Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection

John M. Fossaceca
16th Annual Systems Engineering Conference
October 28-31, 2013

Co-Authors:
Dr. Thomas A. Mazzuchi, Professor and Department Chair
Dr. Shahram Sarkani, Professor and Director of EMSE Off-Campus Programs
What I’ll Cover

• How this talk fits into Net Centric Operations/Interoperability
• Review of Network Intrusion Detection Systems Challenges
• Inspiration for my research
• Overview of Ensemble Decision Making
• Some Other Tools
  ▫ Boosting
  ▫ Support Vectors
  ▫ Kernel Methods & Multiple Kernel Methods
  ▫ Extreme Learning Machines
• My Approach to the Problem
• Preliminary Findings & Future Work
• Discussion & Questions
"Interoperability must be synergized with Information Assurance to assure obtaining the best of both" – Jack Zavin, Beyond Technical Interoperability, NDIA SE Conference, October 2012
Some Challenges in Network Intrusion Detection

- High False Alarm Rates
- High Dimensionality
- Data Imbalance
- Online Processing
- Zero Day & New Attacks (thwart Signature Based Systems)

For Anomaly Based “Learning” NIDS above are True Plus:

- Danger of Over-Fitting Data
- Opportunity for “Incremental Leaning” for “Concept Shifts”
- Opportunity to incorporate human expertise
- Opportunity to incorporate multiple algorithms
Net Centric View: Practical Considerations relative to Network Intrusion Detection

- Data from Disparate Sources
- Multiple Perspectives/Multiple Formats
- Real-time Data Fusion & Decision Optimization
- Dealing with Uncertainty
- Agility & Resilience
- Online Approach vs. Batch Processing
- How do we do all of this quickly?

Adapted from and added to “Beyond Technical Interoperability” Context for the Net Centric Operations & Interoperability Track @ 2011 NDIA SE Conference 26 – 27 October 2012, Jack Zavin NCO/I Chair
Inspiration for Research


Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection
NDIA 16th Annual Systems Engineering Conference
Decision Making with Ensembles

- **Intuitive Basis** for why Ensembles should provide a better decision than other approaches
- Good performance on training data does not predict good **generalization** performance (i.e. performance of the classifier on data not seen during training)
- **Too much data:** training different classifiers with different partitions of data, and combining their outputs using an intelligent combination rule is often a more efficient approach
- **Too little data:** resampling techniques resampling techniques can be used for drawing overlapping random subsets of the available data, each of which can be used to train a different classifier, creating the ensemble
- **Data Fusion:** data from different sources are combined to make a more informed decision

**Ensemble Learning:**
A machine learning paradigm where multiple learners are used to solve the problem

- The generalization ability of the ensemble is usually significantly better than that of an individual learner.
- Boosting is one of the most important families of ensemble methods.

---

Rachlin, Glen. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting” — Rachlin’s updated version presentation by Zhi-Hua Zhou, ICDM 2006 based on Freund & Schapire’s original paper.

Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection
NDIA 16th Annual Systems Engineering Conference
Boosting

- In 1990 it was proven that a *weak learner*, an algorithm that generates classifiers that can merely do better than random guessing, can be turned into a *strong learner* that generates a classifier that can correctly classify all but an arbitrarily small fraction of the instances.

- Creates an ensemble of classifiers by resampling the data, which are then combined by majority voting.

- **Basic Algorithm:**
  - Classifier $C_1$ is trained with a random subset of the available training data.
  - The training data subset for the second classifier $C_2$ is chosen as the most informative subset, given $C_1$. That is, $C_2$ is trained on a training data only half of which is correctly classified by $C_1$, and the other half is misclassified.
  - The third classifier $C_3$ is trained with instances on which $C_1$ and $C_2$ disagree.
  - The three classifiers are combined through a three-way majority vote.

