Intrusion Detection with Neural Networks – Combination of Self-Organizing Maps and Radial Basis Function Networks for Human Expert Integration

Timo Horeis

Email
horeis@fmi.uni-passau.de

1 Introduction

With the increasing number of computers being connected to the Internet, the security of network systems becomes more and more important. This importance is underlined by the recent largest and most sophisticated attack (October, 2002) against the 13 official DNS root servers [1], which provide the primary road map for almost all Internet communications. Among other techniques such as authentication methods, data encryption, or firewalls, the use of Intrusion Detection Systems (IDS) is essential to protect against attacks.

Neural Networks are tolerant of imprecise data and uncertain information. With their ability to generalize from learned data they seem to be an appropriate approach to Intrusion Detection. Current research in the area of Intrusion Detection based on Neural Networks shows encouraging results [2]. However, any successful intrusion on a single host can question the data integrity of the host and the security of the whole computer network. To detect unknown attacks and even subtle variations of known attacks is a problem that is extremely hard to address.

A testbench of a Neural Network based IDS is described here. To detect subtle variations of known attacks or even unknown attacks this IDS integrates a human expert during the application phase. The system describes here combines the advantages of Radial Basis Function (RBF) Networks[3] and Self-Organizing Maps (SOM) [4]. It is able to support an human expert during the classification process by visualizing the relationship of suspicious data to known attacks.

The evaluation of this approach will be based on data sets provided by the Defense Advanced Research Projects Agency (DARPA) IDS evaluation in 1998 [5].

2 Research Work

Our research work consists of a testbench of a Neural Network based IDS. This work contributes to the research fields of Neural Network and Intrusion Detection. An overview of the state of the art in these fields is given in section 3.

The IDS described here is based on an Bayesian approach for pattern classification. The output of the Neural Network is interpreted as approximations to posterior probabilities for class membership. Section 4 contains an overview of Bayesian classification.

Our Neural Network based testbench uses a combination of RBF Networks and SOM, further referred to as RBF-SOM. RBF-SOM are three layered feed-forward networks with an input, map and output layer. The neurons of the map layer have radial basis activation functions. The activation of the output layer will be calculated by a linear combination of the activation of the map and input neurons. Section 5 contains a detailed formal definition of RBF-SOM.

According to RBF Networks RBF-SOM can be trained with a combination of unsupervised and supervised training. An appropriate training algorithm is described in section 6.

RBF-SOM and RBF Networks evaluated in our work are evolved by an genetic algorithm. Genetic algorithms can be seen as an approach to stochastic optimization. The genetic algorithm used in our work applies feature selection and model optimization to find "good enough" trained Neural Networks. Section 7 contains a short overview.

The evaluation of our research work is based on data sets provided by the DARPA IDS evaluation in 1998. An overview of the DARPA IDS evaluation can be found in section 8. To limit the effort of the research work only six attack types (Back, Dict, Guest, Ipsweep, Nmap, Warezclient) and only TCP/IP header information are considered.

The classification ability of RBF-SOM is compared to RBF Networks. The comparison is based on an existing database containing features extracted from the DARPA IDS evaluation in 1998. Neural Networks are evolved by a genetic algorithm and results are contained in section 9.

To classify input data which cannot be classifed automatically with a satisfying reliability, a human expert is integrated during the application phase. To measure the reliability of patterns propagated by a neural network four different functions, further referred to as score function, are proposed in section 10. All four score functions are evaluated based on data provided by the DARPA IDS evaluation in 1998.

3 State of the Art

Intrusion Detection can be divided into two main categories: Misuse Detection and Anomaly Detection. Misuse Detection is trying to discover intrusions by searching for distinguishing patterns or signatures of known attacks, whereas Anomaly Detection is based on the assumption that intrusive behavior deviates significantly from a previously learned normal behavior. Network paradigms which are trained with a supervised training procedure (e.g. RBF Networks) are often used for Misuse Detection. Network paradigms which are trained with an unsupervised learning procedure (e.g. SOM) are often used for Anomaly Detection. Lots of other distinguishing features for IDS, for example the classification into host-based and network-based systems, are set out in [6]

Since the first steps in Intrusion Detection research have been made in 1972 by Anderson [7], various IDS have been developed [6]. The most comprehensive evaluations of IDS performed to date were supported by the DARPA in 1998 and 1999 [5]. A relatively simple network architecture and background traffic have been used for the first DARPA IDS evaluation in 1998. More than 300 instances of 38 attack types have been launched against UNIX and Linux hosts.

The data sets of the DARPA IDS evaluation in 1998 serve as a basis for research in the field of intrusion detection at the University of Passau. Research related includes automated feature extraction [8] and a comparison of different Neural, Fuzzy and Neuro-Fuzzy paradigms for intrusion detection [2].

As mentioned above, the system described here combines the advantages of RBF Networks and Self-Organizing Maps. RBF Networks model hypherspherical clusters in the input space. In contrast to other neural paradigms they can be trained with a combination of unsupervised and supervised training. RBF Networks have led to best classification results in comparison to Multilayer Perceptrons, Classifying Fuzzy-k-Means and NEFCLASS systems for intrusion detection [2]. SOM are topology-preserving mappings, generally from a high-dimensional input space on a lower-dimensional map. Typically they have a rectangular, fully connected, fixed map topology and can be used to visualize relationships between input data.

Jirapunimm et. al[9] combine SOM and MLP trained with RPROP, an accelerated version of Backpropagation, for Intrusion Detection. SOM is used as a preprocessing stage and its output is used as input for MLP. The combined network can be seen as a feed-forward 5-layered neural network. The first layer is the input layer, the second the map layer of the SOM and the three succeeding layers the layers of the MLP. The network is trained with a combination of unsupervised SOM training and supervised RPROP training. The network is able to visualize relationships between patterns through the SOM part.

Growing Cell Structures (GCS) [10] can be seen as an extension of SOM. GCS eliminate some drawbacks of SOM (e.g. fixed number of map neurons specified by the user). A combination of RBF Networks and GCS with a supervised learning procedure is presented in [10]. To determine where new map neurons should be inserted, the supervised learning procedure is specifically adapted to a combination of RBF Networks and GCS. Therefore the proposed learning procedure cannot be used for a combination of RBF Networks and SOM.

Girardin[11] applies SOM for a visual approach to analyze network data to support the decision process of a human expert in the field of intrusion detection. A SOM is trained with an unsupervised training algorithm and no prior knowledge of the data being analyzed is needed. The intrusion detection system presented is not able to make classifications but Girarding emphasizes that the integration of an human expert allows to detect new sophisticated and structured attacks.

4 Classification

Classification is the process of assigning a class to an event according to a set of measurements from this event, called patterns. Using a Bayesian approach for classification uncertainty can be described by probability density functions. Given a finite number of patterns and their corresponding class, class-conditional density functions can be modeled. This in turn can be used with Bayes' Theorem to estimate approximations to posterior probabilities, which can be used to make a classification[3].

Using a Bayesian approach the classification process has two seperate stages[3]. The first is inference, where data is used to determine values for posterior probabilities. In the second stage referred to as decision making, those probabilities are used to make decisions such as assigning a new pattern to one of the classes.

4.1 Inference

Every pattern $\mathbf{x}$ is element of a feature space $\mathcal{L}$ with dimension d. The set $\mathcal{C} = \{C_0, ..., C_{n-1}\}$ contains all classes which a pattern can be classified to. Every pattern is assigned to exactly one class. A function

$$\zeta : \mathcal{L} \to \mathcal{C},$$

is called *Classifier on* $\mathcal{L}$. A classifier provides a rule for assigning each point of the feature space to one of the classes $C_i \in \mathcal{C}$.

Using a Bayesian approach uncertainty is described by probabilities and probability density functions. A probability density function p($\mathbf{x}$) specifies the probability of the variable $\mathbf{x}$ lying in the interval between any two points $\mathbf{x} = a$ and $\mathbf{x} = b$ and is given by

$$P(\mathbf{x} \in [a, b]) = \int_a^b p(\mathbf{x}) d\mathbf{x}$$

The function $p(\mathbf{x})$ is normalized such that $P(\mathbf{x} \in [a, b]) = 1$, if the interval $[a, b]$ corresponds to the overall feature space. Note that we use upper-case letters for probabilities and lower-case letters for probability density functions. For classification problems we are interested in posterior probabilities. Posterior probabilities $P(C_k|\mathbf{x})$ represent the probability of the event belonging to class C_k given the pattern $\mathbf{x}$. Using Bayes' Theorem

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)P(C_k)}{p(\mathbf{x})}$$

we can express posterior probabilities in terms of prior probabilities together with the class-conditional density functions for each class C_k. The prior probabilities $P(C_k)$ can be estimated from the proportions of the training data which fall into each of the classes. Similarly the class-conditional density functions could be approximated using e.g. histograms. The probability density function $p(\mathbf{x})$ plays the role of a normalization factor. We can express the prior probability density function by

$$p(\mathbf{x}) = \sum_{k=0}^{n-1} p(\mathbf{x}|C_k)P(C_k)$$

which ensures that the posterior probabilities sum to unity

$$\sum_{k=0}^{n-1} P(C_k|\mathbf{x}) = 1$$

4.2 Decision Making

As mentioned above, a classifier provides a rule for assigning each point of the feature space to one of the classes C_k. We can therefore regard the feature space as being divided up into n decision regions $R_0, ..., R_{n-1}$. Note that each of these regions need not be continguous, but may itself be divided into several disjoint regions, all being associated with the same class. The boundaries between these

decision regions are known as decision boundaries. A point of the feature space is falling into the decision region R_k if the point is mapped to class C_k by the classifier.

$$\mathbf{x} \in R_k \Leftrightarrow \zeta(\mathbf{x}) = \mathcal{C}_k$$

The probability of a new pattern being correctly classified is given by

$$P(correct) = \sum_{k=0}^{n} P(\mathbf{x} \in R_k | C_k) P(C_k) = \sum_{k=0}^{n} \int_{R_k} p(\mathbf{x}|C_k) P(C_k) d\mathbf{x}$$

This probability is maximized by choosing R_k so that each $\mathbf{x}$ is assigned to the class for which the integrand is a maximum. This is equivalent to :

$$\forall_{j \neq k} : p(\mathbf{x}|C_k)P(C_k) > p(\mathbf{x}|C_j)P(C_j) for all j \neq k$$

5 Formal Definition of RBF-SOM

RBF-SOM can be seen as a combination of RBF Networks and SOM. To be able to apply research results valid for RBF Networks RBF-SOM are presented here as an extension to RBF Networks. The formal definition is based on[12].

