南京信息工程大学

《数据挖掘与安全》课程论文(设计)

题　　目　网页后门 webshell 检测

学生姓名　李辰缘

学　　号　20178314015

学　　院　计算机与软件学院

专　　业　信息安全专业

指导教师　闫雷鸣

二〇二〇 年 六月 十七 日

目　录

网页后门 webshell 检测

李辰缘

南京信息工程大学计算机与软件学院，江苏 南京 210044

摘要： 随着互联网的发展，由此引发的安全事件日益涌现，系统的受攻击面广以及攻击技术的多样化，导致 Web 应用系统易被攻击入侵，严重影响了国家的经济安全、社会稳定。当前，随着机器学习、深度学习等技术的发展，也可以应用到 WebShell 检测中，通过一种智能检测 WebShell 的机器学习算法，对已知存在 WebShell 和不存在 WebShell 的页面进行特征学习，完成对未知页面的预测。数据集包含正常 PHP 文件 9000 个，恶意 PHP 后门文件 3000 个，来自互联网上常见的 WebShell 样本。对数据进行预处理，使用 VLD 拓展提取 PHP 在 Zend 虚拟机中的 opcode 指令，并使用 2-Gram 对 opcode 进行分组，用 TF-IDF 进一步处理。使用朴素贝叶斯、RF、SVM、xgboost、MLP、CNN 等多种方法训练模型并测试。

关键词： opcode；webshell；深度学习

Web Backdoor Webshell Detection

Chenyuan Li

School of computer and software，NUIST，Nanjing 210044

Abstract： With the development of the Internet, the security incidents caused by it are emerging day by day. The wide range of attacks and the diversification of attack technology lead to the web application system vulnerable to attack and invasion, which seriously affects the economic security and social stability of the country. At present, with the development of machine learning, deep learning and other technologies, it can also be applied to the detection of webshell. Through a machine learning algorithm of intelligent detection of webshell, we can learn the characteristics of known and non-existent webshell pages and complete the prediction of unknown pages. The dataset contains 9000 normal PHP files and 3000 malicious PHP backdoor files, which are from the common webshell samples on the Internet. The data is preprocessed, the opcode instructions of PHP in Zend virtual machine are extracted by VLD extension, opcode is grouped by 2-gram, and further processed by TF-IDF. Using naive Bayes, RF, SVM, xgboost, MLP, CNN and other methods to train the model and test.

Key words： Opcode; Webshell; Deep Learning

1 引言

1.1 应用背景

基于网页及网页所连接的各种资源的安全性问题就构成了 Web 安全的实体。Web 安全由于其平台的特殊性，通常归纳为网站安全、数据安全以及 Web 平台上运行的应用安全三大类。WebShell 就是以 asp、php、jsp 网页文件形式存在的一种命令执行环境，通俗来讲，即一种网页后门。恶意用户在入侵了一个网站后，通常会上传 WebShell 文件至服务端，后门文件与网站服务器 Web 目录下的正常的网页文件混在一起，然后就可以使用浏览器来访问 WebShell，得到一个命令执行环境，以控制目标网站服务器，也为后续的提权操作做准备。

通常来说，WebShell 可以简单分为三类：大马、小马和一句话木马。大马，体积大、支持的功能全面，包括文件管理、命令执行、数据库操作等；可通过代码加密等手段使其具有一定隐蔽性。小马，体积小，支持功能少，一般只具备特定功能，如文件上传功能服务于上传大马。一句话木马，代码简短，通常只有一句代码，结合菜刀等 WebShell 管理工具使用；基于 B/S 架构，仅用于向服务端提交控制数据；使用灵活，可单独使用也可嵌入其它正常文件。

1.2 文章结构

第一章是引言部分，包括应用背景和全文结构，详细叙述了有关 webshell 识别的研究现状和应用背景，并给出本文的文章结构。

第二章为数据集分析，介绍了数据来自互联网上常见的 WebShell 样本和开源 CMS 系统，包括 WordPress、PHPCMS 和 PHPMyAdmin。

第三章为特征提取，介绍了词袋和 TF-IDF 模型，并根据 opcode 总类别不超过 200 个，选用 N-Gram 模型进行特征选择的优化。详细说明了他们的原理和实现过程。

第四章为研究方法分析，包括朴素贝叶斯、支持向量机、随机森林、xgboost、卷积神经网络和人工神经网络，详细说明了他们的原理、实现过程和模型评价。

第五章为各模型的性能比较，用准确率、精度、召回率、F1-score 等机器学习常见指标，比较各分类方法的性能优劣。

第六章为总结与展望。该部分对本人的主要工作进行总结，说明了目前分类算法存在的一些问题和缺点，并提出了改进的方法，指出进一步的研究方向。

2 数据集

数据集中 3000 个黑样本是 WebShell，9000 个白样本是开源 CMS 的 PHP 文件。

2.1 黑样本

WebShell 数据来自互联网上常见的 WebShell 样本，为了演示方便，全部使用了基于 PHP 的 WebShell 样本。数据来自 GitHub 上相关项目，如图 1 所示。

