

---

# 网络流量分类系统总体设计说明书

版本<1. 1. 160116. beta >

## 修订历史记录

日期	版本	说明	作者
2016/1/12	1. 0. 160112. beta	beta 版，完成基本算法、用户交互，简单容错；	ChirlChen
2016/1/16	1. 1. 160116. beta	1、添加连续型数据离散化功能，当前版本可使用连续型数据进行分类 2、优化数据统计方法，数据统计结果更加精确；	ChirlChen



## 目录

概 要 .....	- 1 -
第一章 需求分析.....	- 2 -
1.1 功能要求.....	- 2 -
第二章 总体结构.....	- 3 -
2.1 体系结构.....	- 3 -
2.2 系统总体功能设计.....	- 3 -
2.2.1 系统流程图.....	- 4 -
2.2.2 数据流图.....	- 4 -
第三章 系统详细设计.....	- 6 -
3.1 各功能模块流程图.....	- 6 -
3.1.1 GUI 模块 .....	- 6 -
3.2 类结构设计.....	- 9 -
3.2.1 类结构及功能说明.....	- 9 -
3.2.2 UML 类图 .....	- 13 -
3.3 数据结构设计.....	- 13 -
3.3.1 贝叶斯算法存储容器表结构.....	- 13 -
3.3.2 程序数据结构定义.....	- 15 -
第四章 关键技术.....	- 18 -
4.1 关键技术的应用.....	- 18 -
4.2 关键技术的一般说明.....	- 19 -
第五章 方案实施的技术路线和实施计划.....	- 22 -
5.1 实施的技术路线.....	- 22 -
5.2 实施计划.....	- 24 -
缩写词表.....	- 25 -
参考文献.....	- 26 -



## 概 要

网络流量分类器是基于朴素贝叶斯算法实现的一套网络流量分类系统。该系统通过获取网络流量包的各特征值在假设某种网络包类型下的概率以及该网络类型出现的概率的乘积，依次对每种可能出现的网络类型进行假设，求得乘积，然后比较出所得到的最大乘积，这个乘积的假设条件(即前面假设的网络类型)便是该流量包的网络类型。

整个软件系统主要包含两大主要模块。一、GUI 系统；二、贝叶斯算法实现系统。其中 GUI 系统是基于 QT 框架而实现。贝叶斯算法，目前主要实现了其中的离散型数据的算法(即统计法)，而对于连续型数据的处理算法，有待后续开发。就目前的离散型数据分类情况来看，只要建模数据量充足，分类准确率基本可以达到 99.8%。但是如果建模数据量不够，则算法分类准确率便不那么稳定。

## 第一章 需求分析

### 1.1 功能要求

设计实现 GUI 界面,能够在界面上选择训练数据集和测试数据集,能够在界面上勾选用于训练和分类的特征。

读入 arff 格式的训练和测试数据,并用程序实现基于朴素贝叶斯的流分类算法,能够在界面上点击运行朴素贝叶斯分类算法进行训练。

用测试数据集对算法进行测试,并将分类的结果显示在 GUI 界面上。

## 第二章 总体结构

### 2.1 体系结构

整个系统的功能概要已在概要部分进行讲解。在本节中将以系统框图的形式展示整个系统的总体结构。如图 2.1 所示：

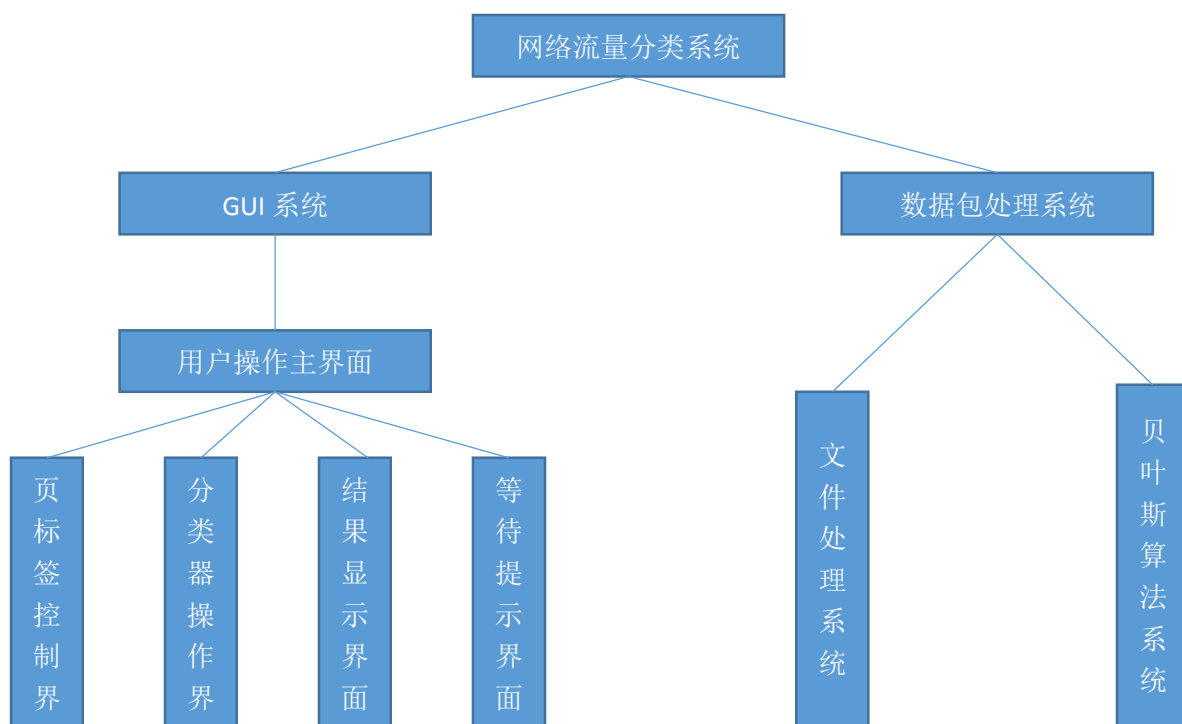


图 2.1 网络流量分类系统框图

### 2.2 系统总体功能设计

本节主要从系统的包含的各功能模块的角度对系统总体功能进行设计。系统流程图描述了该系统的主要执行流程，数据流图描述了系统的数据流大体流向，以及系统的各模块对数据流的处理情况。

