Host-Based Intrusion Detection Using Self-Organizing Maps

Peter Lichodzijewski, A. Nur Zincir-Heywood, *Member, IEEE*, Malcolm I. Heywood, *Member, IEEE*
Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada

Abstract – Hierarchical SOMs are applied to the problem of host based intrusion detection on computer networks. Unlike systems based on operating system audit trails, the approach operates on real-time data without extensive off-line training and with minimal expert knowledge. Specific recommendations are made regarding the representation of time, network parameters and SOM architecture.

Keywords – SOM, Intrusion Detection.

I. Introduction

Defensive information operations and computer intrusion detection systems (IDS) are primarily designed to protect the availability, confidentiality, and integrity of critical networked information systems. These operations protect computer networks against denial-of-service (DoS) attacks, unauthorized disclosure of information, and the modification or destruction of data. The automated detection and immediate reporting of these events are required in order to provide a timely response to attacks [1].

The two main classes of intrusion detection systems are those that analyze network traffic and those that analyze operating system audit trails. These systems typically use either rule-based misuse detection or anomaly detection. Rule-based misuse detection systems attempt to recognize specific behaviors that represent known forms of abuse or intrusion. On the other hand, anomaly detection attempts to recognize abnormal user behavior [2]. Examples of these techniques include, pattern templates, threatening behavior templates, traffic analysis, state-based detection and statistical methods [1].

In all of these approaches, however, the amount of monitoring data generated is extensive, thus incurring large processing overheads. For instance, threatening behavior templates, as used by general rule-based systems, aim to search/match for any "known abnormal behavior" within the monitored data. This process is often too inefficient to conduct without parallel hardware. In addition, such systems will not be able to identify any "new abnormal behavior". A statistical anomaly detection approach will actually identify the "normal behavior" by mining the monitored behavior of each user (for example, each command that is typed by every user) so that "abnormal behaviors" can be characterized. Such systems unfortunately further increase the processing overheads.

A balance therefore exists between the use of resources and the accuracy and timeliness of intrusion detection information. The objective of the research presented in this paper is to construct a UNIX-based anomaly detection system that will highlight "abnormal behavior" without incurring extensive computational overheads. Unlike the previous works, where every user is monitored, the system uses only "session information" of the users of a host to detect potential intruders or abusers among the "common users" of the system.

The remainder of the paper is organized as follows. Section II introduces the problem addressed by this work and makes a case for solving this problem using an unsupervised learning algorithm. Section III presents the methodology for constructing the system and the testbed on which experiments are performed. Results are presented in section IV and conclusions drawn in section V.

II. Intrusion Detection with Self Organizing Maps

In this work, we aim to automate the process of detecting intrusive actions as much as possible. In order to develop such a system, we first try to identify the "characteristics of the common user" for the target host. This information is then used to raise a flag for any user identified as having a "different characteristic". To achieve this, the framework of figure 1 is followed, in which the core of the approach is to automate the identification of typical user profiles. The first problem is to establish the nature of initial information on which the rest of the system is based. A lot of systems utilize off-line information (UNIX log-files) that, although exhaustive in the information collected, is also rather unwieldy. In the case of this work UNIX "session information" is used as the features of the system and the characteristics of a common user are defined based on this information.

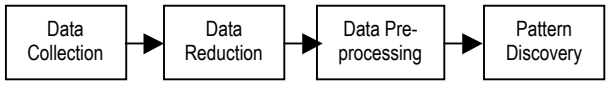

Fig 1. System flowchart.

Steps to achieve Data Reduction and Pre-processing are driven by the needs of the Pattern Discovery component. In this case Pattern Discovery employs an unsupervised learning system – Self Organizing Maps (SOM) – to detect and visualize the characteristics of a common user. Previous research has used this technique on user account log files augmented with the use of additional information detailing the host service used by each user [2]. One of the main differences in the system proposed here is we only use "session information" as the data set. This minimizes resource management and computational considerations, i.e. it can be used as an on-line real-time system. Moreover,

specific recommendations regarding the structure of the Pattern Discovery component are made.

SOMs represent one of the unsupervised learning techniques for data analysis and visualization. They are of particular interest here on account of their efficient update scheme and ability to express topological relationships. This property of an SOM makes it very convenient for expressing relationships between different groups of users. The hypothesis is that typical user characteristics will be emphasized – densely populated regions of the map – whereas atypical activities will appear in sparse regions of the topology.