RBF Network are three layered neural networks with an input, hidden and output layer. Consecutive layers are totally connected, furthermore the input and output layer can also be totally connected (so called shortcut connections). The net input of the input layer is the same as the external input (input pattern). The hidden layer uses the distance between weight and input vector as net input. The output layer uses a linear combination as net input. Neurons of the input layer use the identical function as activation function. Neurons of the hidden layer use radial symetric basis functions and neurons of the output layer sigmoid functions (generally the identical function) as activation function, respectively.

1. A set of neurons $\mathcal{U} = \mathcal{U}_I \cup \mathcal{U}_H \cup \mathcal{U}_O$ with $\mathcal{U}_I \neq \emptyset$, $\mathcal{U}_H \neq \emptyset$ and $\mathcal{U}_O \neq \emptyset$. $\mathcal{U}_I$, $\mathcal{U}_H$ and $\mathcal{U}_O$ pairwise disjunct. $\mathcal{U}_I$ is further referred to as input layer, $\mathcal{U}_H$ as hidden layer or map layer and $\mathcal{U}_O$ as output layer.
2. A scalar $w_{(i,j)}^{(n)} \in \Re$ is assigned to each pair of neurons $(i,j) \in \mathcal{U}_m \times \mathcal{U}_n$ with $m \neq n$ and $m \in \{I, H\}$ and $n \in \{H, O\}$. The scalar is also called weight of the connection between neuron i in layer m and neuron j in layer n. The neurons and the connections together define the structure of the neural network.
3. For every neuron of the input layer $j \in \mathcal{U}_I$ the j-th external input of the network for pattern k is defined by $x_j(k) \in \Re$. For every neuron of the output layer $j \in \mathcal{U}_O$ the j-th external output of the network for pattern k is defined by $y_j(k) \in \Re$.
   The vector $\mathbf{x}(k) =^{def} (\mathbf{x}_1(k), ..., \mathbf{x}_{|\mathcal{U}_I|}(k))$ is called input pattern and the vector $y(k) =^{def} (y_1(k), ..., y_{C|\mathcal{U}_O|})$ is called output pattern of the network.
4. A propagation function $s_j^{(l)}(k)$ is assigned to every neuron $j \in \mathcal{U}$ to calculate its net input (with $l \in \{I, H, O\}$ and $j \in \mathcal{U}_l$).

   - For each neuron $j \in \mathcal{U}_I$:

   $$s_j^{(I)}(k) = x_j(k)$$

   - For each neuron $j \in \mathcal{U}_H$:

   $$s_j^{(H)}(k) = \| \mathbf{x}(k) - w_j^{(H)} \|$$

   - For each neuron $j \in \mathcal{U}_O$:

   $$s_j^{(O)}(k) = \sum_{i \in \mathcal{U}_I} w_{(i,j)}^{(O)} \cdot a_i^{(I)}(k) + \sum_{i \in \mathcal{U}_H} w_{(i,j)}^{(O)} \cdot a_i^{(H)}(k) + w_{(B,j)}^{(O)}$$

$\| \cdot \|$ is the length of a vector. The length of a vector is defined as $\| x \| = \sqrt{\langle x, x \rangle}$ with $\langle \cdot, \cdot \rangle$ being the standard scalar product in $\Re$. $w^{(O)}_{(B,j)}$ is called bias. $w^{(H)}_j =^{def} (w^{(H)}_{(1,j)}, w^{(H)}_{(2,j)}, ..., w^{(H)}_{(|\mathcal{U}_I|,j)}) \in \Re^{|\mathcal{U}_I|}$ is called the prototype or map vector

5. An activation function $a : \Re \to \Re$ is assigned to every neuron $j \in \mathcal{U}$.

   - For each neuron $j \in \mathcal{U}_I$:

   $$a^{(I)}_j(k) = s^{(I)}_j(k) = x_j(k)$$

   - For each neuron $j \in \mathcal{U}_H$:

   $$a^{(H)}_j(k) = \phi_j(s^{(H)}_j(k), p_j)$$

   - For each neuron $j \in \mathcal{U}_O$:

   $$a^{(O)}_j(k) = \sigma(s^{(O)}_j(k)) = y_j(k)$$

   $a^{(l)}_j(k)$ is called activation of neuron j in layer l for pattern k. Thereby σ is a generally sigmoid activation function and ϕ_j a basis function (generally a radial symetric basis function). ϕ_j has an additional parameter p_j, further referred to as radius of the basis function.

According to SOM a D-dimensional (generally 1- or 2-dimensional) a topology is defined on the hidden layer. A coordinate vector is assigned to every neuron of the hidden layer. This enables the RBF-SOM to generate a topology preserving mapping during the training process. The hidden layer of RBF-SOM is further called map layer.

1. A D-dimensional coordinate-vector $h_j \in \Re^D$ is assigned to every neuron $j \in \mathcal{U}_H$ of the map layer. The coordinate vector defines the position of the map neuron j on the map (D-dimensional space). In addition the topology depends on the neighborhoud, distance and radius function defined in the next chapter, containing the training algorithm for RBF-SOM.

5.1 RBF-SOM Networks for classification

As mentioned above, the Bayesian approach to statistical pattern recognition is to model the class-condition density function and the prior probabilities seperately. Using Bayes' Theorem they can be combined to give posterior probabilities which in turn can be used to make classifications. An alternative approach is to estimate the posterior probabilities directly. The outputs of Neural Networks, in this case RBF-SOM networks, can be interpreted as approximations to posterior probabilities. This section is based mainly on[3].

Using a pool of M basis functions, being labelled by an index j, we can write :

$$p(\mathbf{x}|C_k) = \sum_{j=0}^{M-1} p(\mathbf{x}|j)P(j|C_k)$$

An expression for the unconditional density $p(\mathbf{x})$ can be found by summing over all classes

$$p(\mathbf{x}) = \sum_{k=0}^{n-1} p(\mathbf{x}|C_k)P(C_k) = \sum_{j=0}^{M-1} p(\mathbf{x}|j)P(j)$$

where we have defined prior probabilities for the basis functions given by

$$P(j) = \sum_{k=0}^{n-1} P(j|C_k)P(C_k)$$

For classification purposes we are interested in posterior probabilities of class membership. These can be obtained by substituting the expressions into Bayes' Theorem to give

$$P(C_k|\mathbf{x}) = \frac{\sum_{j=0}^{M-1} P(j|C_k)p(\mathbf{x}|j)P(C_k)}{\sum_{i=0}^{M-1} p(\mathbf{x}|i)P(i)} = \frac{\sum_{j=0}^{M-1} P(j|C_k)p(\mathbf{x}|j)P(C_k)}{\sum_{i=0}^{M-1} p(\mathbf{x}|i)P(i)} \frac{P(j)}{P(j)} = \sum_{j=0}^{M-1} w_{(j,k)}^{(O)} \phi_j(\mathbf{x})$$

The expression represents a radial basis function network, in which the normalized basis functions are given by

$$\phi_j(\mathbf{x}) = \frac{p(\mathbf{x}|j)P(j)}{\sum_{i=0}^{M-1} p(\mathbf{x}|i)P(i)} = P(j|\mathbf{x})$$

and the second layer weights are given by

$$w_{(j,k)}^{(O)} = \frac{P(j|C_k)P(C_k)}{P(j)} = P(C_k|j)$$

6 Training of RBF-SOM

As mentioned above an RBF-SOM can be seen as a combination of an RBF network and a SOM. The training of an RBF-SOM can be easily adapted from the training of RBF networks and SOM. The training is a two stage process of unsupervised and supervised training.

1. Stage: Unsupervised training of the weights between input and map layer. This training stage can be adapted from the learning algorithm of SOM. This stage creates a topology preserving mapping from input space to an e.g. two dimensional map space.
2. Stage: Supervised training of the weights between map and output layer (e.g. singular value decomposition). This training stage can be adpated from the learning algorithm of an RBF network.

RBF Networks are usually trained with a three stage process. The first two stages are similiar to the stages mentioned above. In a third stage the weights between the input layer and the hidden layer and the radi are further optimized using non-linear optimization, e.g. backpropagation. This stage is not used for RBF-SOM, because the training of the weights may destroy the topology preserving mapping created in stage one.

6.1 First stage of the training process

In the first stage of the learning process the weights between the input and the map layer are adjusted. The algorithm is identical to the algorithm described in [4] for a self-organizing map. The algorithm can be seen as an unsupervised learning algorithm. For the definition of the training algorithm we need the following definitions based on [12] :

Radius Function of RBF-SOM

A radius function of an RBF-SOM is a function $\Re^D \rightarrow \Re_{\geq}$ with D generally being 2 (two-dimensional map layer). A radius function measures the distance of a point in $\Re^D$ to the origin of the coordinate system with the help of a norm.

