

华中科技大学

2018

数据挖掘

课程设计报告

题目：病毒防入侵数据挖掘

专业：计算机科学与技术

班级：计卓 1501

学号：U201511086

姓名：张铭

指导教师：莫益军

华中科技大学课程设计报告

目 录

1	摘要	2
2	数据介绍	3
2.1	数据来源	3
2.2	数据特点	3
2.3	数据结构	3
2.4	挖掘目标	3
3	数据提取	4
3.1	混合模型	4
3.2	二进制文件 N-GRAM.....	4
3.3	特征去噪	4
3.4	反汇编文件 N-GRAM.....	5
3.5	动态链接库 N-GRAM.....	5
4	数据挖掘	6
4.1	总体设计	6
4.2	数据处理	7
4.3	算法原理	8
4.4	训练及测试	11
5	运行结果	14
6	总结与心得.....	16
6.1	课设总结	16
6.2	课设心得	16
	参考文献.....	17

1 摘要

病毒检测一直是计算机安全领域的焦点话题，在这方面的杀毒软件也是层出不穷，例如国外的 Avira 以及国内的 360 杀毒等软件，可以使用特征代码法、校验和法、行为检测法、软件模拟法等技术查杀病毒。在如今的大数据时代，可以使用数据挖掘的方法，利用机器学习的算法工具，以众多病毒软件的特征代码为基础，训练出病毒软件识别器，在一定程度上为病毒软件的识别提供更简便的方法。

2 数据介绍

2.1 数据来源

本数据来源于 UCI Machine Learning Repository，数据集名为 Detect Malicious Executable(AntiVirus) Data Set，数据集 URL^[1]见参考文献。其中病毒程序的数据来源于 VX-Heavens: <http://vx.netlux.org/>。

2.2 数据特点

- 1) 此数据集为多变量数据，属于二分类问题；
- 2) 共有 373 个数据，包括 100+非病毒程序数据，250+病毒程序数据，总特征数为 513 个；
- 3) 对每个不同的数据提取了不同数量的特征，但每个数据的特征总数不超过 513 个。

2.3 数据结构

训练/测试数据的数据结构描述如下：

+1 1:1 3:1 5:1 7:1 9:1 11:1 13:1 15:1 19:1 21:1 23:1 25:1 27:1 29:1 31:1 33:1 36:1 38:1
40:1 42:1 44:1 ... 505:1 -1

开头的+1 表示该程序为非病毒程序，若为-1 则表示该程序为病毒程序。后续 1:1, 3:1, 5:1, ..., m:1 表示该数据拥有第 1, 3, 5..., m 个特征，最后的-1 表示数据的终结符。

2.4 挖掘目标

确定需要分类的数据，划分出训练集以及测试集，通过训练数据集训练得出分类器，通过此分类器正确识别病毒程序。

3 数据提取

3.1 混合模型

通过分析可执行程序以及反汇编代码和使用的库函数，可以判断该程序是否为病毒程序，详细叙述如下：

- 1) 二进制特征；
- 2) 反汇编代码特征；
- 3) 动态链接库调用特征。

通过分析上述三种混合的特征代码，得到描述该程序的总体特征的混合向量模型。

3.2 二进制文件 N-gram

N-gram 一般用于文本数据挖掘，但此处二进制代码也是字符流，故可以使用 N-gram 方式从二进制文件中得出特征代码，举例如下：

若有一段 6 字节代码为：a1b2c3d4e5f6，通过 4-gram 方式提取出的特征代码为“a1b2c3d4”，“b2c3d4e5”，“c3d4e5f6”。

N-gram 方法的具体实施步骤：

- 1) 通过一个 N 字节的滑动窗口扫描每个二进制文件；
- 2) 如果得到了一个 N 字节的代码串，则将其加入到结果列表中，否则忽略。

问题在于当文件很大时所得到的 N-gram 代码串数量相当多，如何存放以及如何检查被重复扫描的代码串，若存放在内存中则内存空间有限，若对每一个新加进的代码串扫描一遍表中是否已存在，对于 N 和代码串，每个代码串都需要扫描一趟，时间复杂度为 $O(N^2)$ ，如此时间空间的开销问题亟待解决。解决方法是采用磁盘 I/O 解决内存存放空间不足的问题，在磁盘有序存放 N-grams，但会降低存取效率；在内存中使用 AVL 树结构取代列表结构，这样插入一个新 N-gram 的时间复杂度为 $O(N\log(N))$ ，降低了时间复杂度，同时增大了代码编写难度。

