基于集成神经网络的计算机病毒检测方法

张波云[1,2],殷建平[1],张鼎兴[1],蒿敬波[1],王树林[1]
ZHANG Bo-yun[1,2],YIN Jian-ping[1],ZHANG Ding-xing[1],HAO Jing-bo[1],WANG Shu-lin[1]

1.国防科技大学 计算机学院,长沙 410073
2.湖南公安高等专科学校 计算机系,长沙 410138
1.School of Computer Science,National University of Defense Technology,Changsha 410073,China
2.Dept. of Computer Science,Hunan Public Security College,Changsha 410138,China
E-mail:hnjxzby@yahoo.com.cn

ZHANG Bo-yun,YIN Jian-ping,ZHANG Ding-xing,et al.Computer viruses detection based on ensemble neural network. Computer Engineering and Applications,2007,43(13):26-29.

Abstract: Motivated by the standard signature-based technique for detecting viruses,we explore the idea of automatically detecting malicious code using the n-gram analysis.After selecting features based on information gain,the BP neural network is used in the process of building and testing the proposed multi-classifiers system.Experimental results produced by the proposed detection engine shows improvement of accuracy and generalization compared to the classification results of the individual classifier.
Key words: computer viruses;ensemble learning;information gain;BP neural network

摘 要:在借鉴传统的特征扫描技术的基础上,提出了一种基于n-gram分析的计算机病毒自动检测方法。将基于信息增益的特征选择技术引入集成神经网络的构建中,结合Bagging算法,同时扰动训练数据和输入属性生成精确且差异度大的个体分类器,在此基础上以集成的BP神经网络为模式分类器实现对病毒的检测。该法并不针对某一特定病毒,是一种通用的病毒检测器。实验表明提出的检测方法具有较强的泛化能力和较高的精确率。
关键词:计算机病毒;集成学习;信息增益;BP神经网络
文章编号:1002-8331(2007)13-0026-04 **文献标识码**:A **中图分类号**:TP309.05

1 引言

当前的计算机病毒检测技术主要基于特征检测法,其基本方法是提取已知病毒样本的特征,并将此特征数据添加到病毒特征库中,在病毒检测时通过搜索病毒特征库查找是否存在相匹配的病毒特征来发现病毒。这种检测办法只能用于检测已知的病毒,对于新出现的病毒的检测无能为力[1]。为了解决这一问题,各反病毒研究机构专家都在努力探讨病毒检测的智能方法。在IBM WATSON病毒研究中心,曾经成功地将神经网络用于判断引导型病毒[2],他们声称在不久的将来把人工免疫方法应用于病毒检测产品中去。1998年,Symantec公司宣布把IBM公司专利的神经网络引导检测技术集成到诺顿防病毒(Norton Anti-Virus)产品中。其后Symantec公司又引入了基于启发式查毒的Bloodhound技术融入到其反病毒产品中来[3]。何申等[4]对恶意网页代码从统计分析方法角度进行检测,并对分类算法的错误率上限进行了估计。Kolter[5]等人研究了多种模式识别算法如Narve Bayes、IBk和J48等用于检测未知恶意代码。

基于上述思想的启发,本文提出一种基于神经网络集成的病毒检测方法。集成学习技术是机器学习的研究热点之一,而对神经网络的集成是集成学习的一个重要的研究方向。神经网络集成[6]利用多个神经网络对同一个问题进行学习,集成在某输入示例下的输出由构成集成的各神经网络在该示例下的输出共同决定。该方法可以显著地提高神经网络系统的泛化能力,被视为一种非常有效的学习方法。Krogh和Vedelsby[7]在1995年指出,神经网络集成的泛化误差等于集成中个体网络的平均泛化误差和平均差异度之差,因此,要增强神经网络集成的泛化能力,一方面应尽可能提高个体网络的泛化能力,另一方面应尽可能地增大集成中各网络之间的差异。受此启发,本文将基于信息增益的特征选择技术引入神经网络集成的构建中,将扰动训练数据和扰动输入属性结合起来,以生成精确且差异度大的个体神经网络。实验表明本文提出的检测方法具有较强的泛化能力和较高的精确率。