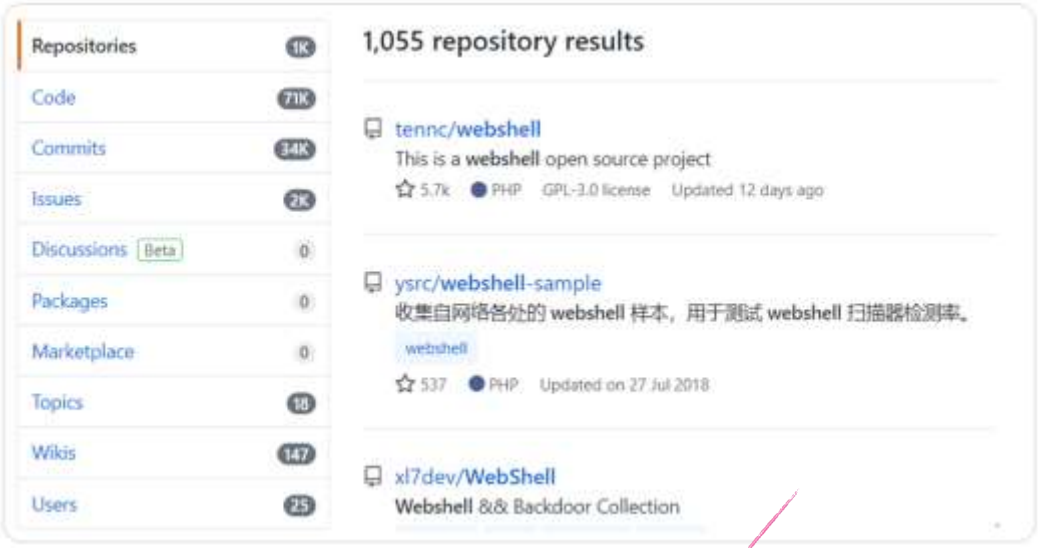

图 1 GitHub 上 webshell 项目

2.2 白样本

白样本主要使用常见的基于 PHP 开发的开源软件，主要包括以下几种。

2.2.1 WordPress

WordPress 是一种使用 PHP 语言开发的平台，用户可以在支持 PHP 和 MySQL 数据库的服务器上架设属于自己的博客，也可以把 WordPress 当作一个内容管理系统来使用。WordPress 有许多第三方开发的免费模板，安装方式，简单易用，拥有成千上万的各式插件和不计其数的主题模板样式。

2.2.2 PHPCMS

PHPCMS 是一款网站管理软件。该软件采用模块化开发，支持多种分类方式，使用它可方便地实现个性化网站的设计、开发与维护。它支持众多的程序组合，可轻松实现网站平台迁移，并可广泛满足各种规模的网站需求，可靠性高，是一款具备文章、下载、图片、分类信息、影视、商城、采集、财务等众多功能的强大、易用、可扩展的优秀网站管理软件。

2.2.3 phpMyAdmin

phpMyAdmin 是一个基于 PHP 开发的 MySQL 数据库管理端，通过 phpMyAdmin 提供的图形化的 Web 界面，可以完成绝大多数针对 MySQL 数据库的日常运维，包括但不限于针对数据库、表以及数据库用户的增、删、查、改。

3 特征提取

3.1 词袋和 TF-IDF 模型

使用最常见的词袋和 TF-IDF 模型提取文本特征。把一个 PHP 文件作为一个完整的字符串处理，定义函数 load_one_file 加载文件到一个字符串变量中返回，文件中的回车换行需要过滤掉

由于开源软件中包含大量图片、JavaScript 等文件，所以遍历目录时需要排除非 PHP 文

件。另外开源软件的目录结构相对复杂，所以需要递归访问指定目录并加载指定文件类型。

加载搜集到的 WebShell 样本，并统计样本个数，将 WebShell 样本标记为 1，加载搜集到的开源软件样本，并统计样本个数，将开源软件样本标记为 0。将 WebShell 样本和开源软件样本合并，使用 TF-IDF 进行处理。

3.1 opcode 和 N-Gram 模型

PHP 的 opcode 一共只有 153 种，因此可以使用 N-Gram 模型。2-Gram 是词袋模型的一个细分类别，N-Gram 基于这样一种假设，第 n 个单词只和它前面的 n-1 个词有关联，每 n 个单词作为一个处理单元。

通过设置 CountVectorizer 函数的 ngram_range 参数和 token_pattern 即可实现 N-Gram，其中 ngram_range 表明 N-Gram 的 n 取值范围，2-Gram 要设置成（2，2）。token_pattern 表明词切分的规则，设置为 r'\b\w+\b'即可。实现过程如图 2 所示。