- **AdaBoost:** is a more general version of the original boosting algorithm (1997 – AT&T Labs Research)

High Level Description of Adaboost

Weighted combinations of classifiers

• “Committee” decisions
  – Trivial example
  – Equal weights (majority vote)
  – Might want to weight unevenly – up-weight good experts

• Boosting
  – Focus new experts on examples that others get wrong
  – Train experts sequentially
  – Errors of early experts indicate the “hard” examples
  – Focus later classifiers on getting these examples right
  – Combine the whole set in the end
  – Convert many “weak” learners into a complex classifier

Adaboost – Block Diagram


Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection
NDIA 16th Annual Systems Engineering Conference
The AdaBoost Algorithm:

TrainAdaBoost(D, BaseLearn)
    For each example \( d_i \) in \( D \) let its weight \( w_i = 1/|D| \)
    Let \( H \) be an empty set of hypotheses
    For \( t \) from 1 to \( T \) do:
        Learn a hypothesis, \( h_t \), from the weighted examples: \( h_t = \text{BaseLearn}(D) \)
        Add \( h_t \) to \( H \)
        Calculate the error, \( \varepsilon_t \), of the hypothesis \( h_t \) as the total sum weight of the
        examples that it classifies incorrectly.
        If \( \varepsilon_t > 0.5 \) then exit loop, else continue.
        Let \( \beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t} \)
        Multiply the weights of the examples that \( h_t \) classifies correctly by \( \beta_t \)
        Rescale the weights of all of the examples so the total sum weight remains 1.
    Return \( H \)

TestAdaBoost(ex, H)
    Let each hypothesis, \( h_t \), in \( H \) vote for \( ex \)'s classification with weight \( \log(1/\beta_t) \)
    Return the class with the highest weighted vote total.

Weight each classifier and combine them:

\[ 0.33 \times + 0.57 \times + 0.42 \times \]

Combined classifier

Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection
NDIA 16th Annual Systems Engineering Conference
Support Vectors

Strengths of SVM:
• Training is relatively easy.
• No issues with local optima, unlike in neural networks.
• Scales relatively well to high dimensional data

Weaknesses:
• Requires selection of good kernel function
• CPU Intensive

Tutorial on Support Vector Machine (SVM) Vikramaditya Jakkula, School of EECS, Washington State University, Pullman 99164 - eecs.wsu.edu/~vjakkula/SVMTutorial.doc & Ian Witten: https://weka.waikato.ac.nz/dataminingwithweka/unit?unit=4&lesson=5
Kernels

General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: x \rightarrow \phi(x) \]
Non-Linear SVMs: Kernel Trick

- With this mapping, our discriminant function is now:

\[ g(x) = w^T \phi(x) + b = \sum_{i \in SV} \alpha_i \phi(x_i)^T \phi(x) + b \]

- No need to know this mapping explicitly, because we only use the dot product of feature vectors in both the training and test.

- A kernel function is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

\[ K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j) \]
Online Multiple Kernel Classification

• Online learning is designed to sequentially learn a prediction model based on iterative feedback and from prior runs

• Kernel learning aims to learn an effective kernel function for a given learning task from training data

• Kernel Learning (MKL) finds the optimal combination of multiple kernels to optimize learning performance

• **Online Multiple Kernel Classification** (OMKC) aims to learn multiple kernel classifiers and their linear combination simultaneously

• Due to online operations, both the optimal kernel classifiers and their linear combinations need to be learned **simultaneously**

• Solutions to kernel classifiers and their linear combinations are strongly correlated, making this a significantly more challenging problem than a typical online learning problem

Online Multiple Kernel Classification - Approach

Deterministic Algorithm for OMKC ($\text{OMKC}_{(D,D)}$)

1: INPUT:
   - Kernels: $k_i(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, $i = 1, \ldots, m$
   - Weights $w_i(1) = 1$, $i = 1, \ldots, m$
   - Discount weight $\beta \in (0, 1)$.
2: Initialization: $f^1 = 0$, $w^1 = 1$, $\theta^1 = \frac{1}{m}$
3: for $t = 1, 2, \ldots$ do
4:   Receive an instance: $x_t$
5:   Predict $\hat{y}_t = \text{sign}\left(\sum_{i=1}^{m} \theta_i \text{sign}\left(f^t_i(x_t)\right)\right)$
6:   Receive the class label: $y_t$
7:   for $i = 1, 2, \ldots, m$ do
8:     Set $z^t_i = I(y_t f^t_i(x_t) \leq 0)$
9:     Update $w^t_{i+1} = w_i(t) \beta z^t_i$
10:    Update $f^{t+1}_i(x) = f^t_i(x) + z^t_i y_t k_i(x_t, x)$
11:   end for
12: $\theta^{t+1}_i = \frac{w^t_i}{W^t}$, $i = 1, \ldots, m$, where $W^t = \sum_{i=1}^{m} w^t_i$
13: end for

### Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection

NDIA 16th Annual Systems Engineering Conference

**Table: Performance Results - OMKC**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Perceptron</th>
<th>Perceptron (*)</th>
<th>OM-2</th>
<th>OMKC(D,D,D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ionosphere</td>
<td>n=351, d=34, m=16</td>
<td>26.82 ± 1.63</td>
<td>22.07 ± 6.77</td>
<td>16.07 ± 1.42</td>
</tr>
<tr>
<td>Mistake</td>
<td></td>
<td>18.73 ± 1.23</td>
<td>17.41 ± 1.20</td>
<td></td>
</tr>
<tr>
<td>SV (#)</td>
<td></td>
<td>65.8 ± 4.3</td>
<td>128.5 ± 4.4</td>
<td></td>
</tr>
<tr>
<td>Time (s)</td>
<td></td>
<td>0.003 ± 0.000</td>
<td>0.113 ± 0.001</td>
<td></td>
</tr>
<tr>
<td>votes84</td>
<td>n=435, d=16, m=16</td>
<td>8.17 ± 0.73</td>
<td>9.45 ± 1.94</td>
<td>7.21 ± 0.68</td>
</tr>
<tr>
<td>Mistake</td>
<td></td>
<td>8.68 ± 0.62</td>
<td>3.93 ± 0.62</td>
<td>7.38 ± 0.56</td>
</tr>
<tr>
<td>SV (#)</td>
<td></td>
<td>37.8 ± 2.7</td>
<td>41.1 ± 4.2</td>
<td></td>
</tr>
<tr>
<td>Time (s)</td>
<td></td>
<td>0.004 ± 0.000</td>
<td>0.045 ± 0.002</td>
<td></td>
</tr>
<tr>
<td>wdbc</td>
<td>n=569, d=30, m=16</td>
<td>34.51 ± 1.82</td>
<td>12.29 ± 1.01</td>
<td>11.70 ± 1.01</td>
</tr>
<tr>
<td>Mistake</td>
<td></td>
<td>41.52 ± 3.70</td>
<td>41.70 ± 4.06</td>
<td></td>
</tr>
<tr>
<td>SV (#)</td>
<td></td>
<td>236.3 ± 21.0</td>
<td>237.3 ± 23.1</td>
<td></td>
</tr>
<tr>
<td>Time (s)</td>
<td></td>
<td>0.007 ± 0.000</td>
<td>0.065 ± 0.001</td>
<td></td>
</tr>
<tr>
<td>breast</td>
<td>n=683, d=9, m=16</td>
<td>26.73 ± 1.19</td>
<td>6.12 ± 0.79</td>
<td>4.86 ± 0.51</td>
</tr>
<tr>
<td>Mistake</td>
<td></td>
<td>41.90 ± 3.40</td>
<td>44.33 ± 3.88</td>
<td></td>
</tr>
<tr>
<td>SV (#)</td>
<td></td>
<td>286.1 ± 23.2</td>
<td>303.4 ± 26.4</td>
<td></td>
</tr>
<tr>
<td>Time (s)</td>
<td></td>
<td>0.008 ± 0.000</td>
<td>0.068 ± 0.001</td>
<td></td>
</tr>
<tr>
<td>australian</td>
<td>n=690, d=14, m=16</td>
<td>39.54 ± 1.51</td>
<td>38.04 ± 2.38</td>
<td>37.67 ± 1.20</td>
</tr>
<tr>
<td>Mistake</td>
<td></td>
<td>39.50 ± 2.70</td>
<td>39.62 ± 2.88</td>
<td></td>
</tr>
<tr>
<td>SV (#)</td>
<td></td>
<td>272.9 ± 10.4</td>
<td>273.4 ± 19.9</td>
<td></td>
</tr>
<tr>
<td>Time (s)</td>
<td></td>
<td>0.010 ± 0.001</td>
<td>0.091 ± 0.003</td>
<td></td>
</tr>
<tr>
<td>diabetes</td>
<td>n=768, d=8, m=16</td>
<td>44.14 ± 1.86</td>
<td>45.35 ± 2.18</td>
<td>33.69 ± 1.29</td>
</tr>
<tr>
<td>Mistake</td>
<td></td>
<td>45.18 ± 2.19</td>
<td>45.35 ± 2.18</td>
<td></td>
</tr>
<tr>
<td>SV (#)</td>
<td></td>
<td>339.0 ± 14.3</td>
<td>348.3 ± 16.7</td>
<td></td>
</tr>
<tr>
<td>Time (s)</td>
<td></td>
<td>0.012 ± 0.001</td>
<td>0.099 ± 0.006</td>
<td></td>
</tr>
<tr>
<td>fourclass</td>
<td>n=862, d=2, m=16</td>
<td>36.29 ± 1.09</td>
<td>35.92 ± 1.65</td>
<td>3.19 ± 0.38</td>
</tr>
<tr>
<td>Mistake</td>
<td></td>
<td>35.82 ± 1.56</td>
<td>35.92 ± 1.65</td>
<td></td>
</tr>
<tr>
<td>SV (#)</td>
<td></td>
<td>312.8 ± 9.4</td>
<td>309.6 ± 14.2</td>
<td></td>
</tr>
<tr>
<td>Time (s)</td>
<td></td>
<td>0.013 ± 0.001</td>
<td>0.092 ± 0.002</td>
<td></td>
</tr>
</tbody>
</table>