2 基于信息增益的特征选择

2.1 N-gram

将连续的数据流按固定长度n分割成的子串即为n-

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60373023);湖南省教育厅优秀青年基金资助项目(No.05B072)。
作者简介:张波云(1972-),男,副教授,博士生,主研方向:网络与信息安全;殷建平,教授,博导;张鼎兴,蒿敬波,王树林,博士生。

gram。n-gram分析因其直观和易于实现，已成功应用于语言模型和语音识别领域[8]。将n-gram用于恶意代码分析在文献[9]中也有提及，但没做深入研究。

随着恶意代码数量剧增，很多恶意代码作者采用自动工具编写和编译恶意代码。n-gram分析采用概率统计方法从n-gram集中挖掘出隐含其间的特征，可以用于检测出编制病毒所用的自动生产机、编译器和编程环境甚至程序作者的某些编程习惯。该法还可较有效地防范病毒作者的反击。

将固定长度为n的滑动窗口，在数据集的线性比特流滑动，记录每个gram出现次数即可得到n-gram的统计频率分布。根据n-gram的分布特性，可以用于区分不同数据流的一致性。图1显示了一个滑动窗口大小为3 byte，一次向右滑动1 byte，在比特流上获取3-gram信息的一个范例。图中小窗口中的子串即为一个3-gram。

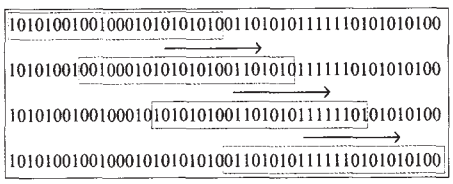

图1　滑动窗口方法(窗口大小3 Byte)

滑动窗口的大小选择依赖于具体的应用。n-gram的计算复杂度随着窗口的长度增加而呈指数级增长。设信息流中字母表大小为X，则易知n-gram覆盖空间为X^n。

本文主要将n-gram分析应用于对Windows PE格式的程序文件的检测，其字母空间为{0，1}。实验中将滑动窗口大小取为16 bit。对数据集进行预处理，可以获得相应的频率矩阵，表1是一个范例。

表1　特征频率分布矩阵(n=2)

样本	特征					
	0080	61C3	638D	9090	E020	…
Win32.Alcaul.a	2	4	21	8	79	
Win32.Alcaul.b	45	6	40	7	20	
Win32.Alcaul.c	11	11	18	8	14	
Win32.Alcaul.e	7	0	12	6	25	…
Win32.Alcaul.f	9	20	15	5	27	
Win32.Alcaul.g	11	0	17	3	20	
Win32.Alcaul.h	19	17	48	4	29	
…	…	…	…	…	…	

2.2　基于IG的特征选取

本文的特征选择是基于信息熵实现。信息增益(Information Gain，记为IG)反映了一个特征在分类中的重要性。先计算各个特征的信息增益值，然后忽略信息增益值小的特征以降低特征空间维数。

根据信息论的定义，变量X的熵为：

$$H(X)=-\sum_i P(x_i)\log_2(P(x_i))$$

变量X关于变量Y的条件熵定义为：

$$H(X|Y)=-\sum_j P(y_j)\sum_i P(x_i|y_j)\log_2(P(x_i|y_j))$$

式中$P(y_j)$为变量Y各分量的先验概率，$P(x_i|y_j)$为x_i关于y_j的后验概率。

信息增益定义为：

$$IG(X|Y)=H(X)-H(X|Y)$$

它说明由于Y的出现映射出X的部分信息从而导致X的信息熵下降。

在文中信息熵的计算公式为：

$$H(X)=-[P(x\text{ is normal})\cdot\log_2 P(x\text{ is normal})+P(x\text{ is abnormal})\cdot\log_2 P(x\text{ is abnormal})]$$

条件熵$H(X|y_i)$由下式计算可得：

$$\begin{aligned}H(X|y_i)=&-P(y_i=0)\cdot[P(x\text{ is normal}|y_i=0)\cdot\log_2 P(x\text{ is normal}|y_i=0)+\\&P(x\text{ is abnormal}|y_i=0)\cdot\log_2 P(x\text{ is abnormal}|y_i=0)]-\\&P(y_i=1)\cdot[P(x\text{ is normal}|y_i=1)\cdot\log_2 P(x\text{ is normal}|y_i=1)+\\&P(x\text{ is abnormal}|y_i=1)\cdot\log_2 P(x\text{ is abnormal}|y_i=1)]\end{aligned}$$