III. Data Reduction and Pre-processing

The significance of effective pre-processing before presentation to a neural network cannot be over-emphasized. The particular interest in this case is in how the concept of time should be expressed. To this end, two approaches are considered. In the first approach, the feature "time" is implicitly input to the system using a "first-in-first-out" sliding buffer algorithm. This is similar to the Finite Impulse Response structure employed in digital filters [3] (except that samples in this case do not appear at equally spaced intervals). Hereafter this is referred to as the implicit scheme. In the second approach, "time" is explicitly fed in as part of the input pattern; hence this approach is hereafter referred to as the explicit scheme. This scheme has found widespread use in online pattern recognition problems such as signature recognition [4]. In the following the parameters of the data set are summarized, the nature of the feature vectors detailed and SOM architectures identified for each of the above schemes for encoding temporal relations.

A. Data Collection, and Reduction

Data was extracted from the log files of the undergraduate UNIX host at the authors' institution. This host contains nearly 2300 user accounts, and is used primarily by undergraduate students for email and course work. However, faculty, graduate students, staff, and students from other faculties also have access to the machine.

For both approaches, the data used to train the system spans a consecutive 11-day period and contains information on approximately 12000 sessions. Table I shows the default session information collected by the operating system. Use of expert knowledge (system administrators') is used to select a subset of this information for composing the training vector on which networks are trained. That is:

1) The identity of the user that logged in or tried to log in, labeled by a UNIX username;
2) The host from which the connection was made;
3) The type of connection that was made, whether it was telnet, rlogin, or created when the user opened a host window after already having logged into a Faculty of Computer Science workstation;
4) The time that the session was started;

TABLE I
DEFAULT FIELDS OF THE SESSION INFORMATION FILE

FIELD NUMBER	FIELD NAME
1	User name
2	Connection type
3	Dev
4	Pid
5 – 10	Time - connection made
11 – 16	Time - logged in
17 – 22	Time - connection terminated
23	Host
24	Class

B. SOM Architecture

A hierarchical SOM architecture consisting of two levels is used. The first level is made up of three maps, with each map summarizing one feature. The first map in the first level is based on the location that the connection is made from. The second map is based on the user account from which the log in connection is made. The final map in the first level is based on the connection type. As indicated above, the difference between the two approaches is the manner in which "time" information is expressed (appears in each first layer SOM).

The first level maps provide a concise summary of features in the three input domains with respect to time. The second level map is composed from a single network, receiving input from the three preceding layer maps, and is therefore responsible for combining results into a unified summary of user activity. Based on this result, network managers decide whether a particular session represents activity that could be considered "abnormal", i.e., worth further investigation.

1. Data Pre-processing for the First Level SOMs

Raw data collected from the log files is not in a format amenable for direct input into the first level SOMs. Certain items of data, such as the usernames used to log into the host, are of a string format. These require expression as a numerical value. In other cases, there was an excess of information. For example, in the case of domain names, knowing the top two levels of a name was deemed sufficient for identifying general trends. Maintaining only this information provided a significant reduction to the total number of domain names encoded. Finally, the dynamic ranges of the parameters input into the same SOM were normalized. After the input vectors were transformed to facilitate training, the patterns to be fed to the maps were generated. There were several differences in the pre-processing of the data for the encoding of temporal information, resulting in the two approaches, as indicated above.

(a) Implicit representation of time:

Grouping and enumeration is used to convert string data types into numeric values. The values for location consisted of domain names, IP addresses, and machine names.

Domain names were enumerated based on the top two levels of the name. That is, all the domain names with the same top two levels are lumped under the same group and given a single (group) number. The IP addresses were enumerated similarly, but based on the first byte. Each machine name received a number on its own. Since the number of locations was in the hundreds, the final values were scaled to reduce their dynamic range. The user data was enumerated based on the user group to which a login was directed. Six groups were defined: undergraduate, graduate, faculty, math/stat, tech-staff, and other. The first five are actual UNIX groups on the host, and the last acted as a catch all for remaining users. Finally, the enumeration for the connection data consisted of mapping the three connection types to distinct numerical values.

Patterns input to the first level SOM were formed using a first-in-first-out (FIFO) buffer. The algorithm is applied to each feature separately. For a window size of n, the basic algorithm proceeds as follows:

1) Session information is forwarded to the FIFO in chronological order;
2) Once all 'n' locations of the FIFO are full, the first pattern is formed. The corresponding SOM sees all 'n' entries;
3) Whenever new session information is collected it is inserted in the first FIFO position (i), remaining entries shuffled one position, and the oldest entry ($i + n - 1$) is lost. The SOM is now said to see a new pattern.