3.3 特征去噪

提取的特征集过于庞大，其中含有很多噪声，不仅存放空间不够，而且训练时间

华中科技大学课程设计报告

会大大加长，并且特征值过多，也不利于训练，故使用信息增益的方式去除噪声。

对于样本 S ，特征 A 的信息增益公式计算如下：

$$Gain(S, A) \equiv Entropy(S) - \sum_{V \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

其中 $Values(A)$ 表示特征 A 可能的取值， S_v 是一个二值数，若特征出现在该样本，则对于该特征取值 S_v 为 1，否则为 0，熵 ($Entropy$) 的计算公式如下：

$$Entropy(S) = -\frac{p(s)}{n(s)+p(s)} \log_2\left(\frac{p(s)}{n(s)+p(s)}\right) - \frac{n(s)}{n(s)+p(s)} \log_2\left(\frac{n(s)}{n(s)+p(s)}\right)$$

$p(S)$ 是 S 中的正样本数量， $n(S)$ 是 S 中的负样本数量。

通过计算每一个特征的信息增益，可选择出具有代表性的特征，在本数据集中共选择出了 513 个特征，大大减少了特征数，去除了无用的特征。

3.4 反汇编文件 N-gram

提取反汇编文件的 N-gram 的方式同 3.2 节，同样提取所有可能的 N-grams，此处提取的不是 N 字节的二进制代码，而是 N 个汇编指令，反汇编代码的 N-gram 提取方式举例如下。

对于三条汇编指令 “push ebp”，“mov eax, ebx”，“add ebx, 1”，使用 2-gram 方式提出的指令为：

“push ebp”，“mov eax, ebx”

“mov eax, ebx”，“add ebx, 1”

再使用 3.3 节中信息增益的方式去除无用的特征，只保留 500 多个最好的特征。

3.5 动态链接库 N-gram

通过解析反汇编文件找到有关于动态链接库的调用信息，如在汇编文件中按顺序出现了如下函数调用信息：

“call functionA”，“call functionB”，“call functionC”

使用 2-gram 方式提取出的特征为

“call functionA”，“call functionB”

“call functionB”，“call functionC”

提取之后通过 3.3 节中计算信息增益的方式选出最好的 500 多个特征即可。

4 数据挖掘

4.1 总体设计

采用 KNN, SVM, NaiveBayes 三种数据挖掘算法对数据集进行分析。总体上可以分为五个模块，分别是数据读取，数据清洗，数据预处理，训练，测试，各模块之间的关系如下图 3.1 所示。

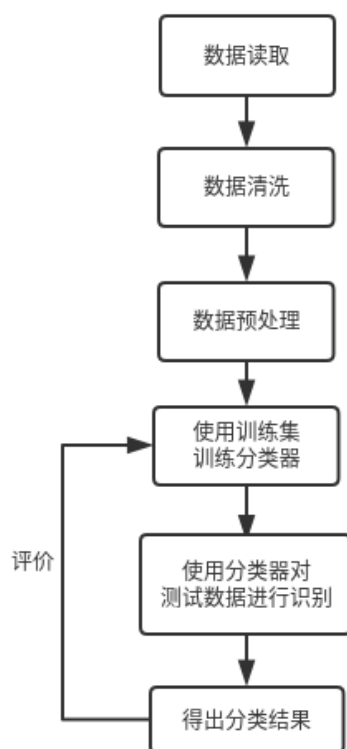


图 4.1 总体模块图

- 1) 使用 Python 语言完成整个部分；
- 2) 根据数据集特有的数据结构读取类别和所有特征；
- 3) 完成数据清洗，去除空数据；
- 4) 对数据做预处理，使其能达到分类器可处理的数据形式；
- 5) 使用训练数据训练，得出分类器；
- 6) 使用分类器对测试集进行分类，预测出测试集中程序是否为病毒程序，将预测结果与实际结果相对比，得到预测结果和评价结果。

4.2 数据处理

4.2.1 数据读取

由于数据结构较为特殊，每一个特征值都使用了 `m:1` 的形式表示该程序具有第 `m` 个属性。首先需要读取训练数据的标签，为 `+1` 或者 `-1`，代表该程序是否为病毒数据，其次需要去除每个属性值后面的 `”:1”` 记号，将每个特征值存入特征向量中，将该程序的类别存入类别向量。如某两个训练数据为：