### 2.2.1 系统流程图

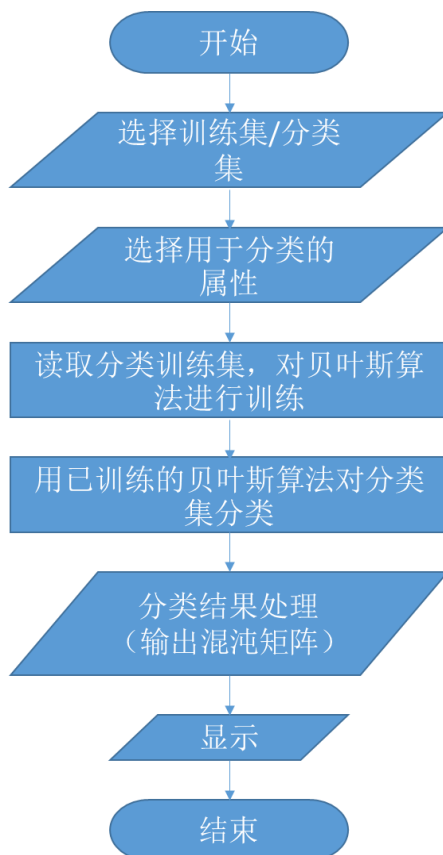


图 2.2 系统总体流程图

### 2.2.2 数据流图

本小节主要以数据流图的形式展示整个系统数据流的具体流向和数据处理的大致过程。如图 2.3:

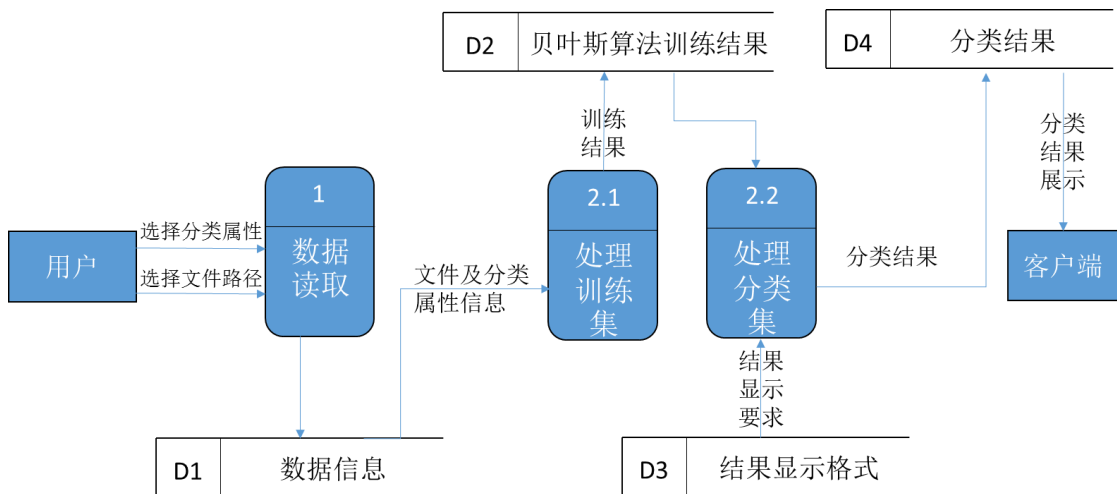


图 2.3 系统数据流图

如图 2.3 所示, 用户需要通过客户端, 选择好所需要的训练集和



分类集文件，并勾选用于分类的属性，然后系统下一步会进行数据读取，并在 D1 中存下所勾选的属性，以及其他必要的信息。接下来这些信息将流向处理训练集模块，对贝叶斯算法进行训练，并将训练结果保存在 D2 中。训练结束后便可以对之前传入的分类集进行分类处理了，同样，分类结果将被保存下来（保存在 D4），同时将结果展示到客户端。

## 第三章 系统详细设计

本章主要对前面的系统总体结构进行细化，具体设计系统各功能模块的流程，各功能模块所用到的数据结构，以及类结构。出于练习的目的，系统模块设计上尽量包含一些知识，促进学习。

### 3.1 各功能模块流程图

系统包含的主要功能模块可分为：GUI 模块、贝叶斯算法实现模块。下面将以流程图的方式对这两个模块的功能进行具体的分析。

#### 3.1.1 GUI 模块

考虑到系统的可扩展性，以及易操作性等。GUI 系统主要设计成了标签页的形式，其中包括主界面、分类器标签页界面、结果展示标签页界面以及等待提示窗等。如果在之后需要对系统的功能进行扩展，可以直接以设计新的界面，然后以标签页的形式嵌入到当前系统中。