Distance Function of RBF-SOM

A distance function of an RBF-SOM is a monotonically decreasing function $\Re_{\geq} \rightarrow \Re_{\geq}$. A distance function is used to model the decreasing neighborhood of neurons with progressing time.

Neighbourhod Function of RBF-SOM

A neighboorhood function of an RBF-SOM is a function $\Re_{\geq} \times \Re_{\geq} \rightarrow \Re$. A neighboorhood function maps a radius r (given by the output of a radius function) and a distance d (given by the output of a distance function) to the degree of neighborhood.

Learning Rate Function of RBF-SOM

A learning rate function $\eta : \Re_{\geq} \rightarrow [0, 1]$ is a monotonically decreasing function with $lim_{t\rightarrow\infty}\eta(t) = 0$. A learning rate function is used to model the decreasing of the learning rate by time.

Winner neuron

The map neuron $j' \in \mathcal{U}_H$ with the lowest net input is called winner neuron. The winner neuron is defined by

$$j' = arg(min_{j \in \mathcal{U}_H}(s_j^{(h)}(k))$$

1. Initialise all weights $w_{(i,j)}^{(l)}$ with $l \in \mathcal{U}_H \cup \mathcal{U}_O$ of the RBF-SOM with 0.
2. For each pattern k of the unsupervised learning process execute the following steps :
   (a) Use $\mathbf{x}(k)$ as external input.
   (b) Propagate the input and determine the net input of each neuron $j \in \mathcal{U}_H$ of the map layer.
   (c) Determine the winner neuron $j' \in \mathcal{U}_H$ and the coordinate vector h_j of the winner neuron.
   (d) Determine for each neuron $j \in \mathcal{U}_H$ the degree of neighbourship to the winning neuron. The deegre of neighbourship is determined by the radius, distance and neighboorhood function in the following way :

$$\lambda_{k,p}(j) = f(r_{norm,D}(h_j - h_{j'}), d(p))$$

   (e) Modify each weight vector $w_j^{(H)}$ for the pattern k in the following way :

$$\nabla_k w_j^{(H)} = (\mathbf{x}(k) - w_j^{(H)}) \cdot \eta(p) \cdot \lambda_{k,p}(j)$$

3. Test, whether a termination condition is met. If this is true, the learning process has ended successfully, otherwise continue the learning process from step 2.

6.2 Second stage of the training process

In the second stage of the learning process the weights between the map and the output layer are determined. The algorithm is identical to the algorithm described e.g. in [3] for RBF Networks. The algorithm uses e.g. singular value decomposition to determine the weights.

The weights between input and map layer were adjusted in stage one. To formulate the learning algorithm we need the following definitions.

Target vector The vector $t_j = (t_j(1), t_j(2), ..., t_j(|L|))^T$ is called target vector which contains the j-th component of each target pattern.

Weight vector The vector $w_j^{(O)} = (w_{(1,j)}^{(O)}, w_{(2,j)}^{(O)}, ..., w_{(\mathcal{U}_H,j)}^{(O)})$ is called the weight vector and contains the weights of the connections between the map layer and output neuron $j \in \mathcal{U}_O$.

Activation Matrix The activation matrix contains the activations of each map neurons for each input pattern and is defined by

$$A^{(H)} =^{def} \begin{pmatrix} a_1^{(H)}(1) & \dots & a_K^{(H)}(1) \\ . & . & . \\ . & . & . \\ . & & . \\ a_1^{(H)}(K) & \dots & a_K^{(H)}(K) \end{pmatrix}$$

For each neuron $j \in \mathcal{U}_O$ the following approximation problem can be formulated

$$A^{(H)} \cdot w_j^{(O)} = t_j$$

The approximation problem can be solved by

$$w_j^{(O)} = \left(A^{(H)}\right)^{-1} \cdot t_j$$

In practical application the explicit matrix inversion is replaced by a efficient numerical robust method, for instance singular value decomposition.

7 Feature Selection and Architecture Optimization

Finding an optimal or a good enough neural classifier for a classification problem is an optimization problem. Non-linear optimization problems, as in the case of neural network training, are in general not trivial. An optimal neural classifer depends on many parameters, e.g. the neural paradigm (MLP, RBF, ..), architectural parameters, training algorithm and features selected. An exhaustive search is in most cases not feasible. For this reason a good enough classifer is considered sufficient in most cases.

Any optimization problem requires a search space, a search algorithm, a starting point in the search space and a termination condition[13]. To reduce the complexity of the optimization problem many algorithms are able to integrate expert knowledge. This knowledge can be used to reduce search space, using a good starting point in search space or guiding the search algorithm.

- **search space**
  The search space is defined by the space spanned by the parameters of the Neural Network. In most cases, the search space is reduced by predefining some parameters, e.g. the neural network paradigm, training algorithm etc. In the simplest case all parameters except the weights are predefined. The weights are then trained by an appropriate learning algorithm. In the case of MLP this learning algorithm might be, for instance, Backpropagation.
- **search algorithm**
  A stochastic search algorithm, more specially a genetic algorithm, is used in this work as search algorithm. Model optimization and feature selection based on a genetic algorithm has achieved promising results for RBF Networks [8, 14, 15]. The genetic algorithm used is described in detail in [15].
- **initial starting point**
  To find a good initial starting point for the search algorithm human expert knowledge can be integrated (e.g. using a small number of hidden neurons). This approach is used to initialize the initial population of the genetic algorithm. Choosing a good starting point reduces in most cases training time and classification error.
- **termination condition**
  A termination condition is needed to determine when a search is finished. E.g. a specified classification rate must be reached or a specified time must have been elapsed.

7.1 Feature Selection

Feature selection is used to select a subset of features that is optimal to seperate the patterns in their target classes. Even though discarding some features, means discarding information feature selection can be used to reduce training time and increase the classification quality of neural classifiers. In the following cases discarding a feature should be considered :

- The feature contains redundant information for separating the patterns in their target classes.
- The feature has little or no correlation with the target class. Due to the localized behaviour of RBF Networks this problem is particularly severe if the feature has a significant variance.
- Within a subset of patterns the feature has an unmeant correlation with the target class. This can appear if only a subset of available patterns is used for training (e.g. cross-validation, use of an independent test set ...).

Feature selection can be seen as a combinatorial optimization problem. A set of features F defines a search space with $2^{|F|}$ possible solutions. Any procedure for feature selection must be based on two components. First, the selection criterion must be defined by which it is possible to judge whether one subset of features is better than another. A possible selection criterion is the classification error of a neural classifier. Second, a systematic search procedure must be defined to search through the set of possible solutions. A possible search procedure is a complete (exhaustive) search that considers all possible solutions. Because of the exponential complexity this is only feasible for very small feature sets.

The quality of a subset of features depends on the model of the classifier. The wrapper approach proposed by [16] for feature selection accounts for this. In addition, the wrapper approach can

incorporate the parameter optimization of the classifier. This allows to adjust the parameters of the classifier according to the features selected. A genetic approach is used here to optimize the architecture of the proposed network and the selected features. The genetic approach is presented in detail in the next section.

7.2 Model Optimization

Training time and classification error of a neural classifier depend on many parameters. Parameters are the neural network paradigm (e.g. MLP, RBF, SOM, RBF-SOM), training algorithm, in the case of MLP and RBF the number of hidden neurons, in the case of SOM the number of map neurons etc. The classification quality of some paradigms (e.g. SOM) is known to depend heavily on the parameter combination. The need for a human expert narrows the applicability of manual approaches. Semi-automatic or automatic approaches have achieved promising results. Semi automatic approaches can, for example integrate hard computing methods for the calibration of some parameters or integrate other soft-computing methods with different parameters which do not depend as heavily on the parameter combination. The use of semi-automatic and automatic approaches can increase the classififcation quality on the cost of execution time and memory requirements.

7.3 Genetic optimization

Genetic optimization is a stochastic optimization approach. Genetic optimization has been applied in the field of neural networks to model optimization and feature selection. An abstract approach called wrapper which integrates the training of the neural classifier and feature selection is proposed by [16]. Work on integrated feature selection and RBF Network model optimization is done by [8, 14, 15]. In [15] a generally applicable abstract model of a genetic algorithm is presented. This work includes theoretical work and an implementation of a testbench specialized on genetic RBF model optimization and feature selection.

Phenotypes are possible solutions of the optimization problem and all phenotypes define the search space of the optimization problem. Because of time restrictions an exhaustive search is not feasible in most practical applications. Good enough solutions, so called heuristic solutions, are then considered as being sufficient. To reduce time effort knowledge available about the optimization problem should be integrated in the search algorithm.

Genotypes are used as a blueprint for phenotypes. A genetic algorithm is applicable to a problem if a set of similiar phenotypes can be described by one genotype. Similiar in this case means similiar considering the fitness of the phenotype. From a genotype different phenotypes can be developed. This is modeled by a development function. Genotypes are also referred to as individuals.

The development function creates an phenotype from a genotype. Phenotypes are evaluated with a fitness function. The fitness function computes a fitness value for an phenotype. Phenotypes with a higher fitness value are more qualified for reproduction.

In each step the genetic algorithm has a set of genotypes referred to as population. The size of the population can be set depending on the problem. An initial population has to be specified. All genetic information available at the beginning of the genetic is contained in the initial population. The initial populations defines the starting point in search space of the gentic optimization algorithm.

Individuals which survive multiple cycles of the genetic algorithm can change over time. This change is defined by the aging function.