`-1 1:1 3:1 5:1 7:1 9:1 11:1 15:1`

`+1 2:1 4:1 6:1 8:1 10:1 12:1 14:1`

表示该程序为病毒程序，则得到的特征向量为：

`[[1, 3, 5, 7, 9, 11, 15], [2, 4, 6, 8, 10, 12, 14]]`，前 7 个数字表示该程序有的特征，得到的类别向量为 `[-1, +1]`，如此即可获得包含所有病毒/非病毒程序的特征向量矩阵，以及对应的类别向量。

核心代码如下所示：

```
for exe in datalist:
    exe_attr = []
    for i in range(1, len(exe)):
        if exe[i] == '-1':
            break
        else:
            exe_attr.append(int(exe[i].split(':', 1)[0]))
```

`exe_attr` 即为每个程序的特征向量

4.2.2 数据清洗

在做特征提取时，发现程序总是数组越界出错，仔细检查发现数据集中有样本数据没有特征，数据中只有 `-1 -1`，表示该数据为病毒数据，随后便是终结符 `-1`。对该类数据清洗方式很简单，只需要判断 `exe_attr` 是否为空即可，若该向量为空，则不将 `exe_attr` 加入到特征向量矩阵中，代码片段如下所示。

```
# 数据清洗，去除无属性值的 exe 信息
if len(exe_attr)!=0:
```



```
type_list.append(int(exe[0])) # add +1 or -1 to type list  
sample_attr.append(exe_attr)
```

4.2.3 数据预处理

SVM 和 KNN 以及 NaiveBayes 算法对于数据的维度要求一致。但经过数据清洗过程后的得到的特征向量维度不同，所以无法直接应用到算法中，必须使得每个特征向量维度都相同才可以。经过思考之后，借用操作系统中描述内存占用情况的“位图法”概念可以完成此标准化工作。思路如下：

- 1) 通过特征矩阵向量计算出维度最大的特征向量，最大维度为 531；
- 2) 为每个程序构造 531 维的向量，以特征值为索引，为出现过的特征值标 1，未出现过的特征值标 0，即可得到“位图”。

位图举例如下：

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

表示程序 1 具有第 1、3、4 个特征，程序 2 具有第 1 个特征，以此类推，根据数据集中共有 372 条数据可以得到 372*531 的位图矩阵，这样也可以使用位图矩阵标识每一个病毒/非病毒程序，未丢失程序的特征值信息。位图矩阵结合数据读取步骤中得到的类别向量，即可得到经过预处理后的训练数据。

4.3 算法原理

4.3.1 KNN 算法

KNN，即为 k-NearestNeighbor，中文译为 K 近邻算法，可用于分类。KNN 算法的核心思想是如果一个样本在特征空间中的 k 个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。该方法在确定分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。由于经过预处理之后的数据为向量数据，故可以使用欧式距离公式计算样本点 x、y 之间的距离 d(x, y)：

$$d(x, y) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$$

实施 KNN 算法步骤：

- 1) 计算测试数据与各个训练数据之间的距离并按照距离的升序排序；
- 2) 选取距离最小的 K 个点并确定前 K 个点所在类别的出现频率；
- 3) 返回前 K 个点中出现频率最高的类别作为测试数据的预测分类。

例如对如下图 3.2 数据采用 3NN 进行计算，则绿色圆被分为红色三角形一类，若按照 5NN 进行计算，则绿色圆被分为蓝色方块一类。

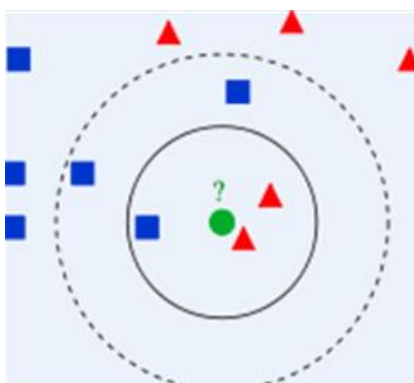


图 4.2 KNN 举例

4.3.2 SVM 算法

SVM，即为 Support Vector Machine，中文译为支持向量机。SVM 算法多被用于分类，对于线性可分或线性不可分的数据都可以起到不错的分类效果，此处病毒/非病毒程序的位图特征向量矩阵是线性不可分的数据，并且 SVM 应对较小数据量能起到不错的分类效果，均满足 SVM 算法的适用条件。

