Intrusion Detection Systems using Unsupervised Neural Networks

Joseph Asante

Project based on paper:

RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks

- *Amini, Jalili,.& Shahriari*

Abstract

In today's progressively internet-based world, security is increasingly vital in our lives. Consequently, those who wish to act maliciously and take advantage of vulnerabilities in systems become more and more skilled, and now simple rule-based intrusion detection systems are no longer viable. Intrusion detection poses an excellent problem for machine learning to solve. Unsupervised neural networks have been found to be one of the most successful methods of tackling this issue. Prior research done by Amini et. al. demonstrated a viable solution that also works in real time. They tested Self-Organizing Maps (SOM) and Adaptive Resonance Theory (ART-1 and ART-2). This project expands on that research and compares the Growing Neural Gas, which is based on the SOM, to the SOM itself. Experiments showed this solution to be vastly superior in performance (for both training and classification time), with similar accuracy.

Introduction

In today's world, computers have been integrated into nearly every aspect of our lives, and we rely on them more than ever, not only to do the difficult tasks that we could not do before, but also to handle everyday tasks, and even storage. Therefore, security is more important than ever. In the first computer security systems, it was sufficient to write a program with a set of rules that it used to classify every program it encountered. If it encountered something that fit the description of an intruder, or did not abide by the set of rules, it was labeled as an intruder. Today, intrusion software is much more intelligent than it used to be. Likewise, the intrusion detection system must become more intelligent, learning to detect more types of intruders and improving over time as intruders also improve.

To do this, Amini et. al. have examined various machine learning algorithms and have determined that this is best done using an unsupervised learning algorithm. In the original paper, Amini et. al. explored a real-time approach to intrusion detection using unsupervised neural networks. In the paper, they explored three different types of unsupervised nets: Self Organizing Map (SOM), Adaptive Resonance Theory 1 & 2 (ART-1 and ART-2). In their original research, they had more components to their model, which included the Sniffer, Preprocessor, UNN-Engine, and Responder. In short, the Sniffer collected packets over a network, the Preprocessor converted everything into numerical data that the models could use, the UNN-Engine consisted of the unsupervised neural network, and the Responder determined what to do with packets after classification [1]. For this project, I scaled it back and focused on the UNN-Engine, while doing the preprocessing part by hand (as opposed to in code) and parts of the responder (to perform all the runs and statistical analysis. To take the research further, I tested another type of self-organizing map: the Growing Neural Gas (GNG).

Initially, I contacted the original authors asking for source code so I could simply build on top of it. However, this code turned out to be difficult to understand and nearly impossible to build. It was based on an extremely old version of Qt that was difficult to procure. Eventually, I decided it wasn't worth any more effort (after spending a few weeks on that alone). I found an open source implementation online of both Self Organizing Map and Growing Neural Gas, and built on top of it for this project.

Generally speaking, using the SOM as a basis for comparison, I found that the GNG performed similarly in terms of accuracy, but was significantly faster during both training and classification phases.

The goal of this project was to test another type of unsupervised neural network, to determine if there are other types that might work even better than ART-2 (which was determined to be the best in that research). This specific goal was perhaps difficult to achieve, because I could not build the original source code. However, it still provides another comparison on which someone can build later for proper comparisons against all other methods.

Approach

For this model, I used a Growing Neural Gas (GNG) in the UNN-Engine. Based on the Neural Gas introduced by Thomas Martinez and Klaus Schulten, the GNG works exceptionally well for intrusion detection. This itself is based on the Self Organizing Map [3].

The model as a whole consists of the data processor (Preprocessor), the neural network (UNN-Engine), and other components to move data around and perform analysis (Responder). In the UNN-Engine lies the Growing Neural Gas, which simply takes in the data (after normalization) and performs clustering per certain parameters (Table 1). Of course, before data is passed into the UNN-Engine it is turned into canonical numerical form and scaled so no one feature overpowers another. This data was split into training and test sets (80/20 ratio); the neural network learned on the training set and testing was done on the test set.

