软件安全

实验五　恶意行为异常检测方法

学号：1170300728 姓名：汤添凝

一、实验项目描述

1、理解基于异常检测的恶意攻击行为检测方法

（1） 掌握异常检测的流程

（2） 学习相关的异常检测算法

2、基于距离的异常检测方法

（1） 掌握欧氏距离的概念

（2） 如何选取数据集的属性集

（3） 选取合适的检测模型

3、基于 KD 树的网络流量异常检测模型

（1） 利用 KD 树构建一个用于多维空间最邻近搜索的数据结构

（2） 建立历史数据集合

（3） 规格化数据

（4） 采用标准分割策略，进行基于维度分割的 KD 树构建

（5） 基于待检测数据 X，利用构建好的 KD 树搜索，找出历史数据集合中与待检测数据 X 最近的数据；计算二者之间的欧氏距离，与阈值比对，确定是否是异常数据点。

二、实验要求

1、实验数据准备。利用 KDD1999 数据集(KD 树异常检测.pdf p31 页-34 页)提供的数据进行实验。选取部分正常数据做为训练集，选择部分攻击数据和剩下的正常数据做为测试集。

2、可以只选择流量属性集(KD 树异常检测.pdf p34 页)。可只针对 DOS(smurf 攻击即可)攻击进行异常检测，其他攻击不考虑。

3、2 人一组完成实验。

4、下载阅读 CMS“相关资料”中的 KD 树异常检测.pdf 文件

5、利用 CMS“相关资料”中的 kddcup.data_10_percent.gz 作为数据（看数据说明）。数据中每条都有标记为：NORMAL 或 ATTACK 类型。利用标记为 NORMAL 的数据建模（构建 KD 树）。利用一部分标记为 NORMAL 的数据