##### 3.1.1.1 系统界面

下面是整个系统的效果展示图。

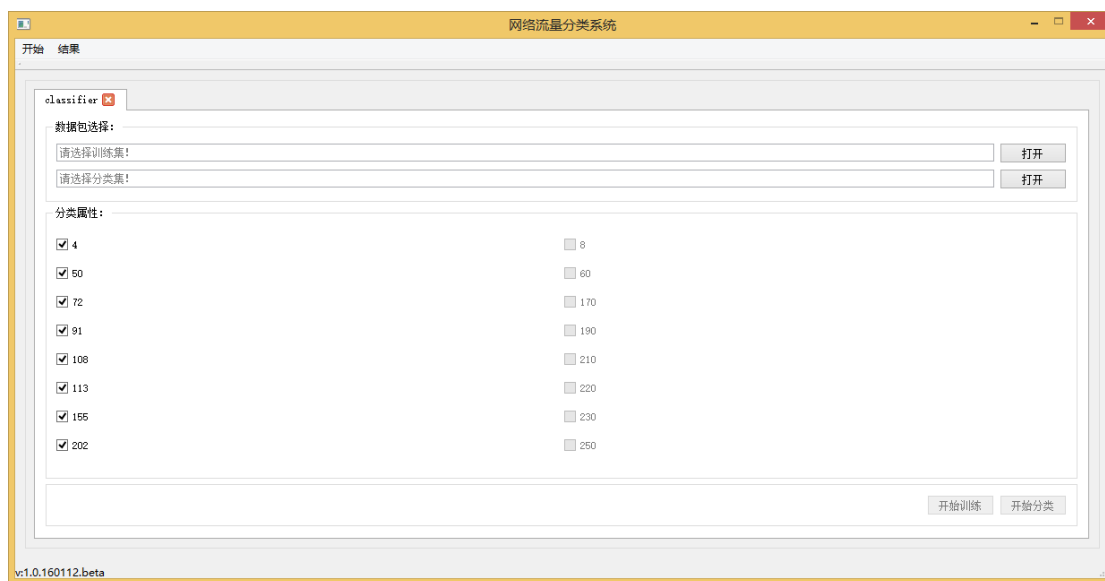
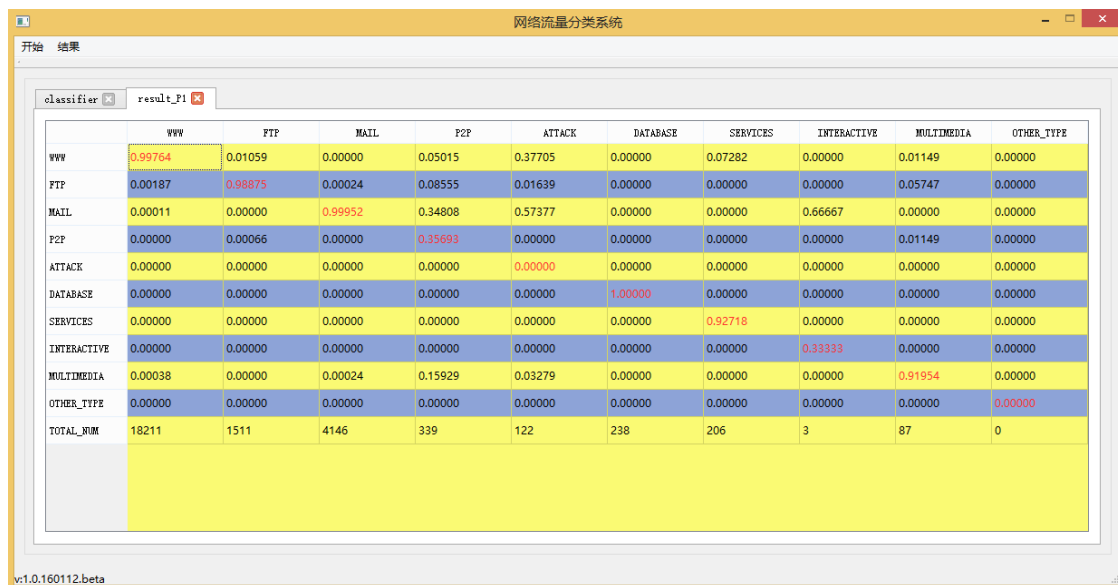


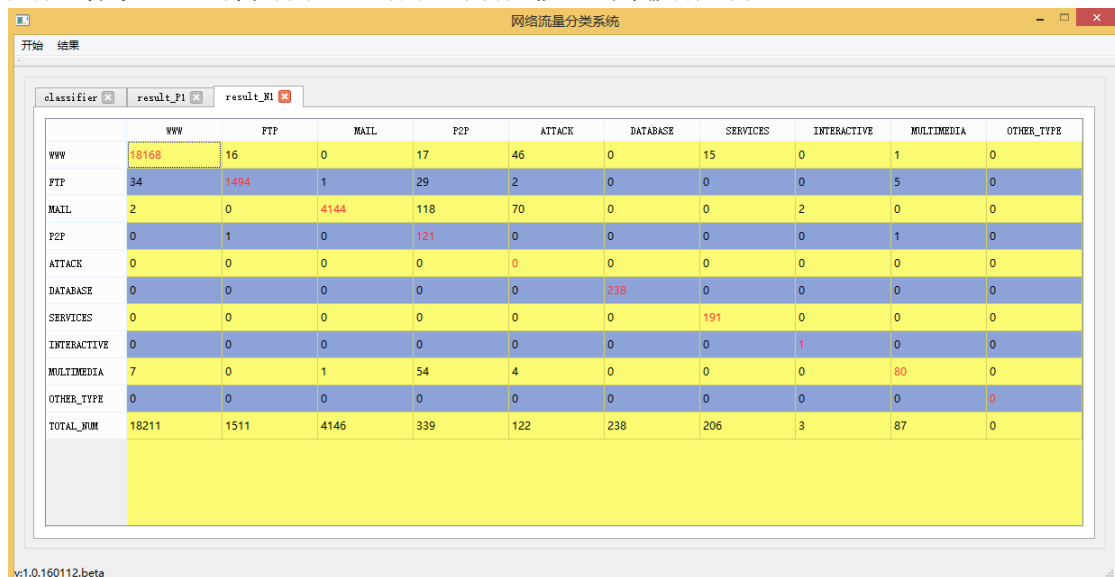
图 3.1 分类器标签页

分类器标签页(图 3.1)是整个系统的核心功能页,可以通过点击开始菜单栏里的“统计法分类”(或 Alt + Shift + C)打开。其中便主要实现了需求分析章节中所提到的所需功能。为了减少用户操作的失误率,这里把部分有逻辑先后关系的按钮设置为了不可触发状态,当



	WWW	FTP	MAIL	P2P	ATTACK	DATABASE	SERVICES	INTERACTIVE	MULTIMEDIA	OTHER_TYPE
WWW	0.99764	0.01059	0.00000	0.05015	0.37705	0.00000	0.07282	0.00000	0.01149	0.00000
FTP	0.00187	0.98875	0.00024	0.08555	0.01639	0.00000	0.00000	0.00000	0.05747	0.00000
MAIL	0.00011	0.00000	0.99952	0.34808	0.57377	0.00000	0.00000	0.66667	0.00000	0.00000
P2P	0.00000	0.00066	0.00000	0.35693	0.00000	0.00000	0.00000	0.00000	0.01149	0.00000
ATTACK	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
DATABASE	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000
SERVICES	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.92718	0.00000	0.00000	0.00000
INTERACTIVE	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.33333	0.00000	0.00000
MULTIMEDIA	0.00038	0.00000	0.00024	0.15929	0.03279	0.00000	0.00000	0.00000	0.91954	0.00000
OTHER_TYPE	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
TOTAL_NUM	18211	1511	4146	339	122	238	206	3	87	0

用户做完适当操作后,相应功能按钮会被激活。



	WWW	FTP	MAIL	P2P	ATTACK	DATABASE	SERVICES	INTERACTIVE	MULTIMEDIA	OTHER_TYPE
WWW	18168	16	0	17	46	0	15	0	1	0
FTP	34	1494	1	29	2	0	0	0	5	0
MAIL	2	0	4144	118	70	0	0	2	0	0
P2P	0	1	0	121	0	0	0	0	1	0
ATTACK	0	0	0	0	0	0	0	0	0	0
DATABASE	0	0	0	0	0	238	0	0	0	0
SERVICES	0	0	0	0	0	0	191	0	0	0
INTERACTIVE	0	0	0	0	0	0	0	1	0	0
MULTIMEDIA	7	0	1	54	4	0	0	0	80	0
OTHER_TYPE	0	0	0	0	0	0	0	0	0	0
TOTAL_NUM	18211	1511	4146	339	122	238	206	3	87	0

