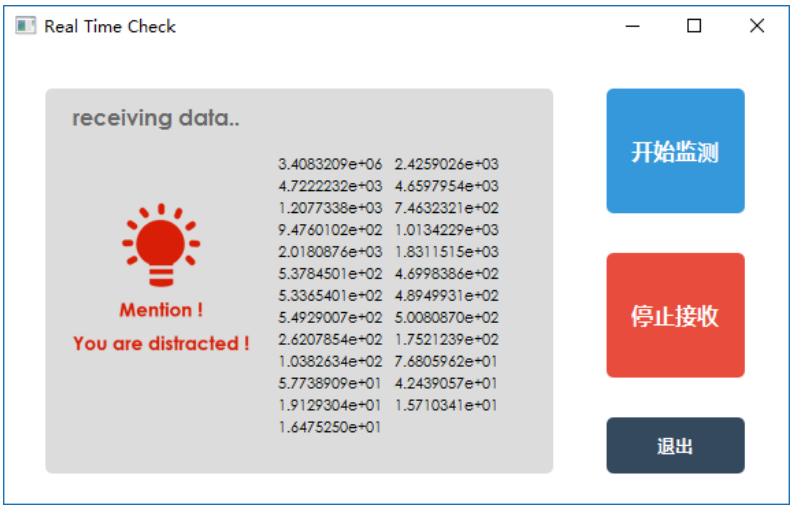
（1）data sender



（2）data analyzer（Real Time Check 窗口）（**focused 状态**）



（3）data analyzer（Real Time Check 窗口）（**distracted 状态**）



4. 项目开发环境

本项目的基本开发环境为 Visual Studio 2015，并通过 Visual Studio 的 Qt 插件集成了 Qt 的开发环境。

二. 各模块实现方案

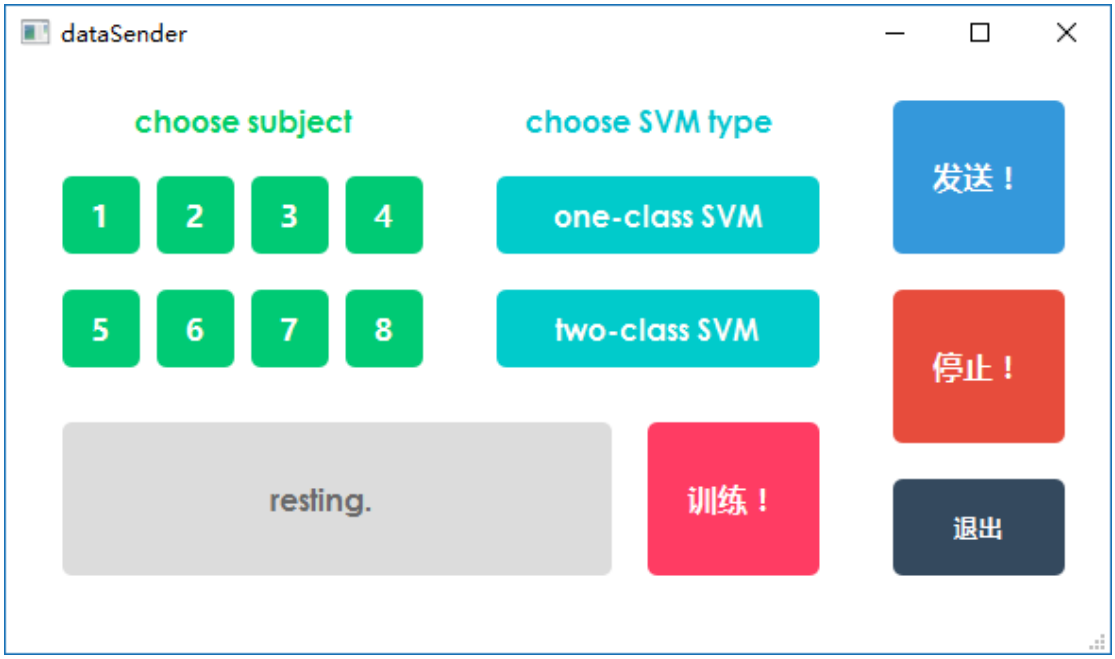
（一）data sender 部分

1. 整体设计说明

dataSender.h，dataSender.cpp，ui_dataSender.h 共同实现了该窗口，其中 ui_dataSender.h 是通过在 Qt designer 中进行绘图后自动生成的。