式中$y_i=0$表示特征y_i不在样本中出现，$y_i=1$表示特征y_i在样本中出现。实验中部分特征的信息增益值如表2所示。

表2　特征的信息增益值(n=2)

特征	正常代码样本集		恶意代码样本集		信息增益
	$y_i=1$	$y_i=0$	$y_i=1$	$y_i=0$	
FF84	98	11	92	0	0.000 954 615
4508	106	3	85	7	0.000 387 123
FDFF	109	0	89	3	0.001 103 624
F33C	101	8	76	16	0.002 371 356
BF28	94	15	91	1	0.000 767 842
…	…	…	…	…	…

3　基于IG-Bagging的神经网络集成

实验证明神经网络等不稳定分类器的设计、配置、训练都是NPC问题，神经网络的成功应用常取决于使用者的实际经验。为了克服上述问题，目前许多学者开始研究神经网络(Ensemble Neural Network)集成的方法。大量的理论和实践上的研究工作表明集成后的神经网络学习器通常比单个神经网络学习器能产生更好的性能[6]。

现在常用的神经网络集成方法都是通过扰动训练数据来获得差异度较大的个体网络，例如Boosting[10]中各网络的训练集决定于之前产生的网络的表现，被已有网络错误判断的示例将以较大的概率出现在新网络的训练集中。Bagging[11]的基础是可重复取样从原始训练集中随机抽取的若干示例来训练网络。通过重复选取训练集增加了神经网络集成的差异度，从而提高了泛化能力。本文结合Bagging算法，将基于信息增益(IG，Information Gain)的特征选择技术引入神经网络集成的构建中，将扰动训练数据和扰动输入属性结合起来，以生成精确且差异度大的个体神经网络(记为IG-Bagging)。

IG-Bagging的具体做法如下：给定训练集S，通过可重复取样获得一系列子训练集S_1，S_2，…，S_T，然后在各个子训练集上用信息增益算法挑选出对分类起重要作用的属性作为输入

神经元训练出个体成员网络，然后通过投票法进行集成。IG-Bagging 算法描述如下：

A 训练过程：

输入：训练集 $S=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$，其中 $x_i\in X, y_i\in Y=\{1,2,\cdots,K\}$；学习机 h；迭代次数 T；每个 Bag 的大小 d

(1) for t=1,2,…,T {

(2) S_t←从 S 中随机重采样 d 个样本

(3) W_j=InfomationGain(S_t) //计算样本各特征的信息增益值

(4) 利用 S_t 对 h 进行学习 h_t←h(S_t,W_j);}

//用选择的特征空间训练成员分类器

输出：h_t

B 测试过程：

输入：测试样本 x

输出：$f(x)=\arg\max_{y\in Y}\left(\sum_{i:h_t(x)=y}1\right)$ //多数投票法集成

设初始训练集 S 中有 m 个样本，进行可重复取样产生的新的训练集 S_t，大小 d=m，则 S 中有 36.8%的样本不包含在 S_t 中：$(1-\frac{1}{T})^T|S|\approx 0.368|S|$。对于没有被取样的样本，可以作为验证集。

IG-Bagging 与标准 Bagging 不同处在于各个子集独立地应用信息增益算法选取特征向量，在训练各个成员分类器时特征向量不是完全相同的，属性空间有部分重叠，这样达到了同时扰动属性和样本的目的，可以生成精确且差异度大的个体神经网络，提高集成的泛化能力。应用 IG-Bagging 进行 Neural Network 集成的具体流程见图 2。