SVM 具有坚实的数学基础，此处简要介绍其数学原理。对于下图 3.3 中实线标识的超平面，记点 A 在超平面投影点为 B，A、B 距离 $\gamma^{(i)}$ ，B 点在超平面上，故满足方程 $w^T x + b = 0$ ，因此

$$w^T \left(x^{(i)} - \gamma^{(i)} \frac{w}{\|w\|} \right) + b = 0.$$

解出间距 $\gamma^{(i)}$

$$\gamma^{(i)} = \frac{w^T x^{(i)} + b}{\|w\|} = \left(\frac{w}{\|w\|} \right)^T x^{(i)} + \frac{b}{\|w\|}.$$

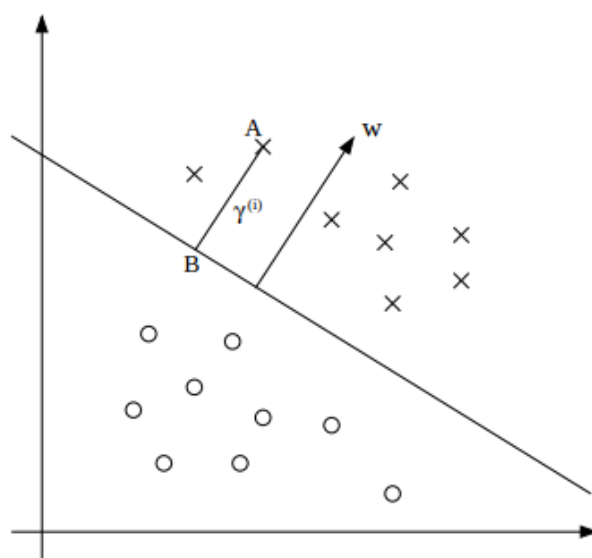


图 4.3 超平面分割两类数据

更进一步，SVM 算法要求超平面距离两类数据的支持向量的样本点间距最大，如下图所示 3.4 所示，中间实线标明的即为所求的超平面。

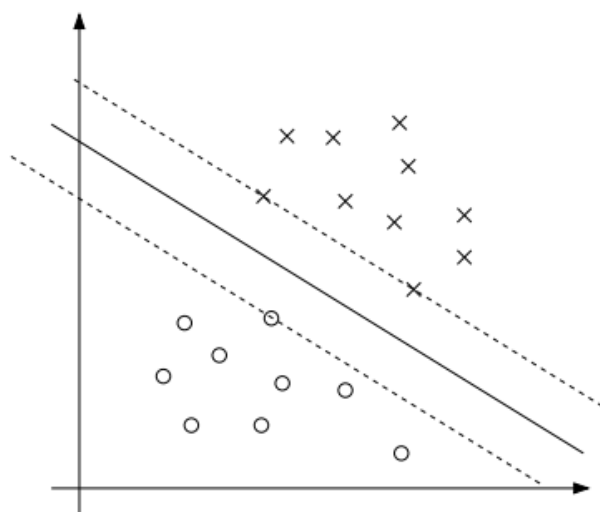


图 4.4 SVM 超平面

对于该超平面 $w^T x + b = 0$ ，该最大间距的求解是 SVM 分类器至关重要的一点。省去繁杂的数学推导，最终需要满足如下条件：

$$\begin{aligned} \min_{\gamma, w, b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \dots, m \end{aligned}$$

根据训练集的样本特征向量和类别，使用 SVM 对进行分类，再使用得到的分类器对测试数据进行分类，判断未知程序是否为病毒程序。

4.3.3 NaiveBayes 算法

NaiveBayes 分类算法，中文译为朴素贝叶斯分类算法，该算法建立在贝叶斯定理基础上，具有概率统计基础。

首先介绍贝叶斯定理：事件 A 发生的概率为 $p(A)$ ，事件 B 发生的概率为 $P(B)$ ，事件 A 发生的前提下，事件 B 发生的概率为 $p(B|A)$ ，事件 B 发生的前提下，事件 A 发生的概率为 $p(A|B)$ ，事件 A 和事件 B 同时发生的概率是 $p(AB)$ ，则有

$$p(AB) = p(A)p(B|A) = p(B)p(A|B)$$

根据上式可以推出贝叶斯定理为

$$p(B|A) = \frac{p(B)p(A|B)}{p(A)}$$

假设 B_i 与 B_j 相互独立，根据全概率公式，对于事件 A，有

$$p(A) = \sum_{i=1}^n p(B_i)p(A|B_i)$$

可以得到广义贝叶斯公式：

$$p(B_i|A) = \frac{p(B_i)p(A|B_i)}{\sum_{i=1}^n p(B_i)p(A|B_i)}$$

其中 $p(B)$ 为先验概率，在已知 $p(B)$ 和 $p(A|B)$ 的情况下即可预测出 $p(B|A)$ ，虽然原理简单，但是朴素贝叶斯算法在分类准确率上表现良好。