The Neural Gas is named so due its characteristically gas-like behavior while it "organizes" itself. It varies from the traditional Self-Organizing Map in that weight updates do not depend on the topological structure of the net, but rather the relative distance among the nodes. Described as a "winner take most" rather than "winner take all" algorithm by Martinez and Schulten, the Neural Gas employs "Hebbian learning" rules. When connections between neurons are excited repeatedly, they tend to stay longer than those that are not excited much, in which case they expire. This model essentially provides for a more dynamic solution than Self-Organizing map, as it only keeps connections it uses, all the while gradually reducing learning rate and distance order [3].

The Growing Neural Gas is a modification to the Neural Gas, differing by the fact that its parameters do not change over time, allowing it to learn continuously as it freely adds and removes nodes [2].

The Setup

Data Preprocessing

As mentioned in the original paper, Amini et. al. used the DARPA dataset along with the data collected from their own sniffer. What they found after beginning their research was that the DARPA dataset was too large unwieldy to deal with, so they just stuck to the data collected by their packet sniffer. Similarly, I attempted to use the DARPA dataset but could not use it easily. However, I did not have a packet sniffer, and did not necessarily have to time to create one to collect data. Therefore, I used data from the KDD Cup Dataset (used in Amini et. al.'s paper before this one).

Originally used for a competition to create IDS's using machine learning, the KDD Cup dataset consisted of a csv file with approximately 500,000 samples with 41 features, plus classification of the kind of attack (or normal). Before using it, I shuffled the entire list randomly and then just picked 50,000 samples from the dataset for use, as dealing with 500,000 samples greatly slowed down development. Three of the features were string representations of things like the type of packet, mode of transportation, etc. Of course, the neural networks could not handle string, so to remedy this I assigned each string an integer ID (starting from 0) and replaced all strings as such. I opted to do this by hand on the dataset itself rather than in code during processing because though it took longer initially, it saved time in the long run. After this, I performed "feature scaling" on all the data. This essentially meant reducing each one to sit somewhere between -1 and 1. This was done 1) to ensure that one feature did not overpower another (for those with large units) and 2) to speed up training and convergence time by using smaller numbers. The precise method is represented by this equation:

$$\text{scaled} = (X - \text{mean}(X)) / \text{stddev}(X)$$

where X represents each column in the dataset.

Finally, I split the dataset into an 80/20 ratio of training and test sets respectively. This was done using SciKit Learn's *model_selection* library for cross validation.

Training

First, I began with the SOM as baseline comparison method. Because this was a different implementation of a SOM than what Amini et. al. used, I needed to do my own testing to determine what parameters were "optimal." Due to a simple lack of resources and time (like they had in their research), I used most of their parameters and only tested for an optimal number of epochs. Using these parameters (Table 2), the SOM performed reasonably well in training and classification. I trained it using the training set and determined classifications based on clusters that were formed. The same process was applied to the GNG.

Some of the parameters for the GNG were determined arbitrarily (with guidance from the SOM), such as the size of the network. Other parameters (such as the number of epochs) were determined through experimentation. Results the experiments to determine optimal number of epochs are shown in Figure 1 and 2.

Cluster units	50
Epochs	5

Table 1: Parameters for Growing Neural Gas

Cluster units	2500
Epochs	100
Learning rate	0.5
Neighborhood size	7
Distance measure	Euclidean

Table 2: Parameters for Self-Organizing Map

Classification

After training, all the different clusters formed on each of the neural networks was saved and labeled according to the most abundant training sample on each winning neuron. These were then used to classify items from the test set. All items in the test set were classified, and results reporting overall accuracy on the test set were recorded.