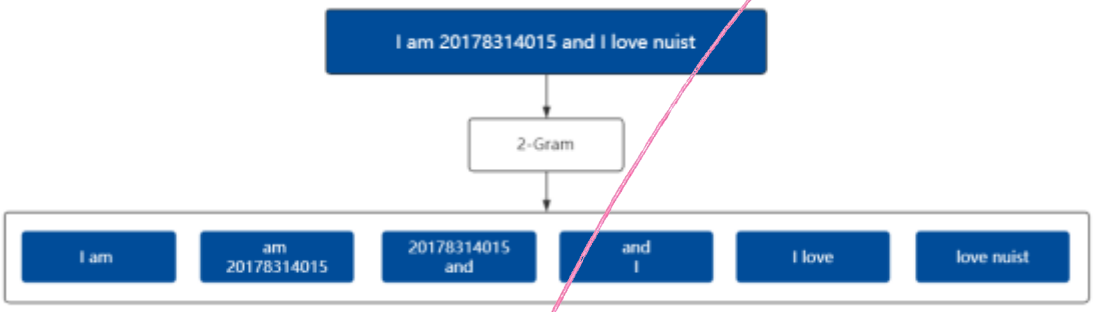

图 2 N-Gram 原理

opcode 是计算机指令中的一部分，PHP 作为一门动态脚本语言，其在 Zend 虚拟机执行过程为：读入脚本程序字符串，经由词法分析器将其转换为单词符号，接着语法分析器从中发现语法结构后生成抽象语法树，再经静态编译器生成 opcode，最后经解释器模拟机器指令来执行每一条 opcode。常见 opcode 见表格 1

表 1 常见 opcode

opcode 代码	编号	语义
ZEND_NOP	0	空操作
ZEND_CONCAT	8	字符串连接
ZEND_BOOL_XOR	14	逻辑异或 XOR
ZEND_ASSIGN	38	赋值=
ZEND_ASSIGN_REF	39	引用赋值 =&
ZEND_ECHO	40	ECHO
ZEND_PRINT	41	PRINT
ZEND_ADD_VAR	56	将变量加到字符串
ZEND_BEGIN_SILENCE	57	错误屏蔽开始
ZEND_INIT_FCALL_BY_NAME	59	初始化通过名称调用函数
ZEND_DO_FCALL	60	函数调用
ZEND_RETURN	62	函数返回
ZEND_RECV	63	函数声明时传递参数
ZEND_NEW	68	new 操作
ZEND_INIT_NS_FCALL_BY_NAME	69	命名空间函数调用
ZEND_ADD_ARRAY_ELEMENT	72	添加数组元素
ZEND_INCLUDE_OR_EVAL	73	include/require/eval 操作
ZEND_UNSET_VAR	74	UNSET 操作
ZEND_FETCH_FUNC_ARG	92	获取函数参数地址

在 php.ini 配置文件中添加 extension=vld.so 来激活 VLD，通过使用 PHP 的 VLD 扩展查看对应的 opcode，其中 vld.active=1 表示激活 VLD，vld.execute=0 表示只解析不执行。以一个非常简单的 PHP 代码图 3 为例，该代码会包含 test.php 并执行，使用 VLD 查看的结果。一个简单的 PHP 文件的实例如图 3.

```php
<?php
include("test.php");
eval("test.php");
?>
```

opcode	operands
INCLUDE_OR_EVAL	'test.php',INCLUDE
INCLUDE_OR_EVAL	'test.php',EVAL
RETURN	1

图 3 提取 opcode 的例子

代码实现方面，首先使用 VLD 处理 PHP 文件，把处理的结果保存在字符串中，PHP 的 opcode 都是由大写字母和下划线组成的单词，使用 re.findall 函数从字符串中提取全部满足条件的 opcode，并以空格连接成一个新字符串，遍历读取指定目录下全部 PHP 文件，保存其对应的 opcode 字符串。依次读取保存 WebShell 样本以及正常 PHP 文件的目录，加载对应的 opcode 字符串，其中标记 WebShell 为 1，正常 PHP 文件为 0。使用 2-Gram 处理 opcode 字符串，通过设置 ngram_range=（2，2）就可以使用 2-Gram。同理，如果使用 3-Gram，则设置 ngram_range=（3，3）即可。使用 TF-IDF 进一步处理。流程如图 4 所示

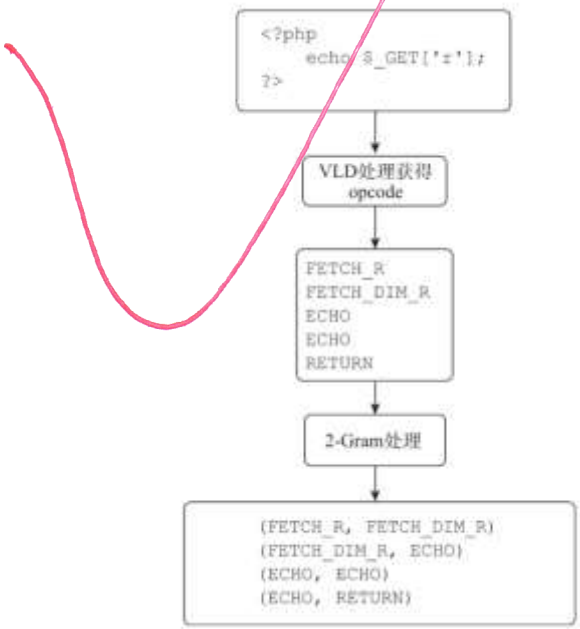

图 4 opcode 处理流程图

4 研究方法和模型评估

4.1 朴素贝叶斯

朴素贝叶斯[1,2,3]是基于统计学的一种分类策略，其核心思想在于贝叶斯公式，贝叶斯公

式是概率与统计中的重要计算公式，即：

$$P(c|x)=\frac{P(x|c)P(c)}{P(x)}=\frac{P(c)}{P(x)}\prod_{i=1}^{n}P(x_i|c) \tag{1}$$