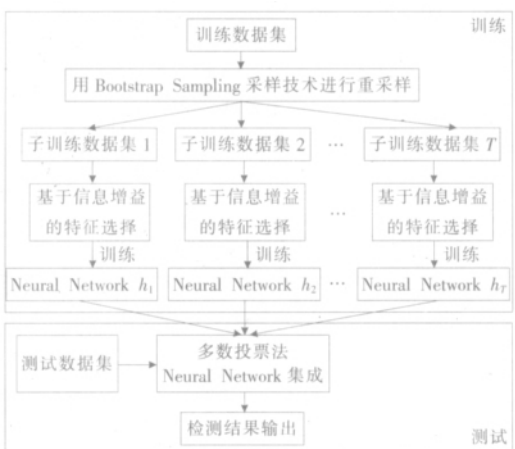

图 2 应用 IG-Bagging 对神经网络集成

4 实验测试

由于不像入侵检测那样有基准测试库（Benchmark Data Set），文献[5]研究的病毒代码集从网站 VX Heavens[12]获得，为便于对比，本文的病毒代码样本也从此处下载，共计 209 个（详见附录），均为 PE 格式。正常程序从操作系统平台中选取，我们选用的是 Windows 2000 Server 首次安装后，机器中的全部 PE 文件共 423 个。

实验测试中使用的神经网络是一个包含输入层、隐层和输出层三层结构的 BP 神经网络，隐层包含 10 个节点，传递函数为对数 S 型函数（logsig），输出层包括一个输出节点，传递函数为线性函数（Purelin）。训练时采用自适应梯度下降法（Traingda），并使用“尽早停止技术（Early stopping）”以防止过配，其参数设为 20，也就是在连续的 20 个训练步长中，如果验证数据集的预测误差没有下降，则停止进一步训练以防止过度训练。采用 5 次交叉验证的方法，通过 Bagging 训练出个体网络；为了提高训练效率，在训练时将变量归一化到[0,1]。个体网络训练和生成过程见图 2。所有实验均以 MATLAB6.5 的神经网络工具箱[13]为基础，通过编程实现。

在模式识别领域中 Receiver Operating Characteristics（ROC）曲线用于比较不同分类算法的性能。曲线下面的面积越大则算法分类性能越好越稳定。图 3 为集成 BP 网络与单个 BP 网络的 ROC 图。从直观上清楚地看出集成 BP 网络 ROC 曲线下方的面积要大于单个 BP 网络的面积，即说明集成网络的性能要优于单个 BP 分类器。这也正好支持了文献[6]的结论。文献[5]对基于 n-gram 的恶意代码检测取得了很好的效果，该文一共测试了 8 种分类器，其结果如图 4 所示，其中 Boosted J48 性能最优。本文设计的检测器性能与该文相当，也得出了与该文一致的结论，即集成的分类器性能要好于单一的分类器。

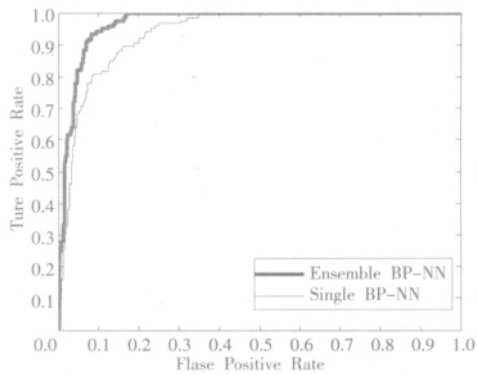

图 3 集成 BP 网络与单个 BP 网络的 ROC 图

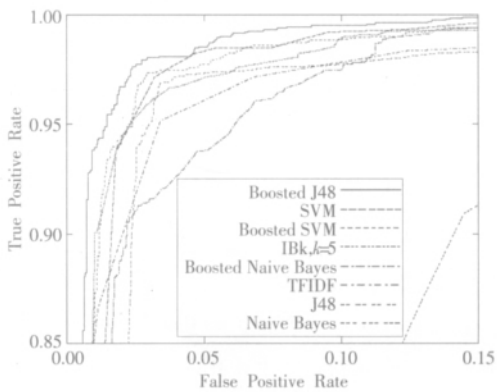

图 4 文献[5]的 ROC 图

5 结论

集成学习技术是机器学习的研究热点之一。本文结合 Bagging 算法，将基于信息增益的特征选择技术引入神经网络集成的构建中，将扰动训练数据和扰动输入属性结合起来，以生成精确且差异度大的个体神经网络。在此基础上尝试以集成的神经网络为模式识别算法来实现对计算机病毒代码的检测，