4.4 训练及测试

4.4.1 划分训练集与测试集

由于 UCI 原数据集中给定的测试集中只有一个测试数据，不足以反应分类器的分类准确率，所以需要在训练集中划分出测试集。根据 2-8 原则，应将训练集中的 20% 数据用于测试，即在病毒程序和非病毒程序中分别随即抽取 20% 的数据用于测试。

4.4.2 训练

使用 Python 的 `sklearn` 库函数进行训练，将预处理后的位图矩阵作为 `X`，包含+1和-1 的类别向量作为 `y` 输入到 `KNN`、`SVM`、`NaiveBayes` 函数中训练，即可得到训练后的分类器。下述各训练函数以及训练参数。

➤ KNN

训练参数：`K=3`，即找到最近的 3 近邻作为分类基础，训练函数编写如下：

```
KNNclassifier = knnclf(n_neighbors=3)
```

```
KNNclassifier.fit(x, y)
```

得到的 `KNNclassifier` 即为训练完毕的 `KNN` 分类器。

➤ SVM

训练参数：`gamma=0.1`，`kernel=poly`。由于数据量小，为保证支持向量样本数，设置 `gamma` 值较小；由于此输入矩阵线性不可分，故使用多项式函数 `poly` 作为核函数，训练函数编写如下：

```
SVMclassifier = svm.SVC(kernel='poly', gamma=0.1)
```

```
SVMclassifier.fit(x, y)
```

得到的 `SVMclassifier` 即为训练完毕的 `SVM` 分类器。

➤ NaiveBayes

使用高斯贝叶斯对数据做训练，训练函数编写如下：

```
Bayesclassifier = GaussianNB()
```

```
Bayesclassifier.fit(x, y)
```

得到的 `Bayesclassifier` 即为训练完毕的 `NaiveBayes` 分类器。

4.4.3 测试

通过已得到的分类器对测试数据做分类，并将分类结果与真实结果相比对得到分类准确率，使用 `sklearn` 库中每个分类器的 `predict` 方法即可实现分类。

测试代码如下所示，`y` 为测试数据中的标签，可以得出分类正确率。

```
rightNum = 0
for i in range(0, len(y)):
    each_test = x[i]
```

华中科技大学课程设计报告

```
tmp = [] # as a list of list
tmp.append(each_test)
res=KNNclassifier.predict(np.array(tmp))
if y[i] == res[0]:
    rightNum += 1
print("正确率: "+str(rightNum/len(y)))
```

5 运行结果

➤ KNN

使用 KNN 算法对测试数据进行分类, $K=5$, 分类准确率为 0.9467, 分类结果如下图 5.1 所示。

```
ming@x1c > ~/repo/MaliciousExeDetect/KNN > master ● > ./knn_train.py
k = 5
正确率: 0.9466666666666667
ming@x1c > ~/repo/MaliciousExeDetect/KNN > master ● > |
```

图 5.1 KNN 分类结果 ($k=5$)

若采用 $K=3$, 分类准确率为 0.9467, 因为数据量较小, 与 $k=5$ 时表现的相同, 分类结果如下图 5.2 所示。

```
ming@x1c > ~/repo/MaliciousExeDetect/KNN > master ● > ./knn_train.py
k = 3
正确率: 0.9466666666666667
ming@x1c > ~/repo/MaliciousExeDetect/KNN > master ● > |
```

图 5.2 KNN 分类结果 ($k=3$)

➤ SVM

使用 SVM 算法对测试数据进行分类, 参数 $\gamma=0.1$, 使用核函数 poly 进行分类, 准确率为 0.9733, 分类结果如下图 5.3 所示

```
ming@x1c > ~/repo/MaliciousExeDetect/SVM > master ● > ./svm_train.py
kernel:poly
gamma: 0.1
正确率: 0.9733333333333334
ming@x1c > ~/repo/MaliciousExeDetect/SVM > master ● > |
```