其中，c 为可能的分类结果，$P(c)$为类别的先验概率，$P(x|c)$为似然概率计算公式，即在 c 类的前提下结果为 x 样本的概率，n 为特征总数，x_i 为 x 样本的第 i 个特征值。

类别的先验概率计算方法为：

$$P(c)=\frac{N_c}{N_t} \tag{2}$$

其中，N_c 为训练集中 c 类别的数目，N_t 为训练集样本总数。

$P(x|c)$针对不同的数据类型，有不同的计算方法，对于离散数据，似然概率按如下公式计算：

$$P(x_i|c)=\frac{D_{tc}(x_i)}{D_{tc}} \tag{3}$$

其中，$D_{tc}(x_i)$ 是训练集 D_{tc} 中第 i 个属性为 x_i 的概率，D_{tc} 为训练集样本总数

对于数值连续型变量，计算公式变为：

$$P(x_i|c)=\frac{1}{\sqrt{2\pi}\sigma_{c,i}}e^{-\frac{(x_i-\mu_{c,i})^2}{2\sigma_{c,i}^2}} \tag{4}$$

其中，$\mu_{c,i}$ 为 c 类样本在第 i 个样本上的均值，$\sigma_{c,i}$ 为 c 类样本在第 i 个样本上的方差。

通过数据分布的状况，朴素贝叶斯算法可以分为高斯分布模型、伯努利分布模型、多项式分布模型，图 2 反应了朴素贝叶斯算法的实现过程，简单来说，在获取训练集样本后，朴素贝叶斯首先计算每个类别或标签出现的概率，再在每个类别中分别计算每个特征出现的概率，对每个属性组合计算其属于每个类别的概率，选取最大概率的类别作为测试样本的分类预测结果。特征提取使用词袋和 TF-IDF 模型的朴素贝叶斯算法处理流程如下：

1）将 WebShell 样本以及常见 PHP 开源软件的文件提取词袋；
2）使用 TF-IDF 处理；
3）随机划分为训练集和测试集；
4）使用朴素贝叶斯算法在训练集上训练，获得模型数据；
5）使用模型数据在测试集上进行预测；
6）验证朴素贝叶斯算法预测效果。

实例化朴素贝叶斯算法，并在训练集上训练数据，针对测试集进行预测。表格 5 展示了在词袋最大特征数为 15000 的情况下，使用词袋和 TF-IDF 模型时，准确率为 94.92%，召回率为 93.19%，TP、FP、TN、FN 矩阵 confusion_matrix 如图 5 所示。

```
metrics.accuracy_score:
0.97361081313
metrics.confusion_matrix:
[[3566   52]
 [  71  972]]
metrics.precision_score:
0.94921875
metrics.recall_score:
0.931927133269
metrics.f1_score:
0.940493468795
```

图 5 朴素贝叶斯

表 2 朴素贝叶斯

次数	Accuracy	Precision	Recall	F1
第一次训练	0.97	0.95	0.93	0.94
第二次训练	0.97	0.95	0.93	0.94

4. 2 随机森林

随机森林[4]是集成学习方法的一种，顾名思义，集成学习是许多小的分类器集合成一个大的分类器的过程。随机森林的基本单元为决策树模型，每一个基本单元都是一个分类器，基本单元个数决定了分类结果的数目，相当于每个单元对可能的类别进行了一次投票，而随机森林对所有的分类投票结果进行统计，最终的输出是被投票最多的类别。

图 6 反映了随机森林算法的实现过程，随机森林算法的核心可以分为随机和森林两个部分，前者包括随机取样形成若干训练集和随机选取特征，使得算法对过拟合有较高的抵抗性；后者就是组合若干基本单元的结果，集思广益，提高了算法的准确率。

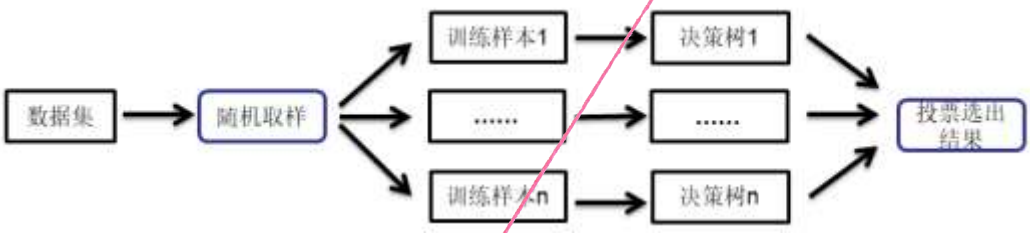

图 6 随机森林实现过程

特征提取使用 opcode 模型的随机森林算法处理流程如下：

1）将 WebShell 样本以及常见 PHP 开源软件的文件提取 opcode。
2）随机划分为训练集和测试集。
3）使用随机森林在训练集上训练，获得模型数据。
4）使用模型数据在测试集上进行预测。
5）验证随机森林预测效果。