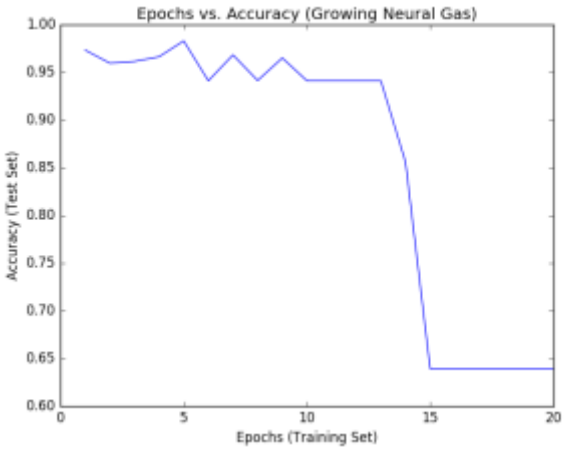

Figure 1: Epochs vs. Accuracy for GNG

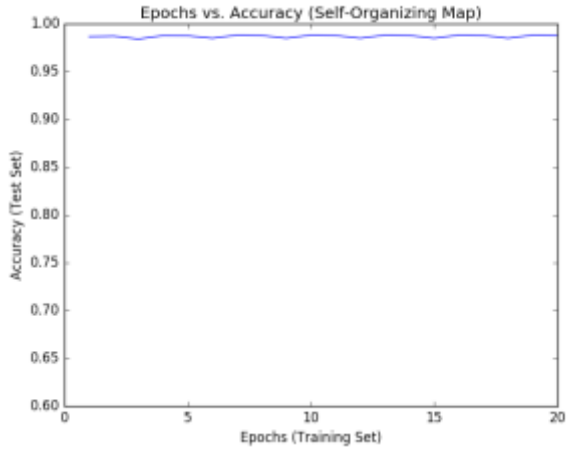

Figure 2: Epochs vs. Accuracy for SOM

Results

After training and experimentation, it was clear that the Growing Neural Gas was vastly superior to the Self-Organizing Map for this purpose. In general, The GNG trained significantly faster, requiring both fewer epochs and less time per epoch. Accuracy was similar between the two. Numerical results are shown in Table 3. Each method was run at least 20 times, and results shown in the table are from the runs with the highest accuracy.

	SOM	GNG
Accuracy (optimal parameters)	98.82%	98.3%
Average time per epoch	125.36s	9.25s
Number of epochs for optimal	8	5
Classification time per sample	1255.990505µs	72.74µs

Table 3: Results

It must be made clear that though results show a slight advantage in accuracy of the SOM over GNG, the difference is too minor to declare GNG definitively better than SOM. Certainly, results from this experiment point to that conclusion, but further, more extensive testing needs to be done on that aspect alone. All experiments used the dataset with 50,000, using 40,000 for training and 10,000 for testing. Though the dataset is large, there are certain degrees of randomness that perhaps call for a significantly larger dataset to differentiate between these small percentages.

Discussion

As shown by the results of experiments, though the SOM slightly outperforms the GNG in accuracy, the GNG vastly outperforms the SOM in the other areas, including training time and classification time. This is especially important because the research done by Amini et. al. was aimed at finding a solution that would work well for real-time intrusion detection. It is clear from this that a Growing Neural Gas is more suitable for a real-time intrusion detection system.

Though we cannot directly compare the results of the Growing Neural Gas here with the results of the ART-2 and ART-1 in the research by Amini et. al., we can still fairly compare the results from the SOM here with their version of the SOM used to classify the data. Datasets were similar, and the concept is the same. Amini et. al. found that the optimal number of epochs for the SOM (given the other optimal parameters) was 100. In my research, I found that I could achieve optimality far earlier, at around

the 8th epoch. This, I believe, can be attributed to the "feature scaling" I perform, which reduces each column to values ranging between -1 and 1. This significantly improved training and classification time, overall improving accuracy. Beyond that, there was not much difference. The accuracy cannot be fairly compared with that minimal of a margin due to the difference in the size of our datasets, the language and environment, and other coding decisions. Generally, we both found that the SOM, while simple and effective, can be incredibly slow compared to other methods.