Brief Introduction to Extreme Learning Machines


Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection

NDIA 16th Annual Systems Engineering Conference
Brief Introduction to Extreme Learning Machines:

- **Basic SLFN Structure**

![Diagram of a Single Layer Feed Forward Network (SLFN)]

- **Hidden Node Output**
  \[ G(a_i, b_i, x) = g(a_i \cdot x + b_i) \]
  - \( a_i \): the weight vector connecting the \( i \)th hidden node and the input nodes.
  - \( b_i \): the threshold of the \( i \)th hidden node.

- **Single Layer Feed Forward Network (SLFN) Output**
  \[ f_L(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x) \]
  - \( \beta_i \): the weight vector connecting the \( i \)th hidden node and the output nodes.


Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection

NDIA 16th Annual Systems Engineering Conference
Brief Introduction to Extreme Learning Machines:

\[ G(a_i, b_i, x) = g(b_i \| x - a_i \|) \]

- \( a_i \): the center of the \( i \)th hidden node.
- \( b_i \): the impact factor of the \( i \)th hidden node.

**Single Layer Feed Forward Network (SLFN) Output**

\[ f_L(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x) \]

- \( \beta_i \): the weight vector connecting the \( i \)th hidden node and the output nodes.

Brief Introduction to Extreme Learning Machines:

$\mathbf{K}(\mathbf{x}, \mathbf{x}_i)$

$\Rightarrow$ SVM Connection to SLFN

$\beta_1 \quad \beta_i \quad \beta_L$

$n$ input Neurons

$L_s$ output Neuron

$\mathbf{K}(\mathbf{x}, \mathbf{x}_L)$

$\Rightarrow$ Kernel Mapping

$\Rightarrow$ Kernel Based Hidden Neurons

Support Vector Machine Decision Function:

$$ f_L(x) = \text{sign} \left( \sum_{s=1}^{L_s} \alpha_s t_s \mathbf{K}(\mathbf{x}, \mathbf{x}_s) + b_i \right) $$

$\beta_i$ : the weight vector connecting the $i$th hidden node and the output nodes.

SVM Optimization Formula

$$ \text{Minimize: } Q_p = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{L} \epsilon_i $$

s.t.: $t_i (\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) \geq 1 - \epsilon_i, \forall i$

$\epsilon_i \geq 0, \forall i$

Brief Introduction to Extreme Learning Machines:  

**Basic ELM Structure**

![Diagram of ELM structure with nodes and connections]

**Hidden Node Output**

\[ G(a_i, b_i, x) = g(a_i \cdot x + b_i) \]

- \(a_i\): the weight vector connecting the \(i\)th hidden node and the input nodes.
- \(b_i\): the threshold of the \(i\)th hidden node.

**Extreme Learning Machine (ELM) Output**

\[ f_i(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x) = t_j \text{ for } j = 1, \ldots, N \]

- \(\beta_i\): the weight vector connecting the \(i\)th hidden node and the output nodes.

Equivalent to:

\[ H\beta = T \]

A Matrix Equation that can be solved With a simple Pseudo Inverse!