The reproduction of individuals is modeled by two functions. First, the mutation function which randonmly changes a genotype. Second, the recombination function which combines two genotypes to one genotype. A selection function determines the individuals which are mutated and combined. The selection of individuals has a great influence on the search direction (search algorithm). Mutation-, recombination- and selection function are the most influential components of the genetic algorithm.

If the size of a population after reproduction is greater than the selected population size a cutting function selects a subset of individuals (with the correct size) which is used for the next cycle of the genetic algorithm.

8 DARPA IDS Evaluation

The Information Systems Technology Group (IST) of the MIT Loncoln Laboratory supported by the Defense Advanced Research Projects Agency (DARPA) and by the Air Force Laboratory (AFRL) has developed the first standard test sets for the evaluation of Intrusion Detection Systems (IDS)[5]. The test sets allow an objective comparison of IDS by measuring the classification rate.

Two comprehensive evaluations of Intrusion Detection Systems were made. The first evaluation made in 1998 was based on data sets gathered from a network with simulated traffic and simulated users built up solely for this purpose. The simulation of traffic and users was based on a statistical evaluation of data gathered for 4 months from real networks.

The network of the IDS evaluation in 1998 was designed to simulate the network of an Air Force Base. The network consisted of two components, an internal and an external component connected by a router. The external component consisted of two "real" workstations. With a modified kernel the first workstation was able to simulate any number of workstations connected by a gateway. In the first evaluation in 1998 the first workstation was used to simulate a gateway to around 100 workstations. The second workstation provided access to more than thousand websites. The content of the websites was updated daily.

The evaluation of our system network will be based on data sets provided by the DARPA IDS Evaluation 1998.

8.1 Available Data

For the off-line detection of attacks BSM-logfiles, tcpdump-files, Harddrivedumps and listfiles are made available by the DARPA. BSM (Basis Security Module) enables to log system activity. BSM fullfills the Orange Book C2 Standard and is delivered with the operating system Solaris. BSM logs system activity at the kernel-level and thus enables more detailed logging than "syslogd", which is part of most linux distributions.

At regular intervals backups of the directories /root, /opt, /usr and /home were made and are made available to be used (e.g. for the detection of modified security-relevant data).

One workstation uses the tool tcpdump to log all network traffic, this includes the TCP header and the TCP body. The TCP header contains the TCP protocol information and the TCP body the application data being transported.

Listfiles are provided for the tcpdump files and the BSM logfiles. All information for one connection is stored in one row. In the case of an attack, the name of the attack is stored in the last column of the row.

8.2 Feature extraction

For a network-based IDS only the tcpdumps files and their corresponding listfiles are of interest. tcpdump files include the content of every packet transmitted between computers in the network of the Air Base simulated. The size of all files is approximately 10 GB. Only TCP connections are considered in this work to restrict the amount of data that needs to be processed.

In order to reconstruct the TCP segments the IP fragments have to be defragmented and the chronological order must be restored. For this purpose the network-based IDS Snort is used. Snort is able to process binary tcpdump files and generate ASCII files. Snort defragments IP fragments and restores the chronological order of the TCP segments. An ASCII file genereted by Snort provides the information for a TCP segment. This information is combined with the corresponding listfiles provided by the DARPA.

To be able to train artifical neural networks features have to be extracted from these ASCII-files. Table 1 gives a summary about all features extracted. The features extracted are stored in a MySQL database.

8.3 Attacks

This section contains an overview of the different attack types used in the simulation. All attacks were launched by an external "virtual" workstation. The training data contains 17 different attack types. All attacks can be divided in 4 different groups.

Field	Description	Data type
Index	Pattern identifier	Integer
0	Source port	Double
1	Target port	Double
2	Duration of the conection in seconds	Double
3	Number of packets	Double
4-15	TTL. Minimum, Maximum, Average and Sum for all three directions (client to server, server to client and both directions)	Double
16-30	TOS. 0x0, 0x2, 0x4, 0x8 and 0x10 for all three directions (client to server, server to client and both directions)	Double
31-42	DgmLength. Minimum, Maximum, Average and Sum for all three directions (client to server, server to client and both directions)	Double
43-54	Reserved flags	Double
55-60	Flag URG. Sum and Average for all three directions (client to server, server to client and both directions)	Double
61-66	Flag ACK. Sum and Average for all three directions (client to server, server to client and both directions)	Double
67-72	Flag PSH. Sum and Average for all three directions (client to server, server to client and both directions)	Double
73-78	Flag RST. Sum and Average for all three directions (client to server, server to client and both directions)	Double
79-84	Flag SYN. Sum and Average for all three directions (client to server, server to client and both directions)	Double
85-90	Flag FIN. Sum and Average for all three directions (client to server, server to client and both directions)	Double
91-102	Window Length. Minimum, Maximum, Average and Sum for all three directions (client to server, server to client and both directions)	Double
103-108	TCP-Options. Sum and Average for all three directions (client to server, server to client and both directions)	Double
109-123	Data Length. Minimum, Maximum, Average and Sum for all three directions (client to server, server to client and both directions)	Double
124-135	TCP Length. Minimum, Maximum, Average and Sum for all three directions (client to server, server to client and both directions)	Double
136	Start Hour	Double
137	Attack	Double

Fig. 1. Table containing an overview of the features extracted from the DARPA IDS evaluation in 1998

DoS (Denial of Service) : "Denial of Service" attacks are trying to block normal authorised access to services offered by a single host or a network. Access is blocked by overloading services or even crashing single hosts or networks. Typically such attacks are started from several hosts and are called "Distributed Denial of Service".

The following attacks are classified as DoS attacks : "Back", "Land", "SYN Flood" (Neptune), "Ping of Death"' (POD), and "Smurf".

User → Root: Having normal user privileges "User → Root" are aiming to gain root accesss (system administrator privileges). Often intruders first try to get normal user privileges before they try to exploit different security flaws to gain root access.

The following attacks are classified as "User → Root" attacks : "Eject", "Ffb", "Loadmodule" and "Perl".

Remote → Local: Attacks of the group "Remote → Local" are aiming at gaining access to a local account from another host or network.

The following attacks are classified as "Remote → Local" : "Dictionary", "Ftpwrite", "Guest", "Imap" and "Phf".

8.4 Probing/Scanning :

To gain valuable information about a host or network the target is scanned. This information is used to find security flaws and services running on the target. "Probing/Scanning" is not considered as an attack, but most sophisticated attacks use scanning as a first step.

The following tools are used for "Probing/Scanning" : "Nmap", "Satan", "Saint", "Ipsweep" and "Portsweep". The first three tools "Nmap", "Satan" and "Saint" are sophisticated security scanners, able to scan the target for security flaws. These tools are also used by system administrators. "Ipsweep" and "Portsweep" are simpler variants.

8.5 Database Design

Recent research work at the University of Passau includes the creation of a database containing extracted features for all TCP-connections of the DARPA IDS Evaluation. All TCP-connections are stored in binary Tcpdump-Files. For feature extraction the binary Tcpdump-Files were converted by the network based IDS Snort to ASCII Files. Finally 137 features for every patterns were extracted from these ASCII-Files and stored in one table of the database, further called FEATURES. Table 2 shows the number of patterns available for the different classes.

Attack type	# of patterns
back	2097
dict	874
guest	50
ipsweep	330
nmap	906
warezclient	1579

Fig. 2. Number of patterns available for each attack type

Related to the database the research work includes two different applications. First, comparison of RBF Networks and RBFSOM by training networks for a single attack. Second, train RBFSOM for all attacks to visualize the relationship of unknown patterns to known patterns.

For the first application Neural Networks are trained for detecting one single attack type. For each attack six tables are created. Five tables are used as validation data for 5-fold cross-validation

and the sixth table for test data. 70% of the attack patterns are used for the validation sets and 30% are used for the test set. The assignment is random. To each validation and test set patterns with normal behaviour are added, so that half of the patterns are attack patterns. Finally 10% (of the attack patterns) are other attacks, labelled as normal user behaviour, added to each data set. E.g. for an attack type with 1000 patterns. 700 patterns are used for validation and 300 for testing. Each of the five validation set contains 140 patterns of the attack and 140 patterns of normal user behaviour. In addition each validation set contains 14 patterns of other attacks labelled as normal user behaviour. The test data set contains 300 attack patterns, 300 normal user behaviour patterns and 30 other attacks labelled as normal user behaviour.

For the second application Neural Networks are trained for detecting all six attack types. Six tables are created, each containing patterns of all six attack types and patterns of normal user behaviour. As in the first application five tables are used as validation data for 5-fold cross validation and the sixth table for test data with one exception : no 10% of other attacks are labelled as normal user behaviour.

9 Comparison of RBF Networks and RBF-SOM

.

RBF-SOM can be seen as an extension to RBF Networks and an empirical comparison to RBF Networks will be made. The evaluation will be based on the DARPA 1998 IDS data. RBF-SOM and RBF Networks will be trained by the evolutionary algorithm described in [15].

The evolutionary algorithm will be used for feature selection and model optimization. Model optimization includes calibration of achitectural parameters (e.g. number of hidden neurons) and parameters for the training algorithm (e.g. learning rate). A 5-fold cross validation and an independent test data set will be used. Validation data sets of the cross validation will be used for evaluation during the training of an individual. The test data set will be used for evaluation of the population.

RBF-SOM and RBF Networks will be trained as a classifier for a single attack type (e.g. ipsweep). For each of the six attacks and each of the two paradigms 20 runs of the evolutionary algorithm will be made. A run of the evolutionary algorithm terminates after 50 cycles. For each attack type one section evaluates the results of these experiments. Each section contains the minimal, maximal and average classification rate on each validation set and the test data set, combined in one table.