实例化随机森林算法，并在训练集上训练数据，针对测试集进行预测。图 7 展示了在词袋最大特征数为 15000 的情况下，使用词袋和 TF-IDF 模型时，准确率为 97.96%，召回率为 96.47%

```
metrics.accuracy_score:
0.979591836735
metrics.precision_score:
0.968503937008
metrics.recall_score:
0.964705882353
metrics.f1_score:
0.966601178782
```

图 7 随机森林

表 3 随机森林

次数	Accuracy	Precision	Recall	F1
第一次训练	0.98	0.97	0.96	0.97
第二次训练	0.98	0.97	0.96	0.97

4.3 xgboost

说起 xgboost，不得不提一下 GBDT，也就是梯度提升决策树，这是一种基于树的集成算法，至于什么方法构成了 GBDT 的决策树，无非就是 ID3、C4.5、C5.0、CART，最常用的就是那个 CART，多棵树的集合就构成了 GBDT。其实 GBDT 是对残差的拟合，假设目标函数是 9，第一棵树预测为 5，剩下 4 的误差，那么下一颗树继续拟合 4，直至误差满足要求，这和 xgboost 的思路是一样的。GBDT 的每一步优化都依赖于上一步的误差，当大数据量的时候就太慢了，xgboost 通过改变目标函数来避免了这个问题。xgboost 是一个树集成模型，他将 K（树的个数）个树的结果进行求和，作为最终的预测值。其中 q 代表每棵树的结构，他将样本映射到对应的叶节点；T 是对应树的叶节点个数；f(x)对应树的结构 q 和叶节点权重 w。所以 XGBoost 的预测值是每棵树对应的叶节点的值的和。我们的目标是学习这 k 个树，所以我们最小化这个带正则项的目标函数。特征提取使用 opcode 模型的 xgboost 算法处理流程如下：

1）将 WebShell 样本以及常见 PHP 开源软件的文件提取 opcode。
2）随机划分为训练集和测试集。
3）使用 xgboost 在训练集上训练，获得模型数据。
4）使用模型数据在测试集上进行预测。
5）验证 xgboost 预测效果。

在训练集上训练数据，针对测试集进行预测。图 8 展示了特征提取使用 opcode 模型，在最大特征数取 100 的情况下 TP、FP、TN、FN 的矩阵 confusion_matrix。

```
load 571 white 557 black
xgboost
              precision    recall  f1-score   support

           0       0.98      0.89      0.93       238
           1       0.89      0.98      0.93       214

   micro avg       0.93      0.93      0.93       452
   macro avg       0.94      0.94      0.93       452
weighted avg       0.94      0.93      0.93       452

[[212  26]
 [  4 210]]
```

图 8 xgboost

表 4 xgboost

次数	Precision	Recall	F1
第一次训练	0.98	0.89	0.93
第二次训练	0.98	0.89	0.93

4.4 支持向量机

支持向量机[5,6]是一种常见的基于统计学的分类分析算法，由于低维空间存在线性不可分的问题，因此 SVM 通过寻找合适的核函数，将其投影到更高维的空间，在高维空间中可以较容易地完成线性划分的目标，进而找到一个最优地超平面，进行数据的二分类。

现实生活中，很多问题不是简单的二分类，而是更为复杂的多分类问题。支持向量机可以通过一对一方法或一对多方法进行多分类的处理。一对一方法在每两类样本间设计支持向量机，共需要 $k(k\text{-}1)/2$ 个向量机，需要分类时，将得票最多的类别作为未知数据的类别；一对多方法将某一类的样本放于一类，剩下所有的样本为另一类别，共有 k 个支持向量机，处理分类任务时选择分类函数值最大的类别作为测试样本的类别。

通常情况下，分类结果会受到核函数的直接影响，常见的核函数包括线性核函数、高斯核函数等。

SVM 算法具有较好的准确率和泛化性能，并且对小样本数据和高维数据的分类效果较好。但是对缺失数据敏感，并且对于非线性问题没有固定的处理方法，不同的核函数得到的分类效果不相同，需要进行调参以确定合适的分类模型。特征提取使用 opcode 模型的 SVM 算法处理流程如下：

1）将 WebShell 样本以及常见 PHP 开源软件的文件提取 opcode。
2）随机划分为训练集和测试集。
3）使用 SVM 在训练集上训练，获得模型数据。
4）使用模型数据在测试集上进行预测。
5）验证 SVM 预测效果。

在训练集上训练数据，针对测试集进行预测。图 9 展示了特征提取使用 opcode 模型，在最大特征数取 100 的情况下，准确率为 80.51%，召回率为 95.65%，TP、FP、TN、FN 矩阵 confusion_matrix 如图 9 所示。

```
load 564 white 499 black
metrics.accuracy_score:
0.8051643192488263
metrics.confusion_matrix:
[[145  74]
 [  9 198]]
metrics.precision_score:
0.7279411764705882
metrics.recall_score:
0.9565217391304348
metrics.f1_score:
0.826722338204593
```