The structure provides a way to implicitly code sequence information – as opposed to explicit temporal stamps. If values for a parameter are input into the SOM one at a time, there is no way the SOM can reflect how different values relate to each other within a particular time frame. A sequence of depth n, on the other hand, shows how the values are ordered. This method only shows the order of occurrences, and not their frequency.

For the actual implementation, a window depth of five was used ($n = 5$). In addition, to increase the time frame covered by each pattern, a slight variation on the algorithm was used. Instead of taking five consecutive values from a sorted list of values for each pattern, every other value was taken. That is, for the first pattern, the first, third, fifth, seventh, and ninth values were taken. For the second pattern, the second, fourth, sixth, eighth, and tenth values were taken. (This is equivalent to a 2n FIFO with taps at every other location.)

(b) Explicit representation of time:

Temporal information is expressed in log files as six fields: year, month, day, hour, minute, and second. A lot of this information is unnecessary for the purpose of intrusion detection. For example, attempts at hacking into the system are unlikely to span years, so the year field is considered to be insignificant.

Three fields are employed to express information to the SOM. The first was either a value for the location, the user, or the connection for the session, depending on the map that the patterns were used to train. The second field indicates the hour in which a session began, whilst the third field represents the minute that the session began. Any more temporal information beyond the hour and minute was deemed unnecessary for the purpose of intrusion detection. On the one hand, login attempts associated with intrusions would most likely fall within hours of each other. On the other, consecutive login attempts within seconds of each other would still be recognized because they would fall under the same minute.

The values for the minute field ranged between zero and fifty-nine. In order to better match this range, the values for the hour field is wrapped around at seventy-two instead of twenty-four. The location, user, and connection values were scaled to match that of the temporal fields. Finally, values for the location data are enumerated slightly differently from the first approach in order to reduce the number of groups to around seventy.

2. Training for the First Level SOMs

Following preprocessing, patterns were fed directly to the first level SOMs. The SOM Toolbox for Matlab was used [5]. For each map, both the dimensions of the map and the length of the training period were considered. A number of trials were dedicated to determining a 'good' size for the map, both with regard to the dimension and actual number of nodes. Once the size of the map was fixed, the map was trained over increasing epoch limits until no significant variation in network weights occurred.

3. Data Pre-processing for the Second Level SOMs

The next step was to use the output of the first level SOMs as input to the second level SOM.

(a) Implicit representation of time:

Each first level map was used to produce a set of vectors that would be used as input to the second level map. These vectors were constructed as follows. On presentation of an input to a layer one map, instead of taking a binary identification of the winning neuron, the Euclidian distance, d, was employed. This was then renormalized to make the closest node represent the largest value and furthest node the smallest value. Thus,

$$d_{ij}` = 1 / (1 + d_{ij}) \quad (1)$$

where d_{ij} is the original distance from input vector i to neuron j. Thus, if i is close to j, the value of $d_{ij}`$ will approach one. For each input vector, each node in the map would have an associated value for $d_{ij}`$. The final output vector was constructed by inserting in position j the value of $d_{ij}`$. This was repeated for all of the input vectors.

Concatenating corresponding vectors, obtained from the first level SOMs, formed the final vector as seen by the second level SOM. The dimension of this final vector was in the range of three hundred, one dimension per node in the lower-level maps.

Before using this as the input to the second layer fields in which the overall variance was small were omitted. The rationale behind this was that the remaining dimensions

would still contain the most vital features of the underlying data. The final dimension of the vector that was input to the second level map was sixty.

(b) Explicit representation of time:

In the second approach, a different method was used to process the output of the first-level SOMs. Given the size of first layer maps, 336 neurons for location, 96 neurons for user, and 625 neurons for connection, prominent features in the first-level maps are identified using a potential function-clustering algorithm [6]. That is to say, nodes are clustered using a clustering algorithm, thus significantly reducing the amount of information passed to the second layer SOM [7].

For each of the first-level maps, the *potential function* clustering algorithm was run several times to establish clustering parameters (α, β, γ_{upper}, γ_{lower}) [6]. A good set of clusters was considered to be one that contained less than ten clusters where these clusters covered the whole data space. Also, the cluster centers had to be sufficiently different so as to avoid redundant information.