Knowledge from recent research on Intrusion Detection with RBF Networks will be integrated in the optimization process to determine a good starting point for the search. 137 features were extracted from the DARPA data sets. A complete initialization consisting of all combinations of feature (2^{137}) is not feasible. Research conducted, based on RBF Networks, shows that small networks with 1-3 features and few basis functions are able to achieve good classification results. To assure that at least one network with each feature is trained at least once, we use a complete initialization restricted to exactly one selected feature. Therefore the initial population does consist of 137 individuals. The size of the hidden layer for RBF Networks is gauss distributed with an expectation of 5 and a variance of 4. The horizontal and vertical size of the map layer for RBF-SOM is gauss distributed with an expectation of 4 and and variance of 3.

9.1 Back

Attack Description Back is a DoS attack for the Apache Webserver. The Apache Webserver will be flooded with requests containing a large amount of slashes, e.g. “GET ///..//index.html”. These requests slow down the server so that it can finally respond to no other requests.

Evaluation Trained RBF Networks achieve an average classification rate on validation resp. test data of at least 0.9947 resp. 0.9920. In contrast RBF-SOM‘s have a slighty better average classification rate on four of the five validation sets and the test set.

All RBF Networks and RBF-SOMs have exactly one selected feature. In most runs RBF Networks resp. RBF-SOM have feature 101 (average Window Length of packets from the server) or feature 106 (average of number of TCP Options setting of packets from the client) resp. feature 101

selected. Both paradigms with feature 101 selected achieve a classification rate of 1.0 on V1 and V4. The best Networks for this attack are trained with feature 101.

RBF-SOM trained with feature 101 have between 9 and 12 basis function. The best one, with the highest average classification rate on the validations sets, is the one with most basis functions. One network with feature 21 selected achieves with 3 basis functions a classification rate of 0.9947 resp. 0.977 on validation resp. training data.

back	RBF	RBFSOM
T min	0,9812	0,9771
T max	0,9958	0,9958
T avg	0,9920	0,9947
V0 min	0,9867	0,9967
V0 max	0,9983	0,9983
V0 avg	0,9965	0,9982
V1 min	0,9833	0,9964
V1 max	1,0000	1,0000
V1 avg	0,9978	0,9998
V2 min	0,9833	0,9933
V2 max	0,9965	0,9983
V2 avg	0,9947	0,9964
V3 min	0,9767	0,9950
V3 max	0,9967	0,9983
V3 avg	0,9950	0,9952
V4 min	0,9767	0,9965
V4 max	1,0000	1,0000
V4 avg	0,9977	0,9997

back	RBF	RBFSOM
#centers min	7	3
#centers max	19	12
#centers avg	11,9500	9,3500
#features min	1	1
#features max	1	1
#features avg	1,0000	1,0000
runTime min	2,1025	51,9892
runTime max	5,5101	101,5860
runTime avg	3,5764	74,2551

back	RBF	RBFSOM
T	0,9951	0,9958
V0	0,9983	0,9983
V1	1,0000	1,0000
V2	0,9965	0,9983
V3	0,9950	0,9950
V4	1,0000	1,0000
#centers	9,0000	12,0000
#features	1,0000	1,0000
#runTime	184,3860	3.119,3500

Fig. 3. Tables contain experimental reuslts for the attack type "back" for RBF Networks and RBF-SOM. The table on the left side contains minimum, maximum and average classification rates. The table in the middle contains minimum, maximum and average for some architectural parameters (number of features and number of centers). The table on the right side contains experimental results for the network with the best average classification rate over all 5 validations sets.

9.2 Dictionary (Dict)

Attack Description The Dictionary attack consists of trying different user/ password combinations to gain access to an user account. The Dictionary attack is automated and there are different variations in choosing user/ password combinations. One variation is brute force by trying e.g. all possible passwords for one user account. Other variations for choosing passwords are based on existing dictionaries. All services using an user/ password authentication are vulnerable to this attack (e.g. telnet, ftp, pop3, rlogin and imap).

Evaluation In exception to V4 : Both paradigms have very high average classification rates of at least 0.9942. RBF Networks compared to RBFSOM have on all data sets a slightly lower average classification rate. It seems that RBFSOM has problems with V4. Minimum and average classification rates are lower (around 1%) than for all other data sets.

There are many different combinations of selected features for the fittest RBF Networks : 13 different combinations in 20 runs, with a at most 2 selected features. RBF-SOM have exactly one selected feature. Feature 129 (Maximal TCP length for packets from the client) is dominating. Both paradigms select different features for classification. Therefore the data sets contain redundant features for classification. One RBF Network achieves, with active features 100 (Maximum "Window Length" for packets from the server) and 118 (Average of the duration of the connection), a classification rate of 1.0 for V4.

The best Networks have in the case of RBF Networks 19 basis functions and in the case of RBF-SOM 20 basis functions. One RBF Network with 8 basis functions achieves a classification rate on validation data resp. test data of at least 0.988 resp. 1.0. A RBF-SOM with 40 basis functions achieves a classification rate on validation data resp. test data of at least 0.992 resp. 1.0.

dict	RBF	RBFSOM
T min	0,9839	0,9915
T max	1,0000	1,0000
T avg	0,9967	0,9977
V0 min	0,9929	0,9964
V0 max	1,0000	1,0000
V0 avg	0,9983	0,9995
V1 min	0,9851	0,9929
V1 max	1,0000	1,0000
V1 avg	0,9971	0,9993
V2 min	0,9786	0,9893
V2 max	1,0000	1,0000
V2 avg	0,9966	0,9946
V3 min	0,9714	0,9929
V3 max	1,0000	1,0000
V3 avg	0,9960	0,9975
V4 min	0,9875	0,9744
V4 max	1,0000	0,9958
V4 avg	0,9942	0,9873

dict	RBF	RBFSOM
#centers min	6	14
#centers max	23	40
#centers avg	14,7500	20,3000
#features min	1	1
#features max	2	1
#features avg	1,3000	1,0000
runTime min	1,9435	54,2813
runTime max	7,2439	76,0807
runTime avg	4,0968	64,3957

dict	RBF	RBFSOM
T	1,0000	1,0000
V0	1,0000	1,0000
V1	1,0000	1,0000
V2	1,0000	0,9964
V3	1,0000	1,0000
V4	0,9958	0,9958
#centers	19,0000	20,0000
#features	1,0000	1,0000
#runTime	434,6320	3.777,9400

Fig. 4. Tables contain experimental reuslts for the attack type "dict" for RBF Networks and RBF-SOM. The table on the left side contains minimum, maximum and average classification rates. The table in the middle contains minimum, maximum and average for some architectural parameters (number of features and number of centers). The table on the right side contains experimental results for the network with the best average classification rate over all 5 validations sets.

9.3 Guest

Attack Description Guest is a variation of the Dictionary attack. Guest accounts are targets of this attack. Guest accounts which often have easy to guess passwords or even no password authentication make this attack effective.

Evaluation The guest attack is the attack with the smallest validation and training sets. We assume that the ability to generalize can depend very strong on the partioning of available data in the different validation and test sets. Each paradigm shows a very huge variance of the classification rate of up to 0.77 on validation and test data. No Network achieves a validation classification rate better than 0.8. Even the best RBF Network has a classification rate for test data of less than 0.5. Same for the fittest RBF-SOM form run #8.

All fittest RBF Networks have exactly 1 selected feature. There are 10 different features for RBF Networks without an outstanding feature. RBF-SOM has up to 5 features (fittest Network from run #8).

The number of basis functions does vary very strong (6 - 72 basis functions) for RBF-SOM. In contrast RBF Networks are small (3 - 14 basis functions). In this case there seems to be no correlation between network size and classification rate. A RBF-SOM with 6 basis functions has comparable classification results to an network with 72 basis functions.

9.4 Ipsweep

Attack Description Ipsweep is a simple portscanner, able to scan multiple hosts or subnets. Alternatively a simple "ping" can be made to get to know which hosts are online.

guest	RBF	RBFSOM
T min	0,3333	0,4667
T max	1,0000	0,9500
T avg	0,7150	0,7625
V0 min	0,5875	0,4875
V0 max	0,7375	0,7875
V0 avg	0,6919	0,6719
V1 min	0,6000	0,5250
V1 max	0,7250	0,8000
V1 avg	0,6731	0,6769
V2 min	0,5779	0,5455
V2 max	0,7468	0,7273
V2 avg	0,7071	0,6510
V3 min	0,5000	0,5000
V3 max	0,7222	0,7778
V3 avg	0,6639	0,6667
V4 min	0,5000	0,5556
V4 max	0,7222	0,7778
V4 avg	0,7028	0,7250

guest	RBF	RBFSOM
#centers min	3	6
#centers max	14	72
#centers avg	8,3500	20,5000
#features min	1	1
#features max	2	5
#features avg	1,1000	1,7500
runTime min	0,1708	8,5321
runTime max	0,2427	20,1627
runTime avg	0,1919	14,4211

guest	RBF	RBFSOM
T	0,4667	0,5667
V0	0,7375	0,7875
V1	0,7250	0,7000
V2	0,7208	0,5909
V3	0,6667	0,7222
V4	0,7222	0,7778
#centers	9,0000	24,0000
#features	1,0000	1,0000
#runTime	12,1940	1.186,2200

Fig. 5. Tables contain experimental reuslts for the attack type "guest" for RBF Networks and RBF-SOM. The table on the left side contains minimum, maximum and average classification rates. The table in the middle contains minimum, maximum and average for some architectural parameters (number of features and number of centers). The table on the right side contains experimental results for the network with the best average classification rate over all 5 validations sets.

