DynamicWEB: Profile Correlation Using COBWEB

Joel Scanlan, Jacky Hartnett, and Ray Williams

School of Computing, University of Tasmania
Hobart, Australia
{jdscanla, j.hartnett, r.williams}@utas.edu.au

Abstract. Establishing relationships within a dataset is one of the core objectives of data mining. In this paper a method of correlating behaviour profiles in a continuous dataset is presented. The profiling problem which motivated the research is intrusion detection. The profiles are dynamic in nature, changing frequently, and are made up of many attributes. The paper describes a modified version of the COBWEB hierarchical conceptual clustering algorithm called DynamicWEB. DynamicWEB operates at runtime, keeping the profiles up to date, and in the correct location within the clustering tree. Further, as there are a number of attributes within the domain of interest, the tree also extends multi-dimensionally. This allows for multiple correlations to occur simultaneously, focusing on different attributes within the one profile.

Keywords: Data Mining, Clustering Algorithms, Intrusion Detection.

1 Introduction

Fundamentally, data mining is the identification of patterns and relationships within large datasets. It has been implemented in a wide variety of domains, including biomedical research, economics, and security. The security domain, including systems such as those designed for intrusion detection, is the focus of the current research discussed in this paper.

Data mining algorithms and techniques are central to the computational power of many security systems. Automated intrusion detection systems have implemented a long list of data mining techniques with great success. The vast bulk of these systems have focused on the detection of malicious activity through matching a user's activity to a signature, or through detecting a pattern of activity that is inconsistent with a normal behaviour profile. Some of the data mining methods include: Instance-Based Learning, Neural Networks, Bayesian, Genetic Algorithms and Clustering.

The work that is outlined in this paper aims to employ clustering to generate profiles of malicious activity upon a network. This is not solely for the detection of unknown malicious users: it also functions to link existing profiles that may represent the same user operating under different IP addresses. This analysis is carried out using a substantially modified version of the COBWEB hierarchical incremental clustering algorithm [1] named DynamicWEB. The modifications allow for the clustering to occur using a dataset that is continuous in nature, with ever changing profiles, requiring the hierarchical structure to be dynamic at runtime.

A. Sattar and B.H. Kang (Eds.): AI 2006, LNAI 4304, pp. 1059 – 1063, 2006.

The paper will explain the initial target dataset, before describing the Cobweb algorithm, and the modification being made to transform the program into DynamicWEB, will be examined.

1.1 The Data

The system is designed to operate on continuous data, with incoming events being used to update developing profiles. Thus, as the attributes that define each profile change with each update, there is a high likelihood of profiles being assigned and reassigned to different clusters. The mechanism for this updating process will be discussed later in the paper; however it is worth noting that the process occurs at the time the data is reviewed.

The most obvious attributes that is examined in the clustering, and one which is also the most readily available, is the time at which events occur. Indeed, similarities in the timing of network events have acted as a catalyst for this work. There are several aspects of time stamps to examine, such as session length, time between events or sessions, and time of session.

Other attributes that contribute to the profile are less variable, and relate to what the event recorded was about. These include things such as source and destination ports (representing the applications), source and destination IP address, and other alert information such as the kind of attack as labelled by the network device (such as an intrusion detection system).

2 Clustering with DynamicWEB

Clustering algorithms aim to divide data into its natural groups. Often these groups are not known beforehand: their discovery is the objective of the algorithm. There are several kinds of clustering algorithms. Some kinds allow an instance to be in only a single group, while others allow a given instance to be in multiple groups [2]. The clustering technique discussed in this paper is known as Incremental Clustering. It uses a hierarchical tree to sort the instances in a much easier way to visualise and understand. As each instance is added the knowledge of the tree grows which then allow information about the data set to be applied. The research being conducted by the authors uses a modified version of the COBWEB [1] which is an Incremental Clustering algorithm. COBWEB was originally designed to function using nominal attributes but was extended to use numerical in CLASSIT [3]. The method in which the clustering occurs within the tree itself was not modified.

2.1 DynamicWEB

The problem being addressed is one that involves the comparison and possible matching of activity profiles. The profiles, however, are not of a static nature as is the case, for example, when an instance is added to the COBWEB tree. Instead, they are dynamic, and require multiple updates at runtime. COBWEB is not designed to be searched efficiently for a previously clustered instance. The COBWEB tree sorts its contents based from the category utility. This in turn is calculated from the attribute

values within an instance, not from an identifier relating to the instance. Therefore, any search of the COBWEB tree would result in a search length of n.

As the COBWEB tree is not designed to be searched for a particular instance or cluster, it also lacks a correct procedure to modify or delete an instance. In the original COBWEB, even if one could have searched the tree, the concept of modifying an instance in its current location is counter-productive. To modify an instance in its current location would in effect change the category utility of the cluster it resides in. This, without correction, would reduce the accuracy of any future category utility calculations, thus degrading the knowledge stored in the tree. Our extensions to COBWEB are designed to overcome these limitations.

The first extension made to COBWEB in its transformation into DynamicWEB was to create an indexing feature to the tree. Several methods were tested with an AVL tree was the most efficient and was adopted (with a hash table as second). The AVL tree is sorted using an identifier assigned to each profile, with a pointer to its location in the COBWEB tree being stored with it; effectively overlaying the COBWEB structure with a second data structure acting as an index.