Having established cluster centers, nearest neighbor clustering could take place, effectively partitioning the input data into several sets. For each of these sets, the standard deviation was calculated. Knowing the mean and standard deviation for the vectors in each cluster meant that it was now possible to use a Gaussian distribution to measure the level of excitation of each cluster center in response to an input vector. This was how the input vector to the second-level map was constructed. In effect, input vectors close to cluster centers resulted in a high level of excitation, while input vectors far from any cluster center produced negligible levels of excitation.

This was repeated for each of the three first-level maps. In each case, a set of vectors whose dimension was equal to the number of clusters in that map was produced. For each input pattern, the corresponding vectors from each map were concatenated to form the input to the second level SOM. As mentioned before, this was performed to reduce the dimension input to the second-level map.

4. Training for the Second Level SOM

Once transformed, the output of the first-level maps form the input to the single second layer map, where no further transforms are necessary to represent time.

IV. RESULTS

As indicated above, the systems for both approaches are trained on the "session information" collected on our UNIX host over an 11-day period (approximately 12000 patterns). Figure 2 and figure 4 are the maps of the second-level SOM for the first and the second schemes respectively. It is immediately apparent that the first temporal encoding provides a much more regular distribution of SOM nodes.

The regions of the map shown in figure 2 can be labeled roughly as follows. Regions with high y-values correspond to sessions using undergraduate and graduate accounts and sessions using telnet. Regions with low y-values correspond to sessions with various connection and user group types. Regions with low x-values correspond to connections from unusual or rare hosts, while sessions with high x-values correspond to sessions with common hosts (such as machines on campus).

Although this alone can be used to characterize unusual behavior (most connections should fall in the regions with high x- and y-values), a better sign that a session is anomalous is if it falls in the middle regions (circled nodes). That is to say, as seen in figure-3, very few patterns from the training data excite this region (half of the circled nodes are not excited by any of the 12000 plus patterns). Assuming the training data exhibits normal behavior, our hypothesis is that this region then represents anomalous behavior.

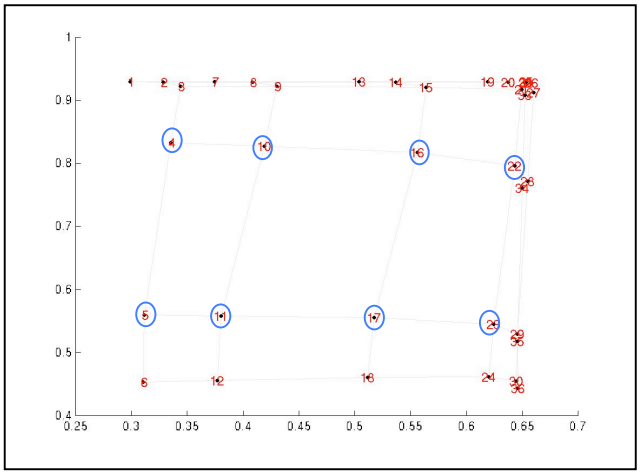

Fig 2. Implicit representation of time – second level SOM

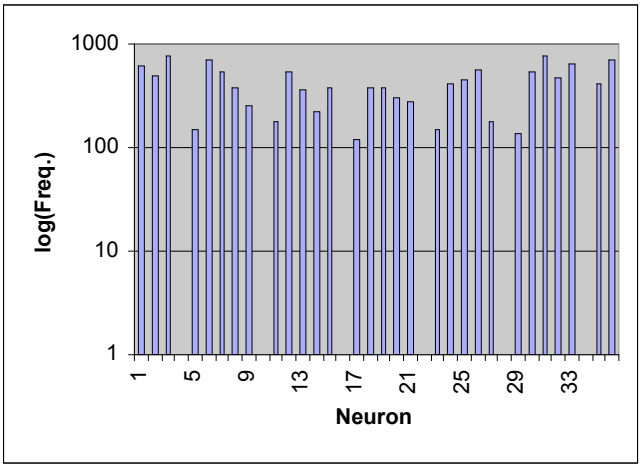

Fig 3. Implicit representation of time – neural pattern frequency in second level SOM (Training Data)

The regions of the map shown in figure 4 cannot be so clearly labeled. Roughly, three extremes can be identified, one near the origin, one in the upper-left, and one in the lower right. All three regions correspond to sessions using undergraduate and graduate accounts. The two regions with lower x-values have connections that use telnet only, while the region with high x-values has various connection types.