The results, frankly, are not particularly surprising. We know that the Growing Neural Gas is based on the Self-Organizing Map. In fact, it was designed specifically to address some of the weaknesses in the SOM, one of which is how long it takes to train for large datasets. The very fact that the GNG is dynamic is what provides that incredible advantage. With the SOM, we always have a fixed size, whether we need it or not. Therefore, we could be spending time during training on unnecessary nodes. The GNG allows us to add and remove nodes as needed throughout the training process, so that we only train what we need to train. This explains why the GNG provided such a great speed boost without necessarily an improvement in accuracy. It is essentially the SOM stripped of unnecessary computation.

One other thing I noticed through experimentation was that the SOM did not have an overfitting problem, which makes sense because that means that the clusters converged. After a certain point, accuracy oscillated among 98.82%, 98.78%, and 98.49% (Figure 2). Strangely enough, the GNG did develop an overfitting problem, in which after a certain number of epochs the accuracy of classification drastically reduced (Figure 1). It is not immediately clear whether this is simply random behavior of the model, or whether it is due to something characteristic of the Growing Neural Gas. Further testing and examination might reveal the reason for this. Nevertheless, this is not a pertinent issue because the goal is to reduce the number of epochs necessary for accuracy, which we have achieved.

Conclusion

Intrusion Detection, in today's progressively internet-based world, is more important than ever. Consequently, it is becoming an increasingly difficult problem that can no longer be solved with a simple algorithm or rule-based system; machine learning is one of the best-known approaches to solving this. Unsupervised neural networks have been determined to be the most effective method of machine learning to solve this problem; this was shown to be true, particularly in real-time situations, in the research done by Amini et. al. They tested a Self-Organizing Map and two variants of Adaptive Resonance Theory (Art-1 & Art-2), asking the question that motivated this research: why was a more complex method than SOM not used? Likewise, this question led me to explore some more.

As has been discovered many times in similar research, the Self-Organizing Map works well for this purpose because it is unsupervised, powerful, and simple. Its simplicity, however, presents a weakness in that it is too slow for good real-time detection. As shown by experiments in this project, the Growing Neural Gas addresses that issue precisely, still offering nearly identical accuracy along with all the other advantages, but is significantly faster and lighter-weight. With over 1355% training speed advantage and even more impressive 1726% classification speed advantage over the traditional Self-Organizing Map, the Growing Neural Gas is the clear winner and is vastly superior as the neural network for the UNN-Engine in this system.

Future Work

As mentioned earlier, the initial goal of this project was to take the original source code from the research done by Amini et. al. and add one more unsupervised neural network to test against the others. The difficulty that building and understanding that code presented itself to be an obstacle that could not be overcome, and so different versions of this system were used. Much of the work done in this project, therefore, cannot be fairly compared to the results of earlier research, though it provides an excellent contribution on which a proper comparison can be built.

Future work for this project includes either implementing a Growing Neural Gas in the original C code from earlier research, or re-implementing ART-1 and ART-2 in this python code to properly compare. Moreover, due to time and resource constraints, I could not perform extensive testing to determine *all* optimal parameters for the neural networks, so this is something that can and should be done in the future.

Lastly, some future work for this research would be to find the remaining "optimal" parameters for the GNG. For example, the number of cluster units chosen initially was fairly arbitrary, based on that of the SOM. Perhaps by adding more cluster units, one could match accuracy of the SOM with the GNG, or even surpass it. Of course, this would eat into the speed advantage, but not enough to justify not doing it.

References

[1] Amini, M., Jalili, R., & Shahriari, H. R. (2006). RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. Computers & Security, 25(6), 459–468. doi:10.1016/j.cose.2006.05.003

[2] Fritzke, Bernd (1995). "A Growing Neural Gas Network Learns Topologies". *Advances in Neural Information Processing Systems*. 7: 625–632.

[3] Thomas Martinetz and Klaus Schulten (1991). "A "neural gas" network learns topologies" (PDF). *Artificial Neural Networks*. Elsevier. pp. 397–402.

Links

Full source code:

https://github.com/jnasante/IDS

Kohonen library used in this project:

https://github.com/lmjohns3/kohonen