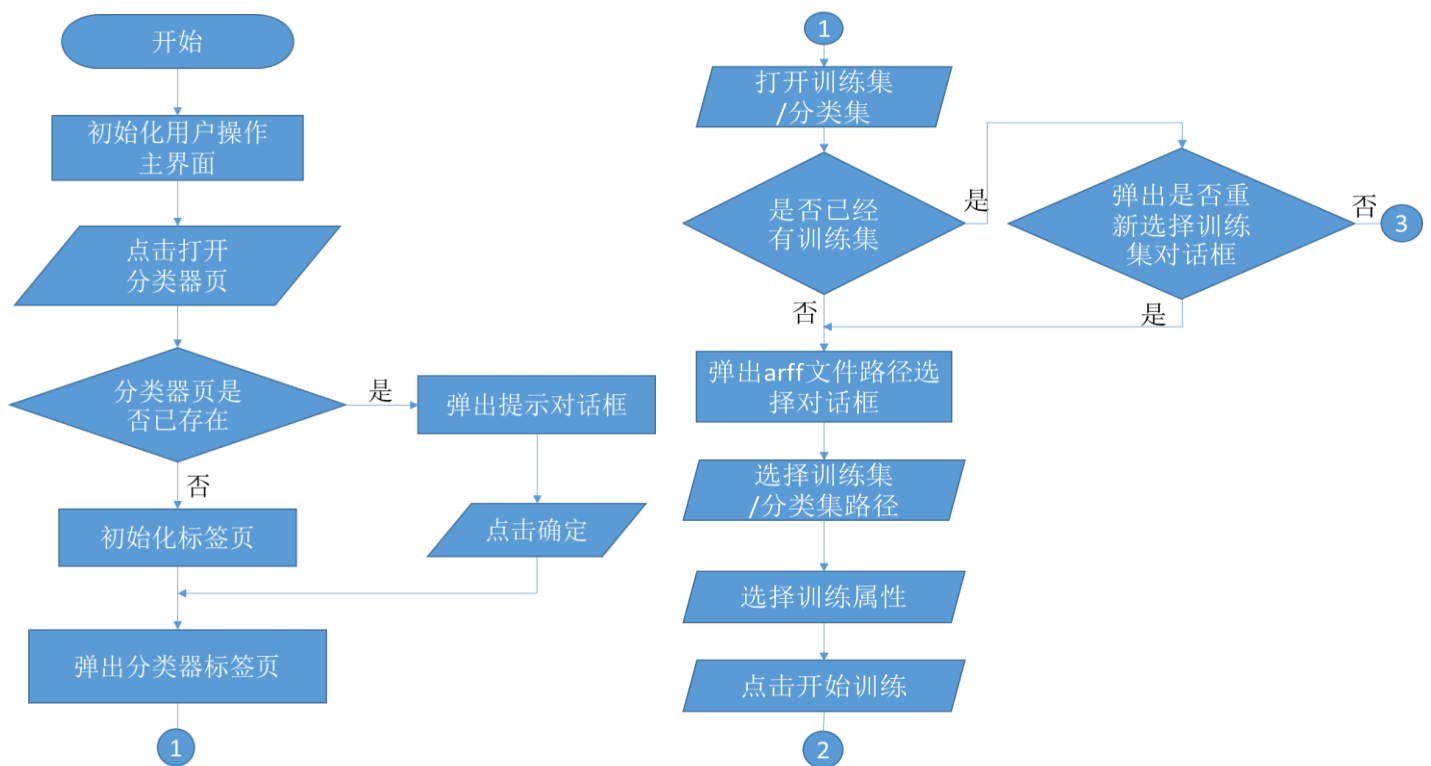
图 3.2(a) 结果展示界面

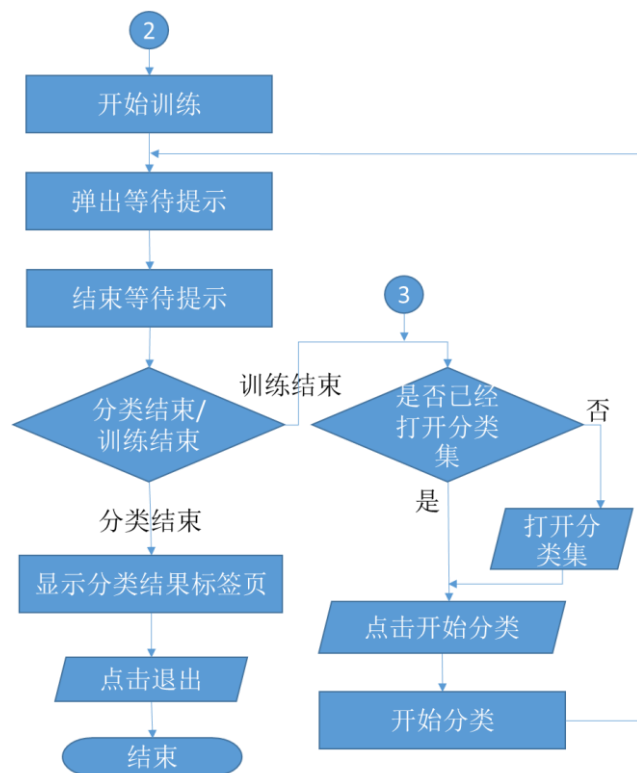
图 3.2(b) 结果展示界面

如图 3.2(a)和 3.2(b),是系统的结果展示界面(可分别通过点击结果菜单里的“分类结果(概率)”/“分类结果(数值)”或 Ctrl +

S/Ctrl + Shift + S 打开)，根据需求，将整个数据包的分类结果做成混沌矩阵的形式展示，其中横排表示每组数据的原始类型，纵排表示该组数据分类出来的类型。其中(a)是按照分类概率展示的，(b)是按数量展示的。其中分类正确部分以红色字体标记出来。最下面一排为该分类集的总数据数量。