Evaluation There is a great variance of the classification rate on each data set of up to 0.5. For each data set a satisfying maximal classification rate of 0.9964 (in one case) and 1.0 (in all other cases) can be achieved. The minimal classification error is for RBF-SOM Networks on all data sets higher, in 5 of the 6 data sets significantly higher (around 2-3%). One RBF-SOM has a classification rate of 1.0 for all data sets (validation and test data).

All Networks are trained with one or two selected features. Two features are significant because they are selected very often. First, is feature 9 (Maximum of TTL for packets from the client) and second is feature 136 (Start Hour).

The number of hidden neurons is less for RBF Networks compared to RBF-SOM. The average number of hidden neurons is 10,8 compared to 16,6. The number of neurons seems to depend on the number of selected features. Only RBF Networks with one selected feature have less than 10 hidden neurons

9.5 Nmap

Attack Description Nmap (Network Mapper) is a very sophisticated and very powerful tool to scan single hosts or whole networks. Based on normal or stealthy portscans open ports, running services and even the operating system can be detected. Compared to other similiar tools Nmap allows very fast port scans and offers a great functional range. Nmap is freely availabe and has been ported to all popular operating systems.

Evaluation Compared to other attacks (except Guest) the average classification rate on test and validation sets is low. RBFSOM achieve significantly better maximal classification rates and therefore also average classification rates.

RBF Networks are only trained with exactly one selected feature. Significant are the features 92 (Maximum Window length for packets of both directions) and 96 (Maximum Window Length for packets from the client). Interesting is : RBFSOM choose different features for classification. 18 of the 20 trained RBFSOM choose one selected feature. Significant are features 86 (Average of Flag FYN for both directions) and feature 88 (Average of Flag FYN for packets from the client). RBF

ipsweep	RBF	RBFSOM
T min	0,9321	0,9821
T max	0,9964	1,0000
T avg	0,9812	0,9931
V0 min	0,9348	0,9891
V0 max	1,0000	1,0000
V0 avg	0,9860	0,9984
V1 min	0,5000	0,5000
V1 max	1,0000	1,0000
V1 avg	0,9245	0,8916
V2 min	0,9239	0,9658
V2 max	1,0000	1,0000
V2 avg	0,9787	0,9887
V3 min	0,9375	0,9565
V3 max	1,0000	1,0000
V3 avg	0,9818	0,9902
V4 min	0,9565	0,9783
V4 max	1,0000	1,0000
V4 avg	0,9905	0,9962

ipsweep	RBF	RBFSOM
#centers min	3	5
#centers max	29	36
#centers avg	10,8000	16,6000
#features min	1	1
#features max	2	2
#features avg	1,6000	1,5500
runTime min	0,4332	17,5947
runTime max	2,1880	31,0245
runTime avg	0,9082	24,9592

ipsweep	RBF	RBFSOM
T	0,9964	1,0000
V0	0,9891	1,0000
V1	0,9891	1,0000
V2	1,0000	1,0000
V3	1,0000	1,0000
V4	1,0000	1,0000
#centers	8,0000	21,0000
#features	1,0000	2,0000
#runTime	45,0611	1.455,0000

Fig. 6. Tables contain experimental reuslts for the attack type "ipsweep" for RBF Networks and RBF-SOM. The table on the left side contains minimum, maximum and average classification rates. The table in the middle contains minimum, maximum and average for some architectural parameters (number of features and number of centers). The table on the right side contains experimental results for the network with the best average classification rate over all 5 validations sets.

Networks resp. RBFSOM choose one time resp. two times feature 10 (Average TTL for packets from the client). Only Networks with at least two selected features achieve classification rates on all data sets of at least 0.989. We assume that RBF Networks with at least two selected features would achieve similiar results.

All RBF Networks have a small number of hidden neurons, between 5 and 8. RBFSOM are significantly larger and have up to 42 neurons. The best RBFSOM has 42 neurons and 3 selected features.

9.6 Warezclient

Attack Description Warezclient allows to establish an anonymous ftp server on a host. Often copyright protected software and/ or music files are distributed over such an ftp access.

Evaluation In comparison to RBF Networks RBFSOM achieve better average, minimal and maximal classification results on all data sets (except the average on the test set). Most classification rates are significantly better than for RBF Networks. Both paradigms show a huge variance of the classification rate of 0.77 - 0.79 on the test data set. RBF Networks have compared to RBFSOM low minimal classification rates on validation sets. The classification rates of two validation sets is below 0.9. In contrast for RBFSOM the classification rate on all validation sets is higher than 0.98. By RBFSOM on all data sets a classification rate of 1.0 is achieved, but only from different Networks. The best Networks of both paradigms achieve a classification rate of at least 0.995 on all data sets. RBFSOM achieve on 4 of the 5 validation sets a classification rate of 1.0. RBF Networks on 3 validation sets.

RBF Networks resp. RBFSOM are trained with 1-3 resp. 1-2 selected features. RBF Networks trained with feature 97 (Average Window Length for packets from client) achieve significantly low classification rates on test data. We assume that there is a random correlation between patterns in the validation sets and feature 97. In contrast RBFSOM achieve significant low classification rates (0.23 - 0.87) on the test data if features 8 (Minimum TTL for packets from the client) and 93 (Average Window Length for both directions) are selected. Only feature 4 (Minimum TTL for both

nmap	RBF	RBFSOM
T min	0,9478	0,9478
T max	0,9527	0,9936
T avg	0,9524	0,9560
V0 min	0,9478	0,9478
V0 max	0,9478	1,0000
V0 avg	0,9478	0,9530
V1 min	0,9436	0,9394
V1 max	0,9478	0,9917
V1 avg	0,9476	0,9513
V2 min	0,9436	0,9436
V2 max	0,9478	1,0000
V2 avg	0,9476	0,9522
V3 min	0,9478	0,9478
V3 max	0,9478	0,9963
V3 avg	0,9478	0,9522
V4 min	0,9478	0,9436
V4 max	0,9478	0,9925
V4 avg	0,9478	0,9520

nmap	RBF	RBFSOM
#centers min	5	4
#centers max	8	42
#centers avg	7,0500	9,7000
#features min	1	1
#features max	1	3
#features avg	1,0000	1,1500
runTime min	0,7786	40,4705
runTime max	0,9322	58,6883
runTime avg	0,8786	47,5726

nmap	RBF	RBFSOM
T	0,9527	0,9936
V0	0,9478	1,0000
V1	0,9478	0,9917
V2	0,9478	1,0000
V3	0,9478	0,9963
V4	0,9478	0,9921
#centers	8,0000	42,0000
#features	1,0000	3,0000
#runTime	51,0598	3.521,3000

Fig. 7. Tables contain experimental reuslts for the attack type "nmap" for RBF Networks and RBF-SOM. The table on the left side contains minimum, maximum and average classification rates. The table in the middle contains minimum, maximum and average for some architectural parameters (number of features and number of centers). The table on the right side contains experimental results for the network with the best average classification rate over all 5 validations sets.

directions) is selected often in the fittest networks of both paradigms. These Networks achieve a classification rate of at least 0.98 on all data sets. The best Networks of both paradigms are trained with two selected features.

warezclient	RBF	RBFSOM
T min	0,2069	0,2307
T max	0,9991	1,0000
T avg	0,9237	0,9138
V0 min	0,8448	0,9886
V0 max	1,0000	1,0000
V0 avg	0,9832	0,9936
V1 min	0,9855	0,9932
V1 max	1,0000	1,0000
V1 avg	0,9939	0,9964
V2 min	0,9757	0,9818
V2 max	1,0000	1,0000
V2 avg	0,9861	0,9912
V3 min	0,8955	0,9909
V3 max	1,0000	1,0000
V3 avg	0,9868	0,9949
V4 min	0,9748	0,9841
V4 max	1,0000	1,0000
V4 avg	0,9887	0,9943

warezclient	RBF	RBFSOM
#centers min	4	4
#centers max	36	45
#centers avg	10,3000	16,9000
#features min	1	1
#features max	3	2
#features avg	1,3500	1,6500
runTime min	0,8938	48,0428
runTime max	5,9688	83,0550
runTime avg	1,8072	65,7424

warezclient	RBF	RBFSOM
T	0,9991	0,9992
V0	1,0000	0,9952
V1	1,0000	1,0000
V2	0,9977	1,0000
V3	1,0000	1,0000
V4	0,9977	1,0000
#centers	22,0000	16,0000
#features	2,0000	2,0000
#runTime	219,0920	3.794,5000

Fig. 8. Tables contain experimental reuslts for the attack type "warezclient" for RBF Networks and RBF-SOM. The table on the left side contains minimum, maximum and average classification rates. The table in the middle contains minimum, maximum and average for some architectural parameters (number of features and number of centers). The table on the right side contains experimental results for the network with the best average classification rate over all 5 validations sets.

10 Human Expert Integration

Classifications problems are in most cases not trivial. A reliable fully automated classification can in most cases only be made for a subset of all possible patterns. This is the case when there are similiar patterns belonging to different classes or there are simply not enough patterns available to create a reliable classifier. In real world applications it is often very costly to acquire meaningful measurements (e.g. purchasing adequate sensors) to seperate the patterns in their corresponding classes or acquire enough different patterns (e.g. manual assignment from patterns to classes). This section describes an approach to integrate a human expert in a classification process to classify patterns which cannot be classified with a reliable probability. Four different measures for the reliability will be proposed and evaluated based on the DARPA IDS data.