实验表明提出的检测方法具有较强的泛化能力和较高的精确率。

因为选择的特征向量是对程序代码静态分析得来的,故对多态病毒的检测不太有效。我们将考虑基于代码行为分析与静态分析相结合的方法来实现恶意代码的检测。另外,本系统是一个病毒代码检测器,只能隔离病毒而不能实现对病毒的清除,与其他 Anti - Virus 工具的集成是一个可行的解决方案。在我们的测试床中,将计划进行其他的基于机器学习算法的检测实验,并对更多类型的恶意代码如木马、蠕虫、间谍程序等进行测试,对各种方法的性能与开销进行选择和优化以期用于实际的产品中。(收稿日期:2007 年 1 月)

参考文献:

[1] Spinellis D.Reliable identification of bounded-length viruses is NP-complete[J].IEEE Transactions on information Theory, 2003, 49(1): 280-284.

[2] Tesauro G J, Kephart J O.Neural networks for computer virus recognition[J].IEEE Expert, 1996, 8: 5-6.

[3] Symantec Corporation.Understanding heuristics: symantec's blood hound technology[DB/OL].Symantec White Paper Series, Volume XXXIV.http: // www.symantec.com /avcenter /reference /heuristc.pdf.

[4] 何申,张四海,王煦法,等.网络脚本病毒的统计分析方法[J].计算机学报, 2006(6): 969-975.

[5] Kolter J Z, Maloof M A.Learning to detect malicious executables in the wild[C]//KDD'04: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2004: 470-478.

[6] Hansen L K.Salamon P.Neural network ensembles[J].IEEE Trans Pattern Analysis and machine Intelligence, 1990, 12(10): 993-1001.

[7] Krogh A, Vedelsby J.Neural network ensembles, cross validation, and active learning[C]//Tesauro D, Touretzky D, Leen T.Advances in Neural Information Processing Systems 7.Cambridge, MA: MIT Press, 1995: 231-238.

[8] Jurafsky D, James H.Speech and language processing[M].New York: Prentice-Hall, Inc, 2000.

[9] Kephart J, Arnold W.Automatic extraction of computer virus signatures[C]//Proceedings of the 4th Virus Bulletin International Conference, Abingdon, 1994: 178-184.

[10] Schapire R E.The strength of weak learnability[J].Machine Learing, 1990, 5(2): 197-227.

[11] Breiman L.Bagging predictors[J].Machine Learning, 1996, 24(2): 123-140.

[12] Vx heavens[EB/OL].http: //www.vx.netlux.org.

[13] Mathworks.Neural Network Toolbox User's Guide[M].ver:4. Ntick, Massachussets: The Mathworks, Inc., 2001.

附录:样本集中的恶意代码

(样本共 209 个,所有恶意代码均通过 Kaspersky Anti-Virus On-Demand Scanner 检测)