### 3.1.1.2 系统界面操作流程





## 3.2 类结构设计

对于系统的类之间的结构，还是主要从 GUI 系统和贝叶斯算法实现部分去设计。GUI 部分主要包含的类有：用户操作主界面类（CMainWindow）、分类器操作类（CClassifyWin）、结果展示类（CResultWin）、标签页管理类（CTabWidget）、等待提示窗（CWaitDialog）；贝叶斯算法实现部分主要包括：文件操作类（CFileOperator）、贝叶斯算法类（基类：CBayesAlgorithm、派生类：CStatisticBayes、CGaussianBayes）、容器类（CContainer）、特定的字符串类（CMyString）、单例类模板（CSingleTon）等。

### 3.2.1 类结构及功能说明

#### 1、CMainWindow 类：

该类是整个系统界面的主界面，该类继承自 QT 的 QMainWindow。

在该界面下，包含了开始菜单栏、结果展示栏，并包含了一个单例的 CTabWidget 类，然后以标签页的形式包含了 CClassifyWin 和 CResultWin。

## 2、CClassifyWin 类：

该类实现的是整个系统的核心功能（继承自 QT 的 QWidget），本系统基本所有需求都在该类中进行实现。在该界面中用户可以选择打开分类集、训练集；选择分类属性；点击开始训练、开始分类等。界面效果可参照图 3.1。

该类包含了一个 CFileOperator 的成员变量，用于处理用户选择的数据包。其中的训练和分类功能主要是通过 QT 的信号、槽机制将相应的功能与 CFileOperator 类中的槽相绑定，并通过信号发送数据包（SDataPack）到 CFileOperator 的接口中，进行相应的功能调用。

## 3、CResultWin 类：

该类主要用于显示分类结果混沌矩阵，继承自 QT 的 QWidget。其调用主要产生在网络流量包分类结束后，以标签页形式自动展示分类的概率结果，也可通过用户手动点击展示分类的数值结果或概率结果。

其中，该类的结果获取自单例的 CStatisticBayes 类中的分类结果存储器中。

## 4、CTabWidget 类：

该类是整个界面系统的一个单例的标签页管理类，继承自 QT 的 QTabWidget。主要用于添加新标签和关闭某标签等，以标签页形式添加功能窗口有利于程序的进一步扩展。

### 5、CWaitDialog 类:

该类用于对数据处理过程中，对用户进行等待提示，继承自 QDialog。由于贝叶斯分类算法处理的数据包的一般都比较大，处理一个数据包需要的时间也比较长，在最初的设计里，程序是单线程执行的。在处理数据包过程中，GUI 部分就会出现假死现象。为了解决假死问题，便将数据处理过程专门放在另一个线程中处理，这时候 GUI 部分便弹出该对话框，提醒用户等待。

### 6、CFileOperator 类:

该类是文件处理类，为了使用QT的信号与槽机制，继承了QObject。该类向外部提供了一个单一的文件操作槽接口（可通过数据包里的操作设置，实现分类操作和训练操作）`slot_operateFile(SDataPack)`。由于GUI和算法位于不同的线程内，线程实现线程间通信，便主要依靠绑定该槽函数，通过数据包SDataPack进行线程间通信。

### 7、CBayesAlgorithm 类:

该类是一个贝叶斯算法的抽象类，主要要用于提取统计法的贝叶斯算法（CStatisticBayes）和正态分布的贝叶斯算法（CGaussianBayes）中的公共特性。

### 8、CStatisticBayes 类:

该类是本系统当前的主要算法类，继承自 CBayesAlgorithm 类以及 CSingleTon 类。本系统当前是采用对数据包数据按离散型实行统计法计算概率，实现分类。其中数据包中的连续数据，便采用离散化算法，进行序列化，达到算法使用上的统一。该类包含了两个一个三

维的数据结构容器(即 `CContainer<CMyString, uint>***`类型), 容器模型以及功能可参照 3.3.1 节中的表二。

#### 9、CContainer 类:

该类是本系统的主要容器类, 由于系统中的数据需要存储在一个三维的立体结构中, 而且第三维的数据量, 每一个种类都不一样, 为了实现动态的三维数据结构, 并考虑到效率上的问题以及上层代码中使用便捷性等问题。便封装了一个 CContainer 的类模板, 其中主要的数据存储功能实现是依靠本类封装的 `map<key, value>`型成员变量。

#### 10、CMyString 类:

该类是一个类似 C++ 的 `string` 类的一个特例化类, 主要针对本系统的数据进行设计。

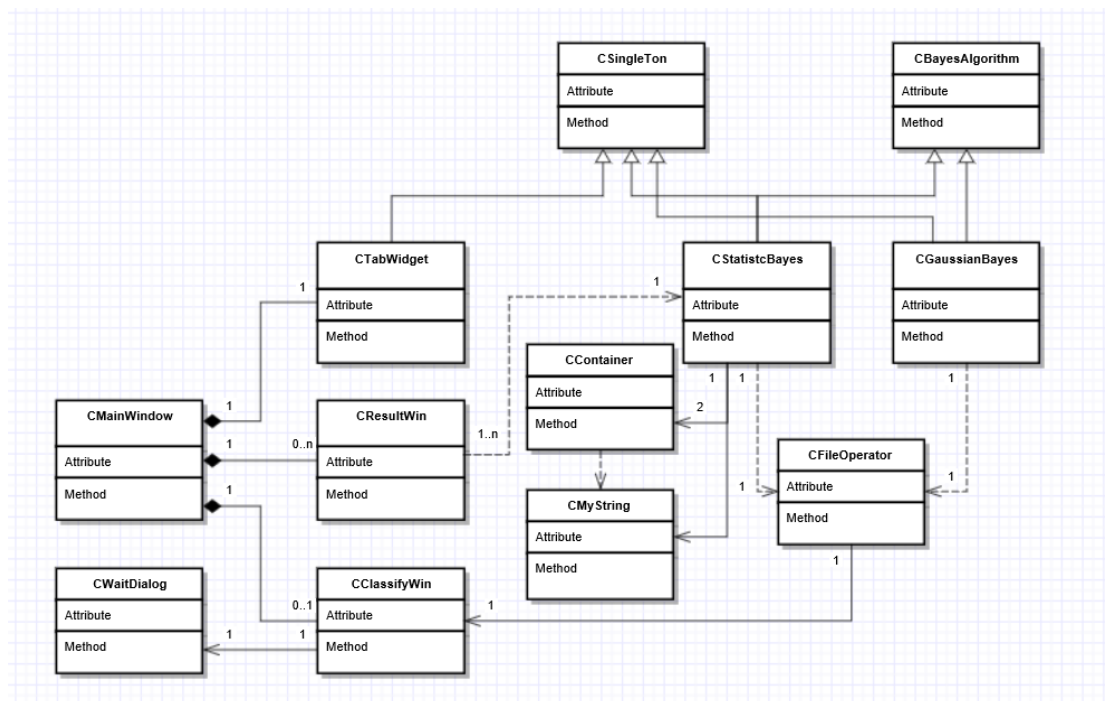
本系统统中的数据统计时, 需要向 `map` 中插值, 以及查找键值, 但是 C++ 内置的 `string` 没有重载“`<`”操作符, 以及其他一些特性不太符合本系统的需求, 因此在系统对 `string` 进行了封装, 以更加符合系统需求, 方便系统其他地方使用。