Now that an instance can be quickly located within the tree, it can then be updated with new information. Updating an instance in its current location without checking for the resulting change in category utility damages the knowledge currently in the tree. While the change may not be dramatic, the variation may be sufficient for the

COBWEB

```
node 0 [Instances 14]
|   node 1
|   |   node 2
|   |   |   leaf 3            5
|   |   |   leaf 4           10
|   |   leaf 5                4
|   node 6
|   |   leaf 7                7
|   |   leaf 8               12
|   |   node 9
|   |   |   leaf 10 [1]       3
|   |   |   leaf 11 [1]      13
|   |   leaf 12 [1]           9
|   node 13
|   |   node 14
|   |   |   leaf 15           1
|   |   |   leaf 16           8
|   |   leaf 17              11
|   |   node 18
|   |   |   leaf 19 [1]       6
|   |   |   leaf 20 [1]       2
|   |   |   leaf 21 [1]      14

Number of merges: 6
Number of splits: 5
```

DynamicWEB

```
node 0 [Instances 14]
|   node 1
|   |   node 2
|   |   |   leaf 3            1
|   |   |   leaf 4            8
|   |   node 5
|   |   |   node 6
|   |   |   |   leaf 7       14
|   |   |   |   leaf 8        4
|   |   node 10
|   |   |   leaf 11           2
|   |   |   leaf 12          11
|   |   |   leaf 13           7
|   node 14
|   |   node 15
|   |   |   node 16
|   |   |   |   leaf 17       5
|   |   |   |   leaf 18      10
|   |   |   leaf 19           6
|   |   node 20
|   |   |   leaf 22          12
|   |   |   node 23
|   |   |   |   leaf 24       3
|   |   |   |   leaf 25      13
|   |   |   leaf 26           9

Number of merges: 9
Number of splits: 6
```

Fig. 1. The two trees above demonstrate how COBWEB and DynamicWEB have clustered the Weather [4] dataset. To illustrate the change 3 items (2, 6, and 7) were updated within the DynamicWEB tree.

instance to become better suited to another cluster. As such, when an instance is updated, it is removed from the tree and then re-added, thus covering the cases where an instance will be clustered to a different class. However, as this changes the content of the cluster. It has also changed the category utility describing the instances that are stored within the cluster. Several options that allow for the preservation of knowledge within the tree when this happens are now considered, with the best result being adopted. These options include removing the cluster (if it was the only profile in the cluster), merging it with a nearby cluster, or splitting the result into multiple clusters. Further operations are then carried out to confirm that no clusters were left parentless, out of place, or empty. In these cases again merging and splitting is considered, or elevating the profiles within the tree to better fitting clusters. The integrity of the tree, and the knowledge that it retains through these stages is the highest priority.

Figure 1 illustrates the change within a tree when 3 items are updated, being removed from the tree and re-clustered. The left-hand tree is the normal tree that is created using COBWEB. When items 2, 6 and 7 are updated using DynamicWEB the right hand tree results. The updating process did not reduce the clustering accuracy, improving it slightly by 7%. This is a result of the additional 3 merges and split that occurred during the 3 updates that resulted in the slight restructuring of the tree.

3 Multi-dimensional DynamicWEB

The profiles to be correlated within this research contain many attributes, some of which are in subgroups. These subgroups contain attributes that relate to one another, but aren't dependant on other attributes within a different subgroup. For example the various attributes relating to time (time, gap between events, session length) are not frequently related to source and destination port or IP address. These differing attribute groups will in effect act as artificial noise within the tree, causing relationships between profiles not to be recognised. As such the problem space appears to contain several dimensions of attributes which are open to correlation. This problem is likely to be present in other knowledge domains.

The most recent addition to the DynamicWEB is to extend the correlation engine to contain multiple trees, each sorted from different attributes of each instance. The AVL tree index contains a reference for each identity to its location in each of the different trees (Figure 3). This allows for correlation of the different subgroups to occur more efficiently, with the least interference from the other attributes. If one tree then clusters two profiles as being related, the corresponding information for them from the other trees can then be examined. This should highlight further links between the two profiles and allow for an accurate classification.

We envisage that the DynamicWEB approach to profile correlation could well be suited to other multiple attribute continuous datasets. The authors intend to trial the method in domains beyond that of intrusion detection which motivated the research.

4 Conclusion

In this paper a problem space has been outlined containing profiles that change over time. The research goal is to correlate these multi-attribute profiles, to the end of identifying any relationships present between them.

A clustering approach is explored, with the intention of grouping similar profiles together. The method described in this paper is called DynamicWEB, and is built upon the COBWEB incremental clustering algorithm described by Fisher [1]. DynamicWEB allows for instances within the tree to be changed, and re-clustered at runtime allowing it work with live data within the context of intrusion detection.

As the problem space contains a broad range of attributes, DynamicWEB also has the facility to allow for multiple clustering trees to function simultaneously upon the same profile, each focusing on different attributes within the profile.

The method described is not context specific and could be used in other domains requiring similar correlation to occur. It is this that the authors are currently exploring, to enable them to benchmark the system in a comparative qualitative manner.

References

1. Fisher, D.H., *Knowledge Acquisition Via Incremental Conceptual Clustering* Mach. Learn. , 1987 **2** (2): p. 139-172
2. Witten, I. and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java implementations*`. 2000, San Francisco: Morgan Kaufmann Publishers. 371.
3. Gennari, J.H., P. Langley, and D. Fisher, *Models of incremental concept formation* Artif. Intell. , 1989 **40** (1-3): p. 11-61
4. Newman, D.J., et al., *{UCI} Repository of machine learning databases*. 1998, University of California, Irvine, Dept. of Information and Computer Sciences.