基于 AdaBoost和概率神经网络的入侵检测算法

陈春玲, 商子豪

(南京邮电大学 计算机学院, 江苏 南京 210003)

[摘要] 将 AdaBoost算法和概率神经网络结合, 提出了一种新的概率神经网络模型 ABPNN, 基于此模型提出一种新的入侵检测算法. 该算法对接收到的网络数据进行分析判断, 实现入侵方式的自动分类, 并且能对新的入侵行为进行分类和记忆. 实验证明该算法在入侵检测系统的检测率和误报率方面都有优越的性能表现.

[关键词] 入侵检测, 概率神经网络, AdaBoost, ABPNN

[中图分类号] TP 393 [文献标识码] A [文章编号] 1672-1292(2008)04-0021-04

Algorithm of Network Intrusion Detection Based on AdaBoost and PNN

Chen Chunling, Shang Zhao

(College of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract By combining AdaBoost algorithm with probabilistic neural network (PNN), a new probabilistic neural network (ABPNN) model is proposed. Based on this model, a new intrusion detection algorithm is suggested. This algorithm analyzes and estimates the network data received, realizes the acctomatic sorting of intrusion methods, and at the same time sorts and memorizes new types of intrusion methods. Experiments show that the proposed algorithm can get better performance in detection rate and alarm rate.

Key words intrusion detection, probabilistic neural network(PNN), AdaBoost, ABPNN

入侵检测(Intrusion Detection)是信息安全研究的重要分支. 它是对系统中未授权的访问或异常的现象、活动与事件进行审计、追踪、识别和检测的全过程. 目前入侵检测的方法和模型多种多样, 主要是统计方法、数据挖掘和专家系统. 它们各有各的优缺点, 但目的都是为了对数据进行分析, 提高入侵检测系统的准确率和运行效率.

基于 AdaBoost和概率神经网络的入侵检测算法就是在概率神经网络的基础上, 结合入侵检测的特点而提出的. 该算法提高了入侵检测系统的检测率, 降低了误报率. 实验表明该算法整体性能优越.

1 基于 AdaBoost和概率神经网络的入侵检测算法的基本原理

1.1 AdaBoost算法

对于人工智能的可学习性, Valiant提出了强学习算法和弱学习算法的概念[1], 即: 如果一个学习算法通过对一组样本的学习, 达到理想的识别率, 则称为强学习算法; 反之, 如果识别率仅好于随机猜测, 则称为弱学习算法. 但是, 对于一般的神经网络, 理想的强学习算法很难得到. 1990年, Shapire提出 Boosting算法[2], 并通过构造此方法证明: 可将一组弱学习算法提升为一个强学习算法. 1995年, Yoav Freund和 Robert E. Schapire又提出了 AdaBoost算法, 相对于 Boosting算法, 效率相当但更适于实际应用.

AdaBoost算法的基本目标为, 利用大量学习能力一般的弱分类器通过一定方法叠加起来, 通过样本训练, 构建一个分类能力更强的强分类器. 简言之, 通过合并许多"弱分类器"的输出以产生有效"投票委员会"的过程. 理论证明, 只要每个弱分类器的分类能力比随机猜测好, 当弱分类器的个数趋于无穷时, 强分

收稿日期: 2008-06-18
基金项目: 国家"863"计划(2006AA01Z219)资助项目.
通讯联系人: 陈春玲, 副教授, 研究方向: 软件技术及其在通信中的应用. E-mail: clchen@njupt.edu.cn

类器的出错率将趋于零. AdaBoost算法的这一特性对于弥补神经网络的不足具有很重要的意义.

AdaBoost算法的主要思想是:

首先, 给出任意弱学习算法和训练集 $\{(X_1, Y_1), \cdots, (X_i, Y_i), \cdots, (X_m, Y_m)\}$. 初始化时, 指定训练集的分布为 $1/m$, 即每个训练样本的权值都同为 $1/m$.

接着, 调用弱学习算法进行 T 次迭代, 每次迭代后, 按照训练结果更新训练集上的分布, 对于训练失败的样本赋予较大的权值, 使得下一次迭代更加关注这些样本, 从而得到一个预测函数序列 $h_1, h_2, \cdots, h_t$ 每个 h_t 也赋予一个权值, 预测效果好的, 其权值越大.

最后, 经过 T 次迭代后, 在分类问题中最终的预测函数 $H(x)$ 经"投票委员会"产生, 即通过代权值的投票法产生. 利用 $H(x)$ 可对新样本进行有效判断.