Considering a RBF Network trained to minimize the euclidean distance between target vector and output vector : The output of the Neural Network can be interpreted as approximations to posterior probabilities. Using a Bayesian approach the classification error can be minimized by selecting the class with the highest posterior probability. A reliable classification can be made when exactly one class as a high approximated posterior probability near 1 and all other classes have low posterior probability near 0. A reliable classification cannot be made when the difference between the winning class and at least one other class is very similiar. For patterns propagated through a RBF Networks this can happen in the following cases :

1. The pattern is not very similiar to any pattern used to train the RBF Network. Because of the localized behaviour of RBF Networks the output of such patterns will be very low for all classes.
2. The pattern is very similiar to patterns of at least two different classes. In this case the output of the corresponding classes is high and similiar.

To measure the reliability of the classification we propose four functions, further called score functions. The output vector $y(k) = (y_0(k), ..., y_{n-1}(k))$ with $n =^{def} |\mathcal{C}|$ contains the approximated posterior probabilities for class membership, and $j \in \{0, ..., n-1\}$ is the class with the highest posterior probability.

1. **Score Function** : $score_1(k) = min(\|y_i(k) - y_j(k)\| : i \in \{0, ..., n-1\} \wedge i \neq j)$
   This score function measures the difference between the approximated posterior probability of the winner class and the class with the second highest approximated posterior probability.
   The score function is high if there is exactly one class (the winner class) with a high approximated posterior probability.
2. **Score Function** : $score_2(k) = y_j(k)$
   This score function measures the approximated posterior probability of the winner class.
   The score function is high if the winner class has a high approximated posterior probability.
3. **Score Function** : $score_3(k) = a * f_0(k) + b * f_1(k)$
   This score functions combines the two score functions above and is defined by their weighted sum. For a and b applies : Both values are greater or equal than zero and they sum to unity.
   For $a = 0.5$ and $b = 0.5$ the score function is high if the winner class has a high approximated posterior probability and if there is no second class with an approximated posterior probability similiar to the winner class.
4. **Score Function** : $score_4(k) = 1 - \|y(k) - t'(k)\| \cdot {max_{score}}^{-1}$
   We assume an unary target vector $t'(k)$ containing the 1 at the position of the maximal posterior probability of $y(k)$. This score function measures the euclidean distance of the output vector $y(k)$ with the assumed target vector $t'(k)$. We define $t'(k)$ by

$$t'(k) = \{t'(k)_0, ..., t'(k)_{n-1}\} \tag{1}$$

$$t'(k)_i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases} \tag{2}$$

   max_{score} is used as a normalization factor in $f_3(k)$. This factor depends on the dimension of the feature space $\mathcal{L}$. We assume $y(k)_0 >= y(k)_i$ without constraining the validity of the following proof. We define $n := dim(\mathcal{L})$. Then max_{score} is defined by

$$
\begin{aligned}
max_{score} = & \\
& = max(\|y(k) - t'(k)\| : \mathbf{x} \in \mathcal{L}) \\
& = max(\sqrt{\sum_i (y(k)_i - t'(k)_i)^2} : \mathbf{x} \in \mathcal{L}) \\
& = max(\sqrt{(y(k)_0 - 1)^2 + (y(k)_1 - 0)^2 + ... + (y(k)_{n-1)} - 0)^2} : \mathbf{x} \in \mathcal{L}) \\
& = max(\sqrt{(y(k)_0 - 1)^2 + y(k)_1^2 + ... + y(k)_{n-1)}^2} : \mathbf{x} \in \mathcal{L})
\end{aligned}
$$

The square root function is monotonic. Therefore the expression is maximal if the sum is maximal. The sum is maximal if each addend is maximal. The following holds : $y(k)_i <= y(k)_1$ for all $i \neq 1$.

$$
\begin{aligned}
& max(\sqrt{(y(k)_0 - 1)^2 + y(k)_1^2 + ... + y(k)_{n-1}} : \mathbf{x} \in \mathcal{L}) \\
& = max(\sqrt{(y(k)_0 - 1)^2 + y(k)_0^2 + ... + y(k)_0^2} : \mathbf{x} \in \mathcal{L}) \\
& = max(\sqrt{(y(k)_0 - 1)^2 + (n-1) \cdot y(k)_0^2} : \mathbf{x} \in \mathcal{L}) \\
& = max(\sqrt{n \cdot y(k)_0^2 - 2 \cdot y(k)_0 + 1} : \mathbf{x} \in \mathcal{L}) \\
& = max(\sqrt{n - 2 + 1} : \mathbf{x} \in \mathcal{L}) \\
& = max(\sqrt{n - 1} : \mathbf{x} \in \mathcal{L}) \\
& = \sqrt{n - 1}
\end{aligned}
$$

This score function is high if exactly one class (the winner class) has a approximated posterior probability near 1.0 and all other classes a value near 0.0.

10.1 Decision Making

A bayesian based classification process hast two seperate stages. The first stage is inference and the second stage is decision making. To integrate a human expert into the classification process the decision making stage will be adapted. Our proposed classifier integrates a traditional classifier. A pattern will be classified automatically by the traditional classifier or classified manually by a human expert. The pattern will be classified automatically if the score value of the pattern is greater than or equal to a predefined score value, further called "score-threshold". If not the pattern will be classified as "undefined" and will be classified manually.

Given a traditional classifier ζ, a set of classes $\mathcal{C} = \{C_0, ..., C_{n-1}\}$ and a score-threshold $\sigma \in [0, 1]$ our proposed classifier is given by :

$$
\zeta : \mathcal{L} \times \Re \times (\zeta \rightarrow \mathcal{C}) \rightarrow \mathcal{C} \cup \{\text{“}undefined''\}
$$

$$
\zeta(\mathbf{x}, \sigma, \zeta_t) := \begin{cases} \text{“}undefined'' & \text{if } score(\zeta_t) < \sigma \\ \zeta_t(\mathbf{x}) & \text{else} \end{cases} \tag{3}
$$

If the score-threshold is set to 0 all patterns will be classified automatically and the classifier behaves like an traditional classifier. Increasing the score-threshold either leaves the number of patterns being classified as "undefined" unchanged or increases it. If the score-threshold is greater than 1 all patterns will be classified manually. Therefore varying the score-threshold creates a so called "Receiver Operator Characteristics" Curve (ROC-Curve) for different variables, for instance the classification rate of a classifier.

10.2 ROC-Curve evaluation

Assuming a adequate trained classifier it can be assumed that most patterns can be classified automatically with high reliability. Assuming our score functions properly measure the reliability of classification most patterns would achieve a high score value. Therefore increasing the "score-threshold" from 0.0 to 1.0 would slowly increase the number of patterns being classified as "undefined".

The quality of a traditional classifier can, for instance, be measured by the classification rate. The quality of the proposed classifier can be measured by :

1. The number of undefined patterns. A high number of undefined patterns needs a high amount of human resources.
2. Considering the set of undefined patterns : The fraction of patterns which would not have been classified correctly by the encapsulated classifier.
3. The classification rate for patterns being classified automatically by the encapsulated classifier. This classification rate differs from the classification rate of the encapsulated classifier, because it is computed on a subset of available patterns. The subset depends on the specified score-threshold and contains all patterns not classified as "undefined".

A RBF-SOM trained by a genetic algorithm will be used for the evaluation. The evaluation will be based on two seperate data sets, the training set used by the genetic algorithm and an independent test set. Both sets contain different patterns. Training set and test set consist of patterns from the following six attack types (back, dict, guest, ipsweep, nmap and warezclient). The training set will be used for ROC-Curve approximation and the test set for validation of our experimental results.

The Neural Network is trained with five selected features determined by the genetic algorithm using 5-fold cross validation. The map layer does consist of 38 map neurons. The output layer does consist of 7 output neurons. One neuron for class-membership for each attack type and one output neuron for membership for normal user behaviour.

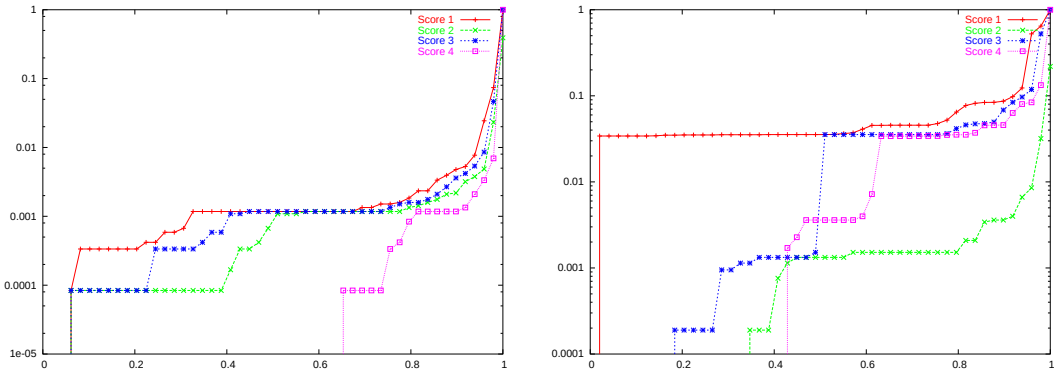
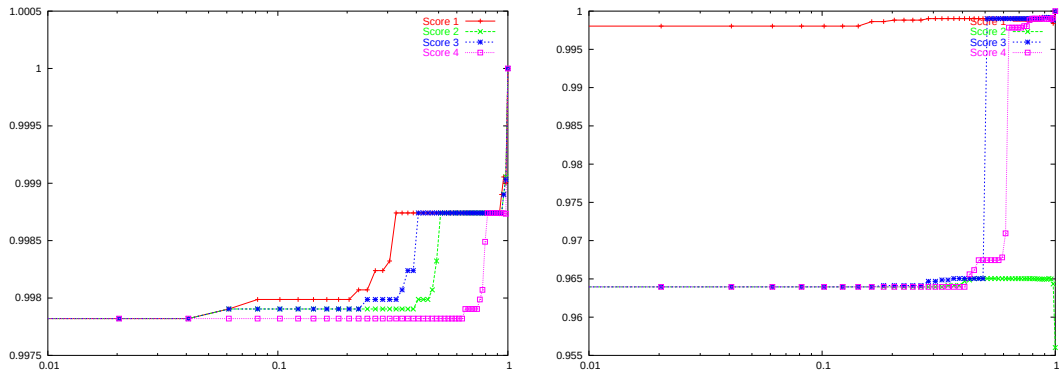


Fig. 9. Figures 9 (a) and (b) contain the ROC-Curves for the fraction of undefined patterns from all patterns. Figure 9 (a) contains the ROC-Curves for each score function based on the training data. The ROC-Curves for each score function based on the test data is contained in figure 9 (a). The x-axis displays the tolerance and the y-axis contains the fraction of undefined patterns. We use for better visualization a logarithmic scaling on the y-axis.