和 ATTACK 数据作为测试数据。 （数据集中可能存在少量有错误的数据，注意）

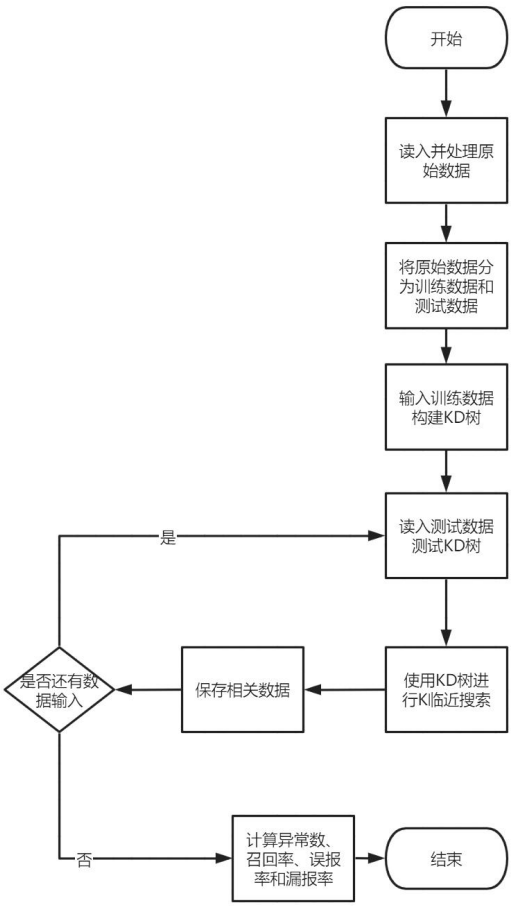
三、实验结果（将来需体现在实验报告中）

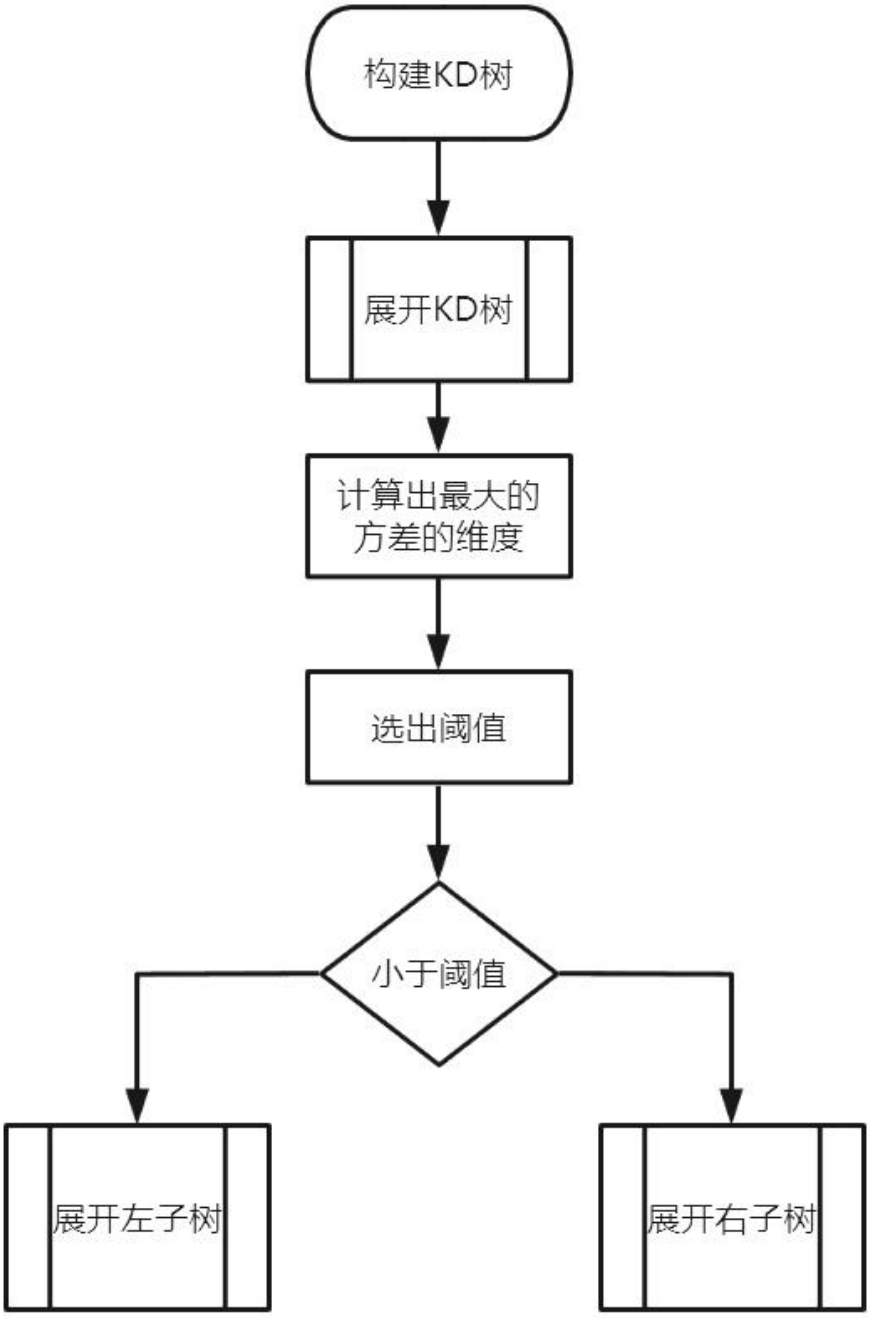
1、程序的流程图

2、列出构建 KD 树所定义的数据结构，简单说明其功能

3、实验结果和实验数据一起给出。如选择多少攻击数据做为测试集？检测出多少攻击？误报率和漏报率是多少？

一、 流程图




构建KD树
展开KD树
计算出最大的
方差的维度
选出阈值
小于阈值
展开左子树
展开右子树

二、数据结构

KDNode：KD 树的节点。其中包括若干属性 value, label, dim, parent, left_child, right_child。每个属性的意义如下：

```
初始化KDNode的属性
:param value: 节点的值
:param label: 标签：0表示smurf，1代表normal
:param dim: 数据的维度
:param parent: 父节点
:param left_child: 左子节点
:param right_child: 右子节点
```

KDTree：KD 树结构。包括树的节点数目 length，树的根节点 root。

三、实验结果与数据

训练数据

```
0, 0, 6, 6, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 3, 3, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 6, 9, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 3, 3, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 10, 14, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 7, 7, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 5, 5, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 2, 2, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 5, 5, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 2, 2, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 8, 23, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 5, 5, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
0, 0, 3, 11, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00,
```

测试数据

```
0, 0, 314, 314, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
0, 0, 511, 511, 0. 00, 0. 00, 0. 00, 0. 00, 1. 00, 0. 00, smurf
```

实验结果：

运行完成!

异常数|399个

召回率|99.75%

误报率|3.00%

漏报率|0.25%