---

ELM Operation

Given a training set $\mathcal{N} = \{(x_i, t_i) | x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i = 1, ..., N\}$ with hidden node output function $G(a_i, b_i, x)$, and the number of hidden nodes $L$:

1 - Assign randomly hidden node parameters $(a_i, b_i)$, $i = 1, \cdots, L$.
2 - Calculate the hidden layer output matrix $H$.
3 - Calculate the output weight: $\beta: \beta = H^+T$.

where $H^+$ is the Moore-Penrose generalized inverse of hidden layer output matrix $H$.

NOTE:

- The hidden node parameters $a_i$ and $b_i$ are independent of the training data and each other.
- Unlike conventional learning methods which MUST see the training data before generating the hidden node parameters, ELM generates the hidden node parameters prior to seeing the training data.
- Unlike traditional gradient-based learning algorithms which only work for differentiable activation functions, ELM works for all bounded non-constant piecewise continuous activation functions.

Advantages of Extreme Learning Machine Approach versus SLFN & SVM

- Need to choose different algorithms & architectures in SLFN (i.e. number of hidden layers)
- Manual Parameter Tuning is Required
- Danger of Over-fitting the Data
- Possibility of Local minima
- CPU Intensive/Time consuming

The ELM learning algorithm is simpler than many other popular learning algorithms: neural networks and support vector machines.

Application of ELM to Network Intrusion Detection – Cheng et. al.

- SVM and ELM are compared (SVM is the Benchmark).
- A Data processing script is used to convert the raw TCP/IP dump data into machine readable form.
- Training Phase: SVM and ELM are trained on normal data and different types of attacks.
- For the binary classification case, the data has 41 features and falls into 2 classes: normal and attack; for the multi-class.
- For the multi-classification case, the data has 41 features and falls into 23 classes: normal and 22 types of attacks. 4 main categories (DoS, U2R, R2L, Probe)
- Testing phase: SVM and ELM are used to predict the type of each data point in the testing dataset, and their individual performances were compared.

**NOTE:** Both SVM and ELM cannot process symbolic data, so a method was employed to convert symbolic data into continuous data.

Application of ELM to Network Intrusion Detection – Cheng et. al.

Multi-Class Classification Performance Results

My Approach to the Problem

• Using a Systems Engineering approach to combine a number of successful techniques in order to optimize the overall decision process

• Basic Requirements:
  ▫ Multiple Algorithm Combination (i.e. “Multiple Perspectives”, “Best of All Worlds”)
  ▫ Online Processing (Suitability for Net Centric Operation)
  ▫ Provision for Data Imbalance
  ▫ Reduce False Alarms
  ▫ Allow for Human Input (i.e. Tuning)
Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection

NDIA 16th Annual Systems Engineering Conference
Very Preliminary Findings on UCI Machine Learning Data Set

<table>
<thead>
<tr>
<th></th>
<th>Ionosphere</th>
<th>OMKC$_{D,D}$</th>
<th>Batch ELM Voting</th>
<th>Online ELM Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mistake %</td>
<td>16.07</td>
<td>1.89</td>
<td>5.89</td>
<td></td>
</tr>
<tr>
<td>Range</td>
<td>± 1.42</td>
<td>± 1.28</td>
<td>± 2.74</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Breast Cancer</th>
<th>OMKC$_{D,D}$</th>
<th>Batch ELM Voting</th>
<th>Online ELM Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mistake %</td>
<td>4.86</td>
<td>2.45</td>
<td>2.91</td>
<td></td>
</tr>
<tr>
<td>Std Dev</td>
<td>± 0.51</td>
<td>± 0.63</td>
<td>± 1.03</td>
<td></td>
</tr>
</tbody>
</table>

*Note: 100 Node ELM with Radial Basis Activation Function*
Planned Future Work

- Pre-processing of KDD Cup ‘99 Data Set
- Analysis and Pre-processing of a Secondary Intrusion Data Set
- Incorporation of scheme for addressing Imbalanced Data issues (e.g. W-ELM, SMOTE)
- Optimal Feature Selection
- Incorporation of Human Expert in the Loop
Discussion/Q & A

John Fossaceca Contact Information
Ph.D. Student, George Washington University
VP, Technology & Product Management
Ultra Electronics, 3eTI

- jfossaceca@gwu.edu
- phone: +1 301.944.1359
- Company website: www.ultra-3eTI.com

Thank You!