svm.h 由 svm.cpp 具体实现，并被包含在了 dataSender.h 中，辅助 SVM 算法的实现。

2. 实现的窗口布局



按键说明：

（1）在左上角“choose subject”处选择监测主体（1-8），在 choose SVM type 处选择 SVM 类型（一类/二类）。

（2）点击“训练!”键后，程序开始训练，并在左下角的提示框显示“training…”，当提示框变回“resting…”说明训练结束。

（3）右侧三个按钮分别用来向 data analyzer 发送数据，停止数据的发送，退出此程序。

注：可在 dataSender.h 中的 public slots 部分查看点击各个按键时触发的槽函数。

3. 主要数据成员/函数成员介绍

注：各成员的含义也可参考"dataSender.h"中的注释。

主要成员介绍

结构体	point	训练数据的存储格式： feature成员存储特征值 value成员存储label值
枚举类型	currentStatus	用于存储程序的当前状态
数据成员	breakLoop	为TRUE时，打破外层循环
	subNo	主体编号（1-8）
	stopSendData	为TRUE时，停止发送数据
	point_list	由point类型结构体构成的单链表
	svmType	SVM种类：1-一类SVM，0-二类SVM
	dimension	EEG特征的维度（25）
	crtSt	当前状态
	nu/gamma/coef0/degree	SVM的参数
函数成员	setBtnQss	设置按钮的样式
	ThreadSend	线程：与dataAnalyzer通信
	ThreadDraw	线程：在GUI上显示程序的当前状态
	ThreadTrain	线程：训练数据

4. 运行机制说明

注：各私有数据成员/私有成员函数的含义可参见 dataSender.h 中的详细注释。

（1）启动后，进行初始化。之后创建用于显示程序当前状态的线程 ThreadDraw。在线程 ThreadDraw 中，根据程序的当前状态（变量 crtSt），在程序界面输出程序状态的提示。

（2）用户点击相关按钮选择了监测主体或者 SVM 种类后，在相应的槽函数中修改相应变量的值（主体：subNo（int 类型）；SVM 种类：svmType（int 类型，1 表示一类支持向量机，0 表示二类支持向量机））。

（3）用户点击“训练”键后，在槽函数 on_train_clicked 创建 ThreadTrain 线程，用于进行数据的训练。ThreadTrain 线程的运行机制详见（4）。

（4）在 ThreadTrain 线程中，根据 SVM 种类的不同，分别调用相关函数依次完成五个任务：生成训练数据所在的文件名称、生成并写好标签(label)文件、设置训练参数、读取训练数据和标签、进行训练并保存模型。完成各功能的函数名称如下表所示：

功能描述	函数名称
生成训练数据所在的文件名称	one-class SVM： generateFileName_oneClass two-class SVM： generateFileName_twoClass
生成并写好标签(label)文件	one-class SVM： writeTrainLabel_oneClass two-class SVM： writeTrainLabel_twoClass
设置训练参数	one-class SVM： setPara_oneClass two-class SVM： setPara_twoClass
读取训练数据和标签	one-class SVM： readTrainData_oneClass two-class SVM： readTrainData_twoClass
进行训练并保存模型	one-class SVM： run_oneClass two-class SVM： run_twoClass

下面对选择一类 SVM 和二类 SVM 时，上述各函数处理时的区别阐述如下（蓝色标注部分为不同 SVM 种类的不同处理方式）：

若选择了**一类**支持向量机：

① 调用 generateFileName_oneClass 函数，生成三个训练数据文件的文件名。

② 调用 writeTrainLabel_oneClass 函数，生成三个 label 文件：**每一行均为一个字符，即‘1’**，表示训练数据对应的状态均为 distracted。

③ 调用 setPara_oneClass 函数，根据指定个体，程序内部指定参数 nu,gamma,coef0,degree。

④ 调用 readTrainData_oneClass 函数，通过读取三个训练数据文件和三个训练 label 文件，将各信号的 feature 和 label 存入 point_list 中。

⑤ 调用 run_oneClass 函数，利用 LibSVM 实现的算法，读取 point_list 并训练出模型，将模型存在"modle.txt"文件中。

若选择了**二类**支持向量机：

① 调用 generateFileName_twoClass 函数，生成四个训练数据文件的文件名。

② 调用 writeTrainLabel_twoClass 函数，生成四个 label 文件：前三个文件每一行均为一个字符‘1’，第四个文件每一行均为一个字符‘0’。