1.2　结合 AdaBoost改进的概率神经网络

PNN可以降低系统敏感并提高运算效率[3], 然而, MPNN 的精确性却不高甚至不如 GRNN. AdaBoost算法本身是通过改变数据分布来实现的, 它根据每次训练集中每个样本的分类是否正确, 以及上次的总体分类的准确率, 来确定每个样本的权值, 然后将每次训练得到的分类器 h_t 最后组合起来, 作为最后的分类器 $H(x)$. AdaBoost在准确度和效率方面都具有很大的优势, 使用 AdaBoost算法, 可以排除一些不必要的特征, 并将重点放在关键的特征上. 为了提高 MPNN 的识别精确度, 本文提出一种结合 AdaBoost算法提高概率神经网络的分类性能和运算效率的网络模型 ABPNN.

在 ABPNN中, AdaBoost算法用于生成 PNN分类器, 这些分类器都是从原始数据集中的不同类型的采样集中训练而来的. 每个分类器都根据循环过程产生一个 h_t, 直到误差 ε_t 超过一定标准才中止循环. 最后, 结合每个分类器的 h_t 而生成一个 h_{final}.

ABPNN的模型结构如图 1所示, ABPNN 模型主要包含两个模块: AdaBoost模块和 PNN 模块. AdaBoost模块中又包含两个子模块, 分别为 D_t 生成器和 H_t 组合器. D_t 生成器根据不同的分布 D_t 对原始数据集进行重新采样而获得新的数据集, 然后将这些新的数据集提供给 PNN 模块用于训练神经网络, 直到满足终止条件才停止这一过程. 针对不同的数据集而产生不同的分类器, 然后被 H_t 组合器组合处理.

ABPNN的处理过程要分模型训练和实际运行两种情况来描述. 在 ABPNN 模型训练时, 其具体处理包括两部分, 训练 PNN 子分类器和构建组合分类器 $H(x)$, 其处理流程如图 2所示. 在模型实际运行时, 其具体处理包括 PNN子分类器分类和组合分类器 $H(x)$ 决策, 处理流程如图 3所示.

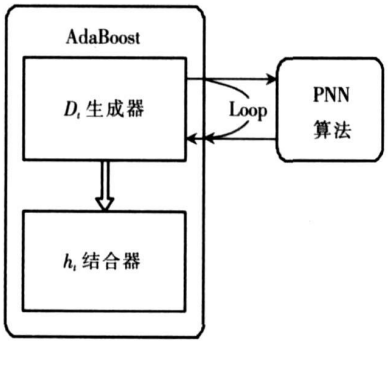

图 1　ABPNN 模型结构图

Fig.1　Structural chart of ABPNN model

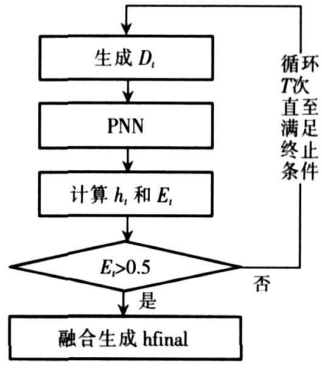

图 2　ABPNN 模型训练流程图

Fig.2　Training process of ABPNN model

1.2.1　ABPNN 模型的训练

假设训练样本集 $S = \{(\boldsymbol{X}_1, Y_1), \cdots, (\boldsymbol{X}_i, Y_i), \cdots, (\boldsymbol{X}_m, Y_m)\}$, 其中: $\boldsymbol{X}_i \in \boldsymbol{X}$ 为输入向量, $Y_i \in Y = \{1, 2, \cdots, k\}$ 代表分类结果, m 为输入样本数, k 代表分类数.

(1) 训练 PNN 弱分类器

AdaBoost算法要重复运行 PNN模块, 假设 T 为循环次数. 在每轮循环中, AdaBoost首先指定一个分布 D_t, D_t 用于确定新的样本如何进行分类并说明了训练集的每个可能的分类中包含的样本数目.

初始情况下, 对于样本集 $S = \{(\boldsymbol{X}_1, Y_1), \cdots, (\boldsymbol{X}_i, Y_i), \cdots, (\boldsymbol{X}_m, Y_m)\}$, 给予 S 的每个样本相同的权

值, 即: $D_t(i) = 1/m$, 这就产生了第一个分类器 h_1. 然后, 计算分类器 h_1、训练集合 S的分类误差 $\varepsilon_1 = \sum (h_1(X_1) \neq Y_1)\, D_t$, 并赋予分类器 h_1 权值 $\alpha_1 = 1/2\ln\{(1-\varepsilon_1)/\varepsilon_1\}$.