图 5.3 SVM 分类结果 (kernel=poly, $\gamma=0.1$)

若采用 sigmoid 函数作为核函数, γ 为 0.1 进行分类, 得到准确率仅为 0.8133, 分类结果如下图 5.4 所示。

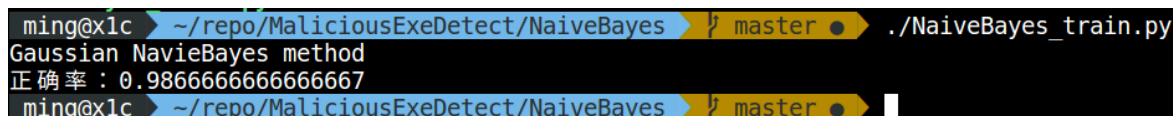
```
ming@x1c > ~/repo/MaliciousExeDetect/SVM > master ● > ./svm_train.py
kernel:sigmoid
gamma: 0.1
正确率: 0.8133333333333334
ming@x1c > ~/repo/MaliciousExeDetect/SVM > master ● > |
```

图 5.4 SVM 分类结果 (kernel=poly, $\gamma=0.1$)

华中科技大学课程设计报告

➤ NaiveBayes

使用高斯朴素贝叶斯对测试数据进行分类，得到分类正确率为 0.9866，分类结果如下图 5.5 所示。



```
ming@xlc ~/repo/MaliciousExeDetect/NaiveBayes master ● ./NaiveBayes_train.py
Gaussian NavieBayes method
正确率：0.9866666666666667
ming@xlc ~/repo/MaliciousExeDetect/NaiveBayes master ●
```

图 5.5 NaiveBayes 分类结果

➤ 结论

- 1) 使用不同的分类算法，分类效果不同；
- 2) 对同一种分类算法，参数设置不同也会得到不同的分类正确率；
- 3) 通过上述三种算法，可以看出高斯朴素贝叶斯算法的分类准确率最高，以多项式函数作为核函数的 SVM 算法其次，KNN 算法的分类准确率稍低，以 sigmoid 函数作为核函数的 SVM 算法的准确率最低；
- 4) 对于该线性不可分的样本，SVM 算法使用多项式函数比使用 sigmoid 函数作为核函数表现要好；
- 5) 由于样本数较少，故使用 3NN 和 5NN 分类结果相同，也体现了 KNN 算法的局限性，在数据量较小的情况下对数据利用效率不高；
- 6) 本数据挖掘结果中，在适当选择参数的情况下，使用朴素贝叶斯算法和 SVM 算法分类效果较好。

6 总结与心得

6.1 课设总结

本数据挖掘课程设计完成了对于病毒/非病毒数据的特征提取与分析，数据读取、数据清洗、数据预处理，使用包含 KNN、SVM、NaiveBayes 三种算法编写的程序训练分类器，并使用分类器对测试程序进行分类，得到每种分类算法的分类正确率，完成数据挖掘的项目目标。

6.2 课设心得

- 1) 在具有理论基础之上，通过实践完成了对 KNN 和 SVM，以及 NaiveBayes 算法的使用，体会到了三种算法对于数据分类的效用；
- 2) SVM 和 NaiveBayes 的算法原理虽然简单易懂，但是分类的效果却很好，不愧为数据挖掘中的经典算法，对具有坚实数学基础的算法有了更具体的感悟；
- 3) 研读论文^[2]，加深了对于 N-gram 提取特征方法的理解。从不同角度提取特征，并且考虑到时间和空间上的可行性和问题，给出优化措施。这些研究方法和研究思路是我要深刻领会和学习的，在这过程中体会到了数据挖掘的乐趣；
- 4) 通过操作系统中位图的概念自行设计了特征向量位图矩阵，使得训练数据更加直观，并且容易处理，体会到解决问题不能拘泥于某一科的知识，学科间交叉更能帮助解决问题；
- 5) 以课程设计的方式训练以及完善了对于数据挖掘的理解，有助于更牢固地掌握课堂所学，对于我们学习《数据挖掘》这门课是非常有益的。

华中科技大学课程设计报告

参考文献

- [1] <https://archive.ics.uci.edu/ml/datasets/Detect+Malicious+Executable%28AntiVirus%29>.
- [2] MM Masud , Latifur Khan, Bhavani Thuraisingham, A hybrid Model to detect malacious executable
- [3] <https://see.stanford.edu/course/cs229>