③ 调用 setPara_twoClass 函数，根据指定个体，程序内部指定参数 nu,gamma,coef0,degree。

④ 调用 readTrainData_twoClass 函数，通过读取四个训练数据文件和四个训练 label 文件，将各信号的 feature 和 label 存入 point_list 中。

⑤ 调用 run_twoClass 函数，利用 LibSVM 实现的算法，读取 point_list 并训练出模型，将模型存在"modle.txt"文件中。

（5）点击“开始!”按钮后，创建进程 ThreadProc，引入 socket 机制。TreadProc 的具体机制详见（6）。

（6）TreadProc 的具体机制如下：

① 创建协议，向 data analyzer 发送连接请求。

② 收到 data analyzer 的第一个回复后，告知 data analyzer 选择的主体编号（1-8）。

③ 收到 data analyzer 的第二个回复后，告知 data analyzer 选择的 SVM 种类。

④ 再收到回复后，便进入 while 循环，并在循环中重复下述动作：通过 ifstream 从指定文件中读取一行字符串（由 25 个特征值（如 2.5352844e+06）组成）；将该字符串进行发送，等待 data analyzer 的回复，收到回复后，读取同一文件的下一行字符串或者（若此文件已经读取完毕，则）下一个文件的第一行字符串，周而复始。

⑤ 若遇到以下三种情况之一，则关闭协议，跳出循环：

• 所有文件中所有内容都已经读取完毕(stopSendData == TRUE)。

• 用户点击“停止!”键（stopSendData == TRUE）。

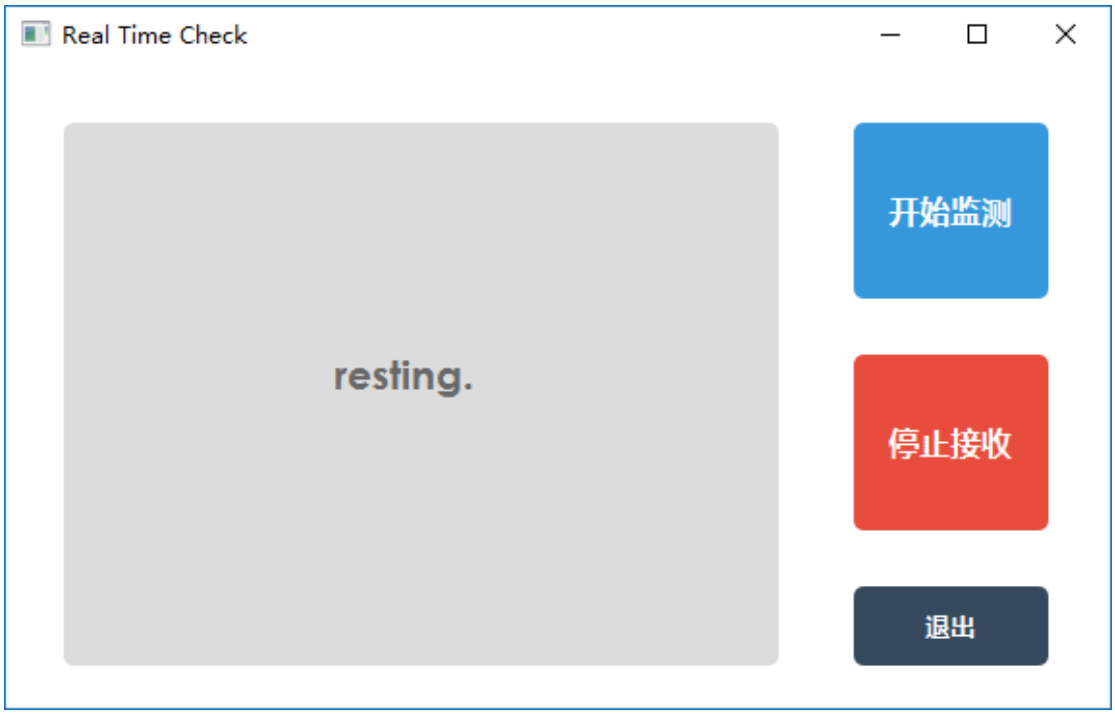
• 收到 data analyzer 的回复为"Stop now!"（breakLoop == TRUE，由用户在 data analyzer 的窗口点击“停止接收”导致）。