The regions near the origin contain more of the uncommon hosts, while the other two regions contain more common hosts. From these descriptions, it was not possible to clearly identify regions that could possibly correspond to anomalous behavior. This effect is attributed to the domination of the input variance by the temporal components, whereas inputs representing other properties are comparatively static. As a consequence, in figure 5, neurons are effectively encoding temporal aspects of the data instead of the other features. This suggests that the second approach results in a system that will not accurately identify intrusions.

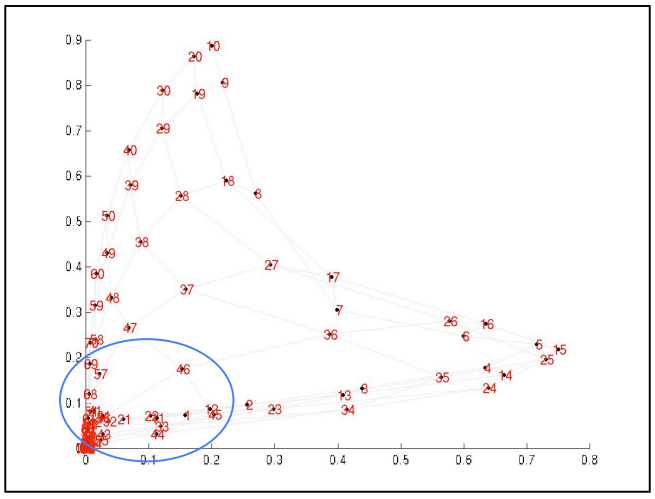

Fig 4. Explicit representation of time – Second level SOM

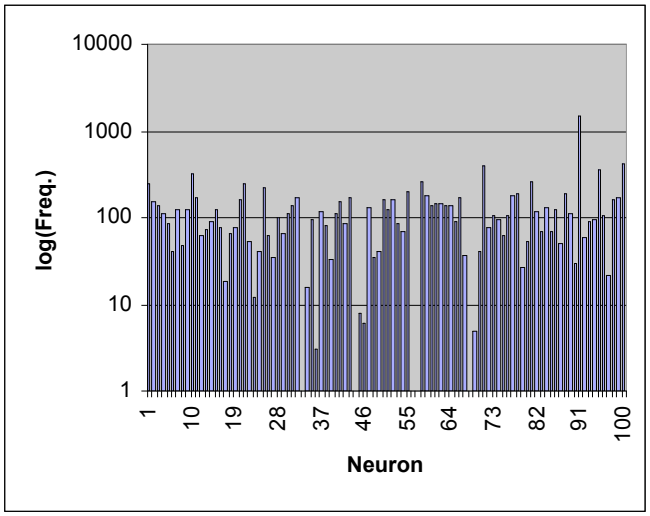

Fig 5. Explicit representation of time – Neural pattern frequency in Second level SOM (Training Data)

Testing was conducted using a data set containing patterns that exhibited potentially "suspicious behavior". This data set was not seen by either of the systems during training. The test data set contained 262 sessions spanning a period of 11 hours. Seven of these sessions represented suspicious behavior (detailed in table II). The results of the test data set are seen in figures 6 and 7. Notice that the peaks in figure 6 often correspond to the hollows in figure 2.

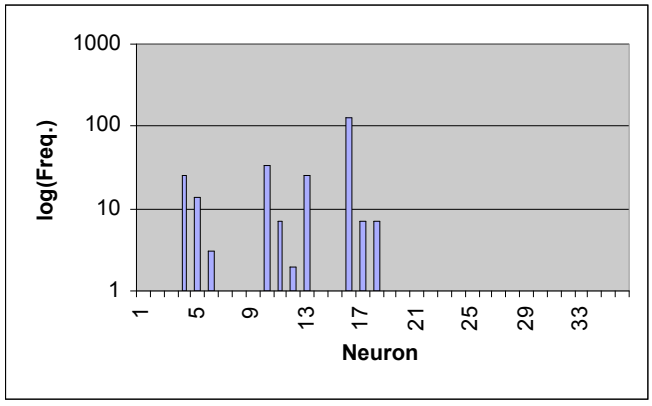

Fig 6. Implicit representation of time – neuron pattern frequency in second level SOM (Test Data)

In the first approach, patterns representing "suspicious behavior" in the test data excited neurons 4, 5, 10, and 11 *only*. That is, of the twenty patterns that contained a component that corresponded to an "abnormal" session, all of them fell in this region. An interesting and unexpected result is that neuron 16, which was excited by none of the training patterns, was excited the most by the test patterns. Despite this anomaly, the fact that the abnormal sessions were restricted to such a narrow area on the map is encouraging.