图 9 SVM

表 5 SVM

次数	Accuracy	Precision	Recall	F1
第一次训练	0.81	0.73	0.96	0.83
第二次训练	0.81	0.73	0.96	0.83

4.5 人工神经网络

MLP[7]也称人工神经网络，是人工智能领域的焦点，也就是通常我们说的神经网络，他将抽象的机器工作的过程和人生理上的神经活动联系起来，通过大量节点的运算，让机器自主学习数据之间的联系。本文主要介绍神经网络用于分类的过程。

人工神经网络的结构可以分为输入层、输出层以及隐藏层。图 6 反映了神经网络的实现过程。数据的特征从输入层传递至隐藏层，通过激活函数运算，将每个节点的信息再传送到输出层，得到输出结果。

在神经网络中，我们常常采用 bp 算法即反向误差传播算法，上一段体现的过程是正向的，正向的传递信息可以得到一个结果，但是这个结果未必是比较好的。于是我们计算误差，并将误差从最后一层依次传递到上一层，从而更新每个神经元节点的权重，使其向着更好的分类效果逼近。

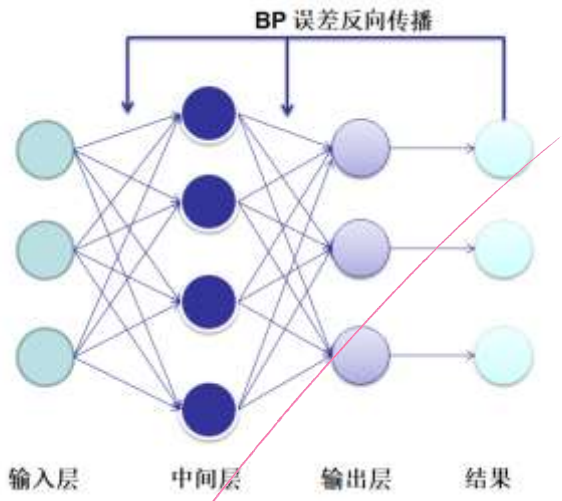

图 10 神经网络结构示意图

激活函数是在节点上计算和运行的函数，将信息映射到下一层，这使得神经网络可以进行非线性的分类，是神经网络中不可缺少的一部分。常用的激活函数包括 Sigmod 函数、Tanh 函数、Relu 函数、Swish 函数等。

Sigmod 函数的计算公式如下：

$$f(x)=\frac{1}{1+e^{-x}} \tag{5}$$

Tanh 函数的计算公式如下：

$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{6}$$

Relu 函数常用于中间层的信息输出，计算公式如下：

$$f(x)=\max(0,x) \tag{7}$$

Swish 函数[6]是 2017 年由 google 提出的新型激活函数，被证明有优于当前激活函数的性能，其计算公式如下：

$$f(x)=x\bullet\mathrm{sigmod}(x) \tag{8}$$

在分类问题中，softmax 常被用作输出层，它可以让输出映射到(0,1)区间上，类似于比较概率以完成多分类任务，softmax 函数定义如下：

$$S_i = \frac{e^{V_i}}{\sum_i^C e^{V_i}} \tag{9}$$

其中，V_i 为上一层的输出，C 为类别总数，这样神经网络的输出便被转换为概率分布。

在 BP 算法中，需要有一个向前传递的损失，在分类问题中，通常使用交叉熵损失函数，按如下公式计算：

$$H(p,q) = -\sum_x p(x)\log q(x) \tag{10}$$

其中，$H(p,q)$ 为交叉熵，p 和 q 分别为期望输出和实际输出的概率分布。

神经网络在很多领域尤其是图像处理了方面有着广泛的应用，但在训练集上具有卓越的分类效果并不总是一件有益的事情，以 CNN 为例，卷积神经网络通过卷积层提取图片的特征，这有可能导致该网络对于训练集样本具有依赖性，可以极好地提取训练集特征从而实现好的分类效果，但是对于测试样本不具有泛化性，测试集准确率远小于训练集正确率，也就是常说的过拟合现象。针对这种问题，目前常用的方法有：增加 l1 或 l2 正则化进行权重缩减，设置 dropout 随机丢弃，增加 batch_size 等。

人工神经网络分类具有多方面的优点，其准确率高，并且可以通过不断加深网络层数和改良学习过程以获得更好的分类效果，对于噪声数据的鲁棒性较强。

但人工神经网络也存在诸多缺点，首先和支持向量机算法类似，对于不同的数据集会有不同的较为适合的参数集，必须通过不断调参以获得更好的分类效果。其次，神经网络的训练时间远超过上述的各类分类算法，甚至会出现欠拟合即没有有效学习的现象；最后，人工神经网络的结果难以用数学理论解释，影响分类的可信度。特征提取使用 opcode 和 N-Gram 模型的 MLP 算法处理流程如下：