以后每轮循环中, 将会更新 S 的每个样本的权值, 对于前一轮分类错误的样本将被赋予较大的权值, 其数学表达式为:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta & h_t(x_i) = y_i \\ 1 & \text{其它} \end{cases}, \qquad (1)$$

其中, $\beta_1 = \varepsilon_1/(1-\varepsilon_1)$.

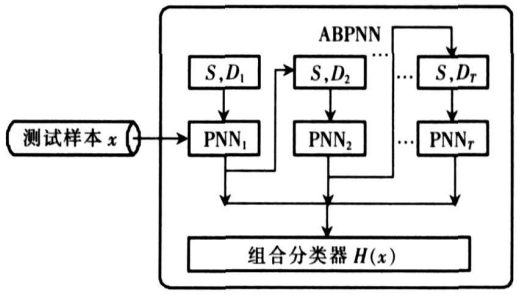

图 3 ABPNN 运行流程图

Fig.3 Running process of ABPNN model

从公式 (1) 我们可以得出计算 D_{t+1} 的方法. 如果 h_t 能对 $\boldsymbol{X}_i$ 正确分类, 则 $D_{t+1} = D_t \times \beta_t$, $\beta_t \in [0, 1]$; 否则, 对应权值不做改变. 然后, 这些权值 D_{t+1} 再被标准化常数 Z_t 相除, 即得新的 D_{t+1}. 利用 D_{t+1}, 更新训练样本集 S中各样本的权值, 并重新送入 PNN模块用于生成分类器 h_{t+1}: $\boldsymbol{X} \rightarrow Y$. 在满足最小训练误差的情况下, 即 $\varepsilon_t = \sum (h_t(\boldsymbol{X}_i) \neq Y_i)\, D_t$, 利用 h_t 可以对训练数据集 S 进行正确分类.

训练 PNN 分类器的过程将反复循环, 直到满足公式 (2) 的循环终止条件.

$$\varepsilon_t = \sum_{i:\, h_t(x_i) \neq y_i} D_t(i) < \frac{1}{2}, \qquad (2)$$

式中, $D_t(i)$ 为第 t轮循环中样本 i的权值. 如果 $\varepsilon_t > 1/2$, 则设 $T = t-1$并放弃此轮循环.

Fruend和 Schapire在文献[4] 中已经证明, 如果 h_t 的误差范围在小于 0.5的范围内, 那最终 h_{final} 的训练误差将接近零. 文献[4] 中提到一种更复杂的误差测量方法, 称为 pseudo-loss法, 其利用各分类的事后概率来用动态判定方式进行分类.

在此处理过程中, 容易正确分类的样本将获得较低的权值, 而易导致错误分类的样本将获得较高的权值. 这些样本权值表明了下一轮的训练中对应样本的优先级, 权值越高, 优先级越高. 这个过程每经过一次训练后, 被分类错误的事件其权值会一直增加, 也就是说, 经由这轮循环训练, 处理的焦点将集中在较难分类的事件上. 每轮循环构建一个新的分类器 h_t, 每个分类器 h_t 的构建则在于修改前一轮生成的分类器 h_{t-1} 的不足, 如此循环, 当新构建的分类器的误差 ε_t 超过随意猜测的误差 0.5时, 或已构建的分类器数目达到分类器数目上限时则停止循环. 整个循环结束后, 将生成一个分类器序列 $h_1, h_2, \ldots, h_T$.

(2) 构建组合分类器 $H(x)$

这一步中, 分类器序列 $h_1, h_2, \ldots, h_T$ 将构成一个分类器"委员会", "委员会"的决策结果将考虑各个分类器 h_t 的权值 α_t, 并根据公式 (3) 融合生成 $H(x)$.

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right). \qquad (3)$$

1.2.2 *ABPNN* 模型的实际运行

假设测试样本集 $\boldsymbol{T} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n\}$, 其中: $\boldsymbol{X}_i \in \boldsymbol{X}$ 为输入向量, n为输入样本数. $Y_i \in Y = \{1, 2, \ldots, k\}$ 代表分类结果; K 代表分类数.

(1) PNN 子分类器分类

测试集 T的样本 $\boldsymbol{X}_i$ 输入 ABPNN模型后, 首先被依次送入各 PNN 子分类器 h_1, h_t 对其进行分类, 得到结果 $Y(t) \in Y$. $Y(t)$ 表示子分类器 h_t 产生的结果. 这样就产生一个分类结果序列 $Y(1), Y(2), \ldots Y(T)$.