On the other hand, when we input the same test data set to the second level map of the second approach, the neurons that were excited by the "suspicious behavior" patterns were generally neurons 35, 58, 71, 81, 88 and 95 (see figure 7). It is unclear whether these nodes correspond to suspicious behavior or not because the features in the map are not sharp enough.

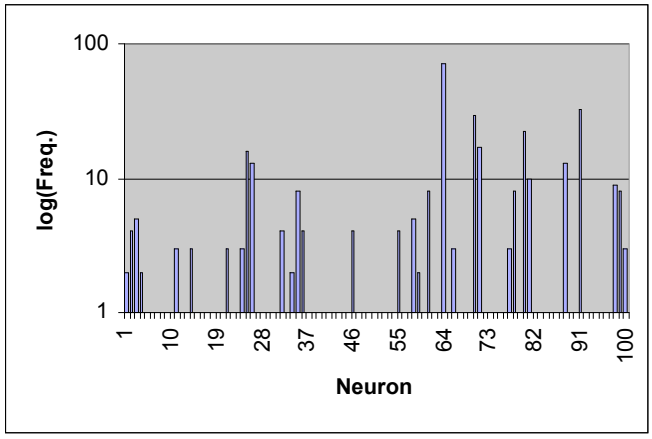

Fig 7. Explicit representation of time – neural pattern frequency in Second level SOM (Training Data)

TABLE II
DESCRIPTION OF TEST DATA SET

DESCRIPTION	Percentage in the Test Set as a whole
Same user connecting (telnet) from different hosts/domains simultaneously.	3%
Same user connecting (telnet) from different hosts/domains at different times.	2%
Different user connecting (telnet) from a different domain simultaneously.	2%

V. CONCLUSION

A system has been developed for aiding network management personnel in the task of computer intrusion detection. Unlike past approaches this method emphasizes the use of host "session information". In contrast to methods based on operating system audit trails, such a scheme significantly reduces computational overheads in identifying a model for 'normal' user profiles. Two methods are investigated for capturing the temporal nature of session information: implicit and explicit. The implicit method uses a FIFO or shift register approach in which each additional event causes the contents of the FIFO to shift along one position. There is therefore no explicit relation to time of day or duration between events. The explicit method on the other hand does provide a time stamp for each event. Contrary to initial expectations, the implicit method for representing time is found to provide a much better separation between user types. In doing so, SOMs trained under an implicit coding of data are demonstrated to provide a much clearer identification of abnormal behaviors.

Future work will apply the technique to a wider cross-section of benchmark problems from the intrusion detection community. In addition we are also interested in the utilization of SOM models capable of incorporating temporal relations in the network itself.

ACKNOWLEDGEMENTS

This work was conducted whilst Peter Lichodzijewski was the recipient of an NSERC USRA grant. Nur Zincir-Heywood and Malcolm I. Heywood both gratefully acknowledge the support of the Individual Research Grants from the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] Bass T., "Intrusion Detection Systems and Multisensor Data Fusion", Communications of the ACM, Vol. 43, No. 4, pp 99-105, April, 2000.

[2] Hoglund A.J., Hatonen K., Sorvari A.S., "A computer Host Based User Anomaly Detection System Using the Self Organizing Map", Proceedings of the International Joint Conference on Neural Networks, IEEE IJCNN 2000, Vol. 5, pp 411-416.

[3] Hamming R.W., Digital Filters, 3rd Edition, Prentice-Hall, 0-13-212895-0, 1989.

[4] Tappert C.C., Suen C.Y., Wakahara T., "The state of the art in on-line handwriting recognition," IEEE Transactions in Pattern Recognition and Machine Intelligence, Vol. 12, pp 787-808, Aug. 1990.

[5] Demuth H., Beale M., Neural Network Toolbox – Users Guide, Version 4. Mathworks Inc., September 2000.

[6] Davé R.N., Krishnapuram R., "Robust Clustering Methods: A Unified View," IEEE Transactions on Fuzzy Systems, 5(2), pp 270-293, 1997.

[7] Vesanto J., Alhoniemi E., "Clustering of the Self-Organizing Map," IEEE Transactions on Neural Networks, 11(3), pp 586-600, 2000.