Undefined patterns First, we will evaluate the ROC-Curves for the fraction of undefined patterns depending on the specified score-threshold. The approximated ROC-Curves are visualized in figures 9 (a) and (b). The gradient (first derivation) of all four functions is very low for tolerance values smaller than 0.8. The gradient is increasing very fast for a tolerance near 1.0. For a score-threshold value of 0.939 less than 1% of all patterns are classified as "undefined". Therefore most patterns

can be classified with a high score value. This is important, because classifying a pattern manually needs human resources.

In addition, it is important that the encapsulated classifier has a high classification rate on all patterns not classified as “undefined”. The approximated ROC-Curves for the classification rate are visualized in figures 10 (a) and (b). For a score-threshold value of 0.939 we have a classification rate of at least 0.9987. For a score-threshold of 0 the classification rate is 0.9978. Therefore the classification rate for patterns not classified as “undefined” is increased. The difference seems small, but in e.g. the field of Intrusion Detection even one single succesfull attack can threaten the security of a whole network. In the next subsection we are interested in the undefined patterns.



(a) (b)

Fig. 10. Figures 10 (a) and (b) contain the ROC-Curves for the classification rate of the encapsulated classifier. Figure 10 (a) contains the ROC-Curves for each score function based on the training data. The ROC-Curves for each score function based on the test data is contained in figure 10 (a). The x-axis contains the tolerance and the y-axis the classification rate. For better visualization purposes we use a logarithmic scaling on both axes.

Fraction of patterns not classified correct In addition to the number of undefined patterns we are interested in the fraction of those patterns which would not have been classified correctly. For a score-threshold value of 0.939 we have a fraction between 0.14 (for score function 1) and 0.44 (for score function 4). With this score-threshold between 0.78% (for score function 1) and 0.21% (for score function 4) of all patterns are classified as “undefined”.

11 Summary and Outlook for future research

In conclusion our research has two main contributions: First, the combination of RBF and SOM for Intrusion Detection and second, the evaluation of human expert integration to Intrusion Detection to reduce the classification error.

The Neural Network paradigm RBF-SOM and an appropriate training algorithm is formulated in section 5 and 6. According to RBF Networks RBF-SOM can be trained with a combination of unsupervised and supervised training. The training algorithm formulated generates according to SOM a topolgy-preserving mapping.

Compared to the training of RBF Networks the training of RBF-SOM is very slow. This can be ascribed to the unsupervised part of training which generates a topology-preserving mapping. [17] describes an algorithm for fast initialization of SOM based on k-means. Because RBF-SOM is formulated as an extension to SOM, this approach can be easily adpated to reduce training time of RBF-SOM.

We compared RBF Networks with RBF-SOM based on data provided by the DARPA IDS evaluation in 1998. In recent research RBF Networks have in comparison to MLP, Classifying Fuzzy-k-Means and NEFCLASS systems led to best classification results. In experiments conducted in the

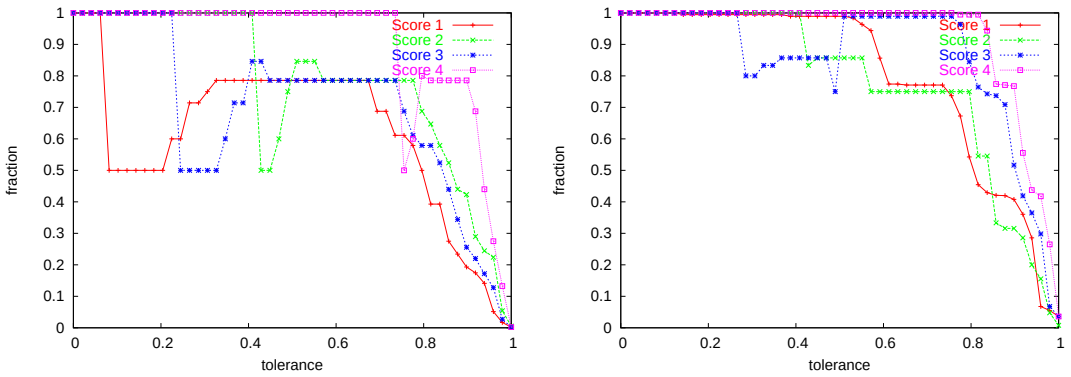


Fig. 11. Figures 11 (a) and (b) contain the ROC-Curves for the fraction of undefined patterns which would have not been classified correct by the encapsulated classifier. Figure 11 (a) contains the ROC-Curves for each score function based on the training data. The ROC-Curves for each score function based on the test data is contained in figure 11 (a). The x-axis displays the tolerance and the y-axis contains the fraction of patterns not classified correct.

context of this research work RBF-SOM achieves, compared to RBF Networks, similiar or even better results. Experimental results are contained in section 9.

Our testbench enables to integrate a human expert in the decision process. Patterns which cannot be classified with a satisfying reliabilty are visualized to a human expert. In section 10 we propose four score functions to measure the reliability of a classification made by a bayesian based classifier. ROC-Curves for all score funtions are approximated and evaluated.

Due to the integration of an human expert the proposed IDS can detect subtle variations of known attacks and even unknown attacks. One of the strengths of using a combination of RBF Networks and SOM is that the IDS can be easily extended to automatically adapt to classification results of an human expert without complete re-training. This can be achived by adding a new map neuron with a radial basis activation function to the proposed Neural Network. The center of the new radial basis function is initialized with a center equal to the input classified by the human expert. To be able to add nodes at the boundaries of the map a technique called Incremental Grid Growth (IGG) [18] can be used. The remaining parameters must be adapted accordingly. This enables to integrate iteratively the knowledge of the human expert during the application phase and reduce the classification error and the frequency of human expert integration.

References

1. WashingtonPost: Attack on internet called largest ever. http://www.washingtonpost.com (2002)
2. Hofmann, A., Schmitz, C., Sick, B.: Intrusion Detection in Computer networks with Neural and Fuzzy classifiers. Submitted to the International Conference on Artificial Neural Networks ICANN 2003 (2003)
3. Bishop, C.M.: Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1995)
4. Teuvo, K.: Self-Organizing Maps. 3 edn. Springer Series in Information Sciences Vol. 30 (2001)
5. Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., Zissman, M.A.: Evaluating Intrusion Detection Systems: The 1998 darpa Off-Line Intrusion Detection Evaluation. In: Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), Hilton Head. Volume 2., IEEE Press (1999) 12 – 26
6. Axelsson, S.: Intrusion Detection Systems: A Survey and Taxonomy. Technical Report 99-15, Department of Computer Engineering, Chalmers University of Technology, Göteborg (2000)
7. Anderson, J.P.: Computer security threat monitoring and surveillance. Technical Report, James P. Anderson Co., Fort Washington (1980)
8. Hofmann, A.: Einsatz von Soft-Computing-Verfahren zur Erkennung von Angriffen auf Rechnernetze. Master's thesis, Universität Passau, Lehrstuhl für Rechnerstrukturen (2002)
9. Jirapummin, C., Wattanapongsakorn, N., Kanthamanon: Hybrid Neural Networks for Intrusion Detection system. (XXX)

10. Fritzke, B.: Growing Cell Structures—a Self-Organizing Network in k Dimensions. In Aleksander, I., Taylor, J., eds.: Artificial Neural Networks, 2. Volume II., Amsterdam, Netherlands, North-Holland (1992) 1051–1056
11. Girardin, L.: An Eye on Network Intruder-Administrator Shootouts. Proceedings of the Workshop on Intrusion Detection and Network Monitoring (1999)
12. Sick, B.: Technische Anwendungen von Soft-Computing Methoden. Lecture Notes, University of Passau, Faculty of Mathematics and Computer Science (2001)
13. Ginsberg, M.: Essentials of Artifical Intelligence. Morgan Kaufmann Verlag, San Mateo, Ca. (1993)
14. Hofmann, A., Sick, B.: Evolutionary Optimization of Radial Basis Function Networks for Intrusion Detection. Submitted to the International Joint Conference on Neural Networks 2003 (2003)
15. Klimek, M.: Evolutionre Architekturoptimierung von RBF-Netzen. Master's thesis, Universität Passau, Lehrstuhl für Rechnerstrukturen (2003)
16. Kohavi, R., John, G.: The wrapper approach. Feature Selection for Knowledge Discovery and Data Mining (1998) 33 – 50
17. Su, M.C., Chang, H.T.: Fast self-organizing feature map algorithm. (2000)
18. Blackmore, J., Miikkulainen, R.: Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map. In: Proc. ICNN'93, International Conference on Neural Networks. Volume I., Piscataway, NJ, IEEE Service Center (1993) 450–455