(2) 组合分类器 $H(x)$ 决策

上一步中得到的分类结果序列 $Y(1), Y(2), \ldots Y(T)$ 构成一个"投票委员会", 并根据类似于公式 (3) 得到组合分类器 $H(x)$ 及其分类结果 Y_i.

1.3 基于 *ABPNN* 的入侵检测算法

结合前面所提出的 ABPNN, 本文提出一种基于 ABPNN 的入侵检测算法, 其算法描述如下:

Step 1 初始化参数: 包括迭代次数 T、PNN的 spread、PNN 各层的节点数等.

Step 2 输入训练样本 $S = \{(\boldsymbol{X}_1, Y_1), \cdots, (\boldsymbol{X}_i, Y_i), \cdots, (\boldsymbol{X}_m, Y_m)\}$, 并初始化各样本权值 $\omega(1) = 1/m$.

Step 3 开始迭代训练, $t = 1, 2, \cdots, T$:

1) 根据输入样本及其权值 ω训练 PNN, 产生分类器 PNN_t, 得到分类结果 $Y_t = \{Y_t(1), Y_t(2), \cdots, Y_t(m)\}$;

2) 根据分类结果 Y_t 计算 PNN_t 的误差 ε_t;

3) if $\varepsilon_t \leqslant 0.5$ break;

4) 计算 PNN_t 的权值 $\alpha_t = 1/2\ln\{(1-\varepsilon_t)/\varepsilon_t\}$;

5) 更新样本权值 $\omega_{t+1}(i) = \omega_t(i) \times F(\varepsilon_t)$.

Step 4 根据公式(3)构造生成结合分类器 $\mathrm{PNN}_{\mathrm{final}}$.

2 实验结果与比较

为了验证算法的性能. 本文使用的实验数据是目前入侵检测领域通用的测试数据集: KDD CUP 99[5]. 使用 MATLAB对本文提出的算法进行仿真实验.

表 1和表 2是针对不同神经网络和攻击类型的检测结果.

从表 1可以看出, ABPNN 对正常数据、刺探攻击和拒绝服务攻击数据的检测性能要明显优于 BP和 RBF算法, 只是对获取根权限和远程攻击类型的检测略逊于 RBF算法, 但仍大大优于 BP算法.

从表 2可以看出, 在误报率方面, ABPNN 的性能要明显优于 BP和 RBF算法.

表 1 各种攻击类型的检测结果

Table 1 Instruction detection result of all kinds of attack

网络类型	检测率				
	刺探攻击	拒绝服务攻击	获取根权限	远程攻击	正常
BP	88.37	82.84	42.73	75.68	86.78
RBF	92.67	90.07	88.74	89.36	91.20
ABPNN	98.12	96.58	86.78	88.12	96.83

表 2 不同算法的误报率比较

Table 2 Compare alarm rate of different algorithm

检测算法	误报率				
	刺探攻击	拒绝服务攻击	获取根权限	远程攻击	正常
BP	5.86	3.28	12.35	10.02	22.6
RBF	3.42	4.89	3.97	4.27	6.79
ABPNN	0.19	0.23	0.03	0.08	4.81

3 结论

本文提出了一种结合 AdaBoost算法和概率神经网络的网络模型 ABPNN, 并基于此模型提出了一种新的入侵检测算法, 实验表明该算法能够对刺探攻击(Probing)和拒绝服务攻击(Dos)有很高的检测率, 并且算法的误报率也达到了比较理想的水平. 实验表明算法的综合性能优越, 适合应用与入侵检测系统中.

[参考文献](References)

[1] Valiant L G. A theory of learnable[J]. Communication of ACM, 1984, 27(11): 1134-1142.

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

[3] Zaknich A. Introduction to the modified probabilistic neural network for general signal processing applications[J]. IEEE Transactions on Signal Processing, 1998, 46(7): 1980-1990.

[4] Freund Y, Schapire R. Experiments with a new boosting algorithm[C]// Saitta L. Proc of the 13th Int'l Conf on Machine Learning. Austin: Morgan Kanfmann, 1996: 148-156.

[5] Liu B, Hsu W, Ma Y. Pruning and summarizing the discovered associations[Z]. KDD-99, 1999.

[6] 刘道群. 基于遗传神经网络的入侵检测模型的研究[D]. 重庆: 重庆大学, 2005.
Liu Daoqun. Study of IDS model based on genetic neural network[D]. Chongqing: Chongqing University, 2005. (in Chinese)

[7] Jackson D A, Yong C. Robust principal component analysis and outlier detection with ecological data[J]. Environmetrics, 2004, 15(2): 129-139.

[责任编辑: 刘 健]