#### 11、CSingleTon 类:

该类是一个单例模式类模板, 主要用于生成系统中需要使用单例的类。



### 3.2.2 UML 类图



## 3.3 数据结构设计

### 3.3.1 贝叶斯算法存储容器表结构

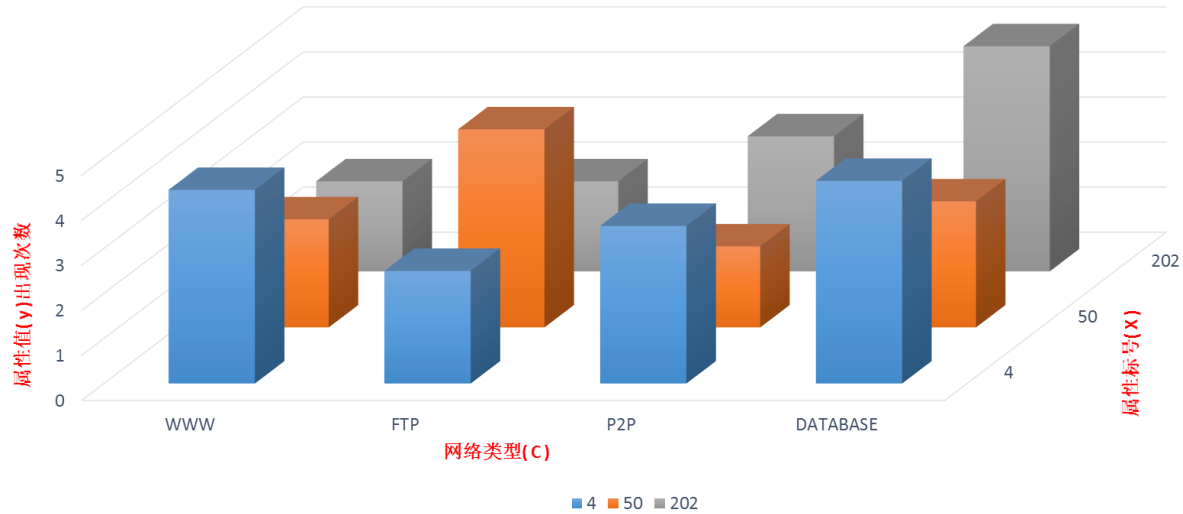
关于贝叶斯算法的实现，系统中采用离散型数据，进行数据统计方法实现的。对于其中的连续型数据，根据取值范围，进行了离散化。基于此统计法求概率，系统中需要存储的数据包括三部分，并分别设计了三个对应的数据结构去存储这些数据。

第一部分，各网络类型出现次数表（表一），用于统计各网络类型出现的次数以及所有网络类型出现的总次数。同时用另一个与此表类型相同的表格存储各网络类型出现的概率（即先验概率  $P(C_i)$ ，其中  $C_i$  表示第  $i$  种网络类型）。

流量类型	该类型出现的总数
所有类型	所有类型总数
WWW型	
SERVICES型	
FTP-PASV型	
.....	

表一 各网络类型出现次数表

第二部分，各网络类型下，每个属性里各个值出现的次数表（表二）。即为一个三维数据。同时也用一个同类型的表去存储对应位置数据的概率（即  $P(X_k = y|C_i)$ ， $C_i$  表示第  $i$  中网络类型， $X_k$  表示数据组  $X$  下的第  $k$  号属性， $y$  表示  $X_k$  的值）。



表二 网络类型及属性下各属性值统计表

第三部分，分类结果存储表（表三），横排表示分类集各数据的原始类型，纵排表示分类出来的类型。表格存储的值即表示某网络类型分到各类型的数量，最后一排的该类型总数表示。同样，用另一个同类型的表格来表示分类结果概率。

原始类型 分类结果	WWW	SERVICES	FTP	MAIL	INTERACTIVE	.....		
WWW								
SERVICES								
FTP								
MAIL								
INTERACTIVE								
.....								
该类型总数								

表三 分类结果存储表

### 3.3.2 程序数据结构定义

由于网络类型相对固定，所以在程序中将所有常出现的网络类型进行枚举(网络类型枚举名:ETrafficType)，方便后面使用。

```
enum ETrafficType    //枚举所有网络类型;
{
    ETT_WWW,           //WWW类型;
    ETT_FTP,           //FTP类型;
    ETT_MAIL,          //MAIL类型;
    ETT_P2P,           //P2P类型;
    ETT_ATTACK,        //ATTACK类型;
    ETT_DATABASE,      //DATABASE类型;
    ETT_SERVICES,      //SERVICES类型;
    ETT_INTERACTIVE,   //INTERACTIVE类型;
    ETT_MULTIMEDIA,    //MULTIMEDIA类型;
    ETT_ERR_TYPE,      //错误网络类型;
    ETT_ALL_TYPE,      //所有网络类型总数
};
```

由于读取文件过程的返回值相对复杂，为了方便程序的调试，对文件读取返回结果进行了枚举（枚举名：EFileReadResult）：

```
enum EFileReadResult    //读取操作文件返回值;
{
    EFRR_ReadFileSucceed,    //读取成功;
    EFRR_FilePathIsNull,    //文件路径为空;
    EFRR_OpenFileFailed,    //打开文件失败;
    EFRR_NoNecessaryAttri,    //分类属性值为空;
};
```