（二）data analyzer 部分

1. 整体模块设计说明

（1）同样，realTimeCheck.h，realTimeCheck.cpp，ui_realTimeCheck.h 共同实现了该**窗口（realTimeCheckWindowClass 类）**，其中 ui_realTimeCheck.h 是通过在 Qt designer 中进行绘图后自动生成的。

2 实现的窗口布局



按键说明：

（1）开始监测：点击后，开始等待 data sender 发来的连接请求。

（2）停止接收：点击后，停止接收数据并告知 data sender 不必再进行发送。

（3）退出：退出程序

注：可在 realTimeCheck.h 中的 public slots 部分查看点击各个按键时触发的槽函数。

3. 主要数据成员/函数成员介绍

注：各成员的含义也可参考"dataAnalyzer.h"中的注释。

主要成员介绍

结构体	point	训练数据的存储格式： feature成员存储特征值 value成员存储label值
枚举类型	currentStatus	用于存储程序的当前状态
数据成员	svmType	SVM种类：1-一类SVM，0-二类SVM
	point_list	由point类型结构体构成的单链表
	dimension	EEG特征的维度（25）
	exitCheckLoop	为1时，结束与dataSender的socket通信
	recvBuf	暂存dataSender发送的数据
	subNo	主体的编号（1-8）
	crtSt	当前状态
函数成员	setBtnQss	设置按钮的样式
	ThreadDetect	线程：与dataSender通信，监测driver的状态
	ThreadDraw	线程：在GUI上显示程序的当前状态

4. 运行机制说明

注：各私有数据成员/私有成员函数的含义可参见 realTimeCheck.h 中的详细注释。

（1）开始程序后，创建进程 ThreadDraw，用于显示当前程序所处的状态。

（2）点击“开始接收”后，创建进程 ThreadProc，用于创建协议与 data sender 通信。ThreadProc 的具体机制详见（3）。

（3）ThreadProc 具体机制：

① 首先创建协议，等待连接请求。

② 收到连接请求后，发送"Which subject will I detect?"，并接收由 data sender 发送的监测主体编号，并保存在变量 subNo（int 类型）中。

③ 再发送"which type of SVM did you use?"，收到由 data sender 发送的 SVM 类型（0 或 1），并保存在变量 svmType（int 类型）中。

④ 根据这两个信息，确定需要读取的模型的文件名（string 类型变量 modelFileName）。

⑤ 进入 while 循环。在循环中，读取由 data sender 发送的特征值并根据模型进行判断，并将判断结果和收到的数据在界面上实时显示。

⑥ 若遇到以下两种情况之一，则关闭协议，跳出循环：

• 用户点击“停止接收”键（exitCheckLoop == TRUE）。

• 收到 data sender 的发送字符串为"I will stop send data."（因所有文件均已读取完毕或在 data sender 端人为终止了数据发送）。

三. 补充说明

1. 此程序中，用每个主体的 phase1,2,3（一类 SVM）或 phase1,2,3,5（二类 SVM）的数据进行训练，用 phase4 和 phase6 的数据进行模拟监测（即，data sender 先后发送指定主体的 phase4 和 phase6）。

2. 此项目中，因引入了线程机制，大多数成员为静态成员，并在.h 头文件中定义，在.cpp 文件的最前处进行初始化。"

3. dataSender 将训练好的模型（txt 文件）存储在"E:\\"下。若想改变路径，修改 dataSender.cpp 中的 modelDir 成员（的初始化语句）即可。

四. 存在的问题/项目的后续工作

1. 各参数还不能完美的适应各个 subject 的各个 phase，仍需进一步调整（需要调整的参数主要有：svm_type, kernel_type, nu, gamma, coef0, degree）。

五. 相关资源

1. 可在 **LibSVM 官网**下载 **LibSVM 工具包**，并参考其中的 readme 文件了解各参数的含义，以便调整训练的参数：

LibSVM 官网：http://www.csie.ntu.edu.tw/~cjlin/libsvm/

2. data sender 和 data analyzer 的 **socket 实现部分**，主要参考以下博文：

C++ Socket 编程：http://blog.csdn.net/rexuefengye/article/details/12145569