1）将 WebShell 样本以及常见 PHP 开源软件的文件提取 opcode。
2）使用 N-Gram 处理。
3）随机划分为训练集和测试集。
4）使用 MLP 算法在训练集上训练，获得模型数据。
5）使用模型数据在测试集上进行预测。
6）验证 MLP 算法预测效果。

实例化 MLP 算法，并在训练集上训练数据，针对测试集进行预测。图 11 展示了特征提取使用 opcode 和 N-Gram 模型，n 取 4，在最大特征数取 2000 的情况下，准确率为 83.30%，召回率为 96.03%，TP、FP、TN、FN 矩阵 confusion_matrix 如图 11 所示。

```
0.963460337289
metrics.confusion_matrix:
[[2601   97]
 [  20  484]]
metrics.precision_score:
0.833046471601
metrics.recall_score:
0.960317460317
metrics.f1_score:
0.892165898618
```

图 11 人工神经网络

表 6 人工神经网络

次数	Accuracy	Precision	Recall	F1
第一次训练	0.96	0.83	0.96	0.89
第二次训练	0.96	0.83	0.96	0.89

4.6 卷积神经网络

卷积神经网络是近几年发展起来，被不断重视的一种用于高效识别的网络引擎。20 世纪 60 年代，Hubel 和 Wiesel 研究猫脑皮层中用于局部敏感，以及方向选择的神经元时发现了其独特的网络结构可以有效降低反馈神经网络的复杂性，继而提出卷积神经网络(Convolutional Neural Networks, CNN)。

卷积神经网络是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现。卷积神经网络由一个或多个卷积层和顶端的全连通层（对应经典的神经网络）组成，同时也包括关联权重和池化层。这一结构使得卷积神经网络能够利用输入数据的二维结构。与其他深度学习结构相比，卷积神经网络在图像和语音识别方面能够给出更好的结果。这一模型也可以使用反向传播算法进行训练。相比较其他深度、前馈神经网络，卷积神经网络需要考虑的参数更少，使之成为一种颇具吸引力的深度学习结构。

卷积神经网络利用输入是图片的特点，把神经元设计成三个维度 ： width, height, depth，如图 12 所示。

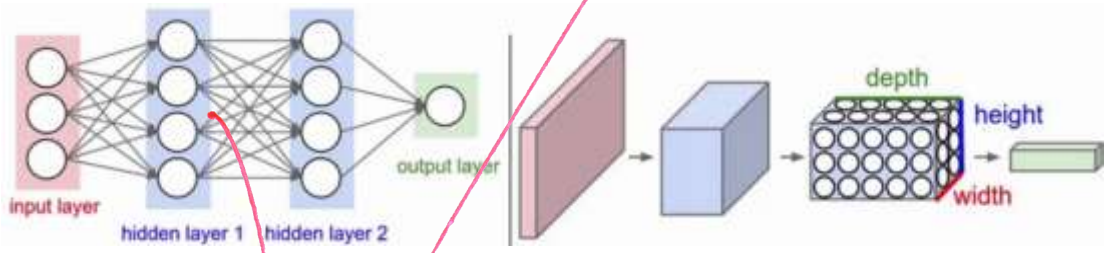

图 12 传统神经网络与卷积神经网络的对比

卷积神经网络通常包含以下几种层：卷积层、线性整流层、池化层、损失函数层。特征提取使用 opcode 调用序列模型的 CNN 算法处理流程如下：

1）将 WebShell 样本以及常见 PHP 开源软件的文件提取 opcode。
2）使用词袋处理，针对 opcode 进行编号，生成 opcode 调用序列。
3）随机划分为训练集和测试集。
4）使用 CNN 算法在训练集上训练，获得模型数据。
5）使用模型数据在测试集上进行预测。
6）验证 CNN 算法预测效果

运行程序，经过 5 轮训练，在 opcode 序列长度为 3000 的情况下，使用 opcode 序列模型，TP、FP、TN、FN 矩阵如图，整个系统的准确率为 96.96%，召回率为 82.34%。

```
metrics.accuracy_score:
0.968144909432
metrics.confusion_matrix:
[[2685   13]
 [  89  415]]
metrics.precision_score:
0.969626168224
metrics.recall_score:
0.823412698413
metrics.f1_score:
0.890557939914
```

表 7 卷积神经网络

Accuracy	Precision	Recall	F1
0.96	0.96	0.82	0.89

5 性能比较

5.1 实验环境设置

本文的所有实验均使用 python 语言编程实现，编程环境使用 pycharm，分类算法使用 python 的机器学习框架 sklearn 实现。具体的实验环境配置如表 8 所示。

本文所有实验均采用交叉验证的方式，将数据样本划分为训练集和测试集，训练集和测试集的比例为 4:1。

表 8 实验环境设置

实验环境	配置与版本
CPU	Intel(R) Core(TM) i5-8250U CPU 1.60GHz
操作系统	Windows 10
编程语言	Python 3.6.4
编程环境	PyCharm Community Edition 2017.1.3
机器学习框架	sklearn