在 GUI 模块与文件处理模块交互的时候，是通过发送数据包进行数据传递的。数据包（SDataPack）的封装内容如下：

```
typedef struct _SDataPack
{
    EOperateFlag aFlag;    //数据包需要操作的类型;
    unsigned short *aSrcArray;    //勾选的属性值数组指针;
    int aSrcArraySize;    //勾选的属性值数组大小;
    char *aFilePath;    //数据包文件路径;

    _SDataPack()
    {
        aFlag = EOF_NULL;
        aSrcArray = 0;
        aSrcArraySize = 0;
        aFilePath = 0;
    }
} SDataPack;
```

其中的 EOperateFlag 是一组数据包操作类型枚举，其枚举内容为：

```
enum EOperateFlag    //标记数据包传递的数据是用于训练集的，或是用于
分类集的。
{
    EOF_Training,    //训练集标记;
    EOF_Classify,    //分类集标记;
    EOF_TrainingFinished,    //训练结束标记;
    EOF_ClassifyFinished,    //分类结束标记;
    EOF_NULL,    //空数据;
};
```

对于3.3.1中的容器表格，在程序中使用的数据结构分别为：表一对应一个一维数组（`uint m_TrafficType[ETT_ALL_TYPE + 1]`）；表二对应一个指向`CContainer<CMyString, uint>*`的二级指针，相当于一个二维数组，在数组中存放的`CContainer<CMyString, uint>`对象的指针，（即`CContainer<CMyString, uint>***getAllTypeAttri()`  
`const`），程序中用这种方式实现了表二中的可以存储三维数据的数据结构；表三对应一个二维数组（`uint m_ClassifyResultArray`  
`[TRAFFIC_TYPE_NUM][TRAFFIC_TYPE_NUM]`）。其中`TRAFFIC_TYPE_NUM`即为一个常量，值为所有网络类型的总数加一（`ETT_ALL_TYPE + 1`）。如上面所讲述的，就是本系统中所用到的所有数据结构类型。

## 第四章 关键技术

### 4.1 关键技术的应用

本系统所用到的算法，主要是朴素贝叶斯算法中的离散数据处理部分；

系统的实现上，则主要是基于 Qt 框架实现的。Qt 框架中用到的主要技术有信号和槽的机制、多线程机制以及自定义对话框的实现等；

编程语言使用的 C++，其中用到的主要 C++ 技术有：类模板、函数模板、STL、设计模式中的单例模式等；

#### 4.1.1 算法选择和实现说明

关于正态分布法和离散数据统计法，两种分类贝叶斯算法建模方式，虽然是分别用于处理离散数据和连续数据的，但是为了在系统的实现上更加简单，统一。本系统则主要在两种算法中选择其一进行实现。由于考虑到离散型数据以及 Y/N 型数据在使用正态分布法处理时算法本身的适用性不如离散统计法。而且对于连续型数据，也同样可以离散化，从而使其是用于统计法。所以本系统采用了统计的方式进行贝叶斯算法建模。

但是统计法计算数值出现概率的情况，为了得到比较精确的结果，需要的建模数据量比较大。如果建模数据量不够的话，对于分类的结果准确性上，便不是那么稳定。对此，本系统实现过程中为了对建模数据进行优化，提高数据包分类结果的稳定性，当对于某种属性 ( $X_k$ ) 的某一值在建模数据中没有出现时，便不是按照常规的情况返回一个为零的概率。这里采用返回一个合适的极小值，这个值得意义等价于

该属性的这个值出现的概率很小，虽然当前建模数据没统计到。由于该数据极小，也不会干扰到正常的分类情况。

## 4.2 关键技术的一般说明

### 1、信号和槽机制：

Qt 的信号和槽的机制类似于 Windows 下的消息机制，是通过事先绑定两个对象的信号与槽，当信号被触发时，系统就会主动调用之前绑定的对象的槽函数。

信号：当对象的状态改变时，信号就由该对象发射（emit）出去，而且对象只负责发送这个信号，而不管另一端是由谁去接收这个信号。这样就做到了真正的信息封装，能确保对象被当作一个真正的软件组件来使用。

槽：用于接收到相应对象的信号后，触发调用相应函数的机制（槽函数相当于一个回调函数）。槽必须使用相应的槽函数，其中槽函数与普通的成员函数基本一致，也可以当作普通的成员函数调用。槽函数的声明需要在访问类型符后面加 slots（即 public slots:）。

信号和槽的绑定：所有继承自 QObject 的子类都可以通过信号和槽的绑定函数 connect（）进行绑定，使用方式即 connect(sender, SIGNAL(signal()), receiver, SLOT(slot()))。

### 2、多线程实现：

Qt 中的多线程实现方式有多种。第一种是继承 Qt 的 QThread，并重写它的 run() 函数。run() 函数是一个纯虚函数，是线程执行的入口，在 run() 中的代码将在新开辟的这个线程中执行。

另外，也可以通过继承自QObject的子类的moveToThread()函数，将该子类移到创建好的线程中执行。然后通过该子类的槽函数，实现在新线程中运行所需代码。本系统中便是采用的这种方式。其基本使用方法如下：

```
m_WorkThread = new QThread;

m_FileOperator = new CFileoperator; //继承自QObject;

m_FileOperator->moveToThread(m_WorkThread);

connect(this, SIGNAL(signal_SendData(SDataPack)),

         m_FileOperator, SLOT(slot_operateFile(SDataPack)));
```

上面代码中的槽函数slot\_operateFile()便是在m\_WorkThread线程中执行的。同样的方式通过绑定信号和槽，还可以实现更多的主线程和子线程间的通信。

### 3、自定义对话框：

在本系统中，主要是等待提示对话框使用了自定义方式处理。Qt中提供了一些基本的对话框供我们使用，如 QMessageBox 一类的，可以满足我们的基本需求。但是如果有自定义需要时，我们可以通过继承 Qt 的对话框类(QDialog)来实现对话框自定义。

### 4、类、函数模板：

在 C++中，模板是为了实现对一批仅仅数据类型不同的代码进行抽象的一个概念，模板的引入使得 C++的泛型编程得以实现，从而大大的提高了编程的效率。

类模板/函数模板的定义和使用：类模板和函数模板的定义都需要



在前面加 `template<class T>` 类型的关键字，其中的 `class` 也可以换成 `typename`。使用时，模板函数跟普通函数的调用方式一样，而模板类则需要在类名后面加上相应的类型说明，如 `map<int, int> MAP`。