Win32.Adson.1559	Win32.HLLW.Xelif	Win32.Redart.2796
Win32.Aidlot	Win32.Hortiga.4800	Win32.Redemption.a
Win32.Akez	Win32.Htrip.a	Win32.Refer.2939
Win32.Alcaul.a	Win32.Htrip.c	Win32.Resur.a
Win32.Aldebaran.8365.a	Win32.Idele.2108	Win32.Revaz
Win32.Aliser.7825	Win32.Idyll.1468	Win32.Rever
Win32.Alma.2414	Win32.IKX	Win32.Rhapsody.2619
Win32.Andras.7300	Win32.Insom.1972	Win32.Riccy.a
Win32.AOC.3649.a	Win32.Intar.1151	Win32.Rivanon
Win32.Apathy.5378	Win32.InvictusDLL.101.a	Win32.Ruff.4859
Win32.Apparition.a	Win32.InvictusDLL.b	Win32.Rufoll.1432
Win32.Arianne.1052	Win32.Ipamor.a	Win32.Rutern.5244
Win32.Aris	Win32.Jethro.5657	Win32.Sadon.900
Win32.Artelad.2173	Win32.Kala.7620	Win32.Sandman.4096
Win32.Asorl.a	Win32.Kanban.a	Win32.Sankei.1062
Win32.Atav.1939	Win32.Kaze.2056	Win32.Santana.1104
Win32.Atav.2073	Win32.Kenston.1895.a	Win32.Savior.1680
Win32.Auryn.1155	Win32.Ketan	Win32.Score.3072.a
Win32.utoWorm.3072	Win32.Kiltex	Win32.Segax.1136
Win32.Awfull.3571	Win32.Klinge	Win32.Sentinel.a
Win32.Bakaver.a	Win32.KME	Win32.Seppuku.2763
Win32.Barum.1536	Win32.KMKY	Win32.Shaitan.3550
Win32.Bee	Win32.Knight.2350	Win32.Shan.1842
Win32.Beef.2110	Win32.Koru	Win32.Shown.538
Win32.Belial.2609	Win32.Kriz.3660	Win32.Silcer
Win32.Belial.a	Win32.Kuto.2058	Win32.Slaman.i
Win32.Belial.b	Win32.Ladmar.2004	Win32.Slow.8192.a
Win32.Belod.a	Win32.Lamchi.a	Win32.Small.1144
Win32.Bender.1363	Win32.Lamebyte	Win32.Spelac.1008
Win32.Benny.3219.a	Win32.Lamewin.1751	Win32.Spit.a
Win32.Bika.1906	Win32.Lamicho.a	Win32.Staro.1538
Win32.BingHe	Win32.Lanky.3153	Win32.Stepar.b
Win32.Blakan.2016	Win32.Lash.a	Win32.Stupid.a
Win32.Emotion.a	Win32.LazyMin.31	Win32.Taek.1275
Win32.Enar	Win32.Legacy	Win32.Tapan.3882
Win32.Enerlama	Win32.Levi.3236	Win32.This31.16896
Win32.Eva.a	Win32.Libertine.d	Win32.Thorin.b
Win32.Evar.3587	Win32.Magic.1590	Win32.Tinit.a
Win32.Evol.a	Win32.Matrix.817.a	Win32.Tirtas.5675
Win32.Evul.8192.b	Win32.Mauz.a	Win32.Tolone
Win32.Evyl.b	Win32.Mental	Win32.Tosep.1419
Win32.Fighter.b	Win32.Miam.5164	Win32.Trion.a
Win32.Flechal	Win32.Milen.3205	Win32.Ultratt.8152
Win32.Fosforo.a	Win32.Minit.b	Win32.Undertaker.4883.a
Win32.Freebid	Win32.MircNew	Win32.Vampiro.a
Win32.FunLove.4070	Win32.Mix.1852	Win32.VCell.3504
Win32.Gaybar	Win32.Mockoder.1120	Win32.VChain
Win32.gen	Win32.Mogul.6800	Win32.Velost.1186
Win32.Genu.c	Win32.Mooder.g	Win32.Viset.b
Win32.Ghost.1667	Win32.Mystery.2560	Win32.Vorcan
Win32.Ginra.3334	Win32.Nicolam	Win32.Vulcano
Win32.Ginseng	Win32.Noise.410	Win32.Wabrex.a
Win32.Giri.4937.b	Win32.Nox.2290	Win32.Wanhope.1834
Win32.Gloria.2928	Win32.Opdoc.1204	Win32.Weird.c
Win32.Glyn	Win32.Oporto.3076	Win32.Wide.b
Win32.Gobi.a	Win32.Orez.6291	Win32.Wit.a
Win32.Godog	Win32.Oroch.5420	Win32.Wolf.b
Win32.Halen.2593	Win32.Padic	Win32.Xorala
Win32.Haless.1127	Win32.Paradise.2168	Win32.Yasw.924
Win32.Hatred.a	Win32.Parite.b	Win32.Yerg.9412
Win32.Henky.504	Win32.Parvo	Win32.Younga.2384.a
Win32.Heretic.1986	Win32.Peana	Win32.Zawex.3196
Win32.Hidrag	Win32.Perrun.a	Win32.ZHymn.a
Win32.Highway.b	Win32.PGPME	Win32.ZHymn.Host
Win32.HIV.6386	Win32.Pilsen.4096	Win32.ZMist
Win32.HLL.Fugo	Win32.Positon.4668	Win32.Zombie
Win32.HLLC.Winatch	Win32.Projet.2404.b	Win32.ZPerm.a
Win32.HLLO.52736	Win32.Qozah.3361	
Win32.HLLP.Thembe	Win32.RainSong.4266	
Win32.HLLP.VB.a	Win32.Ramdile	
Win32.HLLP.Vedex.b	Win32.Razenya	