5.2 性能比较

5.2.1 准确性分析

表 9 反映了不同方法的准确率情况，通过数据对比，可以发现分类准确率最高的方法为随机森林，其次表现较好的算法有卷积神经网络。

表 9 分类算法准确率

	Accuracy	Precision
CNN	0.96	0.96
朴素贝叶斯	0.97	0.95
SVM（RBF）	0.81	0.73
随机森林	0.98	0.97
xgboost	0.98	0.89
人工神经网络	0.96	0.83

5.2.2 精度和召回率分析

精度和召回率是除准确率外，机器学习模型需要注意的另外两个重要指标，对于二分类问题，需要知道真正例、假正例、真反例、假反例的概念。

真正例指实际是正例的数据被分为正例，是正确分类的结果，用 TP 表示。
假正例指实际是反例的数据被分为正例，是错误分类的结果，用 FP 表示。
真反例指实际是反例的数据被分为反例，是正确分类的结果，用 TN 表示。
假反例指实际是正例的数据被分为反例，是错误分类的结果，用 FN 表示。
由此引出精度和召回率的概念，精度是对数据精确性的衡量方法，计算方式为：

$$precision = \frac{TP}{TP + FP} \tag{11}$$

召回率是对完全性的度量，其计算公式为：

$$recall = \frac{TP}{TP + FN} \tag{12}$$

在机器学习中，常用 F1 score 结合精度和召回率，其计算方法为：

$$F_1 = 2 \bullet \frac{precision \bullet recall}{precision + recall} \tag{13}$$

精度和召回率有多种可视化方法，包括混淆矩阵、ROC 曲线和 AUC[16]值。ROC 曲线以 FP 为横轴，以 TP 为纵轴，反应了召回率和精度之间的变化，其线下面积为 AUC 的值，反应了分类模型的总体性能指标。本文各分类算法在数据集上的 F_1 得分如表 10 所示。

表 10 各分类算法的 F1-score

	Recall	F_1
CNN	0.82	0.89
朴素贝叶斯	0.93	0.94
SVM（RBF）	0.96	0.83
随机森林	0.96	0.97
xgboost	0.93	0.93
人工神经网络	0.96	0.89

可以发现总体性能最高的方法为随机森林。

6 总结与展望

随着互联网的发展，由此引发的安全事件日益涌现，系统的受攻击面广以及攻击技术的多样化，导致 Web 应用系统易被攻击入侵，严重影响了国家的经济安全、社会稳定。当前，随着机器学习、深度学习等技术的发展，也可以应用到 WebShell 检测中，通过一种智能检测 WebShell 的机器学习算法，对已知存在 WebShell 和不存在 WebShell 的页面进行特征学习，完成对未知页面的预测。数据集包含正常 PHP 文件 9000 个，恶意 PHP 后门文件 3000 个，来自互联网上常见的 WebShell 样本。对数据进行预处理，使用 VLD 拓展提取 PHP 在 Zend 虚拟机中的 opcode 指令，并使用 2-Gram 对 opcode 进行分组，用 TF-IDF 进一步处理。使用朴素贝叶斯、RF、SVM、xgboost、MLP、CNN 等多种方法训练模型并测试。

在未来的工作中，我们认为本作品还可以从以下几个方面进行改进。一，增加检测方式，增加基于流量与日志的检测方式进一步提升该系统的检测准确率。二，不断更新扩大训练样本集，改进学习算法，进一步提高本系统识别的精准度。三，加大对于动态检测模块的改进，采用 server hook 检测与 opcode 检测结合， 进一步发挥动态检测技术的优势。

参考文献

[1] Chen W, Yan X, Zhao Z, et al. Spatial prediction of landslide susceptibility using data mining-based kernel logistic regression, naive Bayes and RBFNetwork models for the Long County area (China)[J]. Bulletin of Engineering Geology and the Environment, 2019, 78(1): 247-266.

[2] Wood A, Shpilrain V, Najarian K, et al. Private naive bayes classification of personal biomedical data: Application in cancer data analysis[J]. Computers in biology and medicine, 2019, 105: 144-150.

[3] Chen J, Dai Z, Duan J, et al. Naive Bayes with Correlation Factor for Text Classification Problem[J]. arXiv preprint arXiv:1905.06115, 2019.

[4] Wyner A J, Olson M, Bleich J, et al. Explaining the success of adaboost and random forests as interpolating classifiers[J]. The Journal of Machine Learning Research, 2017, 18(1): 1558-1590.

[5] Liang Jing-Wei, Wang Ming-Yang, Wang Shan, Li Shi-Long, Li Wan-Qiu, Meng Fan-Hao. Identification of novel CDK2 inhibitors by a multistage virtual screening method based on SVM, pharmacophore and docking model. Journal of enzyme inhibition and medicinal chemistry, 2020, 35(1): 235-244

[6] Abdollahi S, Pourghasemi H R, Ghanbarian G A, et al. Prioritization of effective factors in the occurrence of land subsidence and its susceptibility mapping using an SVM model and their different kernel functions[J]. Bulletin of Engineering Geology and the Environment, 2019, 78(6): 4017-4034.

[7] Crisosto C. Autoregressive Neural Network for Cloud Concentration Forecast from Hemispheric Sky Images[J]. International Journal of Photoenergy, 2019, 2019.

已阅！

2020/6/30