需要注意的是，模板的成员函数的定义和声明都只能在头文件中进行，如果定义在 `cpp` 文件中，编译会出现无法解析的外部符号问题。

## 5、STL：

标准模板库是一组非常方便的 C++ 库，里面包含了各种好用的容器。包括 `list`、`queue`、`deque`、`set`、`vector`、`map` 等。只需要在程序中对相应的头文件进行包含，便可以对它们进行方便快捷的引用。

在本系统中，对于所需存储的数据结构，选择了 `map` 用于数据的存储，为了方便对数据的统计操作，对 `map` 进行了封装，以便每找到一个同类型的键值，便对该键值后面的值进行加一，实现数据统计。

## 6、单例模式：

单例模式是一种常用的软件设计模式，在它的核心结构中只包含一个被称为单例的特殊类。通过单例模式，可以保证整个系统中一个类只有一个实例，并方便外界访问，从而方便对实例个数的控制并借阅系统资源。

在本系统中，对于贝叶斯算法类（`CStatisticBayes`）和标签页控制类（`CTabWidget`）需要全局唯一，并方便系统不同部位调用，因而都继承了单例类（`CSingleton`）。

## 第五章 方案实施的技术路线和实施计划

### 5.1 实施的技术路线

朴素贝叶斯算法中对网络流量分类的总体思路是，首先通过一组拥有不同属性（本文采用  $X_k$  表示不同属性）的数据对贝叶斯算法进行建模。分类时，首先假设该组分类数据的类型（本文中采用  $C_i$  标识不同数据类型），再用建模好的数据求需要分类的数据中的各个属性的值的概率  $P(X_k = y \mid C_i)$ 。并将各概率值相乘最后乘以所假设类型的先验概率（ $P(C_i)$ ）得到该数据包的  $P(C_i) * P(X \mid C_i)$  的值

$(P(X \mid C_i) = \prod_{k=1}^n P(X_k \mid C_i) = P(X_1 \mid C_i) \times P(X_2 \mid C_i) \times \dots \times P(X_n \mid C_i))$ ，又因

为分类需要的值便是求  $X$  的最大后验概率（即

$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$ ）且  $P(X)$  对于所有的类型的概率是相同的，

所以，只需要  $P(C_i) * P(X \mid C_i)$  最大即可。然后依次假设完所有数据类型得到  $P(C_i) * P(X \mid C_i)$ ，比较其中最大的概率值，该概率值所对应的假设类型  $C_i$  便是该组数据的分类结果类型。

而对于对贝叶斯算法建模的过程，根据每组数据包里面的值不同，又分为连续型数据和离散型数据。连续型数据的建模方式是求该属性的各值在某类型下的均值和方差，分类时便是通过正态分布的概率密

度公式  $g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$  求得该类型下属性的值  $x$  所出现的概

率。而离散型数据便是对各类型下的不同属性的值出现的概率进行统计，在分类时，便是去查找该值出现的概率。所以两种方式在程序实现上有很大的差别。在本系统中，为了方便对所有数据进行处理，统

一采用离散数据法进行分类。对于其中的连续型数据，便采用分段的方式，进行离散化。

对于离散型数据的贝叶斯算法分类具体过程，可用下面的实例进行分析。

例：

两个类别：

C1:buys\_computer = 'yes'

C2:buys\_computer = 'no'

求数据样本是否买电脑：

X = (age ≤ 30,

Income = medium,

Student = yes

Credit\_rating = Fair)

age	income	student	credit_rating	buys_computer
≤30	high	no	fair	no
≤30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
≤30	medium	no	fair	no
≤30	low	yes	fair	yes
>40	medium	yes	fair	yes
≤30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

算法分析计算：

►  $P(C_i): P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$

$$P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$$

► Compute  $P(X|C_i)$  for each class

$$P(\text{age} = \text{"≤30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$$

$$P(\text{age} = \text{"≤ 30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

$$P(\text{student} = \text{"yes"} \mid \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{student} = \text{"yes"} \mid \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$$

$$P(\text{credit\_rating} = \text{"fair"} \mid \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{credit\_rating} = \text{"fair"} \mid \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$

$$P(X|C_i) : P(X|\text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$

$$P(X|\text{buys\_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|C_i) * P(C_i) : P(X|\text{buys\_computer} = \text{"yes"}) * P(\text{buys\_computer} = \text{"yes"}) = 0.028$$

$$P(X|\text{buys\_computer} = \text{"no"}) * P(\text{buys\_computer} = \text{"no"}) = 0.007$$

Therefore,  $X$  belongs to class ("buys\_computer = yes")

由上例可以明显看出，对于离散型数据，我们的算法实现中需要做的就是统计出  $P(X \mid C_i)$ 。而这个统计过程，便需要结合 3.3 节中的几个存储结构进行。其具体实现过程，请参照第三章。

## 5.2 实施计划

根据前面章节的系统分析设计，实施计划同样主要从两方面入手。

第一、GUI 系统方面。首先在 Qt 框架下，封装好 CTabWidget，写好界面的各个部件（包括 CWaitWin、CClassifyWin 等），方便后面使用。对于 GUI 部分，由于刚接触 Qt，需要一边学习，一边尝试，预计一周时间完成界面部分的基本架构，调用算法的接口等；

第二、算法实现部分。根据前面的分析，统计法的贝叶斯算法实现关键点在于容器的设计，以及所需数据的提取，关于算法本身的实现其实并不难。所以，根据需求，先封装好 CContainer 容器类，以及需要用到的 CMyString 类，还有实现文件读取，操作部分。这一部分，实现起来应该比较快，加上 GUI 系统和算法实现部分的连接调用以及基本测试、调试，预计一周时间。

## 缩写词表

下表将对本系统中常用到的缩写词进行简单阐述。

缩写词	原单词	备注
Attri	Attribute	网络包的各属性
Idx	Index	一般为循环控制变量
Itor	Iterator	一般为迭代器对象的对象名
Bayes	Bayesian	贝叶斯
Calc	Calculate	常用于计算数据的函数名中
Def	Define	定义
Pos	Position	位置
DEFSETGET	DefineSetGet	程序中的一个宏定义，主要用于声明类成员变量，和该变量的对外接口。
Str	String	一般用于表示字符串变量
Dest	Destination	目标位置
DataPack	Data package	数据包

## 参考文献

- [1] Jasmin Blanchette[加拿大] Mark Summerfield[英]. C++ GUI Qt4 编程. 第二版 . 闫锋欣 曾泉人 张志强 译;
- [2] Andrew W.Moore . Internet Traffic Classification Using Bayesian Analysis Techniques